Estruturas de Dados II - 2012-2

Lista 2

Algoritmos de Ordenação e Complexidade de Algoritmos

1) Crie um algoritmo $O(N^2)$ para ordenar apenas os elementos pares de um vetor de inteiros de tamanho N. O algoritmo deve manter os inteiros pares inalterados na mesma posição original. Note que sua solução será considerada incorreta se for $O(N*\log_N)$ ou qualquer complexidade diferente de $O(N^2)$.

2) Uma fila de prioridades é uma estrutura de dados em que elementos são inseridos na fila de tal forma que o próximo a ser retirado é sempre aquele que tem a maior prioridade. Você está implementando uma fila de prioridades com números reais e quanto maior o número, maior sua prioridade. Crie as duas primitivas básicas da fila de prioridade abaixo de tal forma que a complexidade de cada uma delas seja O(logN). Importante: soluções O(N) serão consideradas incorretas.

```
insert(Fila f, Real r); // O(logN)
Real remove(Fila f); // O(logN)
```

3) Exemplifique o funcionamento do algoritmo de countsort ordenando **passo-a-passo** o vetor a seguir. Não precisa demonstrar todos os passos, bastando apenas exemplificar os dois ou três primeiros passos de cada etapa do algoritmo e depois o estado final dessa etapa.

1	5	2	3	4	0	6	2	4	2	6	3	4	0	5	2

4) Há alguns bugs no código a seguir, escrito em C++, que deveria ordenar de forma não-decrescente um vetor de inteiros. Corrija-o.

```
#include <algorithm>
void recursiveSort(int *V, int N) {
   int maior = vmax(V, N);
   int menor = vmin(V, N);
   std::swap(V[0], V[maior]); // essa função troca V[0] por V[maior]
   std::swap(V[N-1], V[menor]); // troca V[N-1] por V[menor]
   recursiveSort(V+1, N-2);
}
```