ATIVIDADE OBJETIVA - M2b - Ordenação em Memória Primária

Entrega Sem prazo **Pontos** 8 **Perguntas** 8 **Limite de tempo** Nenhum **Tentativas permitidas** Sem limite

Instruções

O objetivo desta atividade é proporcionar uma melhor fixação quanto ao conteúdo da unidade 2.
Com isso você será capaz de ter um entendimento das abordagens apresentadas na unidade **ORDENAÇÃO EM MEMÓRIA PRIMÁRIA**.

Instruções:

Clique em "**Fazer o teste**":
Responda todas as questões e depois clique no botão "**Enviar teste**" para finalizar a atividade.

Fazer o teste novamente

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MANTIDO	Tentativa 5	1 minuto	8 de 8
MAIS RECENTE	Tentativa 5	1 minuto	8 de 8
	Tentativa 4	1 minuto	7 de 8
	Tentativa 3	2 minutos	7 de 8
	Tentativa 2	Menos de 1 minuto	0 de 8
	Tentativa 1	7 minutos	5 de 8

Detalhes da última tentativa:

Tempo:	1 minuto
Pontuação atual:	8 de 8
Pontuação mantida:	8 de 8

5 tentativas até o momento
Visualizar tentativas anteriores
Tentativas ilimitadas

Fazer o teste novamente

(Será mantida sua pontuação mais alta)

As respostas corretas estão ocultas.

Pontuação desta tentativa: 8 de 8
Enviado 2 dez em 20:56
Esta tentativa levou 1 minuto.

Pergunta 1

1 / 1 pts

O comando **correto** para efetuar a ordenação de um array de palavras vet utilizando recursos nativos da linguagem c# é:

- Sort(vet);
- Vet.Sort();
- Array.Sort(vet);
- Array.Sort(vet);

Em c#, a correta execução do comando usando recursos da linguagem é Array.sort passando como parâmetro o vetor a ser ordenado.

Pergunta 2

1 / 1 pts

O código abaixo apresenta a ordenação dos elementos de um vetor de inteiros usando recursos nativos da liguagem c#.

```
// ordena o vetor
int[] intVetor = new int[5] { 8, 10, 2, 6, 3 };
Array.Sort(intVetor);

// imprime na tela o conteudo do vetor
foreach (int i in intVetor)
    Console.Write(i + " ");
// saida: 2 3 6 8 10
```

Sobre esses recursos nativos, assinale a afirmativa **incorreta**:

- A execução do comando `Array.Sort(intVetor);` ordena os elementos do vetor em ordem crescente.
- O método Array.sort internamente executa o algoritmo quicksort para ordenar os elementos.
- Esse tipo de recurso utilizado pode ser usado apenas para tipos primitivos de dados da linguagem c#.
- A implementação da interface `IComparable` permite a execução da ordenação em tipos customizados.

A utilização da classe Array permite a ordenação de elementos de tipos primitivos e customizados. Para isso, o programador deve implementar em sua classe a interface `IComparable`.

Pergunta 3

1 / 1 pts

Considerando um array A que contém n+n elementos, onde os primeiros n elementos estão ordenados e os demais n elementos não estão. Qual dos algoritmos abaixo é melhor indicado para completar essa ordenação?

- Algoritmo da Bolha, apesar de seu alto custo de execução, o algoritmo percorre todos os elementos.
- Algoritmo de Seleção, pois o mesmo percorre todos os elementos comparando com os demais elementos da parte ordenada.
- Algoritmo de Seleção, por apresentar baixo custo de execução em sua complexidade.
- Algoritmo de Inserção, pois ele faz uma varredura na parte ordenada para encontrar a posição correta de cada novo elemento.

Se já existe uma parte do vetor ordenada, o algoritmo mais indicado para inserir novos elementos é o Algoritmo de Inserção.

Pergunta 4

1 / 1 pts

O algoritmo Inserção apresentado a seguir recebe como parâmetro um vetor de números inteiros desordenados e rearranja esse vetor de tal forma que seus elementos estejam ordenados de forma crescente:

```
1 public void insercao(int [] vetor, int TAM){
2   int j, chave;
3   for(int i = 1; i < TAM; i++){
4       chave = vetor[i];
5       j = i - 1;
6       while(j >= 0 && vetor[j] < chave) {
7           vetor[j+1] = vetor[j];
8           j = j - 1;
9       }
10      vetor[j+1] = chave;
11  }
12 }
```

Considerando que nesse algoritmo há erros que devem ser corrigidos para a sua correta execução, assinale a opção que apresenta todas as correções necessárias.

