Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP1 - Primeiro Semestre de 2016

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. Defina:

   a. (1,0) Complexidade de pior caso

   Resposta: Seja A um algoritmo e $E = \{E_1, E_2, \cdots, E_n\}$ o conjunto de todas as entradas possíveis de A. Dada a entrada E_i, seja t_i o número de passos efetuados por A, para $1 \leq i \leq n$. Podemos definir a *complexidade de pior caso* como $\max_{E_i \in E}\{t_i\}$.

   b. (1,0) Algoritmo ótimo

   Resposta: Seja g um limite inferior para um problema P. Um algoritmo ótimo A que resolve P é tal que sua complexidade é dada por $f = O(g)$. Dessa forma, o algoritmo A possui complexidade de pior caso $\Omega(g)$ e $O(g)$. Em outras palavras, um algoritmo é ótimo se sua complexidade de pior caso é dada pelo limite inferior para o problema.

2. (2,0) Escreva um algoritmo recursivo para determinar o menor elemento de uma lista. Determine sua complexidade.

Resposta: Seja L uma lista linear representada por um vetor com n posições. Se L possui $n \geq 1$ elementos, então executamos a chamada externa $Menor(L, 1, n)$ de modo a recuperar o menor elemento de L. O Algoritmo 1 determina o menor elemento de L entre as posições f e ℓ de modo recursivo.

Algoritmo 1: $Menor(L, f, \ell)$.

Entrada: Lista linear L composta por n valores.
Saída: O valor do menor elemento de L.

se $f = \ell$ **então**
    retorna $L[f]$;
senão
    menor $\leftarrow Menor(L, f+1, \ell)$;
    se $L[f] <$ menor **então**
        retorna $L[f]$;
    senão
        retorna menor;

Dada uma lista L com n elementos, seja $T(n)$ o numero de passos que o Algoritmo 1 executa. Podemos ver que $T(n) = T(n-1) + \Theta(1)$, onde $T(1) = 1$.

Resolvendo essa equação de recorrência, vemos que a complexidade é igual a $\Theta(n)$.

3. (2,0) Considere os algoritmos ORDENAÇÃO POR SELEÇÃO e ORDENAÇÃO PELO MÉTODO DA BOLHA. Qual dos dois efetua menos TROCAS de elementos quando a lista a ser ordenada encontra-se em ordem inversa de ordenação? (Ex: 10 9 8 7 6 5 4 3 2 1.) Justifique sua resposta.

   Resposta: A ORDENAÇÃO POR SELEÇÃO executa $\lfloor \frac{n}{2} \rfloor$ trocas, uma vez que os elementos trocados ocupam suas posições finais no vetor. Isso pode ser verificado para a primeira troca no exemplo com 10 elementos, onde os números 1 e 10 trocam de lugar. Como 1 é o menor elemento do vetor e 10 o maior, os mesmos não mudam de posição novamente. Podemos aplicar o mesmo raciocínio nos demais passos.

   Por outro lado, a ORDENAÇÃO PELO MÉTODO DA BOLHA executa um numero quadrático de trocas, já que em cada iteração a bolha é trocada com todos os elementos que a precedem até o primeiro elemento menor que a mesma. Ou seja, executa-se $(n-1)+(n-2)+\cdots+1 = \frac{n(n-1)}{2}$ trocas.

   Portanto a ORDENAÇÃO POR SELEÇÃO executa menos trocas que a ORDENAÇÃO PELO MÉTODO DA BOLHA.

4. (2,0) Escreva um algoritmo que leia uma informação x e remova TODAS as ocorrências de nós contendo a informação x de uma lista simplesmente encadeada L. (Obviamente, a lista L pode conter nós com informação repetida.) Discuta a complexidade deste algoritmo.

   Resposta: O Algoritmo 2 representa a remoção de todas as ocorrências de x em uma lista simplesmente encadeada L. Podemos ver que a lista é percorrida exatamente uma vez, onde a cada ocorrência de x removemos o nó com essa informação. Dessa forma o algoritmo executa $\Theta(n)$ passos.

Algoritmo 2: Algoritmo que remove todas as ocorrências de x em uma lista simplesmente encadeada L.

Entrada: Ponteiro PTlista para uma lista simplesmente encadeada L e elemento x a ser removido.

Saída: Lista L sem elementos x.

```
ant ← PTlista;
pt ← ant↑.prox;
enquanto pt ≠ λ faça
    se pt↑.info = x então
        ant↑.prox ← pt↑.prox;
        aux ← pt;
        pt ← pt↑.prox;
        Desaloca(aux);
    senão
        ant ← pt;
        pt ← pt↑.prox;
retorna L;
```

5. Para cada item abaixo, desenhe uma árvore binária de busca T de altura 4 (colocando valores de chaves nos respectivos nós) que atenda às condições descritas:

   a. (1,0) T é uma árvore completa, estritamente binária e com um número mínimo de nós.

   Resposta: Uma árvore binária cheia é aquela em que todos os nós com alguma subárvore vazia estão ou no último ou penúltimo nível. Uma árvore estritamente binária é aquela em que todos os nós possuem exatamente 0 ou exatamente 2 filhos. A Figura 1 representa essa árvore com número mínimo de nós.

   b. (1,0) T é uma árvore estritamente binária, não é cheia e tem um número máximo de nós.

   Resposta: Uma árvore cheia é aquela em que todos os nós com alguma subárvore vazia estão no último nível. A Figura 2 representa a árvore desejada com número máximo de nós.

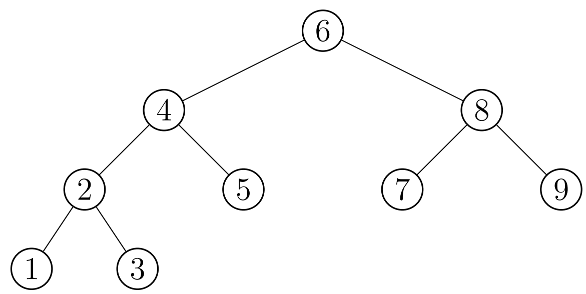

Figura 1: Árvore binária de busca completa estritamente binária com altura 4.

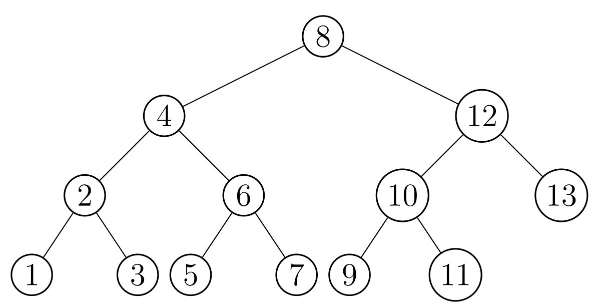

Figura 2: Árvore binária de busca não-cheia e estritamente binária com altura 4.