University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

Undirected connectivity and its space complexity

Théo Borém Fabris

Final Essay

mac 499 — Capstone Project

Supervisor: Prof. Dr. Yoshiharu Kohayakawa

During the development of this work, the author received
financial support from São Paulo Research Foundation (FAPESP) – grant #2022/09280-5

São Paulo
2023

Acknowledgments

Do. Or do not. There is no try.

— Mestre Yoda

I would like to thank Yoshiharu Kohayakawa for advising me since my second semester and for suggesting the study of the topic of this work. I would also like to thank everyone who taught me fundamental concepts in computer science and mathematics. In particular, I am grateful to Marcel Kenji for his amazing course and lecture notes on spectral graph theory, and to Susanna de Rezende for advising me on several topics in computational complexity (though not related to space complexity and this work). Finally, I want to thank my parents for supporting me throughout my life.

Abstract

Théo Borém Fabris. **Undirected connectivity and its space complexity**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

This work aims to investigate two logspace algorithms for the undirected connectivity problem. One of them is a randomized algorithm based on the cover time for random walks on graphs, while the other is a deterministic algorithm based on properties of expander graphs and the zig-zag product. Additionally, this work analyzes a probabilistic construction of universal traversal sequences for graphs using an upper bound on the cover time of random walks on graphs. This work provides most of the theoretical background needed to analyze those algorithms and pseudocode implementations for them. In addition, this work addresses several details omitted in the papers that proposed those algorithms.

Keywords: The undirected *st*-connectivity problem. Space complexity. Random walks on graphs. Spectral graph theory. The zig-zag product of graphs.

Resumo

Théo Borém Fabris. **Conexidade não dirigida e sua complexidade de espaço**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

O objetivo deste trabalho é analisar dois algoritmos com uso de memória logarítmica que resolvem o problema da conexidade em grafos não dirigidos. Um deles é um algoritmo aleatorizado baseado no estudo do tempo de cobertura para passeios aleatórios em grafos, já o outro é um algoritmo determinístico baseado em propriedades de grafos expansores e do produto zig-zag. Além disso, este trabalho examina uma construção probabilística de sequências transversais universais para grafos utilizando uma cota superior para o tempo de cobertura para passeios aleatórios em grafos. Este trabalho provê a maioria dos prerequisitos e resultados básicos necessários para analisar esses algoritmos e implementações em pseudocódigo deles. Além disso, este trabalho considera vários detalhes omitidos nos artigos que propuseram esses algoritmos.

Palavras-chave: O problema da *st*-conexidade não dirigida. Complexidade de espaço. Passeios aleatórios em grafos. Teoria espectral dos grafos. O produto zig-zag para grafos.

Contents

Introduction

A classical result in computational complexity states that the (directed) st-connectivity problem, i.e., the problem of deciding whether two distinct vertices of a digraph are connected by a directed walk, is a complete problem for the complexity class **NL**, the class of all languages accepted by nondeterministic logspace Turing machines. Hence, any superlogarithmic space lower bound or deterministic logspace algorithm for that problem is sufficient to answer the **L** versus **NL** question. A natural first step to understand the space complexity of the st-connectivity problem is to study the space complexity of its restricted version for graphs, called the undirected st-connectivity problem (USTCON). In this problem, we want to decide whether two distinct vertices of a graph are connected by a walk.

One of the first results concerning the space complexity of USTCON was obtained by Aleliunas, Karp, Lipton, Lovász, and Rackoff [Ale+79]. They analyzed the cover time of random walks on graphs and used it to design randomized algorithms to solve USTCON, decide whether a graph is bipartite, and construct universal traversal sequences. After this paper, USTCON received much attention, aiming to understand how this problem is related to other combinatorial problems [LP82] and if it is possible to derandomize those algorithms via space-bounded pseudorandom generators [Nis92; SZ99; Arm+00]. However, the question of whether USTCON $\in$ **L** remained open until the early 2000s when Reingold [Rei08] proposed the first deterministic logspace algorithm for USTCON. His algorithm converts the input graph into an expander graph and exploits the fact that USTCON is easy to solve in logspace for expanders.

The main tool used by Reingold's algorithm is the zig-zag product of graphs, which is a way to combine two graphs G and H into a graph $G \circledz H$ such that $G \circledz H$ has expansion similar to the expansion of G and H. This product first appeared in Reingold, Vadhan, and Wigderson [RVW02] as a tool to explicitly construct expander graphs, and since then, the zig-zag product has appeared in the solution of many other problems [Cap+02; Din07; TaS17].

This work focuses on the randomized algorithms proposed by Aleliunas, Karp, Lipton, Lovász, and Rackoff [Ale+79] to solve USTCON and construct universal traversal sequences, as well as the deterministic algorithm proposed by Reingold [Rei08] to solve USTCON. In particular, we are interested in understanding the probabilistic and combinatorial techniques used by those algorithms, which range from random walks on graphs and their cover time to spectral graph theory and the zig-zag product.

The main contributions of this work are our presentations and detailing of the main arguments given in those papers, providing the background and details needed to understand the algorithms and their analyses, and our implementations in pseudocode of the studied algorithms.

This work is organized as follows. In Chapter 1, we present the general notation and definitions from linear algebra, computational complexity, and graph theory used throughout this work. In Chapter 2, we define Markov chains, state some of their properties, provide a proof of an upper bound for the cover time of random walks on graphs, and then apply this upper bound to establish the correctness of the randomized algorithms in [Ale+79]. In Chapter 3, we first define concepts and prove two classical results from spectral graph theory. Then, we introduce the zig-zag product and its properties, and use them to analyze the multigraph transformations utilized by Reingold's algorithm [Rei08]. Finally, in Chapter 4, we present some conclusions and open problems related to this work.

Chapter 1

Preliminaries

The goal of this chapter is to introduce the basic notations and definitions used in this work.

1.1 Linear algebra

We denote by $\mathbb{N}$ the set $\{1, 2, \ldots\}$ and by $[n]$ the set $\{1, 2, \ldots, n\}$ for each $n \in \mathbb{N}$. For $x \in \mathbb{R}$, we denote by $\lg(x)$ the base 2 logarithm function of x and by $\ln(x)$ the natural logarithm function of x. We denote by $[0, 1]$ the set $\{\, x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ and by $[0, 1)$ the set $\{\, x \in \mathbb{R} \mid 0 \leq x < 1\}$. For a logical predicate P, we denote

$$[P] := \begin{cases} 1 & \text{if } P \text{ is true,} \\ 0 & \text{if } P \text{ is false.} \end{cases}$$

Let R and S be two sets. We denote by S^R the set of all functions from R to S, by $\binom{R}{2}$ the set $\{\, \{u, v\} \mid u, v \in R, u \neq v\}$, by $R^k := \{\, (r_1, \ldots, r_k) \mid r_i \in R\ \forall i \in [k]\}$ for all $k \in \mathbb{N}$, and by $R^* := \cup_{k \in \mathbb{N}} R^k$. We denote by $\mathbb{R}^n$ the set $\mathbb{R}^{[n]}$ and by $\mathbb{R}^{n \times n}$ the set $\mathbb{R}^{[n] \times [n]}$.

Let U and V be finite sets. We say that an element M of $\mathbb{R}^{U \times V}$ is a $U \times V$ matrix, and denote $M_{i,j} := M(i, j)$ for each $(i, j) \in U \times V$. Similarly, an element v of $\mathbb{R}^U$ is a (column) vector, and $v_i := v(i)$ for each $i \in U$. A matrix $M \in \mathbb{R}^{U \times V}$ is a square matrix if $U = V$, and M is symmetric if it is square and $M_{u,v} = M_{v,u}$ for each $(u, v) \in U \times V$. We denote by $0 \in \mathbb{R}^U$ the vector such $0_u := 0$ for all $u \in U$. For $W \subseteq U$, we denote by 1_W the vector defined as follows: $(1_W)_u = [u \in W]$ for each $u \in U$.

For a matrix $M \in \mathbb{R}^{U \times V}$ and a vector $v \in \mathbb{R}^V$, the product $Mv \in \mathbb{R}^U$ of M and v is the vector defined as follows: for each $i \in U$,

$$(Mv)_i = \sum_{k \in U} M_{i,k} v_k.$$

For W a finite set, $M \in \mathbb{R}^{U \times V}$ and $N \in \mathbb{R}^{V \times W}$, the product $MN \in \mathbb{R}^{U \times W}$ of M and N is the matrix defined as follows: for each $(i, j) \in U \times W$,

$$(MN)_{i,j} = \sum_{k \in U} M_{i,k} N_{k,j}.$$

For a square matrix $M \in \mathbb{R}^{U \times U}$ and $l \in \mathbb{N}$, we define $M^l := M$ if $l = 1$ and, $M^l := (M^{l-1})M$ if $l > 1$. For each $u, v \in \mathbb{R}^U$, we denote by $u^\top$ the transpose vector of u, by $\langle u, v \rangle$ the inner product $\sum_{i \in U} u_i v_i$ of u and v, and by $\|u\|$ the norm $\sqrt{\langle u, u \rangle}$ of u. The following theorem is a basic result from linear algebra.

Theorem 1.1 (Cauchy-Schwarz inequality). *Let U be a finite set. Let $u, v \in \mathbb{R}^U$. Then $(\langle u, v \rangle)^2 \leq \langle u, u \rangle \cdot \langle v, v \rangle$. In other words, the following inequality holds*

$$\left(\sum_{i \in U} u_i v_i \right)^2 \leq \left(\sum_{i \in U} u_i u_i \right) \left(\sum_{i \in U} v_i v_i \right).$$

Let $M \in \mathbb{R}^{U \times U}$ be a square matrix. If $\lambda \in \mathbb{R}$ and $x \in \mathbb{R}^U$ satisfy $x \neq 0$ and $Mx = \lambda x$, we call λ an eigenvalue of M and x a λ-eigenvector of M. The following three theorems are basic result from linear algebra.

Theorem 1.2. *Let U be a finite set and $n := |U|$. If $A \in \mathbb{R}^{U \times U}$ is a symmetric matrix, then A has exactly n eigenvalues and they are all real numbers. In this case, we denote the eigenvalues of A by $\lambda_1(A), \ldots, \lambda_n(A)$ satisfying that*

$$\lambda_1(A) \geq \ldots \geq \lambda_n(A).$$

Theorem 1.3 (Spectral Theorem for symmetric matrices). *Let $M \in \mathbb{R}^{U \times U}$ be a symmetric matrix. Let $n := |U|$ and $l \in \{0, \ldots, n\}$. For every orthonormal set $B_0 := \{v_1, \ldots, v_l\}$ of eigenvectors of M, we can extend B_0 to an orthonormal basis $\{v_1, \ldots, v_n\}$ of eigenvectors of M.*

Theorem 1.4. *Let $M \in \mathbb{R}^{U \times U}$ be a symmetric matrix and $l \in \mathbb{N}$. Then, M^l is a symmetric matrix and, for each $i \in [n]$,*

$$\lambda_i(M^l) = \lambda_i(M)^l.$$

1.2 Computational complexity

In this section, we will define the basic concepts from computational complexity used in this work. We used the book Arora and Barak [AB09] as our basic reference.

Let us start by defining the basic terminology.

Definition 1.5. Let $\Omega := \{\epsilon\} \cup \{0, 1\}^*$ be the set of all strings, where ϵ is the empty string. We say that the size (or length) of a string $x \in \Omega$, denoted by $|x|$, is the length of x (that is, the number of bits required to represent x). A language $L \subseteq \Omega$ is a subset of strings. We also call such sets decision problems.

We say that an algorithm $\mathcal{A}$ solves (or decides) a decision problem L if, for any $x \in \Omega$, the output of the algorithm on input x, denoted by $\mathcal{A}(x)$, is equal to

$$\mathcal{A}(x) = \begin{cases} 1 & \text{if } x \in L, \text{ and} \\ 0 & \text{if } x \notin L. \end{cases}$$

For a function $S : \mathbb{N} \longrightarrow \mathbb{R}$, we say that an algorithm $\mathcal{A}$ uses space $S(n)$ for solving a decision problem L if $\mathcal{A}$ solves L and, for every $x \in \Omega$, the sum of the number of bits (or the size of the variables) that are used by $\mathcal{A}$ in its execution on input x is at most $S(|x|)$. If an algorithm $\mathcal{A}$ uses space $S(n)$ for solving L, we say that $\mathcal{A}$ is an $S(n)$-space algorithm for solving L. ■

We will not give formal definitions for mathematical descriptions of algorithms and their space usage, because that would require to introduce the terminology of Turing machines and their tape usage (which is very useful but also very technical). However, once you know those definitions, it is straightforward (and technical) to convert the informal space complexity analyses presented in this work to the framework of Turing machines. In this work, we will only consider algorithms that, on any input, execute a finite number of steps (i.e., our algorithms always halt).

Most of the questions in computational complexity are related to understanding the amount of resources needed to solve a certain problem. In this work, we are interested in understanding the amount of space required to solve the undirected st-connectivity problem, and, in particular, we want to know whether there is an $O(\lg n)$-space algorithm to solve this problem. Note that an $O(\lg n)$-space algorithm is essentially optimal (up to the constant hidden in the big-Oh notation) since any algorithm needs to use at least $\lg n$ bits to store the index (that ranges from 1 to n) of an element of a string of length n.

Definition 1.6. We say that an algorithm $\mathcal{A}$ is a logspace algorithm for a decision problem L if there is a constant $c \in \mathbb{R}$ such that $\mathcal{A}$ is a $(c \lg n)$-space algorithm for solving L.

Let $f : \Omega \longrightarrow \Omega$ be a function from strings to strings. We say that an algorithm $\mathcal{A}$ is a logspace oracle for f if $\mathcal{A}$ has a special variable $v_{\mathcal{A}}$ such that, for every $x \in \Omega$ and when $\mathcal{A}$ receives x as input, the variable $v_{\mathcal{A}}$ has value equal to $f(x)$ at the end of the execution of $\mathcal{A}$, and the sum of the number of bits (or the size of the variables) that are used by $\mathcal{A}$ in its execution on input x is at most $c \lg |x|$ for some constant c that does not depend on x. ■

We can extend the definition of algorithms to the probabilistic setting, that is, we now allow algorithms to sample independent random bits in their execution, and we only require that their output is the correct answer with probability at least 2/3.

Definition 1.7. A randomized algorithm is an algorithm $\mathcal{A}$ that uses random bits during its execution, so, for every $x \in \Omega$, its output $\mathcal{A}(x)$ on input x is a random variable over the probabilistic space that consider all possibilities of values for the random bits used.

We say that a randomized algorithm $\mathcal{A}$ solves (or decides) a decision problem L if, for any $x \in \Omega$, their output $\mathcal{A}(x)$ is the correct answer with probability (over all possible results for the random bits used) at least 2/3, that is,

$$\begin{cases} \Pr[\mathcal{A}(x) = 1] \geq \frac{2}{3} & \text{if } x \in L, \text{ and} \\ \Pr[\mathcal{A}(x) = 0] \geq \frac{2}{3} & \text{if } x \notin L. \end{cases}$$

For a function $S : \mathbb{N} \longrightarrow \mathbb{R}$, we say that a randomized algorithm $\mathcal{A}$ uses space $S(n)$ for solving a decision problem L if $\mathcal{A}$ solves L and, for every $x \in \Omega$, the sum of the number of bits (or size of the variables) that are used by $\mathcal{A}$ in its execution on input x is at most $S(|x|)$ for all possible results of the random bits used.

If a randomized algorithm $\mathcal{A}$ uses space $S(n)$ for solving L, we say that $\mathcal{A}$ is a randomized $S(n)$-space algorithm for solving L. We say that an algorithm $\mathcal{A}$ is a randomized logspace algorithm for a decision problem L if there is a constant $c \in \mathbb{R}$ such that $\mathcal{A}$ is a randomized $(c \lg n)$-space algorithm for solving L. ■

In our pseudocodes for algorithms, for any $x \in \mathbb{N}$, we use the function $\mathrm{Uniform}(x)$ to uniformly sample an element of the set $[x]$, that is, for each $i \in [x]$, the probability of $\mathrm{Uniform}(x)$ returns i is equal to $1/x$ (if $x = 0$, then the probability of $\mathrm{Uniform}(x)$ returns 0 is one). The return value of each call for that function is stochastically independent from the return value of previous calls.

Another type of algorithm that we will consider are randomized logspace algorithms that construct certain objects. As we do not want to require the objects to have logarithmic length, we need to relax the definition of space usage for those algorithms.

Definition 1.8. An algorithm with output tape is an algorithm $\mathcal{A}$ that has a write-once output tape $(o_1, o_2, \dots)$ indexed by $\mathbb{N}$ such that the algorithm can only set a value to an element o_i, for $i \in \mathbb{N}$, of the output tape if o_i was not already set, and o_{i-1} was set (assume that a_0 is always set). In other words, once a variable is written on the output tape, it cannot be changed, and we can only use the output tape sequentially.

A randomized algorithm with output tape is an algorithm $\mathcal{A}$ that has an output tape and uses random bits during its execution. For each $x \in \Omega$, denote by $\mathcal{A}(x)$ the (random) string $(o_1, \dots, o_i)$ where o_i is the last bit of the output tape set when $\mathcal{A}$ receives x as input. Note that $\mathcal{A}(x)$ is a random vector over the probabilistic space that consider all possibilities of values for the random bits used.

Let $f : \Omega \longrightarrow \Omega$ be a function from strings to strings. We say that a randomized algorithm $\mathcal{A}$ with output tape constructs f if, for every $x \in \Omega$ and when $\mathcal{A}$ receives x as input, the string of bits set on the output tape of $\mathcal{A}$ is equal to the string $f(x)$ with probability (over all possible results of the random bits used) at least 2/3, that is,

$$\Pr[\mathcal{A}(x) = f(x)] \geq \frac{2}{3}.$$

For a function $S : \mathbb{N} \longrightarrow \mathbb{R}$, we say that a randomized algorithm $\mathcal{A}$ uses space $S(n)$ to construct a function f if $\mathcal{A}$ constructs f and, for every $x \in \Omega$, the sum of the number of bits (or size of the variables) that are used by $\mathcal{A}$ in its execution on input x (without considering variable in the output tape) is at most $S(|x|)$ for all possible results of the random bits used.

