Algoritmos e Complexidade
LEI (2º ano)

Perguntas do 3° Mini-Teste

Ano Lectivo de 2011/12

1. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   void countBefore(int A[], int N, int p, int *l);
   ```

   Dado um array A, ocupado nas posições 0 a N-1, e um inteiro p, esta função deve retornar o número de elementos de A que antecedem a primeira ocorrência de um valor superior a p. O resultado deve ser devolvido no endereço apontado pela variável l.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.

2. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   void topTwo(int A[], int N, int *l1, int *l2);
   ```

   Dado um array A, ocupado nas posições 0 a N-1, esta função deve retornar os dois elementos de maior valor. Esses valores devem ser devolvidas nos endereços apontados pelas variáveis $l1$ is $l2$, respectivamente. Assuma que $N \geq 2$.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.

3. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   void sumIf(int A[], int N, int p, int *l);
   ```

   Dado um array A, ocupado nas posições 0 a N-1, e um inteiro p, esta função deve retornar a soma dos elementos de A que são superiores a p. O resultado deve ser devolvido no endereço apontado pela variável l.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.

4. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   void under(int A[], int N, int p, int *l);
   ```

   Dado um array A, ocupado nas posições 0 a N-1, e um inteiro p, esta função deve retornar o número de elementos de A que são inferiores a p. O resultado deve ser devolvido no endereço apontado pela variável l.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.

5. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   typedef struct node {
      int a;
      struct node *next;
   } Node, *Lista;

   void countWhile(Lista list, int p, int *l);
   ```

   Dada uma lista ligada *list*, e um inteiro p, esta função deve retornar o número de elementos da lista que antecedem a primeira ocorrência de um valor superior ou igual a p. O resultado deve ser devolvido no endereço apontado pela variável l.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.

6. Responda às seguintes questões.

   (a) Implemente **de forma recursiva** a seguinte função em C.

   ```
   typedef struct node {
      int a;
      struct node *next;
   } Node, *Lista;

   void countAfter(Lista list, int p, int *l);
   ```

   Dada uma lista ligada *list*, e um inteiro p, esta função deve retornar o número de elementos da lista que sucedem a posição p. Considere que o primeiro nó da lista está na posição 1. O resultado deve ser devolvido no endereço apontado pela variável l.

   (b) Escreva uma equação de recorrência que descreva o tempo de execução $T(N)$ do algoritmo que implementou. No caso de identificar claramente um melhor caso e um pior caso, deverá escrever uma equação de recorrência para cada um deles.

   (c) Desenhe uma árvore para descrever graficamente o comportamento de cada uma das equação de recorrência que encontrou na alínea anterior.

   (d) Descreva o comportamento do algoritmo utilizando a notação assimptótica Θ, O e Ω.