Universidade Federal de Goiás – UFG
Instituto de Informática – INF
Turmas: INF0286/INF0447

Algoritmos e Estruturas de Dados 1 – 2024/2

Lista de Exercícios – Algoritmos de Busca e Ordenação Interna

Prof. Wanderley de Souza Alencar
adaptado por Prof. Raphael Guedes

Sumário

1 Definindo conceitos 1

 (+)

Defina, usando as suas palavras, o problema de ordenação.

2 Definindo conceitos 2

 (+)

Defina, usando as suas palavras, o problema de encontrar o menor valor em um vetor.

3 Lápis e Papel

 (++)

Para cada sequência de números abaixo, faça um teste de mesa com os seguintes métodos de ordenação: *Bubble Sort*, *Insertion Sort*, *Merge Sort* e *Quick Sort*.

Mostre o número de comparações e trocas realizadas por cada método:

A. 21, 19, 17, 9, 5, 1.

B. 2, 4, 6, 8, 10, 12, 11, 9, 7, 5, 3, 1.

C. 18, 29, 17, 29, 23, 21, 23, 8, 14, 6.

4 Busca Binária Aproximada

 (++)

Considere um vetor de inteiros cujos valores estão armazenados em ordem crescente. Usando como base o algoritmo de **busca binária**, escreva uma função que, dado o vetor e um valor inteiro x, retorne o elemento do vetor que possua o valor mais próximo de x. Caso x seja equidistante de dois elementos do vetor, sua função deve retornar o valor do menor deles $min(x_1, x_2)$. Por exemplo, considerando o vetor $\{3, 7, 10, 14, 16\}$, sua função deve retornar os valores indicados a seguir para os diferentes casos listados:

Valor de x	Valor retornado
11	10
5	3
14	14
13	14

5 Ordenando a Bicharada

 (+)

O zoológico de Magravanópoolis está passando por uma grande reforma! Para facilitar o manejo dos animais e a vida dos funcionários, a administração decidiu organizar a lista de animais em ordem alfabética. Para isso, contrataram você, um(a) programador(a) experiente, para desenvolver um sistema de ordenação.

Modifique o algoritmo *bubble sort* para que ele possa ordenar um conjunto $n \in \mathbb{N}^*$ de animais em um zoológico por seu nome. Cada animal é representado por uma estrutura contendo seu código identificador numérico, nome, classe (mamífero, ave, réptil, etc).

6 Separando Números Pares de Ímpares

 (++)

Tales, um menino muito levado, pegou na escola uma caixa repleta de números naturais impressos em cartelas de EVA (*Espuma Vinílica Acetinada*) e derrubo-os sobre o chão da sala de aula.
Por estranho que pareça, ao cairem os números formaram uma *fila indiana* de tal maneira que ficaram com seus valores distribuídos aleatoriamente nesta fila.
Sabe-se que na caixa havia $n \in \mathbb{N}^*$ números, com $(1 < n \leq 100)$, mas seus valores são desconhecidos.
Sua tarefa é conceber um programa $\mathbb{C}$ que seja capaz de ordenar esta fila, segundo as seguintes regras:

- primeiro devem vir todos os números pares, em ordem crescente;
- depois devem vir os números ímpares, em ordem decrescente.

Entrada

A primeira linha de entrada contém o número n, quantidade de números existente na caixa que Tales derrubou.
A segunda linha contém os n números naturais, na ordem em que formaram a fila indiana, sempre separados por um único espaço em branco entre eles.

Saída

A saída deverá ter duas linhas. Na primeira são apresentados os números pares e na segunda os números ímpares, sempre separados por um único espaço em branco entre eles, conforme a ordem definida anteriormente.

Exemplos

Entrada	Saída
10 4 32 34 543 3456 654 567 87 6789 98	4 32 34 98 654 3456 6789 567 543 87

Observação: Note que se, como exceção, a saída poderá ter uma única linha, se os números inicialmente fornecidos forem todos pares ou todos ímpares.

