MAC 499 – Trabalho de Formatura Supervisionado

IME-USP – BCC, 2001 – Prof. responsável: Carlos Eduardo Ferreira

Projeto de Iniciação Científica
Autômatos Finitos: Algoritmos e Estruturas de Dados

Aluno: Leo Kazuhiro Ueda

Supervisora: Nami Kobayashi

Por que esse tema?

- No final de 2000 estava procurando por um projeto de iniciação científica.
- Muitos assuntos aparentemente interessantes e pouco conhecimento sobre eles: indecisão.
- Profa. Nami ofereceu ajuda.
- Pesquisei vários projetos em Autômatos, mas continuei com o mesmo problema.
- Formulamos então este projeto.

O Projeto

- Objetivo: adquirir mais conhecimentos fundamentais da Teoria de Autômatos Finitos para poder futuramente estudar aplicações mais complexas.
- Até agora estudamos:
  - Algoritmos de Minimização de Autômatos Finitos
    - Algoritmo de Hopcroft
    - Algoritmo de Revuz para autômatos acíclicos
  - Aplicação num problema de busca de padrões
    - Algoritmo Aho-Corasick
  - Um pouco de Autômato dos Sufixos

Só lembrando...

- Um autômato é uma quíntupla $(Q, \Sigma, \delta, q_0, F)$
  - Q é o conjunto de estados
  - Σ é o alfabeto (conjunto de símbolos)
  - δ é a função de transição
  - q_0 é o estado inicial
  - F é conjunto dos estados finais
- Diremos que n é o número de estados do autômato (n = |Q|).

Algoritmo de Hopcroft

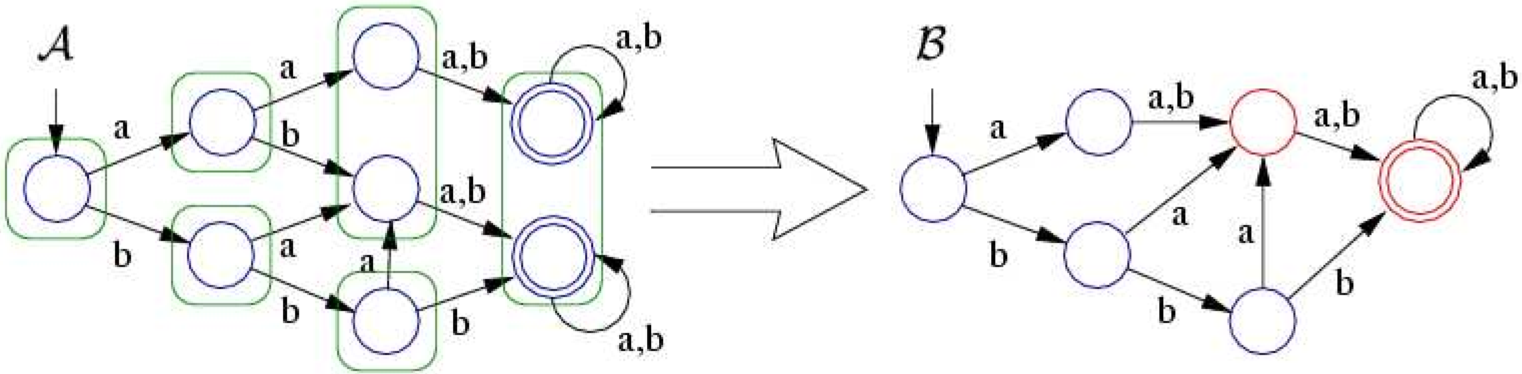
- Em 1971, *John Hopcroft* publica um novo algoritmo $O(|\Sigma|\ n \log n)$.
- O artigo original era muito complicado, então *David Gries* publica uma descrição mais clara (com uma pequena modificação).
- Um outro artigo recente afirma que, ao contrário do que todos acreditam, a versão de Hopcroft não é $O(|\Sigma|\ n \log n)$.

Minimização

- O problema que queremos resolver é minimizar o número de estados de um autômato finito determinístico.
- Em outras palavras, dado um autômato A, queremos encontrar o autômato minimal B com o menor número de estados e tal que $L(A) = L(B)$.
- Muitos já devem ter visto um algoritmo básico $O(|\Sigma|\, n^2)$ (ou pior) no curso de Autômatos. A idéia dos dois algoritmos apresentados aqui é parecida.

