Universidade Federal de Lavras
Departamento de Ciência da Computação

Trabalho Prático

Heap Mínima com Construção em tempo linear

Aluno: Lucas Fiorini Braga
Professor: Sanderson L. Gonzaga de Oliveira

Maio
2019

Universidade Federal de Lavras
Departamento de Ciência da Computação

Heap Mínima com Construção em tempo linear

Solução proposta para construção da estrutura de dados em tempo linear. Trabalho apresentado à disciplina GCC253 - Complexidade e Projeto de Algoritmo.

Aluno: Lucas Fiorini Braga

Professor: Sanderson L. Gonzaga de Oliveira

Maio
2019

Conteúdo

1 Objetivos

O objetivo deste trabalho é implementar e provar que uma estrutura de dados do tipo Heap Mínima pode ser construída em tempo linear. Dessa forma, é necessário implementar a estrutura, realizar testes e provar matemáticamente a hipótese citada acima.

2 Requisitos do Trabalho

- Data da entrega: 16/05/2019
- Usar preferencialmente linguagem de programação C/C++
- Testes com diversas entradas
- Gráficos com resultados obtidos

3 Desenvolvimento

A estrutura de dados Heap, introduzida por J. W. J. Williams em 1964, é representada por uma árvore binária completa, a qual pode ser representada de duas formas: Heap Mínima e Heap Máxima. Na Heap Mínima, estrutura abordada nesse trabalho, todos os nós pai são menores ou iguais que seus filho. Essa estrutura é normalmente implementada em um vetor de tamanho fixo onde o pai pode ser encontrado no índice ((n / 2) - 1) onde n é a quantidade de elementos. Os nós filhos são encontrados no índice ((2 * i) + 1) para o filho da esquerda e ((2 * i) + 2) para o filho da direita onde i representa o índice do pai. Dessa forma, há duas maneiras de construí-la. A primeira seria inserir elemento por elemento e quando uma regra da estrutura for desrespeitada, é necessário um método de balancear os dados de forma que as regras da estrutura sejam novamente respeitadas. AsSim, como nessa abordagem os elementos são inseridos um a um na próxima posição vazia do vetor, o algoritmo corrige o balanceamento trocando esse novo nó filho por seu pai caso o filho seja menor que o pai, ou seja, ele corrige a árvore de baixo para cima. Logo, os nós são adicionados cada vez mais profundamente de acordo com o crescimento da quantidade de elementos e talvez, se o valor dos mesmos forem cada vez menores que os já existentes, eles terão que subir até o topo. Com essa forma de construir, a complexidade de tempo em notação assintótica ficaria **O(n log n)**, pois todas as folhas (segunda metade do vetor) estão inseridas em uma profundidade de log n. A outra forma seria inserir todos os elementos no vetor e arrumar cada sub-árvore por vez. Assim, nesse caso, seria mais relevante a altura da árvore do que a profundidade da mesma, o que nos traz a hipótese de que a complexidade de tempo em notação assintótica dessa abordagem seria de **O(n)**, que será provada em outro tópico subsequente.

3.1 Proposta para construção de uma Heap Mínima em Tempo Linear

Inicialmente foi implementado uma classe chamada MinHeap a qual possui como atributos a capacidade da estrutura, e um ponteiro para um inteiro que vai, posteriormente, apontar para a primeira posição de um vetor de elementos da Heap Mínima. Tal classe possui como métodos o swap que troca dois elementos de posição entre si, os métodos leftSon e rightSon que vão retornar o índice do filho esquerdo ou direito de um determinado pai respectivamente. O método heapfy representa a forma de troca de elementos top-down ou seja corrigir de forma decrescente de níveis. Como o intuito do código é demonstrar a construção em tempo linear da estrutura, construtor

da classe MinHeap, após preencher o vetor com elementos fora de ordem ou aleatórios, já entra em um laço de iteração que passa por cada nó pai aplicando o heapfy para balancear cada sub-árvore. A função que orquestra as ações do programa, a main, permite a instanciação da classe MinHeap e a escolha entre a inserção de ítens ordenados de forma decrescente ou aleatórios.

3.2 Prova

Considerando que a Heap tenha k níveis, o nível da raiz é k = 0 e o nível das folhas é k. Logo, é notável que o nível das folhas não são afetadas pelo algoritmo de balanceamento, pois não há nível inferior nas folhas para que as mesmas possam fazer alguma troca com um nível inferior. Assim, a quantidade de trocas t é igual a zero. Dessa forma, até o nível da raiz teremos:

$$(0*\text{n}) + (1*\text{n}) + (2*\text{n}) \ldots = \sum_{t=0}^{k} t * n$$

Onde n é a quantiade de nós de cada nível sendo $\text{n} = 2^{k+1} - 1$,
ou seja, n representa uma árvoce completa com todos os espaços do vetor preenchidos e a quantidade de nós por nível sendo 2^{k-t}

Logo, temos

$$\sum_{t=0}^{k} t2^{k-t} = 2^k \sum_{t=0}^{\infty} \frac{t}{2^t}$$

Assume-se a série para $x\epsilon(-1, 1)$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 \ldots = \sum_{t=0}^{\infty} x^t$$

Derivando ambos os lados e depois multiplicando-os por x tem-se

$$\sum_{t=0}^{\infty} tx^t = \frac{x}{(1-x)^2}$$

Aplicando x = 0,5

$$2^k \sum_{t=0}^{\infty} \frac{t}{2^t} = \frac{0,5}{(1-(0,5))^2} = 2$$

Então temos

$$T(n) = 2^k \sum_{t=0}^{\infty} \frac{t}{2^t} \leq 2^k * 2 = 2^{k+1}$$

Assim,

$$n + 1 = 2^{k+1}$$

$$T(n) \leq n + 1 \in O(n)$$

4 Resultados

Foram inseridos valores de forma decrescente na estrutura e plotado um gráfico que mostra o tempo de inserção para cada quantidade de nós inserida.

