2º Teste de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA

2 de Junho de 2018

Duração: 2 horas e 15 minutos.

O teste tem **4** páginas e **4** perguntas.

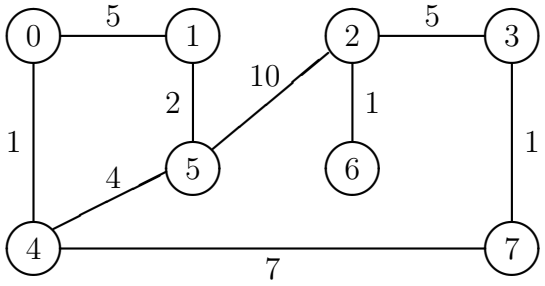
Responda a **perguntas** diferentes em **folhas** diferentes.

Se precisar de folhas, peça ao docente.

Nº de **folhas** entregues (excluindo o enunciado)

Número: _____ **Nome:** _____

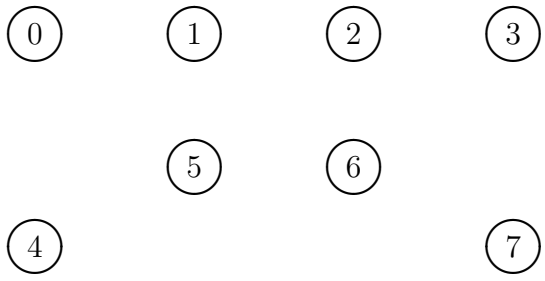
1. [4 valores] Suponha que se executa o algoritmo de Prim com o grafo G esquematizado na figura.



Assuma que a origem é o vértice 0 (ou seja, considere que o método *G.aNode()* retorna 0).

(a) Indique a ordem pela qual os arcos são inseridos no resultado (i.e. no vetor *mst*), representando um arco com peso p entre os vértices v e w pelo par (v, w) ou (w, v).

(b) Represente a árvore mínima de cobertura encontrada, desenhando os arcos do sub-grafo computado na figura abaixo:



(c) Qual é o custo da árvore encontrada? _____

2. [6 valores] A classe *SequenceWithMin* implementa sequências de elementos (comparáveis) que oferecem uma operação para obter o menor elemento da sequência.

```
public class SequenceWithMin<E extends Comparable<? super E>>
{

    private E[] sequence;    // Elements are stored in an array.
    private int curLength;   // Number of elements in the sequence.
    private int analysed;    // Number of analysed elements.
    private E minElem;       // Minimum of the analysed elements.

    @SuppressWarnings("unchecked")
    public SequenceWithMin( int maxLength ) {
        sequence = (E[]) new Comparable[maxLength];
        curLength = 0;
        analysed = 0;
        minElem = null;
    }

    public int length( ) {
        return curLength;
    }

    public void add( E element ) {
        if ( curLength == sequence.length )
            throw new RuntimeException();

        sequence[curLength++] = element;
    }

    public E min( ) {
        if ( curLength == 0 )
            throw new RuntimeException();

        if ( analysed < curLength ) {
            if ( minElem == null )
                minElem = sequence[0];

            for ( int i = analysed; i < curLength; i++ )
                if ( sequence[i].compareTo(minElem) < 0 )
                    minElem = sequence[i];
            analysed = curLength;
        }
        return minElem;
    }

}
```

Considere a função $\Phi(S)$, que atribui a cada objeto S da classe *SequenceWithMin* a diferença entre o comprimento da sequência e o número de elementos já analisados no cálculo do elemento mínimo:

$$\Phi(S) = S.\texttt{curLength} - S.\texttt{analysed}\,.$$

Prove que Φ é uma função potencial válida e calcule as complexidades amortizadas dos métodos *length*, *add* e *min*, justificando. No estudo da complexidade amortizada dos métodos *add* e *min*, assuma que não é levantada a exceção. No estudo da complexidade amortizada do método *min*, analise separadamente os casos em que a condição "`analysed < curLength`" é: falsa; verdadeira. Assuma que o método *compareTo* da classe dos elementos do tipo E tem complexidade constante.

3. [6 valores] No Problema do Escalonamento, há m máquinas (todas iguais) e n processos que vão ser executados nessas máquinas. Cada processo j é representado por um par (j, t_j), que indica que o tempo de processamento de j é t_j (para $j = 1, \ldots, n$). Cada um dos processos terá de ser totalmente executado numa máquina. Uma *atribuição dos n processos às m máquinas* é uma função $f : \{1, 2, \ldots, n\} \longrightarrow \{1, 2, \ldots, m\}$. A *carga* $C_f(i)$ da máquina i é a soma dos tempos de processamento dos processos atribuídos à máquina i:

$$C_f(i) = \sum_{\{\ j \mid 1 \leq j \leq n \,\wedge\, f(j) = i\ \}} t_j\,.$$

Queremos saber se é possível atribuir os processos às máquinas de forma a que nenhuma máquina fique com uma carga superior a um dado valor V.