If a randomized algorithm $\mathcal{A}$ uses space $S(n)$ to construct a function f, we say that $\mathcal{A}$ is a randomized $S(n)$-space algorithm to construct f. We say that a algorithm $\mathcal{A}$ is a randomized logspace algorithm to construct a function f if there is a constant $c \in \mathbb{R}$ such that $\mathcal{A}$ is a randomized $(c \lg n)$-space algorithm to construct f. ∎

In our pseudocodes for algorithms, we use the primitive **output** to set a value in the output tape.

1.3 Graph theory

In this section, we will define the basic concepts from graph theory that are important for the rest of this work. Let us start by defining graphs and multigraphs.

Definition 1.9. A graph G is a pair $(V(G), E(G))$ such that $V(G)$ is a finite set and $E(G) \subseteq \binom{V(G)}{2}$. We call $V(G)$ the set of vertices of G and $E(G)$ the set of edges of G. We usually denote an edge $\{u, v\} \in E(G)$ by uv. For a vertex $u \in V(G)$, we denote by $N_G(u)$ the set $\{\, v \in V(G) \mid uv \in E(G)\}$, and we say that an element of $N_G(u)$ is a neighbor of v in G.

A multigraph G is a triple $(V(G), E(G), \varphi)$ such that $V(G)$ and $E(G)$ are finite sets and φ is a function from $E(G)$ to $V(G) \cup \binom{V(G)}{2}$. We call $V(G)$ the set of vertices of G, $E(G)$ the set of edges of G, and φ the incidence function of G. We say that an edge $e \in E(G)$ is a loop if $\varphi(e) \in V(G)$, and we say that e is a parallel edge if there is some $f \in E(G)$ such that $f \neq e$ and $\varphi(f) = \varphi(e)$. For an edge $e \in E(G)$ that is not a loop, we say that the elements of $\varphi(e) \in \binom{V(G)}{2}$ are the ends of e; for a loop $e \in E(G)$, we say that e has two ends and both are equal to the vertex $\varphi(e)$. We say that an edge e is incident to a vertex v if v is an end of e. We say that two vertices u and v are adjacent if there is an edge $e \in E(G)$ such that u and v are the ends of e.

We define $v(G) := |V(G)|$ and $e(G) := |E(G)|$. For a vertex $v \in V(G)$, we denote by $E(G, v)$ the set $\{\, e \in E(G) \mid v \in \varphi(e)\}$ of edges incident to v, and by $d_G(v) := |E(G, v)|$. We call $d_G(v)$ the degree of v in G. For $d \in \mathbb{N}$, we say that G is a d-regular multigraph if $d_G(v) = d$ for all $v \in V(G)$,

A multigraph $H := (V(H), E(H), \psi)$ is a subgraph of G, denoted by $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, and $\psi(e) = \varphi(e)$ for all $e \in E(H)$.

For a given subset $S \subseteq V(G)$ of vertices of G, the multigraph induced by S in G is the multigraph $G[S] := (S, F, \varphi\restriction_F)$, where $F := \{\, e \in E(G) \mid \varphi(e) \subseteq S\}$ and $\varphi\restriction_F$ is the function restriction of φ to the set F.

We sometimes assume that the vertex set of graphs and multigraphs are equal to the set $[n]$ for some $n \in \mathbb{N}$. Note that any graph G is equivalent to the multigraph $(V(G), E(G), \varphi)$ with $\varphi \colon E(G) \longrightarrow V(G) \cup \binom{V(G)}{2}$ and $\varphi(e) = e$. Thus, all definitions and theorems for multigraphs also hold for graphs.

Let $n, d \in \mathbb{N}$. A multigraph G is an (n, d)-multigraph if G has vertex set $[n]$ and is d-regular. Similarly, a graph G is an (n, d)-graph if G has vertex set $[n]$ and is d-regular. ∎

The following definition specifies a representation of graphs that will be useful to represent the input graph of our graph algorithms.

Definition 1.10. Let $G := (V, E)$ be a graph. The adjacency matrix of G is the matrix $A_G \in \mathbb{R}^{V \times V}$ defined as follows: for each $u, v \in V$,

$$A_G(u, v) := [uv \in E(G)].$$

All graph algorithms analyzed in this work receive the adjacency matrix of the input graph G as the computational representation of G. It is not hard to prove that, given the adjacency matrix of the input graph, one can compute the graph parameters $v(G)$, $e(G)$, $d_G(v)$, and $G(v, i)$ for each $v \in V(G)$ and $i \in [d_G(v)]$ using $O(\log n)$ slot of memory (that is, there is a logspace oracle for those functions).

It is important to note that some multigraph algorithms will use an implicit representation of multigraph via rotational maps, but it will only be important in Chapter 3. ■

Let us now define some special classes of multigraph and subgraphs.

Definition 1.11. Let $G := (V, E, \varphi)$ be a multigraph. We say that G is bipartite if there are two nonempty sets $A, B \subseteq V$ such that $A \cup B = V$ and $A \cap B = \emptyset$, and, for each $e \in E$, we have $\varphi(e) \nsubseteq A$ and $\varphi(e) \nsubseteq B$ (that is, one end of e is in A and the other is in B for all edges $e \in E$). If G is not bipartite, we say that it is nonbipartite.

Let $l \in \{0, 1, \dots\}$. A sequence $w := (w_0, \dots, w_l)$ is a walk on G if $w_i \in V$ for all $i \in \{0, \dots, l\}$, and w_{i-1} and w_i are adjacent for each $i \in [l]$. In this case, we say that w is a (w_0, w_l)-walk and w has length l. We denote by $V(w)$ the set $\{\, w_i \mid i \in \{0, \dots, l\}\}$ of vertices visited by w.

For two vertices u and v of G, denote by $d_G(u, v)$ the smallest length of a uv-walk on G; if there is no uv-walk on G, we define $d_G(u, v) := \infty$. We call $d_G(u, v)$ the distance between u and v. The diameter of G is

$$\operatorname{diam}(G) := \max_{u,v \in V} d_G(u, v).$$

We say that G is connected if $\operatorname{diam}(G) < \infty$. A connected component of G is a maximal subgraph of G (with respect to the subgraph $\subseteq$ relation) that is connected. We denote by $\mathcal{G}(n, d)$ the set of all connected (n, d)-graphs. ■

The following definition will be important for our algorithms and analyses.

Definition 1.12. Let $l, n, d \in \mathbb{N}$. Let $G := (V, E)$ be an (n, d)-multigraph. We say that a function $L \colon V \times [d] \longrightarrow E$ is a label of G if, for each $v \in V$, the map $L_v \colon [d] \longrightarrow E(G, v)$ with $L_v(i) := L(v, i)$ is a bijection. For each $v \in V$ and $i \in [d]$, we call $L(v, i)$ the i-th edge incident to v in G with respect to L, and we say that the other end (that can be equal to v in the case of loops) of $L(v, i)$ is the i-th neighbor of v in G with respect to L. We denote by $\mathcal{L}(G)$ the set of all labels of G.

Let G be a graph with vertex set $[n]$. For each $v \in [n]$ and $i \in [d_G(v)]$, we denote by $G(v, i)$ the i-th smallest (using the usual $<$ relation for $\mathbb{N}$) neighbor of v. We also define $G(v, d_G(v) + 1) := v$. ■

1.4 The undirected st-connectivity problem and universal traversal sequences

Let us now define the two main objects of study of this work: the undirected st-connectivity problem and universal traversal sequences.

Definition 1.13. We define the undirected st-connectivity problem (USTCON) as the following language:

$$\text{USTCON} := \{ \langle G, s, t\rangle | G \text{ is a graph and } s, t \in V(G) \text{ such that } s \neq t \text{ and there is an } st\text{-walk on } G\},$$

where $\langle x \rangle$ denotes the computational representation of x. We will computationally represent a graph G using its adjacency matrix, and vertices of G using a sequence of $\lceil \lg v(G) \rceil$ bits. ∎

Definition 1.14. Let $l, n, d \in \mathbb{N}$. Let G be an (n, d)-graph, and L be a label of G. Let $s \in [d+1]^l$ be a sequence of length l, and $v \in [n]$ be a vertex of G. We denote by $w_{G,L}(s, v) := (w_0, \dots, w_l)$ the walk on G with $w_0 := v$ and, for each $i \in [l]$,

$$w_i := \begin{cases} \text{the } s_i\text{-th neighbor of } w_{i-1} \text{ in } G \text{ with respect to } L & \text{if } s_i \in [d], \\ w_{i-1} & \text{if } i = d + 1. \end{cases}$$

We say that a sequence $s \in [d]^l$ is an (n, d)-universal traversal sequence if, for every $G \in \mathcal{G}(n, d)$, every $L \in \mathcal{L}(G)$, and every $v \in [n]$, we have $V(w_{G,L}(s, v)) = V(G)$. We say that l is the length of the universal traversal sequence s. ∎

Chapter 2

Randomized logspace algorithms

This chapter presents two randomized logspace algorithms proposed by Aleliunas, Karp, Lipton, Lovász, and Rackoff [Ale+79]. One of them (Algorithm 2.1) solves the undirected st-connectivity problem, while the other (Algorithm 2.2) receives two natural numbers n and d as input and constructs an (n, d)-universal traversal sequence of length polynomial in n and d. Our presentation differs from that of Aleliunas, Karp, Lipton, Lovász, and Rackoff [Ale+79] in the sense that we analyse their algorithms without assuming that the input graph is bipartite. This requires more technical arguments, but they closely follow the basic ideas in [Ale+79].

2.1 Markov chains preliminaries

The goals of this section are to define the basic notation used in this chapter and to state some results on probability theory that are relevant to the following sections. We used the book Mitzenmacher and Upfal [MU17] as our basic reference for definitions and standard results on probability theory.

Let us first define finite Markov chains and introduce some notation that we will use throughout this chapter.

Definition 2.1. Let U be a finite set. Let $X := (X_0, X_1, \dots)$ be a sequence of random variables defined over a probabilistic space $(\Omega, \mathcal{F}, \Pr)$. The sequence X is a Markov chain on the set U if the following properties hold:

- for all $\omega \in \Omega$ and for all $t \in \mathbb{N}$, we have $X_t(\omega) \in U$;
- (Memoryless property) for all $t \in \mathbb{N}$ and for all $x_0, \dots, x_t \in U$,

$$\Pr[X_t = x_t | X_{t-1} = x_{t-1}, \dots, X_0 = x_0] = \Pr[X_t = x_t | X_{t-1} = x_{t-1}];$$

- (Time independent property) for all $t \in \mathbb{N}$ and for all $x_t, x_{t-1} \in U$,

$$\Pr[X_t = x_t | X_{t-1} = x_{t-1}] = \Pr[X_1 = x_t | X_0 = x_{t-1}].$$

Throughout this chapter, we usually assume that the probabilistic space $(\Omega, \mathcal{F}, \Pr)$ is implicit and ω is always an element of Ω. We denote by $\mathcal{S}(X)$ the set U of states of X. For $u, v \in U$, we define $p_{u,v} := \Pr[X_1 = v | X_0 = u]$ and $p^t_{u,v} := \Pr[X_t = v | X_0 = u]$ for all $t \in \mathbb{N}$. We say that a vector $\pi \in [0,1]^U$ is a stationary distribution for X if $\sum_{u \in U} \pi(u) = 1$ and, for each $u \in U$,

$$\pi(u) = \sum_{v \in U} \pi(v) p_{v,u}.$$

We define the digraph representation $\mathcal{D}(X) := (V, A)$ of X as follows: $V := \mathcal{S}(X)$ and

$$A := \{ (u, v) \in \mathcal{S}(X) \times \mathcal{S}(X) \mid p_{u,v} > 0 \}.$$

■

Let us now define the hitting time and commute time for state of Markov chains, which are quantities that will play an important role in our analysis.

Definition 2.2. Let $X := (X_0, X_1, \dots)$ be a Markov chain. Let u and v be two states of X. We define the random variable $H_{u,v}$ as follows:

$$H_{u,v}(\omega) := \inf\{ t \in \mathbb{N} \mid X_0(\omega) = u, X_t(\omega) = v \}.$$

Let $h_{u,v} := \mathrm{E}[H_{u,v} | X_0 = u]$. We call the number $h_{u,v}$ the hitting time from u to v and the number $h_{u,v} + h_{v,u}$ the commute time between u and v. ■

The following three theorems provide us sufficient conditions to the existence and uniqueness of a stationary distribution for a Markov chain. We will not give a definition for irreducible or ergodic Markov chain as they are not relevant for our analysis. However, we need to use Theorem 2.5 in Section 2.2. Thus, we give combinatorial conditions for irreducibility (Theorem 2.3) and ergodicity (Theorem 2.4).

Theorem 2.3 (Lemma 7.4 from Mitzenmacher and Upfal [MU17])**.** *Let $X := (X_0, X_1, \dots)$ be a finite Markov chain. Then, X is irreducible if and only if $\mathcal{D}(X)$ is a strongly connected digraph.*

Theorem 2.4 (Corollary 7.6 from Mitzenmacher and Upfal [MU17])**.** *Let $X := (X_0, X_1, \dots)$ be a finite Markov chain. If $\mathcal{D}(X)$ is a strongly connected digraph and has a loop, then X is an ergodic Markov chain.*

Theorem 2.5 (Theorem 7.7 from Mitzenmacher and Upfal [MU17])**.** *Any finite, irreducible, and ergodic Markov chain $X := (X_0, X_1, \dots)$ has a unique stationary distribution $\pi \in [0,1]^{\mathcal{S}(X)}$, and π satisfies*

$$\pi(u) = \lim_{t \to \infty} p^t_{v,u} = \frac{1}{h_{u,u}}$$

for each $u, v \in \mathcal{S}(X)$.

Let us now define random walks on graphs.

Definition 2.6. Let G be a graph. The (lazy) random walk on G is the Markov chain X with the following transition probabilities: for each $u, v \in V(G)$,

$$p_{u,v} := [v \in N_G(u) \cup \{u\}] \frac{1}{|N_G(u) \cup \{u\}|}.$$

■

2.2 Random walks on graphs and their cover time

The goals of this section are to prove Theorem 2.11 and some lemmas used in its proof. Let us start by obtaining an upper bound for the commute time of adjacent vertices of a connected graph.

Lemma 2.7. *Let G be a connected graph and let $X := (X_0, X_1, \dots)$ be a random walk on G. For each $uv \in E(G)$, we have that $h_{u,v} + h_{v,u} \le 8v(G)e(G)$.*

Proof. For every $i \in \{0, 1, 2, \dots\}$, let Y_i be a random vector defined as follows:

$$Y_i := (X_i, X_{i+1}).$$

Note that, for all $w_1z_1, w_0z_0 \in V(G) \times V(G)$, we have

$$\begin{aligned} p_{w_0z_0,w_1z_1} := \Pr[Y_1 = w_1z_1 | Y_0 = w_0z_0] &= \Pr[X_2 = z_1, X_1 = w_1 | X_1 = z_0, X_0 = w_0] \\ &= [w_1 = z_0] \Pr[X_2 = z_1 | X_1 = w_1, X_0 = w_0] \\ &= [w_1 = z_0] \Pr[X_2 = z_1 | X_1 = w_1] \\ &= [w_1 = z_0] p_{w_1,z_1}. \end{aligned}$$

Hence, $Y := (Y_0, Y_1, Y_2, \dots)$ is a Markov chain on the set

$$\{ (w, z) \in V(G) \times V(G) \mid wz \in E(G)\} \cup \{ (w, w) \mid w \in V(G)\} =: U,$$

since it satisfies all properties stated in Definition 2.1:

- Memoryless property: for all $t \in \mathbb{N}$ and for all $y_0, \dots, y_t \in U$ such that $(w_0, z_0) := y_0$, $(w_i, z_i) := y_i$ and $z_{i-1} = w_i$ for each $i \in [t]$ (so the event $\{Y_{t-1} = y_{t-1}, \dots, Y_0 = y_0\}$ is not empty),

$$\begin{aligned} &\Pr[Y_t = y_t | Y_{t-1} = y_{t-1}, \dots, Y_0 = y_0] \\ &\quad = \Pr[X_{t+1} = z_t, X_t = w_t | X_t = z_{t-1}, X_{t-1} = w_{t-1}, (X_{i+1}, X_i) = (z_i, w_i)\ \forall i \in \{0, \dots, t-2\}] \\ &\quad = \frac{\Pr[X_{t+1} = z_t, X_t = w_t | X_{t-1} = w_{t-1}]}{\Pr[X_t = z_{t-1} | X_{t-1} = w_{t-1}]} = \Pr[X_{t+1} = z_t, X_t = w_t | X_t = z_{t-1}, X_{i-1} = w_{t-1}] \\ &\quad = \Pr[Y_t = y_t | Y_{t-1} = y_{t-1}]. \end{aligned}$$

- Time independent property: for all $t \in \mathbb{N}$ and for all $(w_t, z_t) := y_t \in U$ and

$(w_{t-1}, z_{t-1}) := y_{t-1} \in U,$

$$\begin{aligned}
\Pr[Y_t = y_t | Y_{t-1} = y_{t-1}] &= \Pr[X_{t+1} = z_t, X_t = w_t | X_t = z_{t-1}, X_{t-1} = w_{t-1}] \\
&= [w_t = z_{t-1}] \frac{\Pr[X_{t+1} = z_t, X_t = w_t | X_{t-1} = w_{t-1}]}{\Pr[X_t = z_{t-1} | X_{t-1} = w_{t-1}]} \\
&= [w_t = z_{t-1}] \frac{\Pr[X_2 = z_t, X_1 = w_t | X_0 = w_{t-1}]}{\Pr[X_1 = z_{t-1} | X_0 = w_{t-1}]} \\
&= \Pr[X_2 = z_t, X_1 = w_t | X_1 = z_{t-1}, X_0 = w_{t-1}] \\
&= \Pr[Y_1 = y_t | Y_0 = y_{t-1}].
\end{aligned}$$

Furthermore, the Markov chain Y is irreducible and ergodic. The first fact follows from Theorem 2.3, since G is connected and X is a random walk on G (note that, for every $(w_0, z_0) := y_0, (w_1, z_1) := y_1 \in \mathcal{S}(Y)$, there is a walk on G between z_0 and w_1, so $\mathcal{D}(Y)$ is strongly connected). While the second follows from Theorem 2.4, since $\mathcal{D}(Y)$ is a strongly connected digraph with loops (as X is a lazy random walk on G).

