Universidade Federal de Uberlândia

Faculdade de Engenharia Elétrica
Programação Procedimental
Prof. Felipe A. Louza

Lista 13

Ordenação (parte 1)

Questão 1

Escreva uma versão recursiva para o algoritmo de ordenação por seleção (*Selection Sort*).

Questão 2

Implemente uma variação do algoritmo *Selection Sort* que percorre o vetor da última para a primeira posição, isto é, $i = n-1, n-2, \ldots, 1$, buscando o valor de `v[i]` pelo valor máximo em `v[0..i-1]`.

Questão 3

Descreva e analise uma instância de pior caso para o algoritmo *Selection Sort*, ou seja, qual vetor `v[0..n-1]` leva o algoritmo a executar o maior número possível de comparações.

Questão 4

Escreva uma versão recursiva para o algoritmo de ordenação por inserção (*Insertion Sort*).

Questão 5

Implemente uma variação do algoritmo *Insertion Sort* que quando o elemento já está na sua posição correta não percorre mais o vetor testando se `v[j] < v[j-1]`.

Questão 6

Descreva e analise uma instância de pior caso para o algoritmo *Insertion Sort*.

Questão 7

Escreva uma versão recursiva do método de ordenação por bolha (*Bubble Sort*).

Questão 8

Implemente uma variação do algoritmo *Bubble Sort* que pára se em uma das iterações nenhuma troca é realizada (logo a lista já está ordenada).

Questão 9

Qualquer algoritmo de ordenação pode ser utilizado para ordenar listas ligadas ao invés de vetores. Nesse caso, não existe a necessidade de mover os elementos de posição quando um novo é inserido ao subconjunto ordenado, pois isso pode ser feito simplesmente ajustando ponteiros.

Considere o seguinte registro utilizado na lista ligada:

```
1 typedef struct no{
2   int valor;
3   struct no *next;
4   struct no *prev;
5 } Lista;
```

Implemente uma versão dos algoritmos abaixo para listas ligadas.

(a) *Selection Sort*

(b) *Insertion Sort*

(c) *Bubble Sort*

Obs.: Não utilize `free()` ou `malloc()` e não copie o campo `valor` de um nó da lista para outro.

Questão 10

Faça a análise teórica de pior e melhor caso de cada algoritmo implementado no exercício anterior.