Universidade Federal de Minas Gerais

Bacharel em Sistemas de Informação
Algoritmos e Estruturas de Dados 3

Trabalho Prático 1
Agosto 2017

Gabriel Silva Bastos
Matrícula: 2016058204

1 Introdução

O objetivo do trabalho é prover uma comparação que evidencie os ganhos de eficiência ao se estruturar dados de forma inteligente. Para tal, foi proposto um problema e duas formas de se resolver o mesmo. O problema consiste em obter informações sobre máximos, mínimos e somas em intervalos de um vetor mutável. A primeira solução consiste em estruturar os dados de forma trivial, Já a segunda solução implementa uma estruturação mais elaborada. Desta forma, será feita a comparação da eficiência de ambas soluções.

2 Visão Geral das Soluções

Tendo como entrada um vetor de n números, ambas soluções são formas de se pré-computar triplas contendo o mínimo, máximo e a soma dos elementos em dado intervalo $[i, j]$ contido no vetor.

$$T_{i,j} = (\text{Min}, \text{Max}, \text{Soma})_{i,j}$$

2.1 Matriz

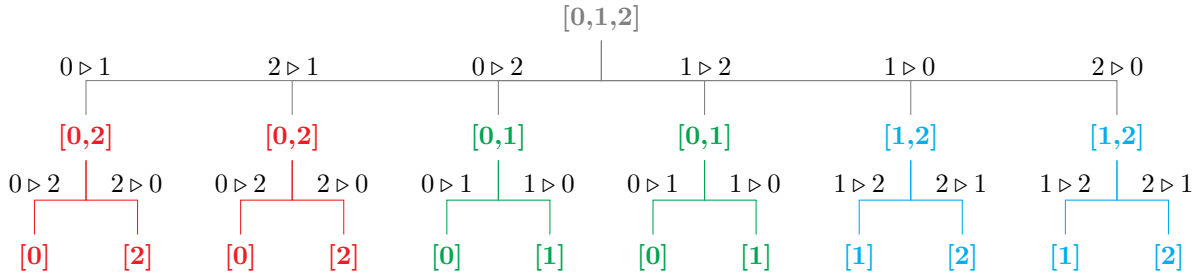
Uma matriz foi escolhida como forma trivial de se estruturar os dados. Cada entrada na posição i, j na matriz contém a tupla correspondente ao intervalo $[i, j]$.

$$\begin{bmatrix} T_{1,1} & \dots & T_{1,n} \\ \vdots & \ddots & \vdots \\ T_{n,1} & \dots & T_{n,n} \end{bmatrix}$$

Para se obter a tripla de um certo intervalo, uma simples consulta na matriz é suficiente.
Para cada alteração no vetor, a matriz deve ser reconstruida.

2.2 Árvore de Segmentos

Uma árvore de segmentos foi escolhida como forma elaborada de se estruturar os dados. Cada nó na arvóre representa um intervalo. As folhas representam todos intervalos de largura 1, já os nós internos representam o intervalo resultante da união dos intervalos de seus descendentes. A raiz representa o intervalo correspondente a todo o vetor.



Para se obter a tripla de um certo intervalo i, é necessário percorrer a árvore a partir da raiz. Para um nó que reprenta um intervalo j, há as seguintes possibilidades:

- $j \subseteq i$: A tripla contida no nó é associada ao resultado final. Os descendentes do nó são desconsiderados.
- $j \cap i \neq \varnothing$: A tripla contida no nó é desconsiderada. Continua-se a iteração em seus descendentes.
- $j \cap i = \varnothing$: O nó e seus descendentes são desconsiderados.

Para alterar dada folha na árvore, é necessário apenas alterar seu valor e recomputar o valor de todos seus antecessores até a raiz.

3 Análise de Complexidade

3.1 Matriz

3.1.1 Espacial

Para um vetor de tamanho n, a matriz com todos intervalos possui dimensão $n \times n$.
Portanto a complexidade espacial da matriz é $\Theta(n^2)$.

3.1.2 Temporal

Três operações são relevantes para análise temporal:

- Criação: A criação da matriz demanda dois passos combinados.
  - É necessário computar cada elemento da matriz de dimensão $n \times n$.
    Portanto, este passo possui complexidade: $\mathcal{O}(n^2)$.
  - Para cada elemento, é necessário calcular a tripla correspondente.
    Tal operação é linear no tamanho do vetor. Complexidade: $\mathcal{O}(n)$.

  A combinação desses dois passos resulta em $\mathcal{O}(n^2) \cdot \mathcal{O}(n) = \mathcal{O}(n^3)$.
- Consulta: Para consultar a tripla correspondente a um intervalo $[i, j]$, basta acessar diretamente o elemento na posição i, j da matriz. Portanto, a complexidade é $\Theta(1)$.
- Atualização: Como a atualização de um elemento no vetor implica na reconstrução de toda a matriz, a complexidade é a mesma da operação de criação: $\Theta(n^3)$.