By Theorem 2.5, we obtain that Y has a unique stationary distribution π and it satisfies $\pi(wz) = 1/h_{wz,wz}$ for each $wz \in U$. Let us prove that the distribution π_U defined as follows is a stationary distribution of Y:

$$\pi_U(wz) = \frac{1}{|U|} \text{ for each } wz \in U.$$

It follows from the following calculations: for each $wz \in U$,

$$\begin{aligned}
\sum_{ab \in U} \pi_U(ab) p_{ab,wz} &= \frac{1}{|U|} \sum_{ab \in U} [w = b] p_{w,z} = \frac{1}{|U|} p_{w,z} \sum_{a \in N_G(w) \cup \{w\}} 1 \\
&= \frac{1}{|U|} p_{w,z} |N_G(w) \cup \{w\}| = \frac{1}{|U|} \frac{|N_G(w) \cup \{w\}|}{|N_G(w) \cup \{w\}|} \\
&= \frac{1}{|U|} = \pi_U(wz).
\end{aligned}$$

Hence, $\pi = \pi_U$ and

$$h_{wz,wz} = 1/\pi(wz) = |U| = 2e(G) + v(G).$$

For each $uv \in E(G)$, let us now find a lower bound for the value of $h_{uv,uv}$. By the definition of $H_{uv,uv}$, we get that, for each $t \in \mathbb{N}$ with $t > 1$,

$$\begin{aligned}
\Pr[H_{uv,uv} = t | Y_0 = uv] &= \Pr[Y_t = uv, Y_j \neq uv \; \forall j \in [t-1] | Y_0 = uv] \\
&= \Pr[X_{t+1} = v, X_t = u, (\{X_{j+1} \neq v\} \cup \{X_j \neq u\}) \; \forall j \in [t-1] | X_0 = u, X_1 = v] \\
&\geq \Pr[X_{t+1} = v, X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = u, X_1 = v] \\
&= p_{u,v} \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = u, X_1 = v].
\end{aligned}$$

By the memoryless and time independent properties of X,

$$\begin{aligned}\Pr[H_{uv,uv} = t | Y_0 = uv] &\geq p_{u,v} \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = u, X_1 = v] \\ &= p_{u,v} \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_1 = v] \\ &= p_{u,v} \Pr[X_{t-1} = u, X_j \neq u \; \forall j \in \{0, \dots, t-2\} | X_0 = v].\end{aligned}$$

As $uv \in E(G)$, we get that $u \neq v$ and then

$$\Pr[H_{uv,uv} = t | Y_0 = uv] \geq p_{u,v} \Pr[X_{t-1} = u, X_j \neq u \; \forall j \in [t-2] | X_0 = v].$$

Note that $H_{uv,uv}(\omega) \geq 2$ for all $\omega \in \Omega$, since if $Y_0 = uv$ and $u \neq v$, it is needed at least one step to reach u and then at least one more step to reach v from u. Thus,

$$\begin{aligned}h_{uv,uv} &= \mathrm{E}\left[H_{uv,uv} | Y_0 = uv\right] \\ &= \sum_{t \geq 2} t \Pr[H_{uv,uv} = t | Y_0 = uv] \\ &\geq p_{u,v} \sum_{t \geq 2} t \Pr[X_{t-1} = u, X_j \neq u \; \forall j \in [t-2] | X_0 = v] \\ &= p_{u,v} \sum_{t \geq 1} (t+1) \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = v] \\ &\geq p_{u,v} \sum_{t \geq 1} t \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = v].\end{aligned}$$

As $\{X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = v\} = \{H_{v,u} = t | X_0 = v\}$, we get that

$$\sum_{t \geq 1} t \Pr[X_t = u, X_j \neq u \; \forall j \in [t-1] | X_0 = v] = h_{v,u}.$$

Hence, $h_{uv,uv} \geq p_{u,v} h_{v,u}$ and

$$h_{v,u} \leq h_{uv,uv} / p_{u,v} = (d_G(u) + 1) / \pi(uv) = |U| (d_G(u) + 1) \leq (v(G) + 2e(G)) v(G).$$

As G is connected, we have $v(G) \leq e(G) + 1$, so $h_{v,u} \leq 4e(G)v(G)$. Using an analogous argument, we get that $h_{u,v} \leq h_{vu,vu} / p_{v,u} \leq 4e(G)v(G)$, and, consequently,

$$h_{u,v} + h_{v,u} \leq 8e(G)v(G).$$

■

Let us now define a generalization of the hitting time of pair of vertices for walks.

Definition 2.8. Let G be a graph and let $X := (X_0, X_1, \dots)$ be a random walk on G. For every $\tau \in \mathbb{N}$ and every $u, v \in V(G)$, define the random variable $H^{\tau}_{u,v}$ as follows:

$$H^{\tau}_{u,v}(\omega) := \inf\{\, t \in \mathbb{N} \mid t > \tau, X_t(\omega) = v, X_\tau(\omega) = u \}.$$

Note that $H^0_{u,v} = H_{u,v}$.

Let $l \in \mathbb{N}$, and let $s := (s_0, \dots, s_l)$ be a walk of length l on G. We define the random variable H_s as follows:

$$H_s(\omega) := H^t_{s_{l-1},s_l}(\omega),$$

where we use $t := 0$ if $l = 1$, and $t := H_{(s_0,\dots,s_{l-1})}(\omega)$ if $l > 1$. Let $h_s := \mathrm{E}\left[H_s | X_0 = s_0\right]$. ■

The next lemma states a relationship between the hitting time of walks and of edges. In fact, the inequality could be replaced by an equality if we use a more technical proof.

Lemma 2.9. *Let G be a connected graph and let $X := (X_0, X_1, \dots)$ be a random walk on G. Let $l \in \mathbb{N}$, and let $s := (s_0, \dots, s_l)$ be a walk of length l on G. Then $h_s \le \sum_{i\in[l]} h_{s_{i-1},s_i}$.*

Proof. We will prove this lemma by induction on $l \in \mathbb{N}$.

If $l = 1$, then, by Definition 2.8,

$$h_s = \mathrm{E}\left[H_{(s_0,s_1)} | X_0 = s_0\right] = \mathrm{E}\left[H^0_{(s_0,s_1)} | X_0 = s_0\right] = \mathrm{E}\left[H_{s_0,s_1} | X_0 = s_0\right] = h_{s_0,s_1}.$$

Now suppose that $l > 1$. Let $t \in \mathbb{N}$ and let $r := (s_0, \dots, s_{l-1})$. By the definition of H_s, we have

$$\Pr[H_s = t | X_0 = s_0] = \Pr[\exists t_1 \text{ s.t. } H^{t_1}_{s_{l-1},s_l} = t, H_r = t_1 | X_0 = s_0].$$

By the disjointness of the events $\{H^{t_1}_{s_{l-1},s_l} = t, H_r = t_1 | X_0 = s_0\}$ for distinct values of t_1, we get

$$\Pr[\exists t_1 \text{ s.t. } H^{t_1}_{s_{l-1},s_l} = t, H_r = t_1 | X_0 = s_0] = \sum_{t_1 \in \mathbb{N}, t_1 < t} \Pr[H^{t_1}_{s_{l-1},s_l} = t, H_r = t_1 | X_0 = s_0].$$

Note that

$$\begin{aligned}\Pr[H^{t_1}_{s_{l-1},s_l} = t, H_r = t_1 | X_0 = s_0] = {} & \Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}, H_r = t_1, X_0 = s_0] \\ & \cdot \Pr[X_{t_1} = s_{l-1}, H_r = t_1 | X_0 = s_0] \\ = {} & \Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}] \Pr[H_r = t_1 | X_0 = s_0],\end{aligned}$$

where the first equality follows from $\{X_{t_1} = s_{l-1}\} \supseteq \{H_r = t_1\}$, and the second follows from $\Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}, H_r = t_1, X_0 = s_0] = \Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}]$, since X is memoryless and the event $\{H_r = t_1, X_0 = s_0\}$ does not depend any X_τ with $\tau > t_1$. By Definition 2.8, we get

$$\begin{aligned}\Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}] &= \Pr[\inf\{\, q \in \mathbb{N} \mid q > t_1, X_{t_1} = s_{l-1}, X_q = s_l\} = t | X_{t_1} = s_{l-1}] \\ &= \Pr[X_{t_1} = s_{l-1}, X_t = s_l, X_q \ne s_l \ \forall q \in \mathbb{N}, t_1 < q < t | X_{t_1} = s_{l-1}],\end{aligned}$$

and, using that X is time independent,

$$\begin{aligned}\Pr[H^{t_1}_{s_{l-1},s_l} = t | X_{t_1} = s_{l-1}] &= \Pr[X_{t_1} = s_{l-1}, X_t = s_l, X_q \ne s_l \ \forall q \in \mathbb{N}, t_1 < q < t | X_{t_1} = s_{l-1}] \\ &= \Pr[X_0 = s_{l-1}, X_{t-t_1} = s_l, X_q \ne s_l \ \forall q \in \mathbb{N}, 0 < q < t - t_1 | X_0 = s_{l-1}] \\ &= \Pr[\inf\{\, q \in \mathbb{N} \mid q > 0, X_0 = s_{l-1}, X_q = s_l\} = t - t_1 | X_0 = s_{l-1}] \\ &= \Pr[H^0_{s_{l-1},s_l} = t - t_1 | X_0 = s_{l-1}].\end{aligned}$$

Hence,

$$\Pr[H_s = t|X_0 = s_0] = \sum_{t_1 \in \mathbb{N}, t_1 < t} \Pr[H^0_{s_{l-1},s_l} = t - t_1|X_0 = s_{l-1}]\Pr[H_r = t_1|X_0 = s_0].$$

As a consequence, we obtain that

$$\begin{aligned}
\mathrm{E}\left[H_s|X_0 = s_0\right] &= \sum_{t\in\mathbb{N}} t\Pr[H_s = t|X_0 = s_0] \\
&= \sum_{t\in\mathbb{N}} \sum_{t_1\in\mathbb{N},t_1<t} t\Pr[H^0_{s_{l-1},s_l} = t - t_1|X_0 = s_{l-1}]\Pr[H_r = t_1|X_0 = s_0] \\
&= \sum_{t_1\in\mathbb{N}} \sum_{t\in\mathbb{N},t>t_1} (t_1 + (t - t_1))\Pr[H^0_{s_{l-1},s_l} = t - t_1|X_0 = s_{l-1}]\Pr[H_r = t_1|X_0 = s_0] \\
&= \sum_{t_1\in\mathbb{N}} \sum_{t_2\in\mathbb{N}} (t_1 + t_2)\Pr[H^0_{s_{l-1},s_l} = t_2|X_0 = s_{l-1}]\Pr[H_r = t_1|X_0 = s_0] \\
&= \sum_{t_1\in\mathbb{N}} t_1 \Pr[H_r = t_1|X_0 = s_0] \sum_{t_2\in\mathbb{N}} \Pr[H^0_{s_{l-1},s_l} = t_2|X_0 = s_{l-1}] \\
&\quad + \sum_{t_2\in\mathbb{N}} t_2 \Pr[H^0_{s_{l-1},s_l} = t_2|X_0 = s_{l-1}] \sum_{t_1\in\mathbb{N}} \Pr[H_r = t_1|X_0 = s_0] \\
&\leq \sum_{t_1\in\mathbb{N}} t_1 \Pr[H_r = t_1|X_0 = s_0] \cdot 1 + \sum_{t_2\in\mathbb{N}} t_2 \Pr[H^0_{s_{l-1},s_l} = t_2|X_0 = s_{l-1}] \cdot 1 \\
&= h_r + h_{s_{l-1},s_l}.
\end{aligned}$$

Therefore, by applying induction on r, we obtain that

$$h_s = \mathrm{E}\left[H_s|X_0 = s_0\right] \leq \left(\sum_{i=1}^{l-1} h_{s_{i-1},s_i}\right) + h_{s_{l-1},s_l} = \sum_{i=1}^{l} h_{s_{i-1},s_i}.$$

■

Before proving the main theorem of this section, we need to define some notations and concepts.

Definition 2.10. Let G be a graph and let $X := (X_0, X_1, \dots)$ be a random walk on G. For every $\tau \in \mathbb{N}$, let $V_\tau(\omega) := \{\, X_t(\omega) \mid t \in \{0, \dots, \tau\}\}$. Define the random variable C_G as follows:

$$C_G(\omega) := \inf\{\, t \in \mathbb{N} \mid V_t(\omega) = V(G)\}.$$

For every $u \in V(G)$, let $c_{G,u} := \mathrm{E}\left[C_G|X_0 = u\right]$. Let $c_G := \max_{u\in V(G)} C_{G,u}$. We call the number c_G the cover time of G. ■

The next theorem gives us an upper bound for the cover time for any connected graph.

Theorem 2.11. *Let G be a connected graph and let $X := (X_0, X_1, \dots)$ be a random walk on G. Then $c_G \leq 8v(G)^2 e(G)$.*

Proof. Let $n := v(G)$ and $m := e(G)$. Let $u \in V(G)$ and let T be a spanning tree of G

rooted at u. Let $s := (s_0, s_1, \dots, s_{2n-2})$ be a walk on T with the following properties:

(1) $s_0 = u$ and, $\forall v \in V(G), \exists i \in [2n-2]$ such that $s_i = v$; and
(2) $\forall vw \in E(T)$, there are exactly two indices $i, j \in [2n-2]$ such that $(s_{i-1}, s_i) = (v, w)$ and $(s_{j-1}, s_j) = (w, v)$.

Such a walk can be found using the sequence of vertices visited by the depth first search algorithm applied to the spanning tree T and starting at the vertex u; Alternatively, one can use an Eulerian trail on the symmetric digraph corresponding to T.

Let H_s be the random variable defined in Definition 2.8. Note that, whenever $H_s(\omega) < \infty$, we have $C_G(\omega) \le H_s(\omega)$, since, by Property (1), every vertex of $V(G)$ appears in s, so $V_{H_s(\omega)}(\omega) = V(G)$. Thus, we obtain that

$$c_{G,u} = \mathrm{E}\left[C_G | X_0 = u\right] \le \mathrm{E}\left[H_s | X_0 = u\right] = \mathrm{E}\left[H_s | X_0 = s_0\right].$$

By Lemma 2.9, we obtain that

$$\mathrm{E}\left[H_s | X_0 = s_0\right] \le \sum_{i \in [2n-1]} h_{s_{i-1}, s_i},$$

and, by Property (2) and using that s is a walk on T, we obtain that

$$\sum_{i \in [2n-1]} h_{s_{i-1}, s_i} = \sum_{uv \in E(T)} (h_{u,v} + h_{v,u}).$$

By Lemma 2.7, we get that

$$\sum_{uv \in E(T)} (h_{u,v} + h_{v,u}) \le \sum_{uv \in E(T)} 8v(G)e(G) = (v(G) - 1)8v(G)e(G) \le 8v(G)^2 e(G).$$

■

2.3 An algorithm for the undirected st-connectivity problem

Algorithm 2.1 is a slight modification of the algorithm analysed in [Ale+79]. Note that it is a logspace algorithm, since it only uses variables n and m to store respectively $v(G)$ and $e(G)$, a variable i to count from 0 to $3 \cdot 8n^2 m$, which requires at most $\lg(24n^2 m) \le \lg(24) + \lg(n^4) = \lg(24) + 4\lg(n)$ bits, a variable j to store at most $\lg(n)$ random bits, and a variable v to store an index for a vertex of G, which requires $\lg(n)$ bits.

Let us now prove that it is a valid randomized logspace algorithm, according to Definition 1.7 in Chapter 1.

Theorem 2.12. *Algorithm 2.1 is a randomized logspace algorithm for solving the undirected st-connectivity problem.*

Proof. Let G be an input graph. Let $n := v(G)$ and $m := e(G)$. Let s and t be two distinct

Algorithm 2.1 Logspace randomized algorithm for undirected st-connectivity

Input: a graph G and two distinct vertices s and t.
Output: True if there is an st-walk on G, and False otherwise.

```
1: $i \longleftarrow 1$, $v \longleftarrow s$, $n \longleftarrow v(G)$, $m \longleftarrow e(G)$
2: while $i \leq 3 \cdot 8n^2m$ do
3:     $j \longleftarrow \mathrm{Uniform}(|N(v)| + 1)$
4:     $v \longleftarrow G(v, j)$
5:     if $v = t$ then
6:         return True
7:     $i \longleftarrow i + 1$
8: return False
```

vertices of G. Let $X := (X_0, X_1, X_2, \dots)$ be a random walk on G as in Definition 2.6. Consider that Algorithm 2.1 receives G, s and t as input, and uses the same source of randomness of X. Hence, conditioned on the event $\{X_0 = s\}$, we get that X_i corresponds to the value of the variable v at the end of the i-th iteration of the **while** loop in the algorithm. Let A be the random variable corresponding to the output of the algorithm, and let T be the random variable corresponding to the value of the variable i in the end of the execution of the algorithm. Hence, for any $j \in \mathbb{N}$ and conditioned on $\{X_0 = s\}$, it holds that if $T(\omega) > j$, then $C_H(\omega) > j$ where H is the connected component of s in G.

Suppose that there is no walk on G between the vertices s and t, so the correct output for the algorithm is False. In this case, A will never be equal to True, since the variable v will never be equal to t. As a consequence, we get that $\Pr[A = \text{False}] = 1 \geq 2/3$.

Now suppose that there is a walk on G between s and t. Hence, the correct output for the algorithm is True and s and t are in the same connected component H of G. By Theorem 2.11, we get that a random walk on H take on average at most $8v(H)^2e(H) \leq 8n^2m$ steps to visit all vertices of H starting from an arbitrary vertex of H. Hence,

$$\begin{aligned}\Pr[A = \text{False}] = \Pr[T > 3 \cdot 8n^2m] &\leq \Pr[C_H > 3 \cdot 8n^2m | X_0 = s] \\ &\leq \Pr[C_H > 3c_H | X_0 = s] \leq \Pr[C_H > 3c_{H,s} | X_0 = s],\end{aligned}$$

where C_H, c_H, and $c_{H,s}$ are defined in Definition 2.10. By Markov's inequality, we obtain that

$$\Pr[A = \text{False}] \leq \Pr[C_H > 3c_{H,s} | X_0 = s] < \frac{1}{3}.$$

Thus, $\Pr[A = \text{True}] \geq \frac{2}{3}$.