- A linha 5 deve ser corrigida da seguinte forma: `j = i - 1;` e a linha 6, do seguinte modo: `while(j >= 0 && V[j] > chave)`.
- A linha 3 deve ser corrigida da seguinte forma: `for(i = 1; i < n-1; i++)` e a linha 10, do seguinte modo: `V[j+1] = chave;`.
- A linha 6 deve ser corrigida da seguinte forma: `while(j >= 0 && v[j] > chave)` e a linha 10, do seguinte modo: `V[j+1] = V[j];`.
- A linha 7 deve ser corrigida da seguinte forma: `V[j+1] = V[j];` e a linha 10, do seguinte modo: `V[j+1] = chave;`.

No algoritmo Inserção correto, o elemento a ser inserido na parte ordenada (chave) é comparado com os já ordenados (posições 0 à i). Enquanto o elemento chave é maior que os já ordenados, deslocamos cada elemento já comparado uma posição à mais do que sua posição atual, permitindo a inserção do elemento chave em sua posição correta.

Pergunta 5

1 / 1 pts

Sobre o algoritmo de ordenação Quicksort, é **incorreto** afirmar que:

- Faz em média n^2 comparações para ordenar n itens.

  Incorreta. Complexidade do algoritmos para ordenar n itens é n × lg (n) comparações na média.
- Escolhe um elemento que será chamado o pivô do vetor.
- Faz em média n × lg (n) comparações para ordenar n itens.
- Seu funcionamento se baseia no paradigma de divisão e conquista.

Pergunta 6

1 / 1 pts

Seja o vetor **[20, 10, 30, 50, 45, 35, 5]** onde o primeiro elemento (20) está na posição 1, assinale a opção que contém o vetor após a execução da primeira interação do algoritmo de ordenação Quicksort :

- [5, 10, 20, 30, 35, 45, 50]
- [20, 10, 30, 5, 45, 35, 50]
- [50, 20, 45, 30, 10, 5, 35]
- [50, 45, 35, 30, 20, 15, 10]

O pivô escolhido foi o elemento da posição 7/2 = 3 => (50). Assim após a primeira verificação todos os elementos abaixo da posição 3 serão menores ou iguais a ele. Apenas um swap será executado, entre o elemento 50 e 5.

Pergunta 7

1 / 1 pts

Considere o arquivo abaixo cujos registros estão representados apenas pelos seus identificadores (ids):

82	42	16	26	23	29	90	43	78	62	79	45

Ordenando o arquivo por meio de um algoritmo de intercalação balanceada de 2 (dois) caminhos em um sistema cuja capacidade de ordenação em memória principal é de apenas 3 (três) registros, qual será o resultado esperado da primeira intercalação?

- Arq 01: 16 23 26 29 42 82
  Arq 02: 43 45 62 78 79 90
- Arq 01: 82 42 16 26 23 29
  Arq 02: 90 43 78 62 79 45
- Arq 01: 16 42 82 43 78 90
  Arq 02: 23 26 29 45 62 79
- Arq 01: 16 42 82 23 26 29
  Arq 02: 43 78 90 45 62 79

A distribuição será:
Arq01) 16-42-82 43-78-90
Arq02) 23-26-29 45-62-79

Após Intercalação:
Arq01) 16-23-26-29-42-82
Arq02) 43-45-62-78-79-90

Pergunta 8

1 / 1 pts

Considerando a capacidade de ordenação em memória de 4 registros e uma intercalação balanceada de 2 caminhos para os registros. Assinale a alternativa que apresenta o resultado após a primeira intercalação das chaves apresentadas a seguir:

14	17	11	20	8	3	32	21	22	30	19	21	24	31	42	35

- Arq 01: 11 14 17 20 19 21 22 30
  Arq 02: 3 8 21 32 24 31 35 42
- Arq 01: 11 8 21 32 19 21 22 30
  Arq 02: 3 14 17 20 24 31 35 42
- Arq 01: 11 14 17 20 24 31 35 42
  Arq 02: 3 8 21 32 19 21 22 30
- Arq 01: 3 8 11 14 17 20 21 32
  Arq 02: 19 31 22 24 30 31 35 42

A distribuição será:
Arq01) 11-14-17-20 19-21-22-30
Arq02) 3-8-21-32 24-31-35-42

Após Intercalação:
Arq01) 3-8-11-14-17-20-21-32
Arq02) 19-21-22-24-30-31-35-42

Pontuação do teste: 8 de 8