O problema do Menor Ancestral Comum

Revisão, Melhorias e Implementação

Daniel N. Ribeiro
danielrb@linux.ime.usp.br

ORIENTADOR: Alair Pereira do Lago
alair@ime.usp.br

1 Histórico

O problema do Menor Ancestral Comum em árvores tem uma longa história de melhoras. A primeira solução não trivial aparece em 1973 e é publicada num trabalho de Aho et al. A primeira solução de consultas em tempo constante após pré-processamento linear é de 1984 e aparece num artigo de Harel e Tarjan. Estes algoritmos são muito complexos e foram simplificados em 1988 por Schieber e Vishkin e em 2000 por Bender e Farach-Colton (que é uma simplificações do algoritmo paralelo presente no trabalho de Berkman et al.).

2 O Problema

Começamos definindo o que é um ancestral de uma árvore.

Dado um vértice v de uma árvore $\mathcal{T}$ de raiz r, existe um único caminho de r até v. Qualquer vértice neste caminho será dito um ancestral *de v.*

Então o problema consiste em, dados dois vértices em $\mathcal{T}$, obter o ancestral de ambos mais distante da raiz, que chamamos de *menor ancestral comum*(MAC).

Segue um exemplo onde o m é o menor ancestral comum de u e v:

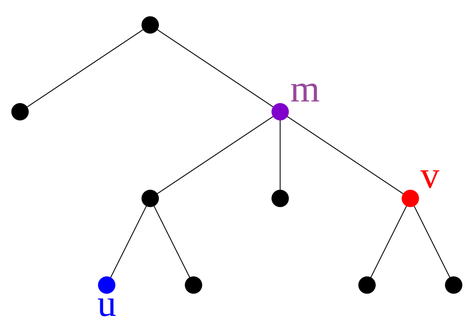

Figura 1: Um exemplo de MAC

3 Primeira Solução

Supomos que conhecemos a *profundidade* dos nós (informação obtível com $O(n)$ de consumo de tempo e espaço, n o número de vértices da árvore). Também supomos que a árvore é dada através de uma função *Pai*. Obtemos o menor ancestral comum dos vértices u e v (supondo, sem perda de generalidade, que a profundidade de u é maior ou igual a de v) buscando o ancestral u'de u que possui a mesma profundidade que v.

Se $u' = v$, este é o menor ancestral comum de u e v. Caso contrário obtemos uma seqüencia de ancestrais de u' e v através da função *Pai* até que encontramos o primeiro par de ancestrais de u' e v que possuem a mesma profundidade e sejam iguais, e sendo portanto o menor ancestral comum de u e v.

4 Uma Redução

O problema do Range Minimum Query (RMQ), recebe um vetor de números $A[1,\ldots,n]$ como entrada. Então, para índices i e j entre 1 e n, quer-se saber qual é o elemento de menor valor no subvetor $A[1,\ldots,n]$.

Uma versão mais restrita para o problema (chamado de RMQ$\pm$1) é aquela onde exigimos que A seja um vetor de inteiros, e todos os elementos adjacentes diferirem entre si de 1 (esta sendo a *restrição* ± 1), temos o problema RMQ$\pm$1.

O problema do MAC é reduzido ao problema RMQ$\pm$1 da seguinte forma:

Pré-processamento:

1. Seja E um vetor que contém todos os vértices que são visitados por um passeio Euleriano em T, começando pela raiz. Então E possui $m := 2n-1$ elementos, e $E[i]$ é o símbolo que representa o i-ésimo nó visitado no passeio Euleriano.
2. Seja a *profundidade* de um nó, a sua distância à raíz. Computa-se o vetor de profundidades $L[1,\ldots,n]$
3. Seja o *representante* de um nó no passeio Euleriano o índice da primeira ocorrência do nó no passeio. Obtém-se um vetor $R[1,\ldots,n]$ tal que $R[i]$ é o índice do representante do nó i.