Therefore, Algorithm 2.1 is a valid randomized logspace algorithm for solving the undirected st-connectivity problem. ■

2.4 An algorithm to construct universal traversal sequences

As defined in Chapter 1, a graph G is an (n, d)-graph if it has vertex set $[n]$ and is d-regular. We denote by $\mathcal{G}(n, d)$ the set of all connected (n, d)-graphs and by $\mathcal{L}(G)$ the set of all labels of a given (n, d)-graph G. Consider the following theorem.

Theorem 2.13. *Let $n, d \in \mathbb{N}$ with $d < n$. Let $\sigma := (\sigma_1, \sigma_2, \dots)$ be a sequence of independent random variables such that, for each $i \in \mathbb{N}$, σ_i is a uniformly chosen element of the set $[d + 1]$. For each $l \in \mathbb{N}$, let $\sigma^l := (\sigma_1, \dots, \sigma_l)$. Let $f(n, d) := 4n^3 d \cdot 2$ and $g(n, d) := (nd + d + 1) \lg n + 2$. Let $l := f(n, d)g(n, d)$. Then*

$$\Pr[\forall G \in \mathcal{G}(n, d), \forall L \in \mathcal{L}(G), \forall v \in [n], V(w_{G,L}(\sigma^l, v)) = [n]] \geq 2/3,$$

where $w_{G,L}$ is the walk defined in Definition 1.14.

Proof. Consider the following lemma.

Lemma 2.14. *Consider the context of Theorem 2.13. Let $G \in \mathcal{G}(n, d)$ be an (n, d)-graph, $L \in \mathcal{L}(G)$ be a label of G and $v \in [n]$. For each $j \in [g(n, d)]$ and for $k := f(n, d) \cdot j$, we have*

$$\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] < 2^{-j}.$$

Let us first use Lemma 2.14 to prove Theorem 2.13. By Lemma 2.14, we know that, for each graph $G \in \mathcal{G}(n, d)$, for each $L \in \mathcal{L}(G)$, and for each $v \in [n]$,

$$\Pr[V(w_{G,L}(\sigma^l, v)) \neq [n]] < 2^{-g(n,d)}.$$

As every (n, d)-regular graph is a graph with exactly $dn/2$ edges, we obtain that

$$|\mathcal{G}(n, d)| \leq \binom{\binom{n}{2}}{nd/2} \leq (n^2)^{nd/2} = n^{nd}.$$

By the definition of label of G, we get that

$$|\mathcal{L}(G)| = nd! \leq n^{1+d}.$$

Thus, by the union bound,

$$\begin{aligned}
&\Pr[\exists G \in \mathcal{G}(n,d), \exists L \in \mathcal{L}(G), \exists v \in [n] \text{ s.t. } V(w_{G,L}(\sigma^l, v)) \neq [n]] \\
&\leq \sum_{G\in\mathcal{G}(n,d)} \sum_{L\in\mathcal{L}(G)} \sum_{v\in[n]} \Pr[V(w_{G,L}(\sigma^l, v)) \neq [n]] \\
&\leq \sum_{G\in\mathcal{G}(n,d)} \sum_{L\in\mathcal{L}(G)} \sum_{v\in[n]} 2^{-g(n,d)} \\
&\leq n^{nd} n^{d+1} n 2^{-g(n,d)} \\
&= 2^{(nd+d+2)\lg n - g(n,d)} \\
&= 2^{-2} < 1/3,
\end{aligned}$$

and $\Pr[\forall G \in \mathcal{G}(n,d), \forall L \in \mathcal{L}(G), \forall v \in [n], V(w_{G,L}(\sigma^l, v)) = [n]] \geq 2/3$. ■

Let us now prove Lemma 2.14.

Proof of Lemma 2.14. By Theorem 2.11, we have that $2c_G \leq 2 \cdot 8v(G)^2 e(G) = 2 \cdot 8n^2 dn/2 = 8n^3 d = f(n,d)$.

Let $W := (W_0, W_1, \dots)$ be a sequence of random variables over the probability space of σ such that $W_0(\sigma) = v$ and, for each $i \in \mathbb{N}$, we have

$$W_i(\sigma) = \begin{cases} \text{the } \sigma_i\text{-th neighbour of } W_{i-1}(\sigma) \text{ in } G \text{ with respect to } L & \text{if } \sigma_i \in [d], \\ W_{i-1}(\sigma) & \text{if } i = d+1. \end{cases}$$

Note that the $w_{G,L}(\sigma^i, v) = (W_0, W_1, \dots, W_i)$ for each $i \in \mathbb{N}$. It is a technical argument to prove that, for each $t \in \{0,1,2,\dots\}$ and for each $u \in [n]$, the sequence $(W_t, W_{t+1}, W_{t+2}, \dots)$ conditioned on the event $\{W_t = u\}$ is equivalent (in some formal sense) to the lazy random walk $(X_0, X_1, \dots)$ on G (defined with respect to ω) conditioned on the event $\{X_0 = u\}$. This fact will play an important role in the proof of this lemma.

We will now prove this lemma by induction on j. Suppose $j = 1$. Using that $(W_0, W_1, \dots)$ conditioned on $\{W_0 = v\}$ is equivalent to a random walk X on G conditioned on $\{X_0 = v\}$ (third equality of the following equation), we get that

$$\begin{aligned}
\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] &= \Pr[V(w_{G,L}(\sigma^k, v)) \neq [n] | W_0 = v] \\
&= \Pr[\{ W_i \mid i \in \{0,\dots,k\}\} \neq [n] | W_0 = v] \\
&= \Pr[\{ X_i \mid i \in \{0,\dots,k\}\} \neq [n] | X_0 = v] \\
&= \Pr[V_k(\omega) \neq [n] | X_0 = v] \\
&= \Pr[C_G(\omega) > k | X_0 = v] \\
&\leq \Pr[C_G(\omega) > 2c_G | X_0 = v] \\
&\leq \Pr[C_G(\omega) > 2c_{G,v} | X_0 = v],
\end{aligned}$$

where V_k, C_G, c_G, and $c_{G,u}$ are defined in Definition 2.10. By Markov's inequality, we obtain that

$$\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] \leq \Pr[C_G(\omega) > 2c_{G,v} | X_0 = v] < 1/2.$$

Now suppose that $j > 1$. Let $h := (j-1)f(n,d)$ (so $k = h + f(n,d)$) and let $\sigma^* := (\sigma_{h+1}, \sigma_{h+2}, \dots, \sigma_k)$. Then

$$\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] \leq \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], V(w_{G,L}(\sigma^*, W_h)) \neq [n] | W_0 = v].$$

For each $u \in V(G)$, we know that

$$\begin{aligned}
&\Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], V(w_{G,L}(\sigma^*, W_h)) \neq [n], W_h = u | W_0 = v] \\
&= \Pr[V(w_{G,L}(\sigma^*, W_h)) \neq [n] | V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u, W_0 = v] \\
&\quad \cdot \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u | W_0 = v] \\
&= \Pr[V(w_{G,L}(\sigma^*, W_h)) \neq [n] | W_h = u] \\
&\quad \cdot \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u | W_0 = v],
\end{aligned}$$

where the last equality follows from the fact that the event $\{V(w_{G,L}(\sigma^*, W_h)) \neq [n]\}$ only depends on the value of W_h and the event $\{V(w_{G,L}(\sigma^h, v)) \neq [n], W_0 = v\}$ does not depend on any variable W_i for $i > h$. Hence,

$$\begin{aligned}
&\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] \\
&\leq \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], V(w_{G,L}(\sigma^*, W_h)) \neq [n] | W_0 = v] \\
&= \sum_{u \in [n]} \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], V(w_{G,L}(\sigma^*, W_h)) \neq [n], W_h = u | W_0 = v] \\
&= \sum_{u \in [n]} \Pr[V(w_{G,L}(\sigma^*, W_h)) \neq [n] | W_h = u] \cdot \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u | W_0 = v].
\end{aligned}$$

Using that, for each $u \in [n]$, the sequence $(W_h, W_{h+1}, \dots)$ conditioned on $\{W_h = u\}$ is equivalent to a random walk X on G conditioned on $\{X_0 = u\}$ (third equality of the following equation), we get that

$$\begin{aligned}
\Pr[V(w_{G,L}(\sigma^*, u)) \neq [n] | W_h = u] &= \Pr[V(w_{G,L}(\sigma^*, u)) \neq [n] | W_h = u] \\
&= \Pr[\{ W_i \mid i \in \{h, \dots, k\}\} \neq [n] | W_h = u] \\
&= \Pr[\{ X_i \mid i \in \{0, \dots, k-h\}\} \neq [n] | X_0 = u] \\
&= \Pr[V_{k-h}(\omega) \neq [n] | X_0 = u] \\
&= \Pr[C_G(\omega) > k - h | X_0 = u] \\
&= \Pr[C_G(\omega) > f(n,d) | X_0 = u] \\
&\leq \Pr[C_G(\omega) > 2c_G | X_0 = u] \\
&\leq \Pr[C_G(\omega) > 2c_{G,u} | X_0 = u],
\end{aligned}$$

and, by Markov's inequality, we obtain that

$$\Pr[V(w_{G,L}(\sigma^*, u)) \neq [n]] \leq \Pr[C_G(\omega) > 2c_{G,u}|X_0 = u] < 1/2.$$

Therefore,

$$\begin{aligned}\Pr[V(w_{G,L}(\sigma^k, v)) \neq [n]] &< \sum_{u\in[n]} (1/2) \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u | W_0 = v] \\ &\leq (1/2) \cdot \sum_{u\in[n]} \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n], W_h = u | W_0 = v] \\ &\leq (1/2) \cdot \Pr[V(w_{G,L}(\sigma^h, v)) \neq [n] | W_0 = v] \leq 2^{-1-(j-1)} = 2^{-j},\end{aligned}$$

where the last inequality follows by induction on $j - 1$. ■

Note that, if we change the function $g(n, d)$ appropriately, we can obtain that, with high probability, a random sequence satisfies the property stated in the conclusion of Theorem 2.13.

Using Theorem 2.13, it is straightforward to prove that Algorithm 2.2 is a valid logspace randomized algorithm that constructs (n, d)-universal traversal sequences, because this algorithm is essentially a generator of a random sequence $s \in [d + 1]^l$ of size $l := f(n, d)g(n, d)$, and ignore elements of s that are equal to $d + 1$, which does not change the property of Theorem 2.13 since those edges with label $d + 1$ correspond to artificial loops on the graph, required for our lazy random walk analysis.

Algorithm 2.2 Logspace randomized algorithm to construct universal traversal sequences

Input: two strings 1^n and 1^d, where $n, d \in \mathbb{N}$ and $d < n$.
Output: An (n, d)-universal traversal sequence.

```
1: $i \longleftarrow 1$, $l \longleftarrow 8n^3d((nd + d + 1)\lg n + 2)$
2: while $i \leq l$ do
3:     $j \longleftarrow$ Uniform$(d + 1)$
4:     if $j \leq d$ then
5:         output $j$
6:     $i \longleftarrow i + 1$
```

Chapter 3

A deterministic logspace algorithm

This chapter presents the deterministic logspace algorithm for the undirected st-connectivity problem proposed by Reingold [Rei08]. This algorithm utilizes the zig-zag product, introduced by Reingold, Vadhan, and Wigderson [RVW02], to transform the input graph into an expander multigraph that, in some sense, preserves the connected components of the input graph, and then solves the undirected st-connectivity problem on this expander. Our presentation is based on the results in Reingold [Rei08], but it differs in that we provide complete proofs for some basic results from spectral graph theory and expand on certain arguments of the main transformation (Section 3.3). In particular, we present self-contained proofs for both the Expander Mixing Lemma for multigraphs and how to use it to establish that expanders have logarithmic diameter. These proof are inspired by proofs presented in Hoory, Linial, and Wigderson [HLW06] in the case of graphs.

3.1 Spectral graph theory preliminaries

The goals of this section are to define the basic notation used in this chapter and to prove some results on spectral graph theory that are relevant to the following sections. Let us start by defining some notation.

Definition 3.1. Let $n, d \in \mathbb{N}$. Let G be an (n, d)-multigraph and let L be a label of G. We define the rotational map $\mathrm{Rot}_{G,L}$ of G with respect to L as the function $\mathrm{Rot}_{G,L} : [n] \times [d] \longrightarrow [n] \times [d]$ such that, for each $u \in [n]$ and $i \in [d]$, $\mathrm{Rot}_{G,L}(u, i) := (v, j)$, where v is the i-neighbour of u and u is the j-neighbour of v, both with respect to L. Thus, we have that, for each $u \in [n]$ and $i \in [d]$, if $\mathrm{Rot}_{G,L}(u, i) = (v, j)$, then $\mathrm{Rot}_{G,L}(v, j) = (u, i)$. We say that a function Rot is a rotational map of G if there is a label L of G such that $\mathrm{Rot} = \mathrm{Rot}_{G,L}$. By Definition 1.12, we can conclude that this label L is unique if it exists for a given function Rot.

Let Rot be a rotational map of G, and let $s \in [d]^l$ be a sequence of length $l \in \mathbb{N}$ and $v \in [n]$ be a vertex of G. We denote by $w_{G,\mathrm{Rot}}(s, v)$ the walk $(w_0, \dots, w_l)$ with $w_0 := v$ and, for each $i \in [l]$,

$$w_i := z, \text{ where } (z, j) = \mathrm{Rot}(w_{i-1}, s_i) \text{ for some } j \in [d].$$

We sometimes call $w_{G,\mathrm{Rot}}(s, v)$ the walk determined by s and v with respect to G and Rot. ■

Recall that all algorithms receive the adjacency matrix of the input graph G as representation of G. Hence, the following fact will be important for Algorithm 3.4.

Fact 3.2. Let G be an (n, d)-graph. Let $L : [n] \times [d] \longrightarrow E(G)$ be a function defined as follows: for each $u \in [n]$ and $i \in [d]$,

$$L(u, i) := uv,$$

where v is the i-th smallest (using the usual $<$ relation for $\mathbb{N}$) neighbor of u; the vertex v is well defined because G has no loops or multiple edges. Thus, the function L is a label of G, since the map $L_u : [d] \longrightarrow E(G, u)$ with $L_u(i) := L(u, i)$ is bijective for every $u \in [n]$.

Note that, for each $u \in [n]$ and $i \in [d]$, one can compute $\mathrm{Rot}_{G,L}(u, i) =: (v, j)$ in the following way: first compute the index v of i-th smallest nonzero element of the u-th row r of A_G; then compute the number j of nonzero elements of the v-th column c with index smaller than or equal to u. This computation of $\mathrm{Rot}_{G,L}(u, i)$ from the adjacency matrix of G only requires $O(\log n)$ memory slots, since we only need to count the number of nonzero elements in r and c (each has exactly d elements) and store the bits of the entry corresponding entries of r and c (each has exactly n entries). ■

Another important aspect of rotational maps is that they can be used to define regular multigraphs.

Definition 3.3. Let $n, d \in \mathbb{N}$. Let $\mathrm{Rot} : [n] \times [d] \longrightarrow [n] \times [d]$ be a bijective function such that, for each $u \in [n]$ and $i \in [d]$, if $\mathrm{Rot}(u, i) = (v, j)$, then $\mathrm{Rot}(v, j) = (u, i)$. We denote by $\mathcal{G}(\mathrm{Rot})$ the multigraph $G := (V, E, \varphi)$ with $V := [n]$,

$$E := \{\, (\{u, v\}, \{i, j\}) \mid u, v \in [n], i, j \in [d] \text{ s.t. } \mathrm{Rot}(u, i) = (v, j)\},$$

and $\varphi(e) := w$ for each $(w, k) := e \in E$. Hence, G is an (n, d)-multigraph. ■

We will now define the adjacency matrix of multigraphs.

Definition 3.4. Let $n, d \in \mathbb{N}$ and let G be an (n, d)-multigraph. Let L be a label of G and $\mathrm{Rot}_{G,L}$ be its corresponding rotational map. The (normalized) adjacency matrix of G is the matrix $A_G \in \mathbb{R}^{n\times n}$ defined as follows: for each $u, v \in [n]$,

$$A_G(u, v) := \frac{1}{d}|\{\, (i, j) \in [d] \times [d] \,:\, \mathrm{Rot}_{G,L}(u, i) = (v, j)\}|.$$

Note that A_G is independent of the choice of L, since, for each label L of G and $u, v \in [n]$, the number $|\{\, (i, j) \in [d] \times [d] \,:\, \mathrm{Rot}_{G,L}(u, i) = (v, j)\}|$ is equal to the number of edges with extremities equals $\{u, v\}$, which only depends on the multigraph G and its incidence function.

As A_G is a symmetric matrix, we obtain, by Theorem 1.2, that all its eigenvalues $\lambda_1(A_G), \dots, \lambda_n(A_G)$ are real numbers. Recall that $\lambda_1(A_G) \geq \dots \geq \lambda_n(A_G)$. We define

$$\lambda_i(G) := \lambda_i(A_G)$$

for each $i \in [n]$, and we denote by $\lambda(G)$ the second largest eigenvalue of A_G in absolute value, that is,

$$\lambda(G) := \max\{|\lambda_2(G)|, |\lambda_n(G)|\}.$$

The number $1 - \lambda(G)$ is usually called the spectral gap of G.

For sets $S, T \subseteq V(G)$, we denote by $e(S,T) := e_G(S,T)$ the number of edges of G with one extremity in S and the other in T, that is,

$$\begin{aligned} e(S,T) &:= |\{\, (u,v,i,j) \in S \times T \times [d] \times [d] \,:\, \mathrm{Rot}_{G,L}(u,i) = (v,j)\}| \\ &= \sum_{u \in S} \sum_{v \in T} |\{\, (i,j) \in [d] \times [d] \,:\, \mathrm{Rot}_{G,L}(u,i) = (v,j)\}| \\ &= d \sum_{u \in S} \sum_{v \in T} (A_G)_{u,v}. \end{aligned}$$

As A_G is independent of L, the value of $e(S,T)$ is also independent of the choice of L.