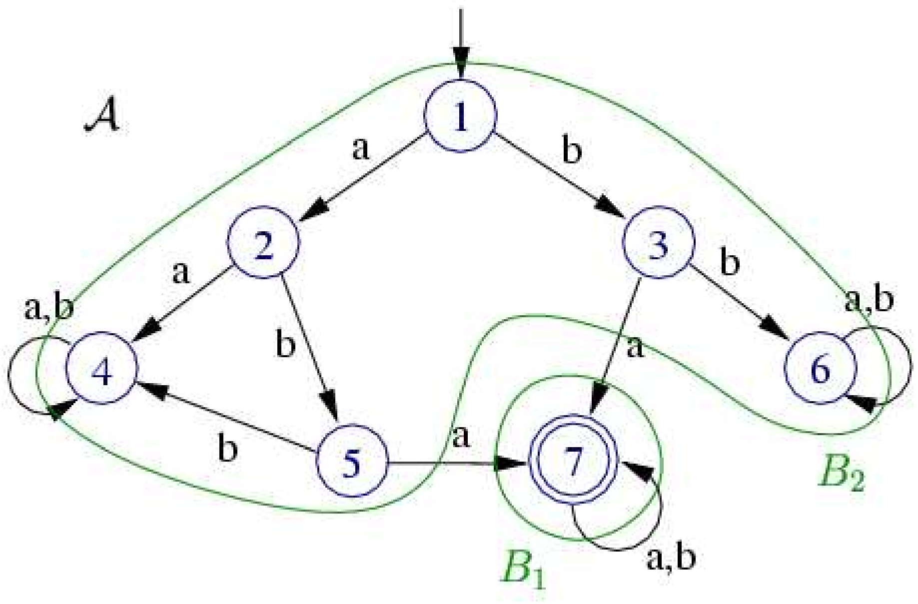
A idéia básica

- Os três algoritmos de minimização procuram pelas classes de equivalência em Q.
- As classes de equivalência representam estados no autômato minimal:



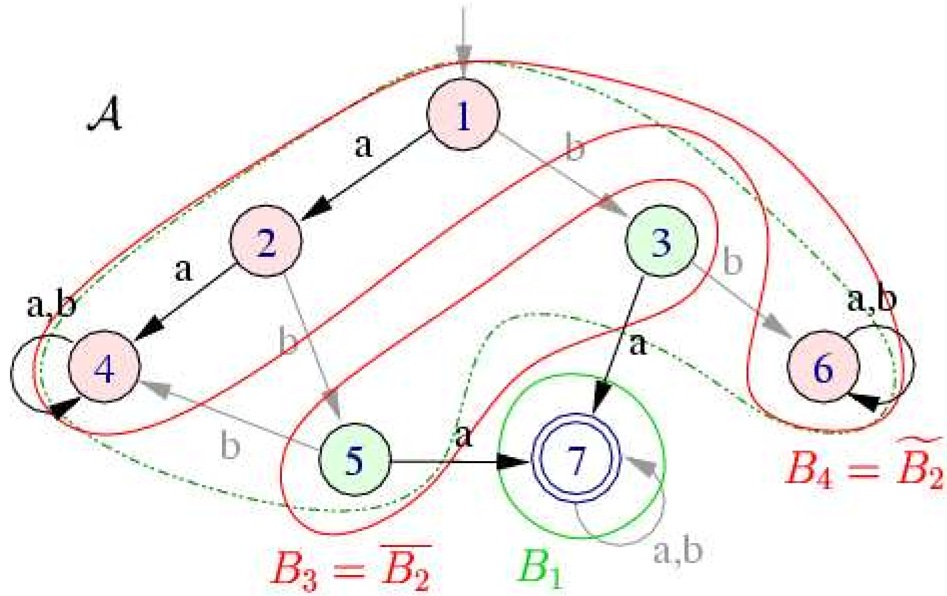
Hopcroft – Início

- O algoritmo começa com uma partição de Q contendo apenas dois blocos:
  - $B_1 = F$
  - $B_2 = Q - F$



Hopcroft – Operação de divisão

- Em seguida, os blocos da partição são divididos de forma que, no final, cada bloco represente uma classe de equivalência.
- Exemplo de divisão de B_2 em relação a (B_1, a):



