Dredd - Juiz Online

Principal | Perfil | Minhas Provas | Sair

Minutos Restantes: 10770

Usuário: Wesley de Jesus Santos

Notas:
Q1: 100
Q2: 100
Q3: 77
Q4: 100
Q5: 100
Q6: 100
Q7: 100
Q8: ?
Q9: ?
Q10: ?
Total: 68

IAlg EAD - Exercícios de Ordenação - 2019/1 - 27/05 à 09/06

Prova Aberta Até: 09/06/2019 23:59:59

Número Máximo de Tentativas: 6

Atenuação da Nota por Tentativa: 0%

Instruções para a prova: A prova é individual. Desligue seu celular. Não converse com os colegas. Não fique olhando para a tela dos colegas.

Contabilização de frequência - Ordenacao

Os exercícios dessa semana que serão utilizados para a contabilização de frequência são: 4 e 5.

Para que a frequência seja contabilizada o aluno deve, ao menos, ter tentado resolver o exercício, não sendo necessário que tenha obtido nota máxima. **Não** serão aceitos para contabilização de frequência tentativas "vazias" - envio de código que compila, mas não tenta resolver o problema. Além disso, **poderão ser utilizadas ferramentas para detecção de plágio e, em caso de plágio, todos os envolvidos ficarão com falta**.

Questão 1: Ordenação - Selection Sort

Uma forma de realizar a ordenação é pesquisando o menor valor presente no vetor e colocá-lo em seu devido lugar. Diante dessa ideia elabore um algoritmo que implemente o método **Selection Sort** e que tenha como entrada o tamanho n do vetor e os números para preenchê-lo.

Entrada

5
6

2
7
1
8

Saida

1
2
6
7
8

Peso: 1

Última tentativa realizada em: 31/05/2019 15:25:46

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 2: Ordenação - Selection Sort passo a passo

Implemente a seguinte variação do algoritmo de ordenação *selection sort*: Procurar o maior valor de um vetor e trocá-lo com o que estiver na última posição do vetor. Em seguida, procurar o segundo maior valor e trocá-lo com o que estiver a penúltima posição do vetor. Assim sucessivamente, até que o vetor esteja ordenado.

Não existe teste para verificar se um valor está trocado com ele mesmo. Veja no exemplo que nesta situação, o vetor é mesmo antes e depois da troca.

Entradas:

1. Tamanho do vetor que será ordenado.
2. Vários número inteiros que serão ordenados.

Saídas:

1. Os elementos do vetor a cada troca de valor.

```
Exemplo de entrada:
5
4 1 7 2 3

Exemplo de saída:
4 1 3 2 7
2 1 3 4 7
2 1 3 4 7
1 2 3 4 7
```

Peso: 1

Última tentativa realizada em: 02/06/2019 11:51:05

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 3: Ordenação - Ordenação nas Metades do Vetor

Implemente um programa que receba vários números inteiros, armazene os números num vetor e ordene a primeira metade deste vetor em ordem crescente e a segunda metade em ordem decrescente. Caso a quantidade de números seja ímpar, o elemento do meio deve ser considerado como parte da primeira metade.

As ordenações devem ser feitas usando algum dos algoritmos vistos em aula.

Entradas:

1. A quantidade de números a ler.
2. Vários números inteiros para armazenamento num vetor (numa mesma linha).

Saídas:

- Os valores do vetor, após as ordenações das metades.

Exemplo de entradas:

```
10
5 3 8 6 10 5 6 3 4 7
```

Exemplo de saídas:

```
3 5 6 8 10 7 6 5 4 3
```

Exemplo de entradas:

```
7
65 91 0 45 87 10 32
```

Exemplo de saídas:

```
0 45 65 91 87 32 10
```

Peso: 1

Última tentativa realizada em: 02/06/2019 12:23:05

Tentativas: 1 de 6

Nota (0 a 100): 77

Status ou Justificativa de Nota: O programa não resolve todas as instâncias do problema.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 4: Matriz - Ordenar Linhas Pares Crescente, Ímpares Descrescente

Faça um programa que receba uma matriz quadrada (NxN) e a preencha. Feito isso, ordene as linhas pares em ordem crescente e as linhas ímpares em ordem decrescente. Imprima a nova matriz.

Entradas:

- `int n` - Tamanho da matriz quadrada.
- `int mat[n][n]` - Valores que preencherão a matriz.

Saídas:

- Matriz ordenada de forma crescente nas linhas pares e decrescente nas linhas ímpares (`int`).