Furthermore, we have, for each $u \in [n]$,

$$(A_G 1_T)_u = \sum_{v \in [n]} (A_G)_{u,v} (1_T)_v = \sum_{v \in T} (A_G)_{u,v},$$

which is the number of edges between the vertex u and the set T (counting loops if $u \in T$), and

$$\begin{aligned} d\left(1_S^\top A_G 1_T\right) &= d \sum_{u \in [n]} (1_S)_u (A_G 1_T)_u \\ &= d \sum_{u \in S} (A_G 1_T)_u \\ &= d \sum_{u \in S} \sum_{v \in T} (A_G)_{u,v} \\ &= e(S,T). \end{aligned}$$

■

The following theorem collects some properties of the adjacency matrix of multigraphs that will be used in this chapter.

Theorem 3.5. *Let $n, d \in \mathbb{N}$. Let G be an (n,d)-multigraph. Then the following properties hold:*

1. *the vector $u_n := 1_{V(G)}/\sqrt{n}$ is a normal 1-eigenvector of A_G;*
2. *there is an orthonormal basis $B := \{b_1, \dots, b_n\}$ of $\mathbb{R}^n$ such that $b_1 = u_n$ and, for each $i \in [n]$, b_i is a $\lambda_i(A_G)$-eigenvector of A_G (it is a consequence of Theorem 1.3).*
3. *for each $i \in [n]$, we have that $|\lambda_i(G)| \leq 1$;*

4. *if G is a bipartite graph, then $\lambda_n(G) = -1$;*

5. *the multigraph G is a connected, nonbipartite graph if and only if $\lambda(G) < 1$;*

We will now define some classes of multigraphs that will be the main objects of this chapter.

Definition 3.6. Let $n, d \in \mathbb{N}$ and $\lambda \in [0, 1]$. We say that an (n, d)-multigraph G is an (n, d, λ)-multigraph if $\lambda(G) \leq \lambda$. We say that an (n, d)-multigraph G is an (n, d)-expander if $\lambda(G) \leq 0.45$. ■

The following theorem gives us an approximation (and an upper bound for its error) for the number of edges between two sets of vertices of G when G is an (n, d, λ)-graph.

Lemma 3.7 (Expander Mixing Lemma)**.** *Let $n, d \in \mathbb{N}$ and $\lambda \in [0, 1]$. Let G be an (n, d, λ)-multigraph. Then, for every $S, T \subseteq V(G)$, we have*

$$\left| e(S, T) - \frac{d}{n}|S||T| \right| \leq d\lambda\sqrt{|S||T|}.$$

Proof. For each $i \in [n]$, let $\lambda_i := \lambda_i(A_G)$. By Theorem 3.5, we obtain an orthonormal basis $B := \{b_1, \dots, b_n\}$ of $\mathbb{R}^n$ such that $b_1 = u_n$ and, for each $i \in [n]$, b_i is a λ_i-eigenvector of A_G. Let $s, t \in \mathbb{R}^n$ be vector of coefficients such that

$$1_S = \sum_{i \in [n]} s_i b_i \text{ and } 1_T = \sum_{i \in [n]} t_i b_i.$$

Note that

$$s_1 = \langle b_1, 1_S \rangle = \frac{1}{\sqrt{n}} \langle 1_V, 1_S \rangle = \frac{1}{\sqrt{n}} \sum_{i \in [n]} (1_S)_i = \frac{1}{\sqrt{n}} \sum_{i \in [n]} [i \in S] = \frac{|S|}{\sqrt{n}},$$

and $\langle 1_S, 1_S \rangle = \sum_{i \in S} 1 = |S|$. Similarly, we have $t_1 = \frac{|T|}{\sqrt{n}}$ and $\langle 1_T, 1_T \rangle = \sum_{i \in T} 1 = |T|$. Using that B is a set of eigenvectors, we get

$$\begin{aligned}
1_S^\top A_G 1_T &= \left(\sum_{i \in [n]} s_i b_i \right)^\top \left(\sum_{j \in [n]} t_j A_G b_j \right) \\
&= \left(\sum_{i \in [n]} s_i b_i^\top \right) \left(\sum_{j \in [n]} t_j \lambda_j b_j \right) \\
&= \sum_{i \in [n]} \sum_{j \in [n]} s_i t_j \lambda_j b_i^\top b_j \\
&= \sum_{i \in [n]} s_i t_i \lambda_i,
\end{aligned}$$

where the last equality follows from the fact that B is an orthonormal set. Hence,

$$e(S, T) - \frac{d}{n}|S||T| = d 1_S^\top A_G 1_T - d s_1 t_1 \lambda_1 = d \sum_{i \in \{2, \dots, n\}} s_i t_i \lambda_i.$$

By applying Theorem 1.1 (Cauchy-Schwarz inequality) to $\left(\sum_{i\in\{2,\dots,n\}} s_i t_i \lambda_i\right)^2$, we get that

$$\begin{aligned}\left(\sum_{i\in\{2,\dots,n\}} s_i t_i \lambda_i\right)^2 &\le \left(\sum_{i\in\{2,\dots,n\}} s_i^2\right)\left(\sum_{i\in\{2,\dots,n\}} (t_i\lambda_i)^2\right)\\ &\le \lambda(G)^2\left(\sum_{i\in\{2,\dots,n\}} s_i^2\right)\left(\sum_{i\in\{2,\dots,n\}} t_i^2\right)\\ &\le \lambda(G)^2\left(\sum_{i\in[n]} s_i^2\right)\left(\sum_{i\in[n]} t_i^2\right)\\ &= \lambda(G)^2\langle 1_S, 1_S\rangle\langle 1_T, 1_T\rangle = \lambda(G)^2|S||T|.\end{aligned}$$

Therefore, we obtain

$$\left|e(S,T) - \frac{d}{n}|S||T|\right| = \sqrt{\left(d\sum_{i\in\{2,\dots,n\}} s_i t_i \lambda_i\right)^2} \le d\lambda(G)\sqrt{|S||T|} \le d\lambda\sqrt{|S||T|}.$$

■

As an application of Expander Mixing Lemma, we will prove that all (n,d)-expander multigraphs have logarithmic diameter.

Theorem 3.8. *Let $n, d \in \mathbb{N}$, and let G be an (n,d)-expander. Then* $\operatorname{diam}(G) \le c\lg n$ *for* $c := 8/\lg 1.05$.

Proof. For each $u \in [n]$ and $r \in \mathbb{R}$, let

$$B_G(u,r) := \{\, v \in V(G) \mid d_G(u,v) \le r \}.$$

Consider the following lemma.

Lemma 3.9. *Let $\lambda := \lambda(G)$ and $\epsilon := 1/2 - \lambda \ge 0.05$. For each $u \in V(G)$ and $r \in [n]$, if $|B_G(u,r)| \le n/2$, then*

$$|B_G(u,r)| \ge (1+\epsilon)|B(u,r-1)| \ge (1+\epsilon)^r.$$

Let us first use Lemma 3.9 to prove Theorem 3.8. Let $r := \lceil 2\log_{1+\epsilon}(n/2)\rceil$ and let $u \in V(G)$. If $|B_G(u,r)| \le n/2$, then, by Lemma 3.9, we get

$$|B_G(u,r)| \ge (1+\epsilon)^r \ge (1+\epsilon)^{2\log_{1+\epsilon}(n/2)} \ge (n/2)^2 > n/2,$$

a contradiction. Hence, $|B_G(u,r)| > n/2$ for all $u \in [n]$. Therefore, for every $u, v \in V(G)$, we have that $B_G(u,r) \cap B_G(v,r) \ne \emptyset$, and, consequently,

$$d_G(u,v) \le 2r \le 8\log_{1+\epsilon} n \le c\lg n,$$

and $\operatorname{diam}(G) \le c\lg n$. ■

We will now prove Lemma 3.9.

Proof of Lemma 3.9. First note that if $S \subseteq V(G)$ is a subset of $V(G)$ with $|S| \leq n/2$, then, by Theorem 3.7, we have $e(S,S) \leq d|S|^2/n + d\lambda|S| = d|S|(|S|/n + \lambda) \leq d|S|(1/2 + \lambda)$. Thus, for $\overline{S} := V(G) \setminus S$,

$$e(S, \overline{S}) = d|S| - e(S,S) \geq d|S|(1/2 - \lambda) = d|S|\epsilon.$$

As $N(S) := \{\, v \in \overline{S} \mid N(v) \cap S \neq \emptyset \}$ has at least $e(S,\overline{S})/d \geq |S|\epsilon$ elements, we get that

$$|S \cup N(S)| \geq |S|(1+\epsilon).$$

Let us now prove the lemma by induction on r. Let u be an arbitrary element of $[n]$. If $r = 1$, then

$$|B_G(u,r)| = |B_G(u,1)| = 1 + d \geq 1 + \epsilon = (1+\epsilon)|B(u,0)|.$$

Now suppose that $r > 1$ and assume that $|B_G(u,r)| \leq n/2$. Let $B_i := B_G(u,i)$ for each $i \in [r]$. As $|B_r| \leq n/2$ and $B_r \supseteq B_{r-1} \cup N(B_{r-1})$, we get that $|B_{r-1}| \leq n/2$, then, by the discussion above for $S := B_{r-1}$,

$$|B_r| \geq |B_{r-1} \cup N(B_{r-1})| \geq |B_{r-1}|(1+\epsilon).$$

By applying induction on $r-1$, we get

$$|B_r| \geq (1+\epsilon)^{r-1}(1+\epsilon) = (1+\epsilon)^r.$$

∎

Using Theorem 3.8, we will design a deterministic logspace algorithm that solves the undirected st-connectivity problem when the input multigraph G is d-regular and each component C of G is a $(v(C), d)$-expander. The correctness of Algorithm 3.1 and its space usage will be proved in Theorem 3.10.

Algorithm 3.1 Logspace deterministic algorithm for USTCON when the components of the input multigraph are expanders

Input: two naturals n, d, a logspace oracle Rot for a rotational map of a multigraph G and two distinct vertices s and t of G. This algorithm assumes that the multigraph G is an (n,d)-multigraph and each component C of G is a $(v(C), d)$-expander.

Output: True if there is an st-walk in G, and False otherwise.

```
USTCONAlgorithmForExpanders($n$, $d$, Rot, $s$, $t$):
$r \longleftarrow \lceil c \lg n \rceil$ ($c$ is the constant that appears in Theorem 3.8)
for all $e \in [d]^r$ do
    $v \longleftarrow s$, $i = 1$
    while $i \leq r$ do
        $(v, j) \longleftarrow \mathrm{Rot}(v, e_i)$
        if $v = t$ then
            return True
        $i \longleftarrow i + 1$
return False
```

Theorem 3.10. *Algorithm 3.1 is correct and uses at most $O_d(\lg n)$ bits of memory.*

Proof. Suppose that $d \in \mathbb{N}$, a graph G on n vertices, and two distinct vertices s and t of G are given as input for Algorithm 3.1. Let us first argue that the algorithm uses at most $O_d(\lg n)$ bits of memory. The **for** loop requires $O(r \lg d) = O(\lg d \lg n)$ bits to generate and store each $e \in [d]^r$, because $O(r \lg d)$ bits are used to store the current sequence, $O(\lg n)$ bits are suffice to compute the next (with respect to the lexicographically order of $[d]^r$) sequence, and no information used in the current **for** iteration will be used in the next **for** iteration after you computer the next sequence (so the algorithm only remembers the sequence of the next iteration). The **while** loop requires $O(\lg n)$ bits, because it only remembers the index of the last visited vertex v, which requires $O(\lg n)$ bits, and the counter $i \in [r]$, which requires $O(\lg r) \leq O(\lg n)$ bits.

Let us now prove that the algorithm is correct. Note that the algorithm outputs True if and only if there is a sequence $e \in [d]^r$ such that $t \in V(w_{G,\mathrm{Rot}}(s, e))$, where $w_{G,\mathrm{Rot}}(s, e)$ is the walk determined by the vertex s and sequence e with respect to G and Rot (Definition 3.1).

Suppose that s and t are not in the same component of G, so the correct output for the algorithm is False. If the algorithm outputs True, then there is a sequence $e \in [d]^r$ such that $t \in V(w_{G,\mathrm{Rot}}(s, e))$, which implies that the walk $w_{G,\mathrm{Rot}}(s, e)$ has a subwalk in G from s to t. This conclusion is a contradiction, because s and t are not in the same component of G. Therefore, the algorithm outputs False.

Now suppose that s and t are in the same component C of G. By the assumptions of the algorithm, we know that G is a d-regular multigraph such that each component of G is an expander. By Theorem 3.8, we know that $\mathrm{diam}(C) \leq c \lg v(C) \leq c \lg n$. Hence, there is a sequence $e \in [d]^r$ such that $t \in V(w_{R,\mathrm{Rot}}(s, e))$; consequently, the algorithm outputs True. ■

3.2 The zig-zag product of multigraphs

The goals of this section are to define and analyse the spectral properties of the zig-zag product. Let us start by defining the zig-zag product of multigraphs.

Definition 3.11. Let G be an (N, D)-multigraph with rotational map Rot_G and H be a (D, d)-graph with rotational map Rot_H. Let $\mathrm{Rot}_{G ⓩ H} : ([N] \times [D]) \times ([d] \times [d]) \longrightarrow ([N] \times [D]) \times ([d] \times [d])$ be the function defined as follows (see Figure 3.1 for an illustration): for each $(v, a) \in [N] \times [D]$ and for each $(i, j) \in [d] \times [d]$,

$$\mathrm{Rot}_{G ⓩ H}((v, a), (i, j)) := ((w, b), (j', i')),$$

where $(a', i') := \mathrm{Rot}_H(a, i)$; $(w, b') := \mathrm{Rot}_G(v, a')$; and $(b, j') := \mathrm{Rot}_H(b', j)$. As Rot_G and Rot_H are rotational maps, we get that

$$\mathrm{Rot}_{G ⓩ H}((w, b), (j', i')) = ((v, a), (i, j)).$$

The zig-zag product $G ⓩ H$ of G and H is the (ND, d^2)-multigraph $\mathcal{G}(\mathrm{Rot}_{G ⓩ H})$ defined by the rotational map $\mathrm{Rot}_{G ⓩ H}$ (see Definition 3.3), where we implicitly use the bijective mapping $(x, y) \in [A] \times [B] \longrightarrow (B(x-1)+y) \in [AB]$, for each $A, B \in \mathbb{N}$, to map the set $[N] \times [D]$ to $[ND]$ and the set $[d] \times [d]$ to $[d^2]$. In the rest of this chapter, we will use interchangeably this mapping. ■

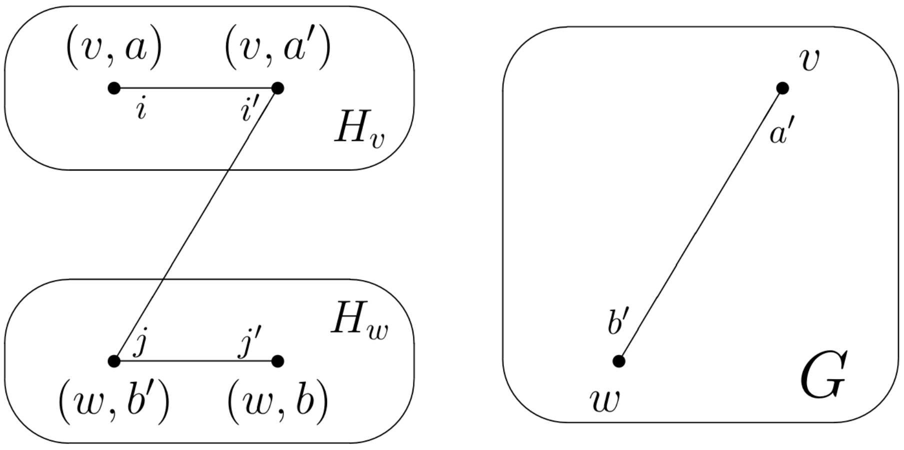

Figure 3.1: *We have the multigraph G on the right and multigraph $G \circledz H$ on the left. We imagine $G \circledz H$ as the graph in which we transform each vertex v of G into a copy $H_v := \{(v, a) \mid a \in V(H)\}$ of the graph H and define exactly d^2 edges to be incident to a given vertex of H_v. For each $(v, a) \in [N] \times [D]$ and $i, j \in [d]$, we define the (i, j)-th edge of a vertex (v, a) in $G \circledz H$ as follows: start at (v, a) inside the cloud H_v; go to (v, a'), where a' is the i-th neighbor of a in H (this first transition inside H_v is sometimes called "zig step" or "small step"); go to (w, b'), where w is the a'-th neighbor of v in G, and v is the b'-th neighbor of w in G (sometimes called "permutation step" or "big step"); go to (w, b), where b is the j-th neighbor of b' in H (sometimes called "zag step" or "small step"); then add an edge between (v, a) and (w, b). Source: Reingold [Rei08]*

Note that the computation of any entry of $\mathrm{Rot}_{G \circledz H}$ requires the value of two entries of Rot_H and one entry of Rot_G. The fact that the zig-zag product is defined by the rotational map $\mathrm{Rot}_{G \circledz H}$ allows us to iterate over the neighborhood of a given vertex using a logspace oracle for $\mathrm{Rot}_{G \circledz H}$, as it is required by Algorithm 3.1. The following theorem states an upper bound for $\lambda(G \circledz H)$ in function of $\lambda(G)$ and $\lambda(H)$.

Theorem 3.12 (Theorem 3.2 from Reingold, Vadhan, and Wigderson [RVW02]). *Let $N, D, d \in \mathbb{N}$ and $\lambda, \alpha \in [0, 1]$. If G is an (N, D, λ)-multigraph and H is a (D, d, α)-multigraph, then*

$$\lambda(G \circledz H) \leq \lambda + \alpha + \alpha^2.$$

Furthermore, $\lambda(G \circledz H) < 1$ whenever $\lambda, \alpha < 1$.

