

A atividade deve ser entregue no formato .DOC ou PDF.

Os exercícios que envolvam codificação podem ser entregues em arquivo zipado ou disponibilizados no GIT.

Atividades iguais serão zeradas.

Atividades em atraso não serão avaliadas.

1) Dado o algoritmo Quick Sort, explique o funcionamento de cada método abaixo, implemente o algoritmo e simule a sua execução para o seguinte domínio de entrada: [11, 15, 32, 43, 28, 17, 79, 18, 33, 99, 88, 75, 45, 82, 42, 55, 78], **realizando a ordenação escolhendo como pivô o elemento central.**
(2.75 pontos)

```
public static void quickSort (int vet[], int ini, int fim){
    int divisao;
    if (ini < fim) {
        divisao = particao(vet, ini, fim);
        quickSort (vet, ini, divisao-1);
        quickSort (vet, divisao+1, fim);
    }
}

public static int particao (int vet[], int ini, int fim){
    int pivo = vet[ini], i = ini+1, f = fim, aux;
    while (i <= f) {
        while (i <= fim && vet[i] <= pivo)
            ++i;
        while (pivo < vet[f])
            --f;
        if (i < f){
            aux = vet[i];
            vet[i] = vet[f];
            vet[f] = aux;
            ++i;
            --f;
        }
    }
    if (ini != f){
        vet[ini] = vet[f];
        vet[f] = pivo;
    }
    return f;
}
```

2) Dado o algoritmo Merge Sort, explique o funcionamento de cada método abaixo, implemente o algoritmo e simule a sua execução para o seguinte domínio de entrada: [11, 15, 32, 43, 28, 17, 79, 18, 33, 99, 88, 75, 45, 82]. (2.75 pontos)

```

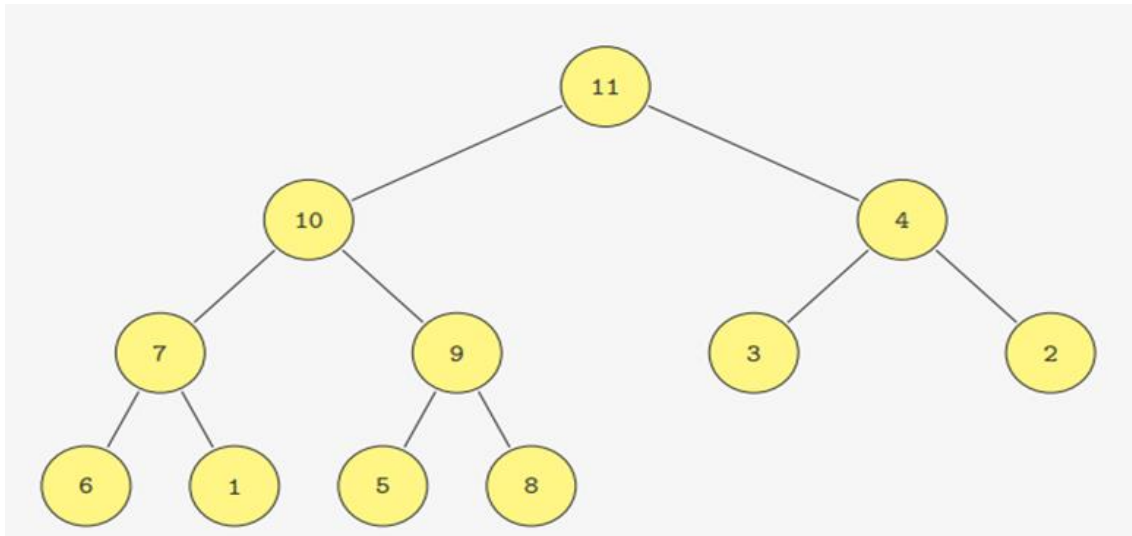
public static void mergeSortRecursivo(int lista[], int inicio, int fim){
    if (inicio < fim){
        int meio = (inicio + fim) / 2;
        mergeSortRecursivo(lista,inicio, meio);
        mergeSortRecursivo(lista,meio + 1, fim);
        mesclar(lista,inicio, meio, meio+1,fim);
    }
}

public static void mesclar(int lista[], int inicioA, int fimA,
    int inicioB, int fimB){
    int i1 = inicioA;
    int i2 = inicioB;
    int iaux = inicioA;
    int aux[] = new int[lista.length];
    while (i1 <= fimA && i2 <= fimB){
        if(lista[i1] <= lista[i2])
            aux[iaux++]=lista[i1++];
        else
            aux[iaux++]=lista[i2++];
    }
    while (i1 <=fimA)
        aux[iaux++]=lista[i1++];
    while (i2 <=fimB)
        aux[iaux++]=lista[i2++];
    for (int i=inicioA;i<=fimB;i++)
        lista[i] = aux[i];
}

```

3) Explique o funcionamento dos algoritmos de ordenação Quick Sort, Merge Sort e Heap Sort, detalhe as principais diferenças entre os três algoritmos de ordenação e apresente um exemplo de teste de mesa para simulação de cada um dos três algoritmos em um conjunto de entrada com no mínimo 8 elementos. (3.5 pontos)

4. O algoritmo Heap Sort utiliza o conceito de Fila de Prioridades para realizar as operações de inclusão e remoção de elementos. Considerando a ordenação pelo Heap Máximo, **demonstre todos os passos** para a reordenação do algoritmo após a remoção de um elemento. (1.0 pontos)



5) Explique qual algoritmo de ordenação se aplica a afirmação abaixo, justifique sua resposta. (0.5 pontos)

“Método de Ordenação que utiliza-se do método da divisão e conquista para ordenação do vetor. Em sua técnica, escolhe um elemento denominado de pivô (um dos elementos a serem ordenados) e separa os elementos em 2 partes, de modo que os elementos menores que o pivô ficam à esquerda e os elementos maiores que o pivô ficam à direita. Esse processo é repetido recursivamente até que todos os elementos estejam ordenados. “