Exemplos de Entradas e Saídas:

Entradas:

```
4
12   42    3    1
 6   14   53   32
44   31   26   52
 7    8    9    2
```

Saídas:

```
 1    3   12   42
53   32   14    6
26   31   44   52
 9    8    7    2
```

Peso: 1

Última tentativa realizada em: 01/06/2019 15:45:41

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o

problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 5: Ordenação - Mediana

Em estatística, a **mediana** é a medida de tendência central, ou seja, metade dos valores são menores que a mediana e metade são maiores que ela. Caso a quantidade de informações seja par, a mediana é a média simples dos dois valores medianos.

Faça um programa que calcula a mediana de um conjunto de valores.

A entrada de dados é composta de vários números positivos. Um número negativo na entrada indica o final dos valores.

Obs: Para programas em C++ considere o tamanho máximo do vetor como 20.

```
Exemplo de entrada:
4.4 5.1 1.2 9.3 -1

Exemplo de saída:
4.75
```

Peso: 1

Última tentativa realizada em: 01/06/2019 15:16:55

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 6: Ordenação - Completar o Merge Sort

Você deverá fazer um programa que recebe um vetor de inteiros V de N posições e o ordena. Para ordenar este vetor, você deverá usar o método **Merge Sort**.

Dica: Abaixo temos uma implementação **incompleta** do Merge Sort. Você deverá completá-la se quiser utilizá-la.

```
void Troca(int &A, int &B)
{
        int aux = A;
        A = B;
        B = aux;
}

void Merge(int V[], int p, int q, int r, int U[])
{
        int a = p;
        int b = q+1;

        for(int i = p; i <= r; i++)
                if( b > r || (a <= q && V[a] < V[b]) )
                        U[i] = V[a++];
                else
                        U[i] = V[b++];

        for(int i = p; i <= r; i++)
                V[i] = U[i];
}

void MergeSort(int V[], int primeira, int ultima, int U[])
{
        if(primeira >= ultima) return;

        int p = primeira;
        int r = ultima;
        int q = (p+r)/2;

}
```

Entradas:

- Tamanho N do vetor.
- Os N elementos do vetor V.

Saídas:

- Vetor V ordenado.

Exemplo de Entrada:

```
5
7 8 5 3 6
```

Exemplo de saída:

```
3 5 6 7 8
```

Peso: 1

Última tentativa realizada em: 01/06/2019 15:20:18

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 7: Ordenação - Completar o Quick Sort 1

Você deverá fazer um programa que recebe um vetor de inteiros V de N posições e o ordena. Para ordenar este vetor, você deverá usar o método **Quick Sort**.

Dica: Abaixo temos uma implementação **incompleta** do Quick Sort. Você deverá completá-la (e não refazer o método).

```
int Rearranja(int V[], int e, int d, int pivo, int U[])
{
        int j = e, k = d;
```

```
        Troca(V[pivo], V[d]);
        pivo = d;

        for(int i = e; i <= d; i++)
                if(V[i] <= V[pivo])
                        U[j++] = V[i];
                else
                        U[k--] = V[i];

        for(int i = e; i <= d; i++)
                V[i] = U[i];

        return j-1;
}

void QuickSort(int V[], int e, int d, int U[])
{
        if(e >= d) return;

        int pivo = e;
        pivo = Rearranja(V, e, d, pivo, U);
        //Complete aqui
}
```

Entradas:

- Tamanho N do vetor.
- Os N elementos do vetor V.

Saídas:

- Vetor V ordenado.

Exemplo de Entrada:

```
5
7 8 5 3 6
```

Exemplo de saída:

```
3 5 6 7 8
```

Peso: 1

Última tentativa realizada em: 01/06/2019 15:22:28

Tentativas: 1 de 6

Nota (0 a 100): 100

Status ou Justificativa de Nota: Nenhum erro encontrado.

Ver Código da Última Tentativa

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 8: Ordenação - QuickSort Soma dos não trocados

O **QuickSort** é um método de ordenação muito rápido e eficiente, que adota a estratégia de divisão e conquista. A estratégia consiste em rearranjar os elementos de modo que os elementos "menores" precedam os elementos "maiores". Em seguida, o QuickSort ordena os dois subvetores de elementos menores e maiores recursivamente até que o vetor completo se encontre ordenado.

Abaixo está uma implementação do método QuickSort **com erros**. Sua missão é corrigi-lo para que ele funcione corretamente e também fazer uma modificação de forma que o algoritmo imprima todo o vetor resultante a cada iteração.