Corollary 3.13. *Let $N, D, d \in \mathbb{N}$ and $\lambda, \alpha \in [0, 1)$. If G is an (N, D, λ)-multigraph and H is a (D, d, α)-multigraph, then the multigraph $G \circledz H$ is connected.*

Proof. By Theorem 3.12, we get that

$$\lambda(G \circledz H) < 1$$

if $\lambda, \alpha < 1$. Hence, by Theorem 3.5, we obtain that $G \circledz H$ is connected. ■

The following theorem improves the upper bound given by Theorem 3.12.

Theorem 3.14 (Theorem 4.3 from Reingold, Vadhan, and Wigderson [RVW02]). *Let $N, D, d \in \mathbb{N}$ and $\lambda, \alpha \in [0,1]$. If G is an (N, D, λ)-multigraph and H is a (D, d, α)-multigraph, then*

$$\lambda(G \text{ⓩ} H) \leq \frac{1}{2}(1-\alpha^2)\lambda + \frac{1}{2}\sqrt{(1-\alpha^2)^2\lambda^2 + 4\alpha^4}.$$

The proofs of Theorem 3.12 and Theorem 3.14 presented in Reingold, Vadhan, and Wigderson [RVW02] are based on the intuition of entropy waves on multigraphs and how the zig-zag product transfers entropy from one cloud of vertices of H to another cloud in each step. They formalize this intuition using algebraic arguments to obtain an upper bound for $\lambda(G \text{ⓩ} H)$. The papers by Reingold, Trevisan, and Vadhan [RTV06] and by Rozenman and Vadhan [RV05] present simplified proofs for similar upper bounds.

In Section 3.3, we will use the following lower bound for the spectral gap of $G \text{ⓩ} H$ rather than Theorem 3.14.

Corollary 3.15 (Theorem 2.9 from Reingold [Rei08]). *Let $N, D, d \in \mathbb{N}$ and $\lambda, \alpha \in [0,1]$. If G is an (N, D, λ)-multigraph and H is a (D, d, α)-multigraph, then*

$$1 - \lambda(G \text{ⓩ} H) \geq \frac{1}{2}(1-\alpha^2)(1-\lambda).$$

Proof. As $\lambda, \alpha \leq 1$, we get that

$$\begin{aligned} \frac{1}{2}\sqrt{(1-\alpha^2)^2\lambda^2 + 4\alpha^4} &\leq \frac{1}{2}\sqrt{(1-\alpha^2)^2 + 4\alpha^2} \\ &= \frac{1}{2}\sqrt{1 - 2\alpha^2 + \alpha^4 + 4\alpha^2} = \frac{1}{2}\sqrt{(1+\alpha^2)^2} \\ &= \frac{1}{2}(1+\alpha^2) = 1 - \frac{1}{2}(1-\alpha^2). \end{aligned}$$

By Theorem 3.14, we obtain

$$\begin{aligned} \lambda(G \text{ⓩ} H) &\leq \frac{1}{2}(1-\alpha^2)\lambda + \frac{1}{2}\sqrt{(1-\alpha^2)^2\lambda^2 + 4\alpha^4} \\ &\leq \frac{1}{2}(1-\alpha^2)\lambda + 1 - \frac{1}{2}(1-\alpha^2) \\ &= 1 - \frac{1}{2}(1-\alpha^2)(1-\lambda), \end{aligned}$$

so $1 - \lambda(G \text{ⓩ} H) \geq \frac{1}{2}(1-\alpha^2)(1-\lambda)$. ■

3.3 Transforming graphs into expanders

The goals of this section are to state and prove the correctness of Reingold's algorithm for generating a multigraph G_{exp}, whose connected components are expander multigraphs, from the input graph G. The basic idea of this algorithm (Definition 3.22 and Algorithm 3.3) is to transform the input graph G into a larger multigraph G_{exp} of constant degree via a logarithmic number of applications of the zig-zag product and powering. This transformation also bijectively maps the connected components of G into the connected components of G_{exp} (Lemma 3.23).

Let us start by defining the l-th power of a multigraph.

Definition 3.16. Let $n, d, l \in \mathbb{N}$. Let G be an (n, d)-multigraph and Rot be a rotational map of G. Let $\mathrm{Rot}_{G^l} : [n] \times [d]^l \longrightarrow [n] \times [d]^l$ be the function defined as follows: for each $u \in [n]$ and $p \in [d]^l$,

$$\mathrm{Rot}_{G^l}(u, p) := (w, q),$$

where $w := a_l$ and $b := (q_l, q_{l-1}, \dots, q_1)$ for $a_0 := u$ and $(a_i, q_i) := \mathrm{Rot}(a_{i-1}, p_i)$ for each $i \in [l]$. As Rot is a rotational map, we get that $\mathrm{Rot}_l(v, b) = (u, a)$.

The l-power $G^l(\mathrm{Rot})$ of G with respect to the rotational map Rot is the (n, d^l)-multigraph $\mathcal{G}(\mathrm{Rot}_l)$ defined by the rotational map Rot_l. It is not hard to prove that the l-th power of G with respect to a rotational map r_1 is isomorphic to the l-th power of G with respect to a rotational map r_2, but we will not use this fact. In our case, the rotational map Rot will be fixed, so we will denote by G^l the graph $G^l(\mathrm{Rot})$. ■

The following theorem states the relationship between $\lambda(G)$ and $\lambda(G^l)$.

Theorem 3.17. *Let $n, d, l \in \mathbb{N}$ and G be an (n, d)-multigraph. Let* Rot *be a rotational map of G. Let* Rot_l *and $G^l := G^l(\mathrm{Rot})$ as in Definition 3.16. Then, for every $u, v \in [n]$,*

$$(A_G^l)_{u,v} = (A_{G^l})_{u,v},$$

and $\lambda(G^l) = \lambda(G)^l$.

Proof. We will prove this theorem by induction on l. If $l = 1$, then, By Definition 3.4,

$$\begin{aligned}(A_{G^1})_{u,v} &= \frac{1}{d^1}|\{\,(p, q) \in [d]^1 \times [d]^1 \;:\; \mathrm{Rot}_1(u, p) = (v, q)\}| \\ &= \frac{1}{d}|\{\,(p, q) \in [d] \times [d] \;:\; a_1 := v, a_0 := u, \mathrm{Rot}(a_0, p) = (a_1, q)\}| \\ &= \frac{1}{d}|\{\,(p, q) \in [d] \times [d] \;:\; \mathrm{Rot}(u, p) = (v, q)\}| = (A_G^1)_{u,v}\end{aligned}$$

for each $u, v \in [n]$. Hence, $A_{G^1} = A_G^1$ and $\lambda(G^1) = \lambda(G)^1$.

Now suppose $l > 0$. Let $u, v \in [n]$. For each $z \in [n]$, $k \in \mathbb{N}$, and $p \in [d]^k$, let

$$\mathcal{W}(z, p) := a_k,$$

where $a_0 := z$ and $(a_i, q_i) := \mathrm{Rot}(a_{i-1}, p_i)$ for each $i \in [k]$. Thus, for each $p \in [d]^k$, we have

$$\exists q \in [d]^k \text{ s.t. } \mathrm{Rot}_k(u, p) = (v, q) \iff \mathcal{W}(u, p) = v,$$

by the definition of $\mathcal{W}(u, p)$ and Definition 3.16. Hence, we get

$$\begin{aligned}
d^l \cdot (A_{G^l})_{u,v} &= |\{ (p, q) \in [d]^l \times [d]^l \ : \ \mathrm{Rot}_l(u, p) = (v, q)\}| \\
&= |\{ p \in [d]^l \ : \ \exists q \in [d]^k \text{ s.t. } \mathrm{Rot}_l(u, p) = (v, q)\}| \\
&= |\{ p \in [d]^l \ : \ \mathcal{W}(u, p) = v\}| \\
&= |\{ (p, p_l) \in [d]^{l-1} \times [d] \ : \ \exists w \in [n] \text{ s.t. } \mathcal{W}(u, p) = w, \mathcal{W}(w, p_l) = v\}| \\
&= |\cup_{w\in[n]}\{ (p, p_l) \in [d]^{l-1} \times [d] \ : \ \mathcal{W}(u, p) = w, \mathcal{W}(w, p_l) = v\}| \\
&= \sum_{w\in[n]} |\{ (p, p_l) \in [d]^{l-1} \times [d] \ : \ \mathcal{W}(u, p) = w, \mathcal{W}(w, p_l) = v\}| \\
&= \sum_{w\in[n]} |\{ p \in [d]^{l-1} \ : \ \mathcal{W}(u, p) = w\}| \cdot |\{ p_l \in [d] \ : \ \mathcal{W}(w, p_l) = v\}| \\
&= \sum_{w\in[n]} |\{ p \in [d]^{l-1} \ : \ \mathcal{W}(u, p) = w\}| \cdot d \cdot (A_G)_{w,v},
\end{aligned}$$

where the last equality follows from

$$d \cdot (A_G)_{w,v} = |\{ (p, q) \in [d] \times [d] \ : \ \mathrm{Rot}(w, p) = (v, q)\}| = |\{ p \in [d] \ : \ \mathcal{W}(w, p) = v\}|.$$

Similarly, for each $w \in [n]$, we have

$$\begin{aligned}
|\{ p \in [d]^{l-1} \ : \ \mathcal{W}(u, p) = w\}| &= |\{ p \in [d]^{l-1} \ : \ \exists q \in [d]^{l-1} \text{ s.t. } \mathrm{Rot}_{l-1}(u, p) = (w, q)\}| \\
&= |\{ (p, q) \in [d]^{l-1} \times [d]^{l-1} \ : \ \mathrm{Rot}_{l-1}(u, p) = (w, q)\}| \\
&= d^{l-1} \cdot (A_{G^{l-1}})_{u,w}.
\end{aligned}$$

By induction on $l - 1$, we get that

$$(A_{G^{l-1}})_{u,w} = (A_G^{l-1})_{u,w},$$

and, consequently,

$$\begin{aligned}
d^l \cdot (A_{G^l})_{u,v} &= \sum_{w\in[n]} |\{ p \in [d]^{l-1} \ : \ \mathcal{W}(u, p) = w\}| \cdot d \cdot (A_G)_{w,v} \\
&= \sum_{w\in[n]} d^{l-1}(A_G^{l-1})_{u,w} \cdot d \cdot (A_G)_{w,v} \\
&= d^l \sum_{w\in[n]} (A_G^{l-1})_{u,w} \cdot (A_G)_{w,v} \\
&= d^l(A_G^{l-1}A_G)_{u,v} = d^l(A_G^l)_{u,v}.
\end{aligned}$$

Therefore, $(A_{G^l})_{u,v} = (A_G^l)_{u,v}$ and $A_{G^l} = A_G^l$. By Theorem 1.4, we get that $\lambda_i(A_G^l) = \lambda_i(A_G)^l$ for each $i \in [n]$. Consequently,

$$\lambda(G^l) = \lambda(G)^l.$$

■

Corollary 3.18. *Let $n, d, l \in \mathbb{N}$. If G is an (n, d)-multigraph with $\lambda(G) < 1$, then, for any rotational map* Rot *of G, the multigraph $G^l := G^l(\mathrm{Rot})$ is connected.*

Proof. By Theorem 3.17, we get that

$$\lambda(G^l) = \lambda(G)^l < 1,$$

and, by Theorem 3.5, we obtain that G^l is connected, since $\lambda(G^l) < 1$. ∎

By Theorem 3.17, we obtain that the power of a multigraph is a natural method for transforming it into a multigraph with large spectral gap. However, powering has the disadvantage of increasing the degree exponentially with exponent of the power. As we will see later (Definition 3.22), the role of the zig-zag product in the main transformation will be to reduce the degree of a multigraph without reducing too much its spectral gap (Corollary 3.15). Recall that the multigraph $G \textcircled{z} H$ has spectral gap lower bound in function of the spectral gap of the small graph H. Thus, for our application of the zig-zag product, we want a multigraph H with large spectral gap. The following theorem states that there is a graph with a good spectral gap with appropriate parameters.

Lemma 3.19 (Alon and Roichman [AR94])**.** *There is some constant $D_e \in \mathbb{N}$ such that there is a $(D_e^{16}, D_e, 1/2)$-graph.*

As D_e is a universal constant, we can find the adjacency matrix of such a graph using a brute force algorithm and it only requires constant space. Using this small expander and the zig-zag product, the transformation will iteratively increase the spectral gap of a multigraph and obtain an expander multigraph after a logarithmic number of iteration. Hence, we need to know a lower bound for the spectral gap of the initial multigraph to estimate the number of steps required to obtain an expander multigraph at the end of the iteration.

By Theorem 3.5, we know that if G is a bipartite graph, then $\lambda_n(G) = -1$ for $n := v(G)$, and $1 \geq \lambda(G) \geq |\lambda_n(G)| = 1$, so the spectral gap of G equals 0. Fortunately, the following theorem gives us a lower bound for the spectral gap of any regular, connected, nonbipartite multigraph.

Lemma 3.20 (Alon and Sudakov [AS00])**.** *Let $N, D \in \mathbb{N}$. If G is a connected, nonbipartite (N, D)-multigraph, then*

$$1 - \lambda(G) \geq 1/DN^2.$$

Thus, we will first transform each component of the input graph into a connected, nonbipartite, regular multigraph and then start the iterative process using this multigraph. One possible transformation of this kind is the following.

Definition 3.21. Let $D \in \mathbb{N}$, $D > 2$, and let G be a connected graph. Let $n := v(G)$. Let $\mathrm{Rot}_{G,D} : [n]^2 \times [D] \longrightarrow [n]^2 \times [D]$ be the function defined as follows: for each $(u, v) \in [n]^2$,

$$\begin{aligned}
\mathrm{Rot}_{G,D}((u,v),1) &:= ((u,(v-1) \mod n),2),\\
\mathrm{Rot}_{G,D}((u,v),2) &:= ((u,(v+1) \mod n),1),\\
\mathrm{Rot}_{G,D}((u,v),i) &:= ((u,v),i) && \text{if } i > 3, \text{ and}\\
\mathrm{Rot}_{G,D}((u,v),3) &:= ((w,z),3),
\end{aligned}$$

where, if $v \leq d_G(u)$, we define w as the v-th neighbor of u, and u as the z-th neighbor of v, and, if $> d_G(u)$, we define $w := u$ and $z := v$. It is not hard to show that the function $\mathrm{Rot}_{G,D}$ satisfies the requirements in Definition 3.3. A logspace oracle for $\mathrm{Rot}_{G,D}$ is implemented in Algorithm 3.2.

We denote by $\mathrm{Reg}(G, D)$ the (n^2, D)-multigraph $\mathcal{G}(\mathrm{Rot})$ defined by the rotational map $\mathrm{Rot}_{G,D}$. This multigraph is essentially G where each vertex u of G is replaced by an n-cycle C_u, and all edges labeled with 1 and 2 are edges of an C_u, all edges labeled with $i \in [D]$ for $i > 3$ are loops, and each edge labeled with 3 corresponds either to an edge in G or to a loop.

Note that, for every vertices s and t in G, there is a walk on G between s and t if and only if there is a walk on $\mathrm{Reg}(G, D)$ between $(s, 1)$ and $(t, 1)$, since every st-walk on G can be transformed into an $((s, 1), (t, 1))$-walk on $\mathrm{Reg}(G, D)$ (by mapping an edge uv of the walk on G into the walk on the n-cycle of u and moving to the n-cycle of v only at the edge corresponding to uv), and every $((s, 1), (t, 1))$-walk on $\mathrm{Reg}(G, D)$ can be transformed into an st-walk on G (by mapping a walk w on the n-cycle of a vertex u to the edge before entering and the edge after leaving the n-cycle of u). ■

Algorithm 3.2 A logspace oracle for $\mathrm{Rot}_{G,D}$

Input: two naturals $n, D \in \mathbb{N}$, a logspace oracle for a rotational map Rot of a graph G, and a pair $(x, i) \in [n]^2 \times [D^{16}]$. This algorithm assumes that the graph G has n vertices and $D > 2$.

Output: the pair $(y, j) := \mathrm{Rot}_{G,D}(x, i)$.

```
1:  LogspaceOracleRegularNonbipMultigraph($n$, $D$, Rot, $(x, i)$):
2:  $(u, v) \longleftarrow x$
3:  **if** $i = 1$ **then**
4:      **return** $((u, (v - 1) \mod n), 2)$
5:  **else if** $i = 2$ **then**
6:      **return** $((u, (v + 1) \mod n), 1)$
7:  **else if** $i > 3$ **then**
8:      **return** $((u, v), i)$
9:  **else if** $i = 3$ and $v > d_G(u)$ **then**
10:     **return** $((u, v), 3)$
11: **else if** $i = 3$ and $v \leq d_G(u)$ **then**
12:     **return** $((w, z), 3)$, where $w$ is the $v$-th neighbor of $u$, and $u$ is the $z$-th neighbor of
        $v$ if $v \leq d_G(u)$, which are computable in logspace using the input logspace oracle for
        Rot
```

We will now define the main transformation of this section.

Definition 3.22. Let $N, D \in \mathbb{N}$ and $\lambda, \alpha \in [0, 1)$. Let G be an (N, D^{16}, λ)-multigraph with rotational map Rot_G and H be a (D^{16}, D, α)-multigraph with rotational map Rot_H. Let $l := 2\lceil \lg(DN^2) \rceil$. We define $G_0 := G$ and, for each $i \in [l]$,

$$T_i(G, H) := G_i := (G_{i-1} \text{ⓩ} H)^8.$$

We denote by Rot_{G_i} the rotational map of G_i, which is defined recursively by $\mathrm{Rot}_{G_i} :=$

$\mathrm{Rot}_{(G_{i-1}ⓩH)^8}$ and only depends on i and the rotational maps Rot_G and Rot_H. We denote by $\mathcal{T}(G,H)$ the graph G_l and by $\mathrm{Rot}_{\mathcal{T}(G,H)}$ the rotational map Rot_{G_l}. ■

We will first prove that solving USTCON for $\mathcal{T}(G,H)$ is sufficient to solve USTCON for G (Property 4 of Lemma 3.23) and that $\mathcal{T}(G,H)$ if a valid input for Algorithm 3.1 (Lemma 3.26 and Lemma 3.27). Then we will argue (Lemma 3.28) that Algorithm 3.3 is a correct implementation of a logspace oracle for $\mathrm{Rot}_{\mathcal{T}(G,H)}$.

