Estrutura de Dados - 2º. período de 2017

Primeira Avaliação a Distância

1. (1,0) Escreva as seguintes funções em notação O:
   $n^2 + n \log n$; $2n + 2$; $n \log n - n$; $8 \log n + \sqrt{n}$; $n! + n^n$.

   Resposta: $O(n^2)$; $O(n)$; $O(n \log n)$; $O(\sqrt{n})$; $O(n^n)$.

2. (1,5) Para cada item abaixo, responda "certo" ou "errado", justificando:

   (a) Se a complexidade de caso médio de um algoritmo for $\Theta(f)$, então o número de passos que o algoritmo efetua no pior caso é $\Omega(f)$.

   Resposta: Certo. A complexidade de pior caso de qualquer algoritmo deve ser pelo menos igual a complexidade de caso médio do mesmo algoritmo, dado que a complexidade de caso médio leva em conta a complexidade de todas as instâncias do problema.

   (b) Se um limite inferior para um problema P é $n \log n$, então nenhum algoritmo ótimo para P pode ter complexidade de pior caso $\Omega(n)$.

   Resposta: Errado. O limite inferior de um problema P diz respeito à complexidade de pior caso do melhor algoritmo que resolve P. Como o limite inferior para P é $n \log n$, então qualquer algoritmo ótimo para P deve executar no pior caso uma função $f = O(n \log n)$ passos. Como $f = \Omega(n)$, a afirmação é falsa.

   (c) Se a complexidade de pior caso de um algoritmo for $\Theta(f)$, então o número de passos que o algoritmo efetua para uma entrada de tamanho n é no máximo igual a $f(n)$.

   Resposta: Errado. Qualquer algoritmo cuja complexidade de pior caso seja $\Theta(f)$ com $f = n \log n$, por exemplo, poderá executar $2n \log n$ passos no pior caso, que continua uma função $\Theta(n \log n)$.

3. (1,5) Considere a seguinte lista ordenada: 0, 2, 5, 7, 11, 17, 20, 26, 41. Utilizando busca binária, determine:

   (a) Um elemento cuja busca resulte em um número mínimo de comparações.

   Resposta: O elemento 11 está no meio do vetor. Logo o mesmo será o primeiro a ser comparado com o elemento a ser buscado gerando uma busca com sucesso com apenas uma comparação.

   (b) Um elemento **pertencente** à lista cuja busca resulte em um número máximo de comparações. Determine quais comparações foram efetuadas.

   Resposta: Para uma lista com n elementos, o número máximo de comparações que a busca binária efetua para elementos pertencentes à lista é igual a $1 + \lfloor \log n \rfloor$. Logo, no exemplo teremos no máximo 4 comparações. Para o elemento 41 vemos que são feitas comparações com os elementos 11, 20, 26 e 41.

(c) Um elemento **não pertencente** à lista cuja busca resulte em um número máximo de comparações. Determine quais comparações foram efetuadas.

Resposta: Para o elemento 40 teremos 4 comparações, que são as mesmas do caso anterior para o elemento 41: 11, 20, 26 e 41.

4. (2,0) Seja V um vetor ordenado. Elabore um algoritmo que crie uma lista encadeada L, também ordenada, que contenha apenas os elementos de V que ocorrem uma única vez. Calcule a complexidade do seu algoritmo.

Resposta: O Algoritmo 1 efetua tal operação.

Algoritmo 1: *Remove_Repeticao*(V, n).

Entrada: Vetor V ordenado de tamanho $n > 0$.

Saída: Lista encadeada ordenada L com nó cabeça PTlista com elementos únicos de V.

```
ocupar(PTlista);
pont ← PTlista;
para i ← 1, ..., n faça
    j ← i + 1;
    enquanto V[i] = V[j]  e  j ≤ n faça
        j ← j + 1;
    se j = i + 1 então
        ocupar(pt);
        pt↑.info ← V[i];
        pt↑.prox ← λ;
        pont↑.prox ← pt;
        pont ← pont↑.prox;
    senão
        i ← j;
retorna L;
```

5. (1,5) Seja L uma lista encadeada com n elementos, com nó cabeça. Escreva um algoritmo que construa um vetor V a partir de L, de forma que os elementos de V sejam os de L em ordem inversa. Por exemplo, se L contiver os elementos 1 7 3 5 8, nesta ordem, o vetor V deverá conter os elementos 8 5 3 7 1, nesta ordem.

Resposta: O Algoritmo 2 efetua tal operação.

Algoritmo 2: $Inverte\_Lista(L, n)$.

Entrada: Lista encadeada L com $n > 0$ elementos e nó cabeça PTlista.
Saída: Vetor V cujos elementos estão em ordem inversa em relação aos de L.

```
V ← Inicializa_Vetor(n);
pt ← PTlista↑.prox;
para i ← n, ..., 1 faça
    V[i] ← pt↑.info;
    pt ← pt↑.prox;
retorna V;
```

---

6. Considere a lista: 30 24 8 3 79 27. Desenhe **todas** as trocas de elementos e determine o número de trocas efetuadas, utilizando:

   (a) (0,7) Ordenação por seleção

   Resposta:–Trocas pela Ordenação por Seleção: são efetuadas 6 trocas.
   30 24 8 3* 79 27 Vetor inicial
   3 24 8* 30 79 27
   3 8 24* 30 79 27
   3 8 24 30 79 27*
   3 8 24 27 79 30*
   3 8 24 27 30 79*
   3 8 24 27 30 79

   (b) (0,8) Ordenação por bolha

   Resposta:–Trocas pelo Método da Bolha: são efetuadas 8 trocas.
   30* 24 8 3 79 27 Vetor inicial
   30 24* 8 3 79 27
   24 30 8* 3 79 27
   24 8* 30 3 79 27
   8* 24 30 3 79 27
   8 24 30 3* 79 27
   8 24 3* 30 79 27
   8 3* 24 30 79 27
   3* 8 24 30 79 27
   3 8 24 30 79* 27
   3 8 24 30 79 27*
   3 8 24 30 27* 79
   3 8 24 27* 30 79
   3 8 24 27 30 79 Vetor ordenado

7. (1,0) Seja $1, 2, \ldots, n$ uma sequência de elementos que serão inseridos e retirados de uma pilha P uma vez cada. A ordem de inclusão dos elementos na pilha é fixa, sendo igual a $1, 2, \ldots, n$. Por outro lado, a ordem de remoção não o é. A sequência de remoção define uma permutação que representa o movimento dos nós na pilha. Por exemplo, com $n = 3$, a sequência de operações "incluir em P, incluir em P, retirar de P, incluir em P, retirar de P, retirar de P" produzirá a permutação $2, 3, 1$, a partir da entrada $1, 2, 3$. Representando por I, R, respectivamente, as operações de inserção e remoção da pilha, a permutação $2, 3, 1$ pode ser denotada por $IIRIRR$. De um modo geral, uma permutação é chamada *admissível* quando ela puder ser obtida mediante uma sucessão de inclusões e remoções em uma pilha a partir da permutação $1, 2, \ldots, n$. Assim, por exemplo, a permutação $2, 3, 1$ é admissível. Pede-se:

   (a) Determinar a permutação correspondente a $IIRIIRIRRR$, $n = 5$.

   Resposta: $2, 4, 5, 3, 1$.

   (b) A permutação $1, 2, \ldots, n$ é sempre admissível ? Justifique.

   Resposta: Sim, pois tal permutação pode ser obtida pela sequência de operações $I_1 R_1 I_2 R_2 \ldots I_n R_n$.