L1 de Estrutura de Dados - IC/UFRJ - 2021.2

Nathan Andrade dos Santos Lobo

DRE: 120082390

Questão 1.

a)

Como a política de expansão será de duas unidades, teremos uma PA no seguinte formato:

$$4 + 6 + 8 + \cdots + 100$$

A soma de todos os termos de uma PA é dado por:

$$S_n = \frac{(a_1 + a_n) \cdot n}{2}$$

Para encontrar n, sabemos que como será inserido 100 elementos e já iniciaremos com 4 destes já adicionados no array, então, temos que:

$$2 \cdot x + 4 = 100$$
$$x = 48$$

Para encontrar o elemento na 48 posição, temos:

$$a_{48} = 4 + (48 - 1) \cdot 2$$
$$a_{48} = 98$$

Por fim, retornando aos termos da PA e substituindo os valores:

$$S_n = \frac{(4 + 98) \cdot 48}{2} = 2448 = O(n^2)$$

b)

Agora, teremos uma PG pois o array terá o seu tamanho dobrado a cada overflow que ele tiver. Além disso, teremos o custo adicional de adicionar os elementos após o array ter o seu tamanho dobrado.
Do que sabemos da soma de todos os termos de uma PG é

$$S_n = \frac{a_1(q^n - 1)}{q - 1}$$

Como precisamos por definição ter $4 \cdot q^{n-1} \leq 100$ e sabendo que $q = 2$, logo:

$$4 \cdot 2^{n-1} \leq 100$$
$$2^{n-1} \leq 25$$
$$n \leq \left\lfloor \frac{\log 2 + 2\log 5}{\log 2} \right\rfloor = 5$$

Substituindo os valores para encontrar o somatório da PG, temos:

$$S_n = \frac{4 \cdot (2^5 - 1)}{2 - 1} = 124$$

Levando em consideração em que nas 5 operações que realizarmos de cópia dos arrays, teremos de escrever os elementos em memória, então, logo

$$124 + 100 = 224 = O(n)$$

Questão 2.

Assumindo que o algoritmo de busca binária já esteja implementado para procurar pelo elemento 82, será realizado os seguintes passos:

1º Passo - Calcular o meio do array, dado o início e o fim.

2º Passo - Buscar no array o elemento que está na posição do meio.

3º Passo - Comparar o elemento com o elemento que estamos buscando e caso seja menor, mudamos o início do array para o meio + 1. Do contrário, mudamos o final do array para o meio - 1.

Executando o algoritmo:

$$\begin{aligned}
\textbf{Início: } & 0 \\
\textbf{Fim: } & 14 \\
\textbf{Meio: } & 7 \\
\text{Array[7]: } & 33 \\
\\
33 & < 82 \\
\textbf{Novo início: } & 7+1=8 \\
\textbf{Fim: } & 14 \\
\textbf{Meio: } & 11 \\
\text{Array[11]: } & 61 \\
\\
61 & < 82 \\
\textbf{Novo início: } & 11+1=12 \\
\textbf{Fim: } & 14 \\
\textbf{Meio: } & 13 \\
\text{Array[13]: } & 88 \\
\\
88 & > 82 \\
\textbf{Início: } & 12 \\
\textbf{Novo fim: } & 13-1=12 \\
\textbf{Meio: } & 12 \\
\text{Array[12]: } & 80 \\
\\
80 & < 82
\end{aligned}$$

Por fim, como o elemento não foi encontrado em nenhuma posição do array podemos considerar este o pior caso de execução e então, ter sido realizado o total de 4 iterações.

Questão 3.

a)

Se formos eliminar um elemento de uma lista encadeada, será necessário percorrer a lista indo de nó em nó até o nó que estamos procurando. Ao encontrar o nó que refere ao elemento que queremos eliminar, precisamos "remendar" esta posição, isto é, ligando o nó anterior do nó que será removido ao nó seguinte do que será removido.

b)

Como já temos um ponteiro como referência para o nó, basta que façamos o "remendo" como dito anteriormente, apontando o nó anterior ao próximo do nó que será removido.

