3ª Prova de Algoritmos e Estruturas de Dados I - Curso: Engenharia de Computação

Nome do aluno: _____ **Data:** 18/09/2013

Resolva as questões abaixo identificando-as claramente na folha de respostas. Mantenha o silêncio na sala (mantendo desligado aparelhos eletrônicos). A interpretação das questões faz parte da prova. **Horário limite:** 15:40

(15 pontos) 1) Sobre algoritmos de ordenação:

a) É possível afirmar que o algoritmo QuickSort é superior aos algoritmos elementares (ex: BubbleSort, InsertionSort, SelectionSort) em todos os casos? Porquê?

b) Explique alguma estratégia para minimizar o problema apresentado no item (a).

c) A respeito da estabilidade de um algoritmo de ordenação, defina o conceito de estabilidade e cite 2 algoritmos estáveis e 2 não estáveis.

(15 pontos) 2) Considere as técnicas de busca sequencial, busca binária e busca baseada em hashing:

a) Descreva as vantagens e desvantagens de cada uma dessas técnicas, indicando em que situações você usaria cada uma delas.

b) Qual é a eficiência de utilização da memória (relação entre o espaço necessário para dados e o espaço total necessário) para cada método?

(20 pontos) 3) Diga-me a Frequência

Dada uma linha de texto, você deve encontrar as frequências de cada um dos caracteres presentes nela. As linhas fornecidadas não conterão nenhum dos primeiros 32 ou dos últimos 128 caracteres da tabela ASCII. É claro que não estamos levando em conta o caracter de fim de linha.

Entrada

A entrada contém vários casos de teste. Cada caso de teste é composto por uma única linha de texto com até 1000 caracteres.

Caso seja necessário ordenar, considere que a função 'Ordena()' está pronta e recebe como argumento um vetor e os indicies de inicio e fim do vetor. Mas a função 'criterio()' você deve implementar. A função critério() recebe dois elementos e retorna qual deles é o maior.

Saída

Imprima o valor ASCII de todos os caracteres presentes e a sua frequência de acordo com o formato abaixo. Uma linha em branco deverá separar 2 conjuntos de saída. Imprima os caracteres ASCII em ordem ascendente de frequência. Se dois caracteres estiverem presentes com a mesma quantidade de frequência, imprima primeiro o caracter que tem valor ASCII maior. A entrada é terminada por final de arquivo (EOF).

Entrada:	**Saída:**

`AAABBC`
`122333`
	`67 1`
	`66 2`
	`65 3`
	
	`49 1`
	`50 2`
	`51 3`

(50 pontos) 4) Um aluno da UNIFESSPA (Universidade Federal do Sul e Sudeste do Pará) Câmpus I - Marabá, está implementando um novo e revolucionário sistema de consulta de dados. Esse sistema será utilizado para fazer consultas no sistema *Rettiwt* onde as pessoas poderão colocar mensagens sobre variados temas e os usuários poderão fazer consultas sobre as mensagens.

O nosso nobre colega está tendo alguns problemas na implementação das consultas de mensagens. Para facilitar a abstração do problema foi nos passado uma versão simplificada do problema, onde temos:

Dado um vetor de inteiros, sua tarefa é encontrar a k-ésima ocorrência (da esquerda para a direita) de um inteiro v no vetor. Para tornar o problema mais difícil (e mais interessante!), você deve responder a m consultas deste tipo.

Entrada

Há vários casos de teste. A primeira linha de cada caso de teste contém dois inteiros n e m ($1 \leq n, m \leq 100.000$), o número de elementos no vetor e o número de consultas a serem respondidas, respectivamente. A próxima linha contém n inteiros positivos não maiores que 1.000.000, que descrevem o vetor. As próximas m linhas contém dois inteiros k e v cada ($1 \leq k \leq n$, $1 \leq v \leq 1.000.000$), descrevendo as consultas.

O arquivo de entrada termina com fim-de-arquivo (EOF). O tamanho do arquivo de entrada não excede 5 Mb.

Lembre que a consulta deve ser eficiente!

Saída

Para cada consulta, imprima o índice do vetor (1-indexado) da ocorrência solicitada. Se tal ocorrência não existe, imprima 0 ao invés.

Entrada:	**Saída:**
8 4 1 3 2 2 4 3 2 1 1 3 2 4 3 2 4 2	2 0 7 0