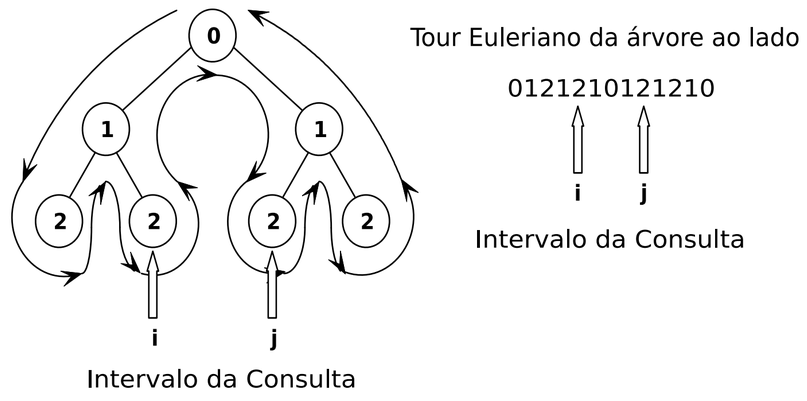

Figura 2: Exempo de uma redução

Portanto, para se obter o $MAC(u,v)$, basta obter $E[_{RMQ_L}(R[u],R[v])$, segundo o lema 1. NOTA: RMQ_L é o RMQ resolvido para o vetor L.

O pré-processamento que faz a redução é feito como consumo de tempo e memória lineares. Já o RMQ$\pm$1 pode também ser resolvido em tempo constante após um pré-processamento linear em memória e espaço, ao dividir o vetor em blocos de mesmo tamamho, e utilizar técnicas de programação dinâmica. Tal resultado vem de dois fatos:

- Se dois vetores de mesmo tamanho satisfazem a restrição ± 1 e diferem apenas de uma constante em algum ponto, eles diferem desta mesma constante em todos os pontos, e toda consulta RMQ tem como resultado um elemento que está na mesma posição dos dois.
- Há $2^{\text{tamanho dos blocos}-1}$ vetores que satisfazem a restrição ± 1 e cujo primeiro elemento é zero.

5 Melhorias e Implementações

Foram implementados uma versão do algoritmo descrito por Bender e Farach-Colton, e uma versão melhorada deste, proposta em [dLS03]. Ambos diferem essencialmente na resolução do problema RMQ$\pm$1 para os blocos. Enquando aquele utiliza matrizes para as computações internas, este utiliza a técnica dos quatro russos, o que permite uma economia na utilização de memória (o que é muito relevante para as aplicações do MAC em problemas de biologia computacional).

Foi também implementada uma segunda melhoria no algoritmo, utilizando o fato de que para resolver o problema do RMQ$\pm$1, basta resolver o mesmo problema num vetor maior que obedece algumas propriedades. Com isto houve um ganho sensível na memória temporária dispendida, e um ganho pequeno na memória utilizada no pré-processamento final.

6 Aplicações

Uma árvore de sufixos de uma palavra é estrutura compacta capaz de armazenar todos os fatores desta palavra, e que pode ser consultada de forma eficiente (consumo linear de tempo e espaço).

Resolvendo o MAC em tempo constante depois de um pré-processamento linear, e aplicando-o à árvores de sufixos, pode-se resolver de forma eficiente os seguintes problemas:

- Encontrar todos os palíndromos maximais/repetições encadeadas de um texto
- Buscas de padrões em textos admitindo erros num dos dois ou nos dois
- Encontrar textos nos quais um padrão aparece ao menos uma vez (que é o mesmo problema dos sites de busca, tais como o Google

Referências

[dLS03] Alair Pereira do Lago and Imre Simon. *Tópicos em Algoritmos sobre Seqüências*. IMPA, Rio de Janeiro, 2003. ISBN 85-244-0202-4.

Apoio pelo processo 05/50796-0 FAPESP