Lemma 3.23. *Consider the context of Definition 3.22. Let C be a connected component of G. For each $i \in \{0,\dots,l\}$, the following properties hold:*

1. *The graph G_i is an (ND^{16i}, D^{16})-multigraph.*
2. *For $S := V(C)\times[D^{16}]^i$, we have $\mathcal{T}_i(C,H) = G_i[S]$ and the subgraph $G_i[S]$ is a connected component of G_i.*
3. *For every component E of G_i, there is a component F of G such that $E = G_i[V(F)\times[D^{16}]^i]$.*
4. *For all vertices s and t of G, the vertices s and t are in C if and only if the vertices $(s,(1,\dots,1))$ and $(t,(1,\dots,1))$ of $V(G_i) = V(G_0)\times[D^{16}]^i$ are in $G_i[V(C)\times[D^{16}]^i]$.*

Proof. Let C be a connected component of G. We will prove this lemma by induction on i.

If $i = 0$, then the following properties hold:

1. The graph $G_0 = G$ is an (N, D^{16})-multigraph.
2. For $S := V(C)\times[D^{16}]^0 = V(C)$, we have $\mathcal{T}_0(C,H) = C = G[V(C)] = G_0[S]$ and the subgraph $G_0[S] = C$ is a connected component of G_0.
3. For every component E of G_0, there is a component $F := E$ of G such that $G_0[V(F)\times[D^{16}]^0] = E$.
4. For all vertices s and t of G, the vertices s and t are in C if and only if the vertices $(s,(1,\dots,1)) = (s)$ and $(t,(1,\dots,1)) = (t)$ of $V(G_0)\times[D^{16}]^0 = V(G_0)$ are in $G_0[V(C)\times[D^{16}]^0] = C$.

Now suppose $i \geq 0$ and, by induction on i, assume that G_i satisfies all properties in Lemma 3.23. We will now prove that G_{i+1} also satisfies that properties for $i+1$.

First note that, by the definition of the zig-zag product and the powering of multigraphs, we get that $G_iⓩH$ is a $(v(G_i)D^{16}, D^2)$-multigraph and $(G_iⓩH)^8$ is a $(v(G_i)D^{16}, D^{16})$-multigraph, so

$$G_{i+1} \text{ is an } (ND^{16(i+1)}, D^{16})\text{-multigraph},$$

and Property 1 holds.

Let us now prove that Property 2 holds. Consider the following lemmas.

Lemma 3.24. *Let $N, D, d \in \mathbb{N}$ and $\lambda, \alpha \in [0,1)$. Let G be an (N,D,λ)-multigraph, H be a (D,d,α)-multigraph, and T be a connected component of G. Let $S := V(T)\times[D]$. Then $TⓩH = (GⓩH)[S]$ and $TⓩH$ is a connected component of $GⓩH$.*

Proof. First note that $T \circledz H$ and $(G \circledz H)[S]$ have the same vertex set. It is not hard to prove that $T \circledz H$ and $(G \circledz H)[S]$ have the same edge set and incidence function by a careful analysis of Definition 3.11. Hence, $T \circledz H = (G \circledz H)[S]$, and, by Corollary 3.13, we get that $T \circledz H$ is a connected graph. Finally, note that $e_{G \circledz H}(S, V(G \circledz H) \setminus S) = 0$, since if there is an edge e of $G \circledz H$ with one end (v, a) in S and the other end (w, b) in $V(G \circledz H) \setminus S$, then, by the definition of the zig-zag product, there is an edge in G with ends v and w, which contradicts the fact that T is a component of G (note that $w \notin V(T)$ since $(w, b) \notin V(T) \times [D]$). Therefore, $(G \circledz H)[S]$ is a connected component of $G \circledz H$. ∎

Lemma 3.25. *Let $N, D, l \in \mathbb{N}$ and $\lambda \in [0, 1)$. Let G be an (N, D, λ)-multigraph, and W be a connected component of G. Let $S := V(W)$. Then $W^l = G^l[S]$ and W^l is a connected component of G^l.*

Proof. First note that W^l and $G^l[S]$ have the same vertex set. It is not hard to prove that W^l and $G^l[S]$ have the same edge set and incidence function by a careful analysis of Definition 3.16. Hence, $W^l = G^l[S]$. By Corollary 3.18, we get that W^l is a connected graph. Finally, note that $e_{G^l}(S, V(G^l) \setminus S) = 0$, since if there is an edge e of G^l with one end v in S and the other end w in $V(G^l) \setminus S$, then, by the definition of the power of graphs, there is a vw-walk on G, which contradicts the fact that W is a component of G. Therefore, $G^l[S]$ is a connected component of G^l. ∎

Let $S := V(C) \times [D^{16}]^{i+1}$. We know by induction that $\mathcal{T}_i(C, H) = G_i[V(C) \times [D^{16}]^i] =: T$ is a connected component of G_i. By Lemma 3.24, we get that $W := T \circledz H = (G_i \circledz H)[S]$ and W is a connected component of $G_i \circledz H$. By Lemma 3.25, we get that $W^8 = (G_i \circledz H)^8[S]$ and W^8 is a connected component of $(G_i \circledz H)^8$. Therefore, we obtain that

$$\mathcal{T}_{i+1}(C, H) = (\mathcal{T}_i(C, H) \circledz H)^8 = (T \circledz H)^8 = W^8 = (G_i \circledz H)^8[S] = G_{i+1}[S],$$

so $G_{i+1}[S]$ is a component of G_{i+1}. Consequently, Property 2 holds.

Finally, let us prove that Properties 3 and 4 hold. Let E be a connected component of G_{i+1} and let $(v, (v_1, \dots, v_{i+1}))$ be an element of $E \subseteq V(G_0) \times [D^{16}]^{i+1}$. Let F be the connected component of v in G_0. By Property 2, we obtain that $E' := G_{i+1}[F \times [D^{16}]^{i+1}]$ is a connected component of G_{i+1}. As $E' \cap E \supseteq (v, (v_1, \dots, v_{i+1}))$, we get that $E = E' = G_{i+1}[F \times [D^{16}]^{i+1}]$. Thus, Property 3 holds.

Let $C' := C \times [D^{16}]^{i+1}$. If s and t are vertices of C, then, by Property 2, the vertices $(s, (1, \dots, 1))$ and $(t, (1, \dots, 1))$ are in $\mathcal{T}_{i+1}(C, H) = G_{i+1}[C']$. Similarly, if the vertices $(s, (1, \dots, 1))$ and $(t, (1, \dots, 1))$ are in $G_{i+1}[C']$, then s and t are vertices of C, because otherwise we obtain that either s or t is a vertex of another component F of G, so $G_{i+1}[F \times [D^{16}]^{i+1}]$ is a component of G_{i+1} and $(F \times [D^{16}]^{i+1}) \cap C' \neq \emptyset$, hence $C = F$, a contradiction. Therefore, Property 4 holds. ∎

Lemma 3.26. *Consider the context of Definition 3.22. If $\lambda(H) \leq 1/2$ and G is connected and nonbipartite, then $\lambda(\mathcal{T}(G, H)) \leq 4/9 < 0.45$.*

Proof. Let $i \in [l]$. We will first prove that $\lambda(G_i) \le \max\{\lambda(G_{i-1})^2, 4/9\}$. Let $\lambda := \lambda(G_{i-1})$. By Corollary 3.15 and Theorem 3.17, we obtain that

$$\lambda(G_i) \le \left(1 - \frac{1}{2}\left(1 - \lambda(H)^2\right)\left(1 - \lambda(G_{i-1})\right)\right)^8 \le \left(1 - \frac{3}{8}(1-\lambda)\right)^8 \le \left(1 - \frac{1}{3}(1-\lambda)\right)^8.$$

If $\lambda \le 4/9$, then

$$\lambda(G_i) \le \left(1 - \frac{1}{3}\cdot\frac{4}{9}\right)^8 = \left(\frac{23}{27}\right)^8 < \frac{4}{9} \le \max\{\lambda^2, 4/9\}.$$

Now suppose that $\lambda > 4/9$. Let $g(x) = 81x - (2+x)^4$. Note that $g(4/9) > 0.29 > 0$; $g(1) = 0$; $g'(x) = 81 - 4(2+x)^3$;

$$g'(x) \ge 0 \iff x \le -2 + (81/4)^{1/3} =: x^* \in (0.725, 0.726);$$

and $g(x^*) > 0$. By elementary calculus, we get $g(x) \ge 0$ for all $x \in [4/9, 1]$, because $g(4/9) > 0$, $g(x^*) > 0$, $g(1) = 0$, $g(x)$ is crescent for $x \in [4/9, x^*]$, and there is no $x \in (x^*, 1]$ with $g'(x) = 0$. Consequently, the inequality

$$\frac{1}{81}(2+x)^4 \le x$$

holds for every $x \in [4/9, 1]$. Thus, we get

$$\lambda(G_i) \le \left(1 - \frac{1}{3}(1-\lambda)\right)^8 \le \left(\frac{1}{81}(2+\lambda)^4\right)^2 \le \lambda^2 \le \max\{\lambda^2, 4/9\}.$$

Let us now prove that $\lambda(G_l) \le 4/9$. Suppose that, for all $i \in [l]$, we have $\lambda(G_i) > 4/9$. Hence, $\lambda(G_i) \le \lambda(G_{i-1})^2$, and, consequently,

$$\lambda(G_i) \le \lambda(G_0)^{2^i}.$$

By Lemma 3.20 and the definition of l, we get that

$$\begin{aligned}\lambda(G_l) \le \lambda(G_0)^{2^l} &\le (1 - 1/DN^2)^{2^{2\lg(DN^2)}}\\ &\le e^{-(DN^2)^2/(DN^2)} = e^{-DN^2} < 4/9,\end{aligned}$$

a contradiction. Thus, there is some $i \in [l]$ such that $\lambda(G_i) \le 4/9$, and, as a consequence,

$$\lambda(G_j) \le \max\{\lambda(G_{j-1})^2, 4/9\} = 4/9$$

for all $j \in [l]$ and $j > i$; in particular, $\lambda(G_l) \le 4/9$. Therefore,

$$\lambda(\mathcal{T}(G,H)) = \lambda(G_l) \le 4/9 < 0.45.$$

∎

Lemma 3.27. *Consider the context of Definition 3.22. If $\lambda(H) \le 1/2$ and G is nonbipartite, then $G_{exp} := \mathcal{T}(G,H)$ is an $(N^{O_D(1)}, D^{16})$-multigraph such that each component D of G_{exp} is a*

$(v(D), D^{16})$-expander.

Proof. First note that $G_{\exp}$ has

$$ND^{16l} \leq ND^{64\lg(DN^2)} = ND^{64\log_D(N^2)/\log_D 2} = N(N^2)^{64/\log_D 2} = N^{1+128/\log_D 2} = N^{O_D(1)}$$

vertices and is D^{16}-regular by Property 1 of Lemma 3.23 (recall that $G_{\exp} = G_i$ for $i := l$). Let D be a component of $G_{\exp}$. By Property 3 of Lemma 3.23, there is a component E of G such that $G_{\exp}[V(E) \times [D^{16}]^l] = D$. By Property 2 of Lemma 3.23, we get that

$$\mathcal{T}(E, H) = G_{\exp}[V(E) \times [D^{16}]^l] = D,$$

and, by Lemma 3.26,

$$0.45 > 4/9 \geq \lambda(\mathcal{T}(E, H)) = \lambda(D).$$

Therefore, D is a $(v(D), D^{16})$-expander. ■

Lemma 3.28. *Algorithm 3.3 is a logspace oracle for* $\mathrm{Rot}_{\mathcal{T}(G,H)}$.

Proof. Let us first argue that Algorithm 3.3 uses at most $O_D(\lg N)$ memory slots. As mentioned in Algorithm 3.3, the variables $v, a_0, \dots, a_{l-1}, a_l$ are shared by all calls for the funtion LOTIExpanderRec and, for each $i \in \{0, \dots, l\}$, the variables $k_{i,1}, \dots, k_{i,16}$ are just shorthand notations for the components of $a_i \in [D]^{16}$. Hence, all those variable requires at most $\lg N + (l+1)\lg(D)^{16} = O(\lg N)$ memory slots. For each $i \in [l]$ and $j \in [8]$, the j-th iteration of the **for** loop in the i-level of recursion of LOTIExpanderRec uses at most $O(\lg D)$ memory slots to compute two entries of Rot_H, and make one recursive call to LOTIExpanderRec for $i-1$. Note that all relevant information of the recursive call is stored in the variable $v, a_0, \dots, a_l$. Hence the computation of the l-level of recursion of LOTIExpanderRec requires at most $O(l \lg D) = O_D(\lg N)$ memory slots. Therefore the algorithm uses at most $O(\lg N) + O_D(\lg N) = O(\lg N)$ memory slots. Appendix A in Reingold [Rei08] has low level implementation details of Algorithm 3.3.

Let us now prove that the functions LogspaceOracleTransformationIntoExpander and LOTIExpanderRec are correct. We will prove it by induction on the recursion level $i \in \{0, \dots, l\}$ of LOTIExpanderRec. If $i = 0$, then LOTIExpanderRec updates (v, a_0) to the value of $\mathrm{Rot}_G(v, a_0)$, which is the expected behavior of LOTIExpanderRec (Equation 3.1).

Suppose now that $i \geq 0$. Assume that the i-th recursion level of LOTIExpanderRec is correct, that is, LOTIExpanderRec updates the tuple $(v, a_0, \dots, a_{i-1}, a_i)$ with the value of $\mathrm{Rot}_{\mathcal{T}_i(G,H)}((v, a_0, \dots, a_{i-1}), a_i)$, see Equation 3.1. Let us prove that the $(i+1)$-th recursion level of LOTIExpanderRec is correct.

We will first prove that, for each $j \in [8]$, the value of the tuple $(((v, a_0, \dots, a_{i-1}), a_i), (k_{i+1,2j}, k_{i+1,2j-1}))$ at the end of the j-th iteration of the **for** loop is equal to

$$\mathrm{Rot}_{(G_i \textcircled{z} H)}(((v', a'_0, \dots, a'_{i-1}), a'_i), (k'_{i+1,2j-1}, k'_{i+1,2j})).$$

where $(v', a'_0, \dots, a'_i, a'_{i+1})$ and $(k'_{i+1,2j-1}, k'_{i+1,2j})$ are respectively the values of $(v, a_0, \dots, a_i, a_{i+1})$ and $(k_{i+1,2j-1}, k_{i+1,2j})$ at the beginning of the j-th iteration.

Let $u_j := (v', a'_0, \dots, a'_{i-1})$ and $b_j := a'_i$. Let $p_{2j-1} := k'_{i+1,2j-1}$ and $p_{2j} := k'_{i+1,2j}$. Intuitively, we interpret the j-th iteration of the **for** loop as computing the tuple $\mathrm{Rot}_{G_i \textcircled{z} H}((u_j, b_j), (p_{2j-1}, p_{2j}))$. Formally, we have the following:

- After the execution of Line 11, the tuple $(a_i, k_{i+1,2j-1})$ is equal to

$$\mathrm{Rot}_H(b_j, p_{2j-1}).$$

  Note that the value of $k_{i+1,2j-1}$ will only change again at the end of the algorithm (Line 14). Denote by b'_j the value of a_i and by q_{2j-1} the value of $k_{i+1,2j-1}$ after the execution of Line 11.

- After the execution of Line 12, we get, by induction on i, that the tuple $((v, a_0, \dots, a_{i-1}), a_i)$ is equal to

$$\mathrm{Rot}_{G_i}(u_j, b'_j).$$

  Denote by c'_j the value of a_i and by w the value of $(v, a_0, \dots, a_{i-1})$ after the execution of Line 12.

- After the execution of Line 13, the tuple $(a_i, k_{i+1,2j})$ is equal to

$$\mathrm{Rot}_H(c'_j, p_{2j}).$$

  Denote by c_j the value of a_i and by $q_{i+1,2j}$ the value of $k_{i+1,2j}$ after the execution of Line 13.

Hence, the following equalities hold:

$$\begin{aligned}(b'_j, q_{2j-1}) &= \mathrm{Rot}_H(b_j, p_{2j-1}),\\ (w_j, c'_j) &= \mathrm{Rot}_{G_i}(u_j, b'_j), \text{ and}\\ (c_j, q_{2j}) &= \mathrm{Rot}_H(c'_j, p_{2j}),\end{aligned}$$

so, by the definition of the zig-zag product, we get that

$$\mathrm{Rot}_{(G_i \textcircled{z} H)}((u_j, b_j), (p_{2j-1}, p_{2j})) = ((w_j, c_j), (q_{2j}, q_{2j-1})).$$

Therefore, the value of $(((v, a_0, \dots, a_{i-1}), a_i), (k_{i+1,2j}, k_{i+1,2j-1}))$ at the end of the j-th iteration of the **for** loop is equal to

$$\mathrm{Rot}_{(G_i \textcircled{z} H)}(((v', a'_0, \dots, a'_{i-1}), a'_i), (k'_{i+1,2j-1}, k'_{i+1,2j})).$$

Furthermore, for $(w_0, c_0) := (u_1, b_1)$, we get that, for each $j \in [8]$,

$$((w_j, c_j), (q_{2j}, q_{2j-1})) = \mathrm{Rot}_{(G \textcircled{z} H)}((w_{j-1}, c_{j-1}), (p_{2j-1}, p_{2j})).$$

Thus, by Definition 3.16, we have

$$\mathrm{Rot}_{(G \textcircled{z} H)^8}((w_0, b_0), ((p_1, p_2), \dots, (p_{15}, p_{16}))) = ((w_8, b_8), ((q_{16}, q_{15}), \dots, (q_2, q_1))).$$

As $((v, a_0, \dots, a_{i-1}), a_i)$ and a_{i+1} are respectively equal to (w_8, b_8) and $(q_{16}, q_{15}, \dots, q_2, q_1)$ at Line 14, we conclude that the $(i+1)$-th recursion level of LOTIExpanderRec updates the value of $((v, a_0, \dots, a_i), a_{i+1})$ to the value of the tuple

$$\mathrm{Rot}_{G_{i+1}}((v, a_0, \dots, a_i), a_{i+1}) = \mathrm{Rot}_{\mathcal{T}_{i+1}(G,H)}((v, a_0, \dots, a_i), a_{i+1}).$$

■

3.4 Reingold's algorithm for the undirected st-connectivity problem

Algorithm 3.4 is a slight modification of the algorithm proposed by Reingold [Rei08]. We will prove its correctness in Theorem 3.29.