A impressão do vetor resultante deverá ser impresso antes das chamadas recursivas.

```
void quickSort(int v[], int esq, int dir) {
        int i, j, aux, pivo;
        i = esq;
        j = dir;
        pivo = v[(esq+dir)/2];
        while (i <= j) {
                while (v[i] < pivo) {
                        i++;
                }
                while (v[j] > pivo) {
                        j--;
                }
                if (i <= j) {
                        v[i] = v[j];
                        v[j] = v[i];
                        i++;
                        j--;
                }
        }

        if (esq < j) {
```

```
                quickSort(v, esq, j);
        }
        if (i < dir) {
                quickSort(v, i, dir);
        }
}
```

Entradas:

1. A quantidade de números a ser ordenada.
2. Os números inteiros positivos entre 1 e 1000 a serem ordenados.

Saídas:

1. A situação do vetor a cada iteração.

Exemplo de Entrada:

```
10
10 1 9 2 8 3 7 4 6 5
```

Exemplo de Saída:

```
5 1 6 2 4 3 7 8 9 10
2 1 6 5 4 3 7 8 9 10
1 2 6 5 4 3 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

Exemplo de Entrada:

```
10
1 10 2 9 3 8 4 7 5 6
```

Exemplo de Saída:

```
1 3 2 9 10 8 4 7 5 6
1 2 3 9 10 8 4 7 5 6
1 2 3 9 10 8 4 7 5 6
1 2 3 4 10 8 9 7 5 6
1 2 3 4 6 8 5 7 9 10
1 2 3 4 6 7 5 8 9 10
1 2 3 4 6 5 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

Peso: 1

Nova Resposta:
Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 9: Ordenação - Estratégia Pokémon Go (arquivo)

Um fã do jogo Pokémon Go resolveu criar um programa para definir quais serão os próximos pokémons que ele tentará pegar. Ele conseguiu um jeito de gerar um **arquivo texto** (*dados.txt*) contendo na primeira linha a quantidade de pokémons e depois, em cada linha, as seguintes informações de um pokémon: **nome**, **tipo** e **distância** para sua posição atual. A ideia é que o usuário informe na **entrada padrão** o **tipo** de pokémon que quer pegar e a **quantidade N** de pokémons desse tipo e o programa, a partir dos dados do arquivo, salve em um **arquivo texto** (*pokemon.txt*) os nomes dos **N** pokémons **daquele tipo mais próximos** de sua posição atual (em **ordem crescente** de distância).

Observações: considere que não existem distâncias repetidas e também que sempre existirá pokémons suficientes para a quantidade **N** passada pelo usuário.

Entradas (*arquivo dados.txt*):

1. Quantidade de pokémons do arquivo.
2. Depois, em cada linha: nome, tipo e distância (inteiro) de um pokémon para a posição atual do jogador.

Entradas (*entrada padrão*):

1. Tipo de pokémon que o jogador quer pegar.
2. Quantidade **N** de pokémons que o jogador quer pegar.

Saídas (*arquivo pokemon.txt*):

1. Nomes dos **N** pokémons mais próximos, do tipo passado, em ordem crescente de distância.

Exemplo de Entrada (*arquivo dados.txt*):

```
10
Bulbasaur Planta 8
Charmander Fogo 9
Squirtle Agua 13
Pikachu Eletrico 10
Chikorita Planta 3
```

```
Treecko Planta 4
Cyndaquil Fogo 7
Torchic Fogo 5
Totodile Agua 6
Mudkip Agua 11
```

Exemplo de Entrada (*entrada padrão*):

```
Fogo
2
```

Exemplo de Saída (*arquivo pokemon.txt*):

```
Torchic
Cyndaquil
```

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta

Questão 10: Ordenação - Pontos de Vitória

Certo jogo de estratégia, com N jogadores, é jogado em volta de uma mesa. O primeiro a jogar é o jogador 1, o segundo a jogar é o jogador 2 e assim por diante. Uma vez completada uma rodada, novamente o jogador 1 faz sua jogada e a ordem dos jogadores se repete novamente. A cada jogada, um jogador garante uma certa quantidade de Pontos de Vitória. A pontuação de cada jogador consiste na soma dos Pontos de Vitória de cada uma das suas jogadas. Dado o número de jogadores, o número de rodadas e uma lista representando os Pontos de Vitória na ordem em que foram obtidos, você deve determinar e imprimir, em ordem decrescente de pontuação cada jogador. Caso mais de um jogador obtenha a pontuação máxima, o jogador com pontuação máxima que tiver jogado por último é o vencedor.

Entradas:

1. Número de jogadores
2. Número de rodadas
3. Pontos de vitórias de todos os jogadores em cada rodada

Saídas:

1. Jogadores em ordem decrescente de pontuação.

Exemplo de Entrada:

```
3
2
1  2  3
3  2  2
```

Exemplo de Saída:

```
3
2
1
```

baseado em: http://maratona.ime.usp.br/hist/2015/primeira-fase/maratona.pdf (Problema J)

Peso: 1

Nova Resposta:

Selecione o arquivo com o código fonte do programa que resolve o problema para enviá-lo.

Escolher Arquivo nenhum arquivo selecionado Enviar Resposta