Hopcroft – Observações

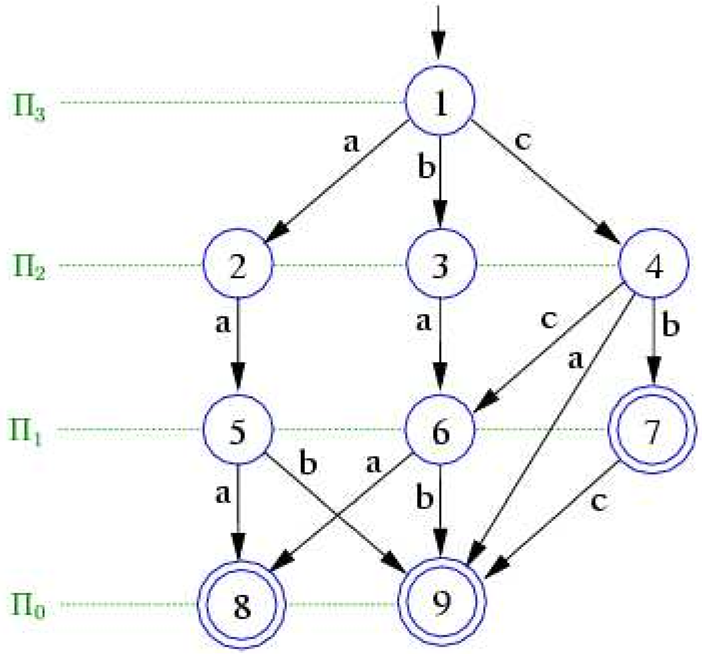
- O que faz de fato o algoritmo ser *n log n* é a ordem em que são feitas as divisões, e essa é principal característica do algoritmo.
- Essa complexidade de tempo depende de uma implementação eficiente de algumas estruturas de dados (listas ligadas e conjuntos)
- O espaço gasto com as estruturas é $O(|\Sigma|n)$.

Algoritmo de Revuz

- É aplicado em autômatos acíclicos.
- Tem complexidade $O(|\Sigma|n)$ de tempo e espaço .
- Um autômato acíclico é um autômato cujo grafo é acíclico.
- Autômatos acíclicos são muito usados para representar dicionários e manipular funções booleanas.

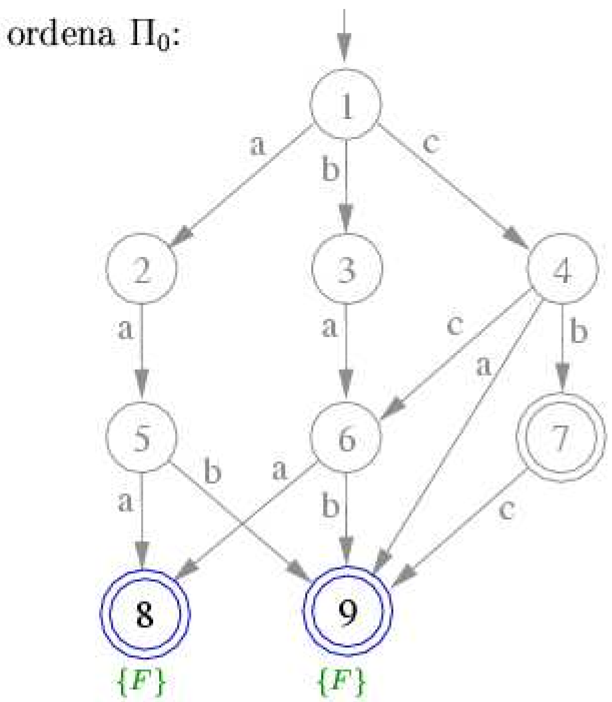
Revuz – Início

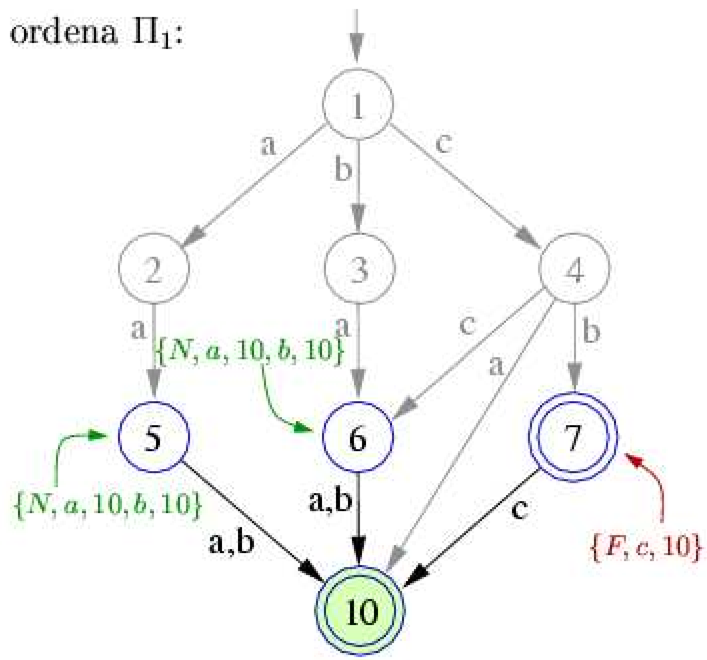
- Inicialmente é calculada uma partição $\Pi = \Pi_1, \Pi_2, \ldots$, onde um estado pertence ao bloco Π_i, sse a sua altura é i. A figura ilustra essa situação:



Revuz – Idéia

- A idéia é nomear cada estado de um bloco de acordo com as suas transições, de forma que um estado é equivalente a outro sse têm nomes iguais.
- Em seguida ordenamos os nomes de cada bloco para encontrarmos os nomes iguais.