Suponhamos que há 3 máquinas ($m = 3$) e 6 processos ($n = 6$) com os seguintes tempos de processamento:

$$P = \{(1, 2),\ (2, 3),\ (3, 4),\ (4, 6),\ (5, 2),\ (6, 2)\}.$$

Se a atribuição g dos processos às máquinas for:

$$g(1) = 1 \quad g(2) = 2 \quad g(3) = 2 \quad g(4) = 3 \quad g(5) = 1 \quad g(6) = 1,$$

ou seja, atribuindo os processos 1, 5 e 6 à primeira máquina, os processos 2 e 3 à segunda máquina e o processo 4 à terceira máquina, então:

- a carga da primeira máquina é $C_g(1) = 2 + 2 + 2 = 6$;
- a carga da segunda máquina é $C_g(2) = 3 + 4 = 7$;
- a carga da terceira máquina é $C_g(3) = 6$.

Portanto, é possível atribuir os processos às máquinas de forma a que nenhuma máquina fique com uma carga superior a 7.

O **Problema do Escalonamento** formula-se da seguinte forma.
Dados três inteiros positivos, m, n e V, e um conjunto $P = \{(1, t_1),\ (2, t_2),\ \ldots,\ (n, t_n)\}$ com os tempos de processamento dos processos (onde t_j é um inteiro positivo, para qualquer $j = 1, \ldots, n$), existe uma atribuição dos n processos às m máquinas tal que as cargas de todas as máquinas são inferiores ou iguais a V?

Prove que o Problema do Escalonamento é NP-completo.

4. [4 valores] No Clube de Livros De-Mão-Em-Mão, os membros não usam serviços postais para enviar livros uns aos outros; passam-nos de mão em mão nos locais onde têm atividades regulares em comum.

Membro	2ªf	3ªf	4ªf	5ªf	6ªf	Sáb	Dom
Ana	Escola	-	Ginásio	Escola	-	-	-
Bela	Ginásio	Escola	-	-	Bar	-	-
Tó	-	Escola	-	-	-	-	-
Xana	Ginásio	-	-	-	Bar	Ginásio	-
Zé	Escola	-	-	Escola	-	Ginásio	-

Suponhamos que as atividades semanais regulares de cada membro do clube se encontram na tabela acima.

- Se o Tó quiser enviar um livro à Ana:
  - o Tó entrega o livro na 3ª feira na Escola à Bela,
  - a Bela entrega-o na 6ª feira da mesma semana no Bar à Xana,
  - a Xana entrega-o no Sábado da mesma semana no Ginásio ao Zé,
  - o Zé entrega-o na 2ª feira da semana seguinte na Escola à Ana.

  Este livro leva **seis** dias a chegar ao destinatário, porque o tempo só começa a contar a partir do momento em que o livro muda de mãos pela primeira vez.
- Se a Ana quiser enviar um livro ao Zé, pode entregar-lho diretamente na Escola (numa 2ª feira ou numa 5ª feira). Neste caso, o envio do livro demora **zero** dias.

Apresente uma função (em pseudo-código) que recebe:

- Uma matriz A de caracteres, com um número (L) positivo de linhas e sete colunas. A matriz tem uma linha por cada membro do clube e uma coluna por cada dia da semana (de 2ª feira a domingo). Em cada posição da matriz, há uma letra de 'A' a 'Z', que representa a atividade desse membro do clube nesse dia, ou o carácter '-', que indica que essa pessoa não tem atividades regulares nesse dia.
- Dois inteiros, R e D (ambos entre 0 e $L-1$), que representam o remetente e o destinatário do livro.

A função deve retornar o número mínimo de dias que um livro enviado pelo remetente demora a chegar ao destinatário, assumindo que o tempo só começa a contar no momento em que o livro muda de mãos pela primeira vez. **O corpo da sua função deve chamar:**

- uma função que constrói um grafo e
- um ou vários algoritmos de grafos estudados, como se eles estivessem numa biblioteca (mesmo que esses algoritmos retornem resultados que não interessam para resolver este problema e sejam menos eficientes do que poderiam ser para este caso).

Não programe a função que constrói o grafo. Desenhe apenas o grafo que seria construído se a matriz correspondesse à tabela acima, o remetente fosse o Tó e o destinatário fosse a Ana. Denote os nós de forma a se perceber o que caracterizam (usando palavras como "Ana", "Escola", "2ªf", etc.) em vez de os identificar com números inteiros.