A complexidade no primeiro caso é $O(n)$, pois precisamos atravessar toda a lista buscando o elemento, isto caso ele exista. No caso onde teremos a referência para o nó, será $O(1)$.

Questão 4.

Seguindo o conceito de pilha, os elementos estarão sempre sendo adicionados ao topo da mesma, logo:

Valor - Indíce
8 removido
13 removido
15 removido
$15 - 4$
$13 - 3$
9 removido
$9 - 3$
$8 - 2$
$7 - 1$
$6 - 0$

Portanto, a pilha final contém os seguintes elementos nas seguintes posições:

0 : 6
1 : 7

Questão 5.

No caso de ser uma fila, a regra é que como os elementos a serem removidos serão os do início, basta inverter a pilha anterior tendo os últimos elementos:

0 : 13
1 : 15

Questão 6.

a)

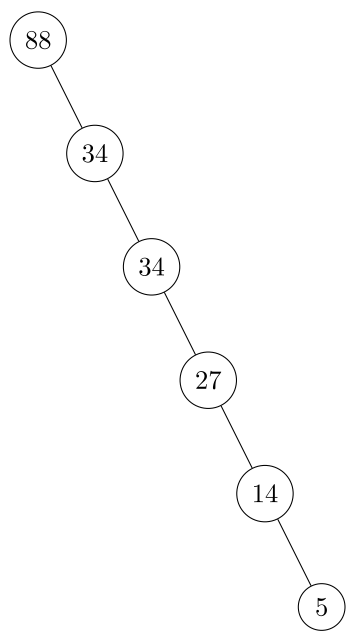
Temos em mãos os seguintes dados de de frequência e chave:

$$(27, 2)$$
$$(34, 3)$$
$$(88, 4)$$
$$(5, 3)$$
$$(14, 1)$$

Logo, o peso máximo é dado por 13.
Para o caso de altura máxima, precisamos calcular o somatório dos pesos vezes a altura respectiva daquele nó, da seguinte maneira:

$$\sum_{i=0}^{n} \frac{k(h_i + 1)}{w}$$

Sendo w o somatório dos pesos e k o peso na determinada altura. Ou seja,



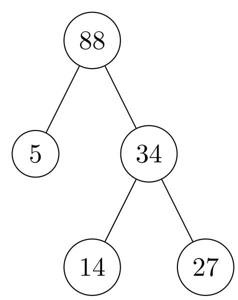
Cada posição em que se encontra o círculo simboliza a altura do mesmo na árvore, indo de 1 à n. Substituindo os elementos pela sua frequência e a altura que foi posto na árvore tomando a equação anterior, sabemos:

$$\frac{4 \cdot 1}{13} + \frac{3 \cdot 2}{13} + \frac{2 \cdot 3}{13} + \frac{1 \cdot 4}{13} + \frac{3 \cdot 5}{13} = \frac{35}{13}$$

Portanto, o custo médio de acesso é $\frac{35}{13}$.

b)

Para calcular a altura mínima, partiremos da definição para organizar os elementos em uma árvore.
Assumindo que n seja o elemento que está na raíz da árvore, o elemento i que estará à esquerda da árvore tem a propriedade tal que $i < n$. Analogamente, para o elemento j a propriedade segue que $j > n$.



Realizando a mesma operação dada a equação da questão anterior, podemos obter o custo médio pelo produto da frequência do elemento pela a altura em que o mesmo se encontra na árvore sobre o peso total, que permanece sendo 13.

$$\frac{4 \cdot 1}{13} + \frac{3 \cdot 2}{13} + \frac{3 \cdot 2}{13} + \frac{1 \cdot 3}{13} + \frac{2 \cdot 3}{13} = \frac{25}{13}$$

Logo, o custo médio de acesso para esta árvore de busca binária mínima é

$$\frac{25}{13}$$

.