Theorem 3.29. *Algorithm 3.4 is a deterministic logspace algorithm for the undirected st-connectivity problem.*

Proof. As argued in Definition 3.21, solving USTCON for the graph G and vertices s and t is equivalent to solving the USTCON for the multigraph $G_{\text{reg}} := \mathrm{Reg}(G, D_e)$ and vertices s_{reg} and t_{reg}. As G_{reg} is a nonbipartite (N, D_e^{16})-multigraph and H is a $(D_e^{16}, D_e, 1/2)$-multigraph, we get, by Lemma 3.27, that $G_{\text{exp}} := \mathcal{T}(G_{\text{reg}}, H)$ is an $(N^{O(1)}, D_e^{16})$-multigraph such that each component C of $\mathcal{T}(G, H)$ is a $(v(C), D_e^{16})$-expander. Thus, the multigraph G_{exp} satisfies the assumptions for the input multigraph of Algorithm 3.1. By Property 4 of Lemma 3.23, we get that solving USTCON for the multigraph G_{reg} and vertices s_{reg} and t_{reg} is equivalent to solving the USTCON for the multigraph G_{exp} and vertices s_{exp} and t_{exp}. Therefore, the output of Algorithm 3.4 is correct, because it is equal to the output of Algorithm 3.1 with input $v(G_{\text{exp}})$, D_e^{16}, G_{exp}, s_{exp} and t_{exp}, which is correct by Theorem 3.10.

As argued in Theorem 3.10, Definition 3.21, and Lemma 3.28, all functions used in Algorithm 3.4 requires memory at most logarithm of the size of their input, and their inputs in Algorithm 3.4 have size at most a polynomial in n, since D_e is a constant independent of n. Therefore, Algorithm 3.4 uses at most $O(\lg n)$ memory slots. ■

Reingold [Rei08] also designed deterministic logspace algorithms to construct (n, d, π)-universal traversal sequences and (n, d)-universal exploration sequences using techniques similar to those studied in this chapter. These sequences are restricted versions of universal traversal sequences previously considered in the literature.

Algorithm 3.3 A logspace oracle for $\mathcal{T}(G, H)$.

Input: two naturals $N, D \in \mathbb{N}$, a logspace oracle for a rotational map Rot_G of a multigraph G, a logspace oracle for a rotational map Rot_H of a multigraph H, and a pair $(x, y) \in V(\mathcal{T}(G, H)) \times [D^{16}]$, where $V(\mathcal{T}(G, H)) = [N] \times [D^{16}]^l$ for $l := 2\lceil \lg(DN^2) \rceil$. This algorithm assumes that the graph G is an (N, D^{16})-multigraph and the graph H is a (D^{16}, D)-multigraph.

Output: the pair $\text{Rot}_{\mathcal{T}(G,H)}(x, y)$.

```
1: LogspaceOracleTransformationIntoExpander($N$, $D$, $\text{Rot}_G$, $\text{Rot}_H$, $(x, y)$):
2: $(v, a_0, \dots, a_{l-1}) \longleftarrow x$, $a_l \longleftarrow y$
3: LOTIExpanderRec($N$, $D$, $\text{Rot}_G$, $\text{Rot}_H$, $(v, a_0, \dots, a_l)$, $l$)
4: **return** $((v, a_0, \dots, a_{l-1}), a_l)$, note that the values of $v, a_0, \dots, a_l$ are update by the recursive calls
```

Input: two naturals $N, D \in \mathbb{N}$, a logspace oracle for a rotational map Rot_G of a multigraph G, a logspace oracle for a rotational map Rot_H of a multigraph H, a tuple $((v, a_0, \dots, a_{l-1}), a_l) \in V(\mathcal{T}(G, H)) \times [D^{16}]$, and the recursion level $i \in \{0, \dots, l\}$. This algorithm assumes that the graph G is an (N, D^{16})-multigraph and the graph H is a (D^{16}, D)-multigraph.

Output: Update the tuple $((v, a_0, \dots, a_{i-1}), a_i)$ with the value of

$$\text{Rot}_{\mathcal{T}_i(G,H)}((v', a'_0, \dots, a'_{i-1}), a'_i), \tag{3.1}$$

where $(v', a'_0, \dots, a'_{i-1}, a'_i)$ corresponds to the values of $(v, a_0, \dots, a_{i-1}, a_i)$ at the beginning of execution of the function.

```
5: LOTIExpanderRec($N$, $D$, $\text{Rot}_G$, $\text{Rot}_H$, $(v, a_0, \dots, a_l)$, $i$):
6: **if** $i = 0$ **then**
7:     **return** $\text{Rot}_G(v, a_0)$
8: $(k_{i,1}, \dots, k_{i,16}) \longleftarrow a_i$, where we use the correspondence between $a_i \in [D^{16}]$ and $(k_{i,1}, \dots, k_{i,16}) \in [D]^{16}$
9: // All the variables $v$, $a_j$, and $k_{i,j}$ defined above are just pointers for the variables they are attributed, that is, they do not consume aditional space because they use the same space as the variable they are attributed.
10: **for all** $j \in [8]$ **do**
11:     $(a_{i-1}, k_{i,2j-1}) \longleftarrow \text{Rot}_H(a_{i-1}, k_{i,2j-1})$
12:     LOTIExpanderRec($N$, $D$, $\text{Rot}_G$, $\text{Rot}_H$, $(v, a_0, \dots, a_l)$, $i - 1$), which recursively implements the operation $((v, a_0, \dots, a_{i-2}), a_{i-1}) \longleftarrow \text{Rot}_{G_{i-1}}((v, a_0, \dots, a_{i-2}), a_{i-1})$.
13:     $(a_{i-1}, k_{i,2j}) \longleftarrow \text{Rot}_H(a_{i-1}, k_{i,2j})$
14: $(k_{i,1}, \dots, k_{i,16}) \longleftarrow (k_{i,16}, \dots, k_{i,1})$
```

Algorithm 3.4 Reingold's algorithm for the undirected st-connectivity problem

Input: a graph G and two distinct vertices s and t.
Output: True if there is an st-walk in G, and False otherwise.

```
1: ReingoldUSTCONAlgorithm($G$, $s$, $t$):
2: Let $n := v(G)$ and $N := n^2$. Let $D_e$ be the constant in Lemma 3.19 and $l := 2\lceil \lg D_e N^2 \rceil$.
3: Let $H$ be a $(D_e^{16}, D_e)$-expander graph. We need at most constant space to find and store
   the adjacency matrix $A_H$ of $H$.
4: Let $\text{Rot}_G$ be a logspace oracle for a rotational map of $G$ and $\text{Rot}_H$ be a logspace oracle
   for a rotational map of $H$. As mentioned in Fact 3.2, we can design a logspace oracle
   for a rotational map of a graph using its adjacency matrix.
5: Let $\text{Rot}_{G,D_e}(x, i)$ be the logspace oracle for the $(N, D_e^{16})$-multigraph $\text{Reg}(G, D_e)$ defined
   using the function LogspaceOracleRegularNonbipMultigraph($n$, $D_e$, $\text{Rot}_G$, $(x, i)$) in
   Algorithm 3.2, where $(x, i) \in [N] \times [D_e^{16}]$ are the only inputs for this function.
6: Let $\text{Rot}_{\text{exp}}(x, i)$ be the logspace oracle for the multigraph $\mathcal{T}(\text{Reg}(G, D_e), H)$ defined
   using the function LogspaceOracleTransformationIntoExpander($N$, $D_e$, $\text{Rot}_{G,D_e}$, $\text{Rot}_H$,
   $(x, i)$) in Algorithm 3.3, where $(x, i) \in V(\mathcal{T}(\text{Reg}(G, D_e), H)) \times [D_e^{16}]$ are the only inputs
   for this function.
7: Let $N_{\text{exp}} := v(\mathcal{T}(\text{Reg}(G, D_e), H))$ and $u := (1, \dots, 1) \in [D^{16}]^l$.
8: Let $s_{\text{reg}} := (s, 1)$ and $t_{\text{reg}} := (t, 1)$, so $s_{\text{reg}}, t_{\text{reg}} \in V(\text{Reg}(G, D_e))$.
9: Let $s_{\text{exp}} := (s_{\text{reg}}, u)$ and $t_{\text{exp}} := (t_{\text{reg}}, u)$, so $s_{\text{exp}}, t_{\text{exp}} \in V(\mathcal{T}(\text{Reg}(G, D_e), H))$.
10: **return** USTCONAlgorithmForExpanders($N_{\text{exp}}$, $D_e^{16}$, $\text{Rot}_{\text{exp}}$, $s_{\text{exp}}$, $t_{\text{exp}}$)
```

Chapter 4

Conclusions

Generally speaking, the goal of this work is to study the techniques utilized to understand the space complexity of the undirected st-connectivity problem. As there is an extensive literature about this problem, we narrowed our focus to the first randomized logspace algorithm [Ale+79] and the first deterministic logspace algorithm [Rei08] solving USTCON. We aimed to provide detailed proofs for the main results and present all the background needed to understand them. However, we left some technical details about the algorithms and their implementations without complete proofs, since those proofs would require more technical mathematical theories (e.g., defining Turing machines and their tape complexity, or using arguments from measure theory/advanced probability theory) to properly formalize them.

Although Reingold's algorithm solved one of the main problems regarding logspace complexity, there are still many other interesting open questions in this area. The following list highlights some of those questions and provides a non-exhaustive list of references for the interested reader:

1. Is there a deterministic logspace algorithm for the directed st-connectivity problem? As mentioned in the introduction, this question is crucial to answer the **L** versus **NL** question. To the best of our knowledge, Savitch's algorithm [Sav70] is currently the best upper bound.

2. Is there a deterministic logspace algorithm for the undirected st-connectivity problem that runs in linear time? By a classical result in complexity theory, we know that every logspace algorithm also runs in polynomial time, so Reingold's algorithm is a polynomial time algorithm for USTCON. However, it is not optimal in the sense that the multiplicative constant for the space usage is extremely high and its running time may be a large polynomial. To the best of our knowledge, the currently most efficient deterministic logspace algorithm for USTCON was proposed by Rozenman and Vadhan [RV05].

3. Are there efficient space-bounded pseudorandom generators? Can we design deterministic logspace algorithms for other problems in **RL**? There has been some recent progress on space-bounded pseudorandom generators (see Hoza [Hoz21]) and on deterministic nearly-logspace algorithms for approximately solving linear systems

given by the Laplacian of undirected graphs (see Murtagh, Reingold, Sidford, and Vadhan [Mur+21]).

4. Is there a deterministic logspace algorithm to construct universal traversal sequences? By the probabilistic proof of Aleliunas, Karp, Lipton, Lovász, and Rackoff [Ale+79] (see Section 2.4 of this work), we know that there are universal sequences of polynomial length and, with high probability, a random sequence of appropriate length is universal. Besides being interesting combinatorial objects, those sequences were used by Babai, Nisan, and Szegedy [BNS92] to construct space-bounded pseudorandom generators, which made researchers interested in explicit constructions for them. To the best of our knowledge, the best construction for those sequences is the $O((\lg n)^2)$-space algorithm by Nisan [Nis92] that constructs universal sequences of length $n^{O(\lg n)}$.

We recommend reading Section 6.2 of Reingold [Rei08] for more details about those questions.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity*. A modern approach. Cambridge University Press, Cambridge, 2009, pp. xxiv+579. ISBN: 978-0-521-42426-4. URL: https://doi.org/10.1017/CBO9780511804090 (cit. on p. 4).

[Ale+79] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. "Random walks, universal traversal sequences, and the complexity of maze problems". In: *FOCS*. 1979 (cit. on pp. 1, 2, 11, 18, 47, 48).

[AR94] Noga Alon and Yuval Roichman. "Random Cayley graphs and expanders". In: *Random Structures Algorithms* 5.2 (1994), pp. 271–284. ISSN: 1042-9832,1098-2418. URL: https://doi.org/10.1002/rsa.3240050203 (cit. on p. 36).

[Arm+00] Roy Armoni, Amnon Ta-Shma, Avi Wigderson, and Shiyu Zhou. "An $O(\log(n)^{4/3})$ space algorithm for (s,t) connectivity in undirected graphs". In: *J. ACM* 47.2 (2000), pp. 294–311. ISSN: 0004-5411,1557-735X. URL: https://doi.org/10.1145/333979.333984 (cit. on p. 1).

[AS00] Noga Alon and Benny Sudakov. "Bipartite subgraphs and the smallest eigenvalue". In: *Combin. Probab. Comput.* 9.1 (2000), pp. 1–12. ISSN: 0963-5483,1469-2163. URL: https://doi.org/10.1017/S0963548399004071 (cit. on p. 36).

[BNS92] László Babai, Noam Nisan, and Márió Szegedy. "Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs". In: *J. Comput. System Sci.* 45.2 (1992). Twenty-first Symposium on the Theory of Computing (Seattle, WA, 1989), pp. 204–232. ISSN: 0022-0000. URL: https://doi.org/10.1016/0022-0000(92)90047-M (cit. on p. 48).

[Cap+02] Michael Capalbo, Omer Reingold, Salil Vadhan, and Avi Wigderson. "Randomness conductors and constant-degree lossless expanders". In: *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*. ACM, New York, 2002, pp. 659–668. ISBN: 1-58113-495-9. URL: https://doi.org/10.1145/509907.510003 (cit. on p. 1).

[Din07] Irit Dinur. "The PCP theorem by gap amplification". In: *J. ACM* 54.3 (2007), Art. 12, 44. ISSN: 0004-5411. URL: https://doi.org/10.1145/1236457.1236459 (cit. on p. 1).

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. "Expander graphs and their applications". In: *Bull. Amer. Math. Soc. (N.S.)* 43.4 (2006), pp. 439–561. ISSN: 0273-0979. URL: https://doi.org/10.1090/S0273-0979-06-01126-8 (cit. on p. 25).

[Hoz21] William M. Hoza. “Better pseudodistributions and derandomization for space-bounded computation”. In: *Approximation, randomization, and combinatorial optimization. Algorithms and techniques.* Vol. 207. LIPIcs. Leibniz Int. Proc. Inform. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2021, Art. No. 28, 23. ISBN: 978-3-95977-207-5 (cit. on p. 47).

[LP82] Harry R. Lewis and Christos H. Papadimitriou. “Symmetric space-bounded computation”. In: *Theoret. Comput. Sci.* 19.2 (1982), pp. 161–187. ISSN: 0304-3975,1879-2294. URL: https://doi.org/10.1016/0304-3975(82)90058-5 (cit. on p. 1).

[MU17] Michael Mitzenmacher and Eli Upfal. *Probability and computing*. Second. Randomization and probabilistic techniques in algorithms and data analysis. Cambridge University Press, Cambridge, 2017, pp. xx+467. ISBN: 978-1-107-15488-9 (cit. on pp. 11, 12).

[Mur+21] Jack Murtagh, Omer Reingold, Aaron Sidford, and Salil Vadhan. “Derandomization beyond connectivity: undirected Laplacian systems in nearly logarithmic space”. In: *SIAM J. Comput.* 50.6 (2021), pp. 1892–1922. ISSN: 0097-5397,1095-7111. URL: https://doi.org/10.1137/20M134109X (cit. on p. 48).

[Nis92] Noam Nisan. “Pseudorandom generators for space-bounded computation”. In: *Combinatorica* 12.4 (1992), pp. 449–461. ISSN: 0209-9683. URL: https://doi.org/10.1007/BF01305237 (cit. on pp. 1, 48).

[Rei08] Omer Reingold. “Undirected connectivity in log-space”. In: *J. ACM* (2008) (cit. on pp. 1, 2, 25, 32, 33, 41, 43, 47, 48).

[RTV06] Omer Reingold, Luca Trevisan, and Salil Vadhan. “Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem”. In: *STOC’06: Proceedings of the 38th Annual ACM Symposium on Theory of Computing*. ACM, New York, 2006, pp. 457–466. ISBN: 1-59593-134-1. URL: https://doi.org/10.1145/1132516.1132583 (cit. on p. 33).

[RV05] Eyal Rozenman and Salil Vadhan. “Derandomized squaring of graphs”. In: *Approximation, randomization and combinatorial optimization*. Vol. 3624. Lecture Notes in Comput. Sci. Springer, Berlin, 2005, pp. 436–447. URL: https://doi.org/10.1007/11538462_37 (cit. on pp. 33, 47).

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. “Entropy waves, the zig-zag graph product, and new constant-degree expanders”. In: *Ann. of Math. (2)* 155.1 (2002), pp. 157–187. ISSN: 0003-486X. URL: https://doi.org/10.2307/3062153 (cit. on pp. 1, 25, 32, 33).

[Sav70] Walter J. Savitch. “Relationships between nondeterministic and deterministic tape complexities”. In: *J. Comput. System Sci.* 4 (1970), pp. 177–192. ISSN: 0022-0000. URL: https://doi.org/10.1016/S0022-0000(70)80006-X (cit. on p. 47).

[SZ99] Michael Saks and Shiyu Zhou. “$\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$”. In: vol. 58. 2. 36th IEEE Symposium on the Foundations of Computer Science (Milwaukee, WI, 1995). 1999, pp. 376–403. URL: https://doi.org/10.1006/jcss.1998.1616 (cit. on p. 1).

[TaS17] Amnon Ta-Shma. “Explicit, almost optimal, epsilon-balanced codes”. In: *STOC’17—Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, New York, 2017, pp. 238–251. ISBN: 978-1-4503-4528-6. URL: https://doi.org/10.1145/3055399.3055408 (cit. on p. 1).