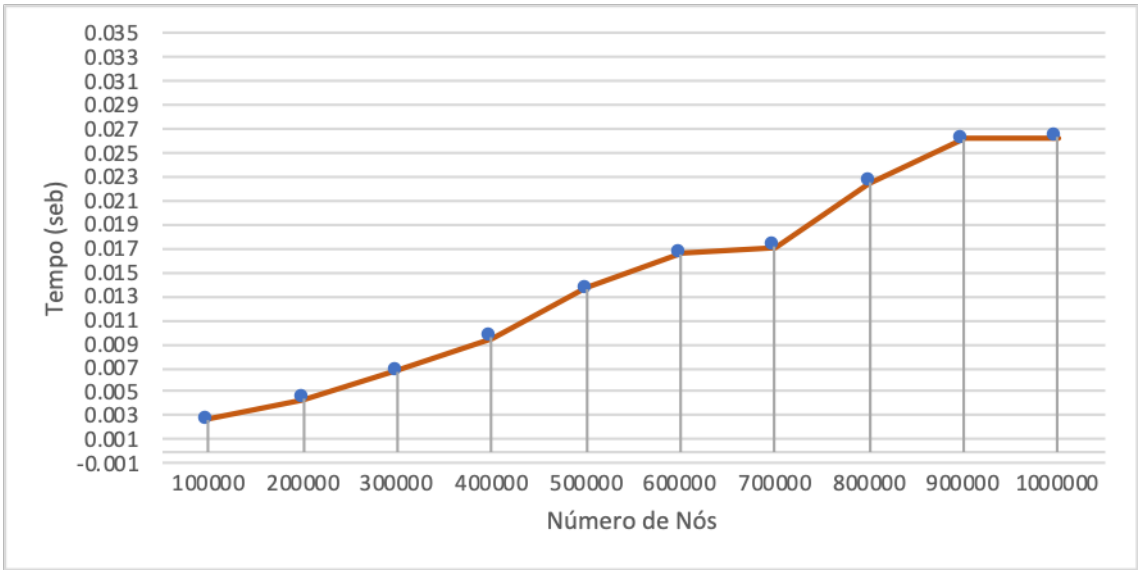

Figura 1: Valores em ordem decrescente

Pode-se constatar a linearidade para construção da Heap Mínima também com a mesma quantidade de valores aleatórios.

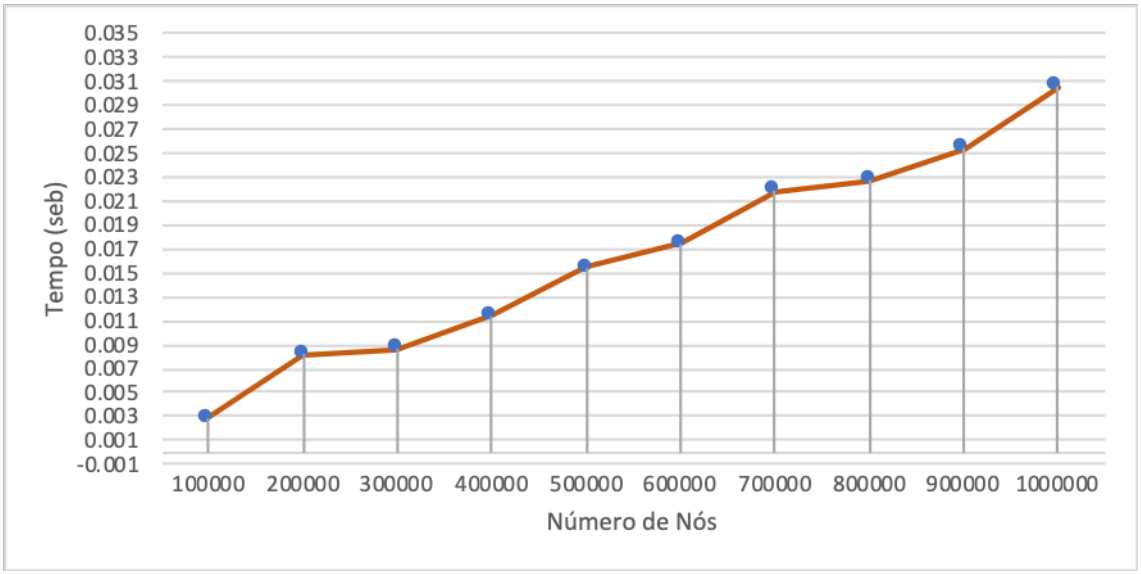

Figura 2: Valores Aleatórios

Os dados exibidos a cima foram obttidos de uma arquitetura com 16GB de Ram 1600Mhz DDR3, 2,7 GHz Intel Core i5.

5 Conclusão

Com base na prova e nos resultados obtidos,pode-se concluir que é possível contruir uma estrutura de dados do tipo Heap Mínima com a complexidade de tempo linear. Nesse caso, O(n) lembrando que nessa abordagem, todos os nós são acessados pelo menos uma vez. Já a disposição dos dados (ordem decrescente e aleatórios) parece não afetar de forma muito significativa o tempo de execução.

Referências

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Algoritmos Teoria e Prática. ISBN 978-0-262-03384-8.

[2] David Scot Taylor, Linear Time BuildHeap
`https://www.youtube.com/watch?v=MiyLo8adrWw`