Simulado da primeira avaliação

Algoritmos e Estrutura de Dados II (AE43CP)

Prof. Jefferson T. Oliva

Questionário

1. Dado o algoritmo abaixo:

```
int faz_algo(int v[], int n){
01.   int i, j, p, aux;
02.   aux = 0;
03.   for (i = 0; i < n / 2; i++)
04.      aux += v[i] + v[n - i - 1];
05.   for (i = n - 1; i > 0; i--){
06.       for (j = i; j > 0; j--)
07.           if (v[j] < v[j - 1]){
08.               aux = v[j];
09.               v[j] = v[j - 1];
10.               v[j - 1] = aux;
11.           }
12.   }
13.   return aux;
}
```

   Faça:

   a) - Calcule a quantidade de instruções, que o algoritmo pode executar no **melhor caso**. Para isso, mostre o passo a passo detalhado para o cálculo.

   b) - Calcule a quantidade de instruções que o algoritmo pode executar no **pior caso**. Para isso, mostre o passo-a-passo detalhado para o cálculo.

   c) - Determine a complexidade do algoritmo utilizando uma das seguintes notações: O, Ω ou Θ. Em seguida, interprete (explique) a complexidade do algoritmo de acordo com a notação que você escolheu. Pré-requisito: ter feito o item (a) ou (b).

   d) - Determine o espaço extra necessário (em unidades de espaço) para executar o algoritmo. Em seguida, determine a complexidade de espaço utilizando uma das seguintes notações: O, Ω ou Θ.

2. Implemente uma função que inverta uma lista encadeada.

3. Faça:

   a) - Implemente uma função iterativa para fazer a inserção de uma chave (número inteiro) em uma árvore binária de busca. Na árvore binária não podem ser adicionadas chaves repetidas. Exemplo de protótipo de função: *Node* insercao_iterativa(Node *tree, int chave);*

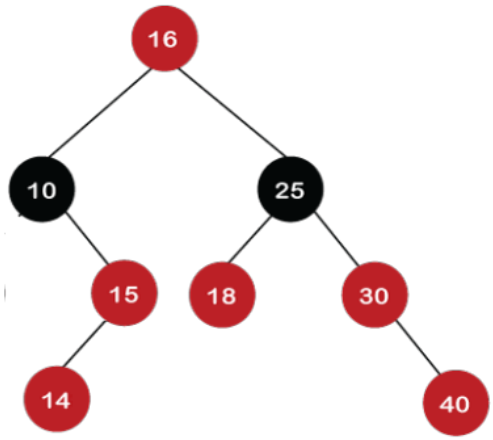
   b) - Em que situação ocorre a inserção em uma árvore binária de busca no tempo médio (caso médio)?

   c) - Calcule a quantidade de instruções executadas no algoritmo implementado por você para o caso médio.

   d) - Com base no item (B), qual a complexidade de tempo do algoritmo para o caso médio?

   e) - Qual a complexidade de espaço para a inserção em árvore binária de busca em um algoritmo recursivo para o caso médio? Como você chegou a tal conclusão?

Continua na próxima página...



4. Dada a árvore rubro-negra abaixo, onde apenas os nós 10 e 25 estão na cor preta:

   a) - Por que a árvore acima está desbalanceada?

   b) - Descreva a(s) operação(ões) necessária(s) para tornar a árvore balanceada. Em seguida, mostre (por meio de desenho) como ficaria a árvore rebalanceada. Na árvore resultante, indique quais nós estão na cor vermelha e quais estão na cor preta.

5. Dada uma árvore B de ordem $N = 4$, responda: (30 pontos)

   a) - Uma árvore de altura 4 pode ter, no máximo, quantas páginas? Mostre como você chegou em tal quantia.

   b) - Uma árvore de altura 4 pode ter, no máximo, quantas chaves? Mostre como você chegou em tal quantia.

   c) - Mostre o porquê do algoritmo busca em uma árvore B de ordem N e com M chaves possui complexidade de tempo na ordem de $O(\log_2(M))$.

Continua na próxima página...

Anexos

- Teorema mestre:
  - Se $f(n) \in O(n^{log_b a - \epsilon})$ para a constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
  - Se $f(n) \in \Theta(n^{log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log_2 n)$
  - Se$f(n) \in \Omega(n^{\log_b a + \epsilon})$ para a constante $\epsilon > 0$, e se $af\left(\frac{n}{b}\right) \leq cf(n)$, para a constante $c < 1$ e n suficientemente grande, então $T(n) \in \Theta(f(n))$

- Estruturas de nó de árvores binárias.

```
typedef struct Node Node;

struct Node{
   int item;
   Node *left, *right;
};
```

- Estruturas de dados e protótipos de função para listas encadeadas:

```
typedef struct Cell{
       int item;
       struct Cell *prox;
   }Cell;

typedef struct{
       Cell *cabeca;
   }Lista;

typedef struct{
       Cell *topo;
   }Pilha;

typedef struct{
       Cell *ini, *fim;
   }Fila;

Cell* criar_celula(int chave);

Lista* criar_lista();

Pilha* criar_pilha();

Fila* criar_fila();

void empilhar(Pilha *p, int chave);

int desempilhar(Pilha *p);

void enfileirar(Fila *p, int chave);

int desenfileirar(Fila *p);

int pilha_vazia(Pilha *p);

int fila_vazia(Fila *f);

void liberar_lista(Lista *l);

void liberar_pilha(Pilha *p);

void liberar_fila(Fila *f);
```

 Continua na próxima página...

Observações

* Não é necessária a implementação dos protótipos acima.
* O acesso indevido aos dados de pilha de fila acarretará no desconto de 25% da nota.

- Séries

$$\sum_{i=1}^{n} 2^i = 2^{n+1} - 1$$

$$\sum_{i=1}^{n} a^i = \frac{a^{n+1}-1}{a-1}$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Final do questionário.