UNIVERSIDADE FEDERAL DE MATO GROSSO
CENTRO UNIVERSITÁRIO DO ARAGUAIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Disciplina: Projeto e análise de algoritmos
Docente: Robson Lopes

EXERCÍCIOS

1. Vamos supor que estamos comparando implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho n, a ordenação por inserção é executadas em $8n^2$ etapas, enquanto a ordenação por intercalação é executadas em $64n \lg n$ etapas. Para que valores de n a ordenação por inserção supera a ordenação por intercalação?
2. Descreva um algoritmo que, dados n inteiros no intervalo de 0 a k, realize o pré-processamento de sua entrada e depois responda a qualquer consulta sobre quantos dos n inteiros reacem em um intervalo [a..b]. Seu algoritmo deve utilizar o tempo de pré-processamento $\Theta(n+k)$.
3. Qual é o menor valor de n tal que um algoritmo cujo tempo de execução é $100n^2$ funciona mais rápido que um algoritmo cujo tempo de execução é 2^n na mesma máquina?
4. Considere a ordenação d n números armazenados no arranjo A, localizando primeiro o menor elemento de A e permutando esse elemento com o elemento contido em A[1]. Em seguida, encontre o segundo menor elemento de A e o troque pelo elemento A[2]. Continue desse maneira para os primeiro n-1 elementos de A. Escreva o pseudocódigo para esse algoritmo, conhecido como ordenação por seleção. Que loop invariante esse algoritmo mantém? Por que ele só precisa ser executados para os primeiros n-1 elementos, e não para todos os n elementos? Forneça os tempos de execução do melhor caso e do pior caso de ordenação por seleção em notação Θ.
5. Cada um dos algoritmos abaixo recebe um inteiro positivo e devolve outro inteiro positivo. Os dois algoritmos são equivalentes: devolvem o mesmo número se receberem um mesmo n.

```
Soma-Quadrados-A (n)
  x ← 0
  para j ← 1 até n faça
    x ← x + j · j
  devolva x

Soma-Quadrados-B (n)
  x ← n · (n+1) · (2n+1)
  x ← x/6
  devolva x
```

Digamos que uma *operação aritmética* é uma adição, subtração, multiplicação ou divisão. Quantas operações aritméticas o primeiro algoritmo faz? Quantas operações aritméticas o segundo

6. Escreva, com recursividade, um algoritmo que encontre o maior elemento de um vetor. Mostre a equação de recorrência. Analise a sua complexidade para o pior caso (último elemento é o maior) e para o melhor caso (primeiro elemento é o maior). Os resultados são diferentes? Por quê?
7. A mediana de uma lista ordenada $L = x_1, x_2, ..., x_n$ de n inteiros é um elemento com índice $\lceil n/2 \rceil$ e pode ser encontrada em tempo O(1). Dadas duas listas ordenadas L1 e L2 de n inteiros cada, a mediana de todos os 2n elementos pode ser encontrada em tempo O(n) através do *merging* das duas listas e do retorno do elemento de índice n. Escreva o algoritmo e determine a sua equação de recorrência. Analise a sua complexidade e prove, através do Método da Substituição, que a sua complexidade é O(n).
8. Explique por que a declaração "O tempo de execução no algoritmo A é no mínimo $O(n^2)$" é isenta de significado.
9. Prove se as seguintes igualdades são corretas ou incorretas aplicando **a definição de notação assintótica** para comparar as ordens de crescimento das funções. Você deve obrigatoriamente apresentar todos os seus cálculos.

- $30n^2 .n + n\lg n^{10} + 100 = \Omega(n\lg n)$
- $5/n^3 = O(1/n^2)$
- $3n\log n + \log n^{10} = \Theta(n\log n)$
- $200^{100} = O(\lg n)$
- $16n^3/(\log n+1) = \Theta(n^3)$
- $n^2\lg n = \Theta(n\lg n)$

10. Considere os algoritmos dados a seguir e para cada um deles responda as seguintes questões:
    - O que o algoritmo calcula?
    - O número de vezes que a operação básica é realizada depende somente do tamanho da entrada? Explique
    - Quantas vezes a operação básica é executada? (Obs. Estabeleça um somatório ou uma relação de recorrência para indicar o número de vezes que a operação básica é executada e resolva este somatório ou relação de recorrência).

```
functionA(n)
    x ← 0
    i  ← n
    enquanto i > 0 faça
        para j ← 1 até i faça
            x ← x + 1
        i ← piso(i/2)
```

```
funB(n)
    // Entrada: um inteiro não
negativo
    if n = 0
        return 1
    else
      return funB(n-1) + funB(n-1) +
```

funB(n-1)

11. Demonstre a prova do teorema mestre.
12. Use uma árvore de recursão para determinar um bom lite superior assintótico na recorrência T(n) = 3T(n/2)+n
13. Demonstre que a solução para a recorrência T(n) = T(n/3) + T(2n/3) +cn, onde c é uma constante, é Ω(n lg n), apelando para uma árvore de recursão.
14. Trace uma arvore de recursão para T(n) = 4T(n/2) + cn, onde c é uma constate, e forneça um limite assintótica sobre sua solução. Verifique o limite pelo método da substituição.
15. Quais são os números mínimo e máximo de elementos em um heap de altura *h*?
16. Conhecendo como o procedimento MAX-HEAPIFY funciona , escreva o pseudocódigo para o procedimento MIN-HEAPIFY(A,i), que executa a manipulação correspondente sobre um heap minimo. Como o tempo de execução de MIN-HEAPIFY se compara ao de MAX-HEAPIFY.
17. Por que queremos que o indice de loop i na linha 2 BUILD-MAX-HEAP diminua de piso(comprimento[A]/2) até1, em vez de aumentar de 1 até piso(comprimento[A]/2)?
18. Qual é o tempo de execução de heapsort sobre um arranjo A de comprimento n que está ordenado em ordem crescente? E em ordem decrescente?
19. Explique por que o QUICKSORT, em um particionamento balanceado, ele apresenta um custo total de O(n lg n).
20. Mostre que o tempo de execução de QUICKSORT é $\Theta(n^2)$ quando o arranjo A contém elementos distintos e está ordenado em ordem decrescente.
21. Suponha que o cabeçalho do loop for na linha 9 do procedimento COUNTING-SORT seja reescrito

    9 for j ← 1 to comprimento[A]

    mostre que o algoritmo ainda funciona corretamente. O algoritmo modificado é estável?