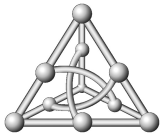


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

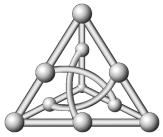
Prova 3

Algoritmos e Programação Orientada a Objetos II — 2018

Instruções para a realização da prova:

1. A prova contém **4 questões**, totalizando **10 pontos**;
2. A prova tem duração de **3 horas e 30 minutos**;
3. **Não altere a disposição do mobiliário da prova**;
4. **Coloque seu celular** em cima do gabinete, desligado. Caso o professor ouça o celular vibrando ou tocando durante a aula, o aluno receberá **nota zero**;
5. **Faça a prova em silêncio**; não converse durante a prova;
6. **Não tente plagiar a prova do(a) seu(sua) colega**, você pode prejudicar você e seu(sua) colega;
7. **Utilize apenas o que foi ensinado em sala de aula**. O uso de qualquer estrutura de programação ou estrutura de dados que não foi ensinada em sala de aula anulará a sua questão.
8. Utilize a IDE ou editor de texto de sua preferência;
9. A questão 2 é dissertativa, e deve ser respondida utilizando algum editor de texto ou ferramenta de sua preferência;
10. Crie uma pasta com o seu nome dentro do diretório *Documentos*, e coloque o seu projeto dentro desta pasta.

BOA PROVA!



Questões

1. **[2,50]** O **piso** de um número inteiro positivo x é o único inteiro i tal que $i \leq x < i+1$. O piso de x é denotado por $\lfloor x \rfloor$.

   Segue uma amostra de valores da função $\lfloor \log_3 n \rfloor$:

n	26.5	27.7	80.1	81.7	242.2	243.9
$\lfloor \log_3 n \rfloor$	3	4	4	5	5	6

   Escreva um método recursivo *pisoLog3(int n)*, que recebe um número real positivo n e devolve $\lfloor \log_3 n \rfloor$. Escreva também um método *main* para testar o método recursivo proposto.

2. **[2,50]** O algoritmo abaixo contém o método separa, utilizando em um dos algoritmos de ordenação vistos em sala de aula.

```
int separa(int p, int r, int v[]) {
    int x, i, j;
    x = v[r];
    i = p-1;
    for(j = p ; j <= r-1 ; j++) {
        if(v[j] <= x){
            i = i + 1;
            troca(v, i, j);
        }
    }
    troca(v, i+1, r);
    return i+1;
}

void troca(int[] v, int i, int j) {
    int aux = v[i];
    v[i] = v[j];
    v[j] = aux;
}
```

(a) Execute o algoritmo para o vetor apresentado na tabela abaixo, deixando claro em cada passo o conteúdo das variáves i e j.

Posição	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Variáveis	i	j									
Vetor		1	20	2	21	3	22	4	23	5	10
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											
Variáveis											
Vetor											

(b) Com base nas análises de pior caso e melhor caso do *Quick Sort*, você diria que o pivô do exemplo anterior é um exemplo típico de boa ou má escolha de pivô? Justifique.

3. **[2,50]** Durante as aulas de Algoritmos e Programação 2, Wallace teve a brilhante ideia de mesclar o *merge sort* e o *heap sort* em um único algoritmo de ordenação, seguindo a seguinte abordagem: quando o algoritmo de ordenação por intercalação dividir o vetor recursivamente e, em um determinado momento, um subvetor ficar com tamanho menor ou igual 4, aplicar o algoritmo de ordenação *heap sort* apenas para esse subvetor.

   (a) Crie um método chamado *heapSortIntervalo*, que corresponde a uma versão método *heapSort* que ordena apenas um intervalo de índices $[i...j]$ do vetor v recebido como parâmetro, utilizando a mesma ideia do algoritmo original *heap sort*. Crie uma classe de testes para validar esse algoritmo.

   (b) Crie uma classe chamada *MergeHeap*, que contenha um método chamado *mergeHeap* com a seguinte assinatura:

   ```
   public static void mergeHeap(int[] v, int p, int r) { ... }
   ```

   Esse método deve implementar a ordenação por intercalação em sua maneira convencional, ordenando o vetor v de maneira crescente. Entretanto, esse método deve aplicar a ideia de utilizar o algoritmo de ordenação *heap sort* quando o tamanho do subvetor se tornar menor ou igual a 4 elementos. Para ordenar esse subvetor, você deve utilizar o algoritmo proposto no ítem (a).

   Crie também um método *main* que teste o método *mergeHeap* com diferentes entradas.

   O arquivo Algoritmos2.java (disponibilizado juntamente com o arquivo da prova) contém algoritmos que podem te auxiliar na resolução dessa questão.

4. **[2,50]** Wallace é um menino passeador e, com a chegada das férias, está planejando sua próxima viagem. Para isso, ele deseja saber informações sobre os locais que já fez viagem, auxiliando assim na escolha do próximo local para seu passeio.

   Wallace é muito organizado e mantém um registro de todas as suas viagens em um arquivo *viagens.csv*. Esse arquivo possui o seguinte formato:

   ```
   nome_do_local,data_da_viagem,tempo_de_estadia,custo_da_viagem
   ```

   Abaixo segue um exemplo desse arquivo.

   ```
   Maceió,11/12/2017,7,4000
   Campo Grande,11/12/2016,1,100
   Bonito,11/12/2014,5,1000000
   Maceió,11/12/2013,5,4000
   ```

   Crie um programa chamado *RelatoriosDoWallace*, que a partir das informações do arquivo *viagens.csv*, oferece dois relatórios referentes às viagens feitas por Wallace:

   (a) Listagem dos locais que Wallace já visitou;

   (b) Quantidade de dinheiro gasto nas viagens feitas para um determinado local. Por exemplo, para o arquivo de entrada de exemplo, você deveria informar que Wallace gastou *R$ 8000* em suas viagens para Maceió, bem com o mesmo tipo de informação para os outros locais visitados.

   Os arquivos Algoritmos4.java e Viagem.java (disponibilizado juntamente com o arquivo da prova) contém classes e métodos que efetuam a leitura do arquivo de entrada, o devolvendo já em um formato apropriado.

   Neste exercício, você deve utilizar as coleções genéricas ArrayList, Map e Set.