Revuz – Observações

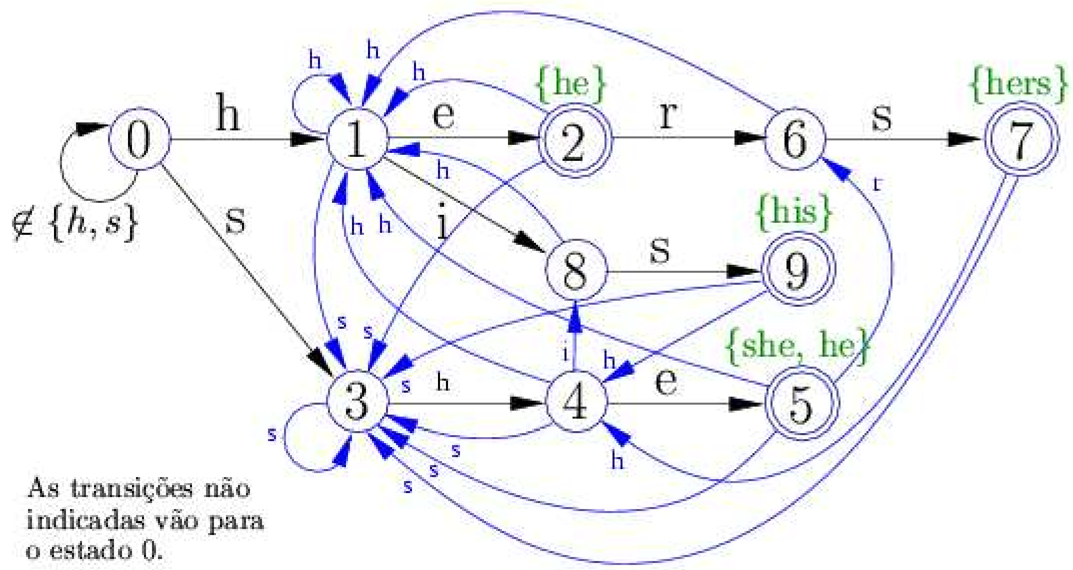
- O algoritmo de ordenação deve ser linear
  - O ordenação proposta por Revuz é semelhante ao radix sort.
- Além disso, os nomes dados aos estados não podem ser muito grandes (têm que ser de alguma forma lineares também).

Um problema de busca de padrões

- Estudamos também a aplicação de Autômatos Finitos num caso de busca de padrões num texto
- Dados um dicionário K com palavras em $\Sigma^{*,}$ e um texto x, também uma palavra em Σ^*, queremos saber quais palavras de K ocorrem em x.
- O algoritmo de Alfred Aho e Margaret Corasick constrói um autômato que reconhece a linguagem $\Sigma^* K$.
- Uma busca de x nesse autômato consome tempo linear no tamanho de x, ou seja, independe de K!

Aho-Corasick - Exemplo

- Exemplo para K = {he, she, hers, his}:



Aho-Corasick - Observações

- O tempo de construção do autômato pode ser linear no tamanho de K.
- O espaço também pode ser linear em K.
- Na verdade, as complexidades de tempo e espaço dependem muito da implementação. Em geral é feita uma troca de espaço por tempo ou vice-versa.
- A construção utiliza as mesmas idéias do algoritmo KMP.

Implementações

- Implementamos os três algoritmos em C.
- O objetivo das implementações era apenas compreender melhor os algoritmos.
- Gerou discussões interessantes.

Frustrações

- Trabalho praticamente individual
- Resultados não muito concretos

Conclusão

- A experiência foi positiva, aprendi bastante.
- A forma de trabalho com a orientadora foi uma novidade interessante.
- As disciplinas que eu apliquei diretamente neste trabalho foram:
  - Autômatos;
  - Análise de Algoritmos e Geometria Computacional, para a parte de análise de algoritmos...
  - As disciplinas básicas e de programação, como ED, Lab1 e 2, POO, Concorrente, XP.