Entrada	Saída
7 2 5 6 51 512 913 375	2 6 512 913 375 51 5

Entrada	Saída
8 6 2 8 12 202 304 18 10	2 6 8 10 12 18 202 304

Entrada	Saída
8 1 3 5 7 11 23 45 81	81 45 23 11 7 5 3

Entrada	Saída
5 10 8 6 4 2	2 4 6 8 10

7 Placar – Quem vai ser reprovado?

 (+++)

O professor Charles Francis Xavier, *Professor X*, aplicou para seus(suas) alunos(as) uma *Lista de Exercícios* contendo um conjunto de 10 (dez) "*problemas*" e concedeu o prazo de um mês para que eles(as) os resolvessem.
No final do mês os(as) alunos(as) enviaram, para o *Profesor X*, o número de problemas resolvidos corretamente.
A promessa do professor era reprovar, sumariamente, o(a) último(a) colocado(a) desta competição, onde os(as) alunos(as) seriam ordenados(as) conforme o número de problemas resolvidos, com empates resolvidos de acordo com a ordem alfabética dos nomes[1].
Este padrão fez com que alunos(as) com nomes iniciados nas últimas letras do alfabeto se esforçassem muito nas tarefas, e não compartilhassem suas soluções com colegas, especialmente aqueles(as) cujos nomes começassem com letras anteriores às deles(as).
Você é o(a) "*monitor(a)*" do *Professor X* e sua tarefa,é escrever um programa $\mathbb{C}$ que leia os resultados dos(as) alunos(as) e imprima a classificação final, marcando com a *hashtag* `#reprovado(a)` aquele(a) estudante que deverá ser reprovado(a).

Entrada

A primeira linha de cada contém um número natural n, $1 \leq n \leq 100$, que indica a quantidade estudantes na competiação.
Cada uma das n linhas seguintes contém o nome do(a) aluno(a) – sem espaço entre as suas palavras formadoras de seu nome – e o número de problemas resolvidos por ele(a). O número de problemas resolvidos está, obviamente, entre 0 e 10, inclusive extremos.

Saída

O programa de computador criado por você deverá imprimir o nome do(a) aluno(a) e o número de exercícios realizados, ordenadamente. Além disso, o(a) último(a) colocado deverá ser seguido pela *hashtag* `#reprovado(a)`.

Exemplos

[1] Suponha, para simplificação, que não há homônimos na turma e que todos os nomes dos(as) estudantes são escritos utilizando uma única palavra, sem espaços em branco entre elas, e que terá, no máximo, 20 símbolos. Por exemplo: AnaLuiza, Guilherme, Alice, LuizFelipe, Marcelo, PedroAugusto, etc.

Entrada	Saída
10 joaozinho 9 marcela 8 marcos 9 frodo 7 jailson 10 jolei 8 andre 10 maria 8 beatriz 10 ana 9	andre 10 beatriz 10 jailson 10 ana 9 joaozinho 9 marcos 9 jolei 8 marcela 8 maria 8 frodo 7 #reprovado(a)

Entrada	Saída
3 pedro 3 jose 3 arnaldo 3	arnaldo 3 jose 3 pedro 3 #reprovado(a)

Entrada	Saída
5 amanda 10 ananda 10 abadia 10 andreia 10 andubio 10	abadia 10 amanda 10 ananda 10 andreia 10 andubio 10 #reprovado(a)

Entrada	Saída
4 cardonha 9 infelizreprovado 3 marcel 9 infelizaprovado 3	cardonha 9 marcel 9 infelizaprovado 3 infelizreprovado 3 #reprovado(a)

8 Insertion versus Selection

 (+)

Escreva um programa ℂ que, a partir de um vetor de números naturais fornecido como entrada, calcule a diferença entre o número de trocas realizadas pelos algoritmos `insertionSort` e `selectionSort`, nesta ordem.
Cada movimentação efetiva de um número no vetor deve ser contabilizada. Os algoritmos devem ser implementados de maneira a realizar o menor número de trocas possível.

Entrada

A primeira entrada é um número natural n, $1 \leq n \leq 1000$, que representa o tamanho do vetor de entrada. A próxima linha contém os elementos do vetor, sempre fornecidos da primeira posição até a última, e separados por um único espaço em branco entre si.

Saída

A saída consiste de uma única linha que contém a diferença entre o número de trocas realizadas pelo `insertionSort` e pelo `selectionSort`, nesta ordem.

Exemplos

Entrada	Saída
20 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	19

Entrada	Saída
20 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33	199

Entrada	Saída
10 8 6 4 3 2 1 7 9 5 10	23