3.2 Árvore

3.2.1 Espacial

Devido à preferencia de Nubby por memória sequencial, a árvore teve de ser implementada em um vetor. Para tal, é necessário que a árvore seja completa. O número de nós em uma árvore binária completa de x folhas é $2x-1$. Porém, o número de elementos n da entrada não necessariamente é uma potência de 2, o que torna necessário obter a potência de 2 mais próxima que é maior que n. Então, o número de nós da árvore de segmentos será

$$2 \cdot 2^{\lceil \log_2 n \rceil} - 1$$

Considerando as propriedades da função teto, tal expressão jamais será superior a expressão[1]

$$2 \cdot 2^{\log_2(n)+1} - 1$$

que pode ser simplificada:

$$\begin{gathered} 2 \cdot 2 \cdot 2^{\log_2(n)} - 1 \\ 2 \cdot 2 \cdot n - 1 \\ 4n - 1 \end{gathered}$$

A complexidade espacial da árvore é portanto $\mathcal{O}(n)$.

[1] https://discuss.codechef.com/questions/85040/size-of-segment-tree

3.2.2 Temporal

Três operações são relevantes para análise temporal:

- Criação: Na criação da árvore, cada nó é computado somente uma vez. Como há $2n - 1$ nós no total, a complexidade é $\Theta(n)$.
- Consulta: Para consultar a tripla correspondente a um intervalo i, basta percorrer a árvore a partir da raiz, associando as triplas dos nós cujo intervalo está totalmente contido no intervalo i. Em cada nível, no máximo dois nós serão expandidos.[2] Como a altura da árvore é $\log n$, no máximo $2 \cdot \log n$ nós serão expandidos no total.
  Portanto, a complexidade é $\mathcal{O}(\log n)$.
- Atualização: Para atualizar uma folha, esta e todos os seus antecessores são atualizados. Como há apenas um antecessor por nível, o número de atualizações é equivalente à altura da árvore. Portanto, a complexidade é $\Theta(\log n)$.

4 Análise Experimental

A análise experimental da implementação é mostrada pela figura 1. Para realizar os experimentos, foi feito um gerador de entradas. Para um valor n, é gerada uma entrada contendo um vetor de tamanho n, e $n/2$ operações de busca e atualização intercaladas. O intervalo em cada operação é aleatório, mas a largura do intervalo é fixa em $n/3$. Para medir o tempo de execução do código, foi utilizada a informação de uso de recursos fornecida pelo sistema operacional linux. Os valores n foram escolhidos estrategicamente para cada teste. A razão inicial é verificar, de forma geral, o comportamento dos algoritmos para casos genéricos.

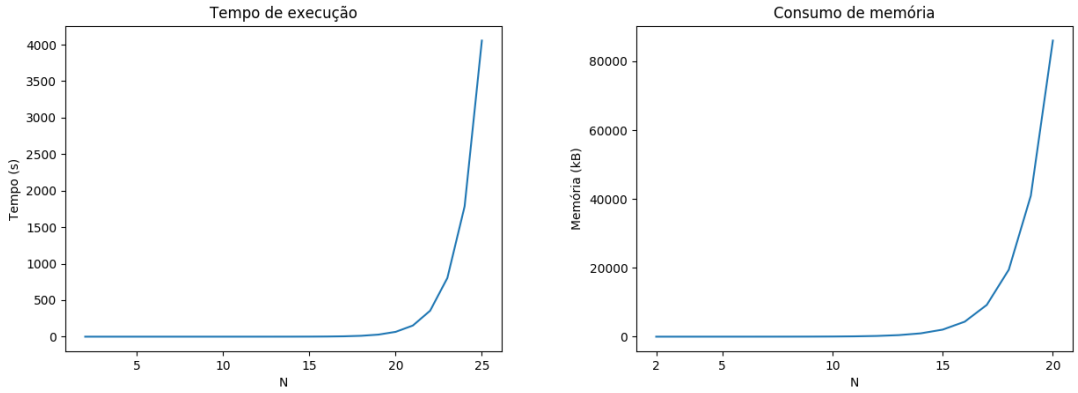


Figura 1: Teste experimental

Nos gráficos da figura 1 é notável o ganho em eficiência da árvore de segmentos sobre a matriz.

5 Conclusão

O objetivo do trabalho foi atingido ao demonstrar as diferenças na eficiência das soluções propostas, tanto teoricamente quanto experimentalmente. Tornou-se evidente o grande ganho obtido na performance ao se utilizar uma estruturação de dados adequada.

[2] https://cs.stackexchange.com/questions/37669/time-complexity-proof-for-segment-tree-implementation-of-the-ranged-sum-problem