Prova de Introdução a Ciência da Computação II
Prof. Eduardo C. Xavier

Questão 1 (2 pontos) – Suponha que você tenha que ordenar um vetor que está quase todo ordenado, com apenas 1% dos números em posições erradas. Você pode usar o QuickSort ou o InsertionSort. Qual algoritmo você usaria? Justifique sua escolha em no máximo 6 linhas.

Usaria o InsertionSort pois com um vetor ordenado (quase ordenado) temos o melhor caso do InsertionSort que é O(n). Um vetor ordenado é o pior caso do QuickSort $O(n^2)$.

Questão 2 (2,5 pontos) – Faça a implementação da interface IPilha abaixo usando nós da classe dada:

```
public interface IPilha<T>{
  //insere objeto no topo da pilha
  public void insere(T o);
  //remove objeto do topo da pilha e o retorna
  public T remove();
}
```

```
class No<T>{
    No<T> prox=null;
    T elemento=null
}
```

```
public class Pilha<T> implements IPilha<T>{
        No<T> topo=null;

        public void insere(T o){
                No<T> novo = new No<T>();
                novo.elemento = o;
                novo.prox = topo;
                topo = novo;
         }

         public T remove(){
                 if(topo != null){ //se tiver algum elemento
                        T aux = topo.elemento;
                        topo  = topo.prox;
                        return aux;
                 }
                 else return null;
         }

}
```

Questão 3 (2,5 pontos) – Para comparar se duas Strings são iguais, deve-se comparar cada um dos caracteres das duas Strings. Você está projetando uma aplicação que armazena objetos que contem como

um de seus campos Strings. A comparação de igualdade de Strings nesta aplicação é muito frequente. Explique em no máximo 6 linhas como utilizar uma função de hash para acelerar o processo de comparação de Strings.

Podemos gerar um número para cada String utilizando uma função de hash que recebe como parâmetro uma String e gera um número. Este número é salvo junto com a String no objeto. Quando precisarmos comparar duas Strings, comparamos primeiro os números. Se forem diferentes, as Strings são diferentes. Se forem iguais devemos comparar as Strings caracter por caracter como antes para saber se são realmente iguais. A comparação de números é bem mais rápida pois é feita uma única vez.

Questão 4 (3 pontos) – Considere uma eleição onde cada candidato possui um número inteiro único que o identifica. Temos um vetor **v** de inteiros tal que **v[i]** possui o número do candidato escolhido pelo i-ésimo eleitor. Explique em no máximo 5 linhas como descobrir o candidato mais votado com tempo O(n log n) onde n é o tamanho do vetor. Faça uma implementação em Java do seu algoritmo e explique o porquê da complexidade (você pode usar, sem implementar, qualquer um dos algoritmos e estrutura de dados vistos em aula).

Primeiramente ordenamos o vetor com o MergeSort. Com isso os votos de cada candidato estão agrupados no vetor. Basta fazer um laço que conta votos e armazena o mais votado a medida que percorremos o vetor.

```
int maisVotado(int[] v){
  int mais_votado = 0, max_votos = 0;
  int votos=0,  candidato_anterior=v[0], i;
  for(i=0;i<v.length;i++){
     if(v[i] == candidato_anterior)
        votos++;
     else{  //se mudar de candidato checamos se tem mais votos que máximo
       if(votos > max_votos){
         max_votos = votos;
         mais_votado = v[i-1];
        }
       //reinicializamos variaveis para comecar a contar votos   do novo candidato
       votos =1;
       candidato_anterior = v[i];
     }
  }

  if(votos>max_votos){//teste para o ultimo candidato
    mais_votado=v[i-1];
    max_votos = votos;
  }
  return mais_votado;
}
```