**FACULDADE CAMPO LIMPO PAULISTA**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**Projeto e Análise de Algoritmos II**
Lista de Exercícios 1

Prof. Osvaldo.

1. Utilize os algoritmos de ordenação estudados em sala para ordenar o conjunto {4, 0, 3, 1, -1, 6, 5, 2}. Ilustre a sua resposta desenhando o vetor e o movimento dos elemntos.
    a) Ordenação por inserção (*Insertion sort*);
    b) Ordenação por seleção (*Selection sort*);
    c) Ordenação pelo método da bolha (*Bubble sort*);
    d) Ordenação por intercalação (*Merge sort*);
    e) Ordenação rápida (*Quick sort*).

2. Desenvolva exemplos de seqüências de números para os quais a complexidade de tempo do algoritmo `Quicksort` é $\Omega(n^2)$ considerando:
    a) o pivô a ser escolhido é o primeiro elemento;
    b) o pivô a ser escolhido é o último elemento;
    c) o pivô a ser escolhido é o elemento do meio da seqüência.

3. As complexidades de pior caso, melhor caso e caso médio do algoritmo de ordenação pelo método da bolha, desenvolvido em sala de aula, são todas iguais a $O(n^2)$. Desenvolva uma nova versão recursiva deste mesmo algoritmo de tal forma que a complexidade de melhor caso seja igual a $O(n)$. Qual a complexidade de caso médio da nova versão. [Sugestão: se em uma passada para verificação de trocas ("um borbulhamento") não houver troca, então o vetor pode ser considerado ordenado.

4. Desenvolva exemplos de seqüências de números para os quais o algoritmo *Bubble Sort* desenvolvido em questão anterior executa, para um vetor de $n$ elementos:
    a) $\Omega(n^2)$ comparações;
    b) *$O(n)$ comparações.*

5. O aluno João da Silva propôs um algoritmo para ordenação que utiliza como critério de redução a busca do menor e do maior elemento do vetor seguido da colocação destes elementos, respectivamente, na primeira e última posições. Assim, o tamanho inicial da solução do problema é reduzido em 2. Desenvolva este algoritmo e calcule a sua complexidade.

6. A aluna Maria das Graças esboçou a idéia de outro algoritmo para ordenação que tem como critério de redução a divisão do vetor em duas partes onde uma delas é

indutivamente ordenada e os elementos da outra são, após isto, inseridos na parte ordenada. Desenvolva este algoritmo e calcule a sua complexidade.

7. Qual entre os algoritmos de José da Silva e de Maria das Graças é o mais eficiente (isto é, tem menor complexidade de pior caso)? Você seria capaz de formular um outro algoritmo de ordenação, diferente daqueles desenvolvidos em sala de aula, mais eficiente que os algoritmos de José da Silva e de Maria das Graças?

8. O algoritmo *Merge Sort* divide o conjunto a ser ordenado em duas partes, ordena indutivamente as duas partes e, por fim, intercala estas partes para obter a solução final. Esta estratégia de "divisão" se mostrou mais eficiente do que as reduções de um em um dos algoritmos *Insertion Sort* e *Selection Sort*. O professor Ganancioso está investigando a divisão do conjunto a ser ordenado em 4 partes em vez de duas, influenciado pela promissora estratégia do *Merge Sort*. O professor Ganancioso será bem sucedido em sua proposta de diminuir a complexidade de tempo de *Merge Sort*?

9. Escreva um algoritmo para intercalar *(merge*) dois vetores ordenados. Qual a complexidade de tempo deste algoritmo?

10. Descreva como se pode ordenar um conjunto de inteiros no intervalo de 1 a $n^2$ em um tempo igual a $O(n)$.

11. Um conhecido jogo consiste em fazer um jogador "advinhar" um certo número pensado por uma pessoa sabendo-se que este número situa-se no intervalo 1 a $n$. Estabeleça uma boa estratégia para "advinhar" o número pensado o mais rapidamente possível. Baseado em sua estratégia, qual é a menor quantidade de tentativas para "advinhar", no pior caso, o número pensado?

12. Resolva a seguinte variação do problema da pesquisa binária que trata de encontrar o menor elemento em uma seqüência ordena circularmente. Dado uma seqüência ordenada de elementos $x_i < x_{i+1} ... < x_n < x_1 < x_2 ... < x_{i-1}$, achar a posição do menor elemento. Por simplicidade, assuma que esta posição é única. Qual a complexidade do seu algoritmo?

13. Considere novamente outra variação do problema da pesquisa binária, escreva um algoritmo para resolvê-lo e calcule a sua complexidade. Desta vez a busca se dará em uma seqüência de tamanho desconhecido conforme descrevemos. Dado uma seqüência teoricamente infinita $x_1 < x_2 < x_3 < x_4 < ...$ e um elemento $z$, achar o índice $i$ tal que $x_i = z$.

14. O problema deste exercício é conhecido como **método da bisseção** ou **método de Bolzano** para cálculo de raízes de equações e também pode ser considerado uma pesquisa binária. Este método é freqüentemente estudado em disciplinas como Cálculo Numérico ou Análise Numérica. São dadas uma função real $f(x)$ e dois números reais $a$ e $b$ tais que $a < b$ e $f(a) \cdot f(b) < 0$. O problema consiste em encontrar um valor de $x$ para o qual $f(x) = 0$ no intervalo $[a .. b]$.

15. Seja A uma coleção de elementos. Desenvolva um algoritmo eficiente para converter A em um conjunto, ou seja, remover todas as repetições em A. Qual a complexidade do seu algoritmo?

16. Considere o seguinte problema. É dado um conjunto S contendo $n$ números reais. Desenvolva um algoritmo $O(n)$ para achar um número que não esteja no conjunto.

17. O problema agora tem como entrada um conjunto S contendo $n$ números reais e um número real $x$.

    a) Desenvolva um algoritmo para determinar se existem dois elementos em S tais que a soma seja exatamente igual a $x$. O algoritmo deve ter complexidade de tempo $O(n \log n)$.
    b) Suponha agora que os elementos de S estão ordenados. Desenvolva um algoritmo para resolver este problema em $O(n)$.

18. Desenvolva um algoritmo para determinar se dois conjuntos, $S_1$ de $m$ elementos e $S_2$ de $n$ elementos, são disjuntos. Dois conjuntos $S_1$ e $S_2$ são disjuntos quando $S_1 \cap S_2 = \varnothing$. Qual é a complexidade do seu algoritmo como função de $m$ e $n$?

19. Desenvolva um algoritmo para computar a união de dois conjuntos de $n$ elementos. Os conjuntos são dados como vetores de elementos. A saída deve ser um vetor de elementos distintos, isto é, nenhum elemento deve aparecer mais do que uma vez. A complexidade de tempo no pior caso do seu algoritmo deve ser $O(n \log n)$.

20. Prove que $\Omega(n)$ é a cota inferior para a complexidade do problema de achar o maior elemento de um conjunto de tamanho igual a $n$.