Estrutura de Dados - MC202 A

1º Semestre de 2018

Tiago de Paula Alves - 187679

Lista 2 - Exercício 2

a) Considerando o heap cheio, cada nó do nível n tem d filhos no nível $n+1$, sendo que o primeiro nó de n tem como filhos os d primeiros nós de $n+1$ (1, 2 até d), o segundo tem os d seguintes ($d+1$, $d+2$ até $2d$), e o m-ésimo ($1 \leqslant m \leqslant n$) tem $(m-1)d+1$, $(m-1)d+2$ até $m\ d$, e, seguindo a mesma ideia, o pai de m é o $\lceil m/d \rceil$-ésimo termo do nível $n-1$. Como o número de elementos de cada nível n de um heap d-ário cresce como uma série geométrica do tipo $a_n = a_0\ r^{n-1}$, em que $a_0 = 1$ e $1 < r = d$, o índice geral de um elemento m do nível n é $S(n-1)+m$, sendo $S(n) = \sum_{i=1}^{n} a_i = \frac{a_0\ (1-r^n)}{1-r}$ a soma geométrica dos n primeiros termos, ou seja, a soma dos n primeiros níveis. Então, o $k-ésimo$ filho de $i = \frac{1-d^{n-1}}{1-d} + m - 1$ (o -1 serve apenas para ajustar para a indexação por zero) é:

$$\begin{aligned}
filho(i,k) &= \frac{1-d^n}{1-d} + d \cdot (m-1) + k - 1 \\
&= \frac{(1+d-d) - d\ d^{n-1}}{1-d} + d\ (m-1) + k - 1 \\
&= \frac{1-d}{1-d} + d\ \frac{1-d^{n-1}}{1-d} + d\ (m-1) + k - 1 \\
&= 1 + d\ i + k - 1 \\
&= d\ i + k
\end{aligned}$$

E o pai:

$$\begin{aligned}
pai(i) &= \frac{1-d^{n-2}}{1-d} + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= \frac{1-d^{n-1}/d}{1-d} + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= \frac{d-d^{n-1}}{d\ (1-d)} + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= \frac{1}{d}\frac{d-1+1-d^{n-1}}{1-d} + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= \frac{1}{d}\left(\frac{d-1}{1-d} + \frac{1-d^{n-1}}{1-d}\right) + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= -\frac{1}{d} + \frac{(1-d^{n-1})/(1-d)}{d} + \left\lceil \frac{m}{d} \right\rceil - 1 \\
&= \left\lceil \frac{(1-d^{n-1})/(1-d)}{d} + \frac{m}{d} - \frac{1}{d} \right\rceil - 1 \\
&= \left\lceil \frac{i}{d} \right\rceil - 1 = \left\lceil \frac{i}{d} - 1 \right\rceil = \left\lceil \frac{(i-d)}{d} \right\rceil \\
&= \left\lfloor \frac{(i-d)+d-1}{d} \right\rfloor \\
&= \left\lfloor \frac{i-1}{d} \right\rfloor
\end{aligned}$$

Por fim, fica que o k-ésimo filho de i, se existente, é (com o ajuste de indexação) $filho(i,k) = d \cdot i + k$, com $k \in [1,d]$, e $pai(i) = (i-1)/d$.

b)

```
Algoritmo HeapSortTernário:

Entradas: Vetor D de tamanho n

-- heap ternário
d ← 3

-- cada elemento desde o final
-- note que aqui os primeiros elementos, de n-1 a pai(n-1),
-- não terão filhos e poderiam ser removidos do laço
para i de n - 1 até 0:

    -- desce o elemento no heap
    j ← i
    faça:

        -- a partir do elemento
        máx ← j
        máx_k ← 0

        -- encontre o maior filho, caso exista
        para k de 1 até d:
            se filho(j, k) < n e V[filho(j, k)] > V[máx]:
                máx ← filho(j,  k)
                máx_k ← k

        -- troque com o filho
        se j > máx:
            troca(V, j, máx)
        j ← filho(j, máx_k)

    -- até não ter filhos maiores ou não ter filhos
    enquanto máx_k > 0 e j < n;
```

Para esse algoritmo, consideremos o pior caso, que é quando cada nó analisado deve descer até o nível mais baixo. Então, em um heap cheio, cada altura h do heap tem no máximo $\lceil n\ (1 - 1/d)^{h+1} \rceil$ elementos e cada um deles deve descer h nós, o que gasta $O(h)$ em tempo, e são $\lfloor \log_d n \rfloor$ níveis de altura. Assim a complexidade para se construir o heap de um vetor fica:

$$\sum_{h=0}^{\lfloor \log_d n \rfloor} \left\lceil n \ (1 - 1/d)^{h+1} \right\rceil \ O(h)$$

$$O\left(\sum_{h=0}^{\lfloor \log_d n \rfloor} \left\lceil n \ (1 - 1/d)^{h+1} \right\rceil \ h \right)$$

$$O\left(\sum_{h=0}^{\log_d n} n \ (1 - 1/d)^{h+1} h \right)$$

$$O\left(n \sum_{h=0}^{\log_d n} h \left(\frac{d-1}{d} \right)^{h+1} \right)$$

$$O\left(n \frac{d-1}{d} \sum_{h=0}^{\log_d n} h \left(\frac{d-1}{d} \right)^{h} \right)$$

$$O\left(n \sum_{h=0}^{\log_d n} h \left(\frac{d-1}{d} \right)^{h} \right)$$

$$O\left(n \sum_{h=0}^{\infty} h \left(\frac{d-1}{d} \right)^{h} \right)$$

Mas, sabemos que ara $0 < x < 1$:

$$\sum_{n=0}^{\infty} x^n = \lim_{n \to \infty} S(n) = \lim_{n \to \infty} \frac{1 - x^n}{1 - x} = \frac{1}{1 - x}$$

Derivando ambos os lados:

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

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

Aplicando esse resultado com $n = h$ e $x = \frac{d-1}{d}$ na equação incial:

$$O\left(n \frac{(d-1)/d}{(1 - (d-1)/d)^2} \right)$$

$$O\left(n \ d(d-1) \right)$$

$$O(n)$$

Que mostra que o algoritmo tem complexidade linear, já que d é constante para o heap escolhido.