2º Teste de Análise e Desenho de Algoritmos

Departamento de Informática, FCT NOVA

11 de Junho de 2019

Duração: 2 horas e 15 minutos

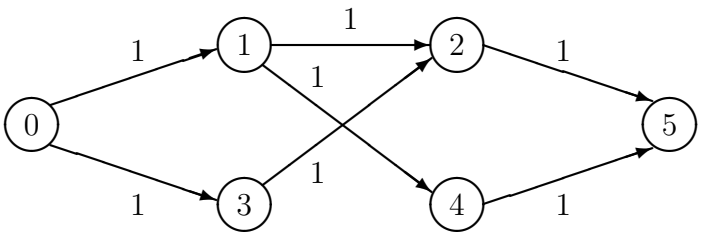
Tem de entregar os 2 cadernos, cada um com 2 folhas.

Os cadernos não podem ser desagrafados.

Identifique os cadernos com o seu número e o seu nome.

Número: _____ **Nome:** _____

Pergunta 1 Suponha que se executa o algoritmo de Edmonds-Karp com o grafo G esquematizado na figura, a fonte 0 e o dreno 5.

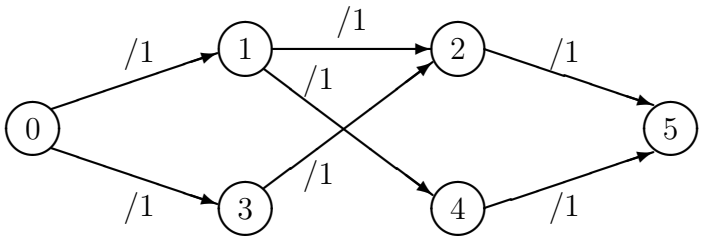


Assuma que o método *outIncidentEdges* itera sempre os arcos por ordem crescente de vértice destino. Por exemplo, $G.outIncidentEdges(0)$ produz os arcos $(0, 1)$ e $(0, 3)$, por esta ordem, ambos com peso 1.

(a) [3 valores] Indique a sequência de caminhos da fonte para o dreno que são encontrados pelo algoritmo.

(b) [0.5 valores] Qual é o valor do fluxo máximo? ☐

(c) [0.5 valores] Indique, na figura abaixo (à esquerda de "/"), o fluxo de cada arco de G quando o algoritmo termina:



Pergunta 2 O **Problema da Soma de Subconjunto** formula-se da seguinte forma. Dados um conjunto finito C de números inteiros e um inteiro t, existe um subconjunto $S \subseteq C$ tal que:

$$\sum_{e \in S} e \;=\; t\,?$$

Por exemplo, a solução da instância $(\{1, 2, -7, 3, -10, -5\},\ 0)$ é sim, porque a soma dos elementos de $\{2, 3, -5\}$ é $2 + 3 + (-5) = 0$.

(a) [3.8 valores] Prove que o Problema da Soma de Subconjunto é NP.

(b) [1.4 valores] Prove que o Problema da Soma de Subconjunto é NP-difícil.

(c) [0.3 valores] O Problema da Soma de Subconjunto é NP-completo? Justifique a sua resposta, assumindo que resolveu as duas alíneas anteriores.

NOTA: O que escrever nesta página não será avaliado.

Pergunta 3 A classe *LazyExpression* implementa expressões aritméticas inteiras (com somas, diferenças e produtos) avaliadas de forma "lazy".

```
public class LazyExpression {

    private int value;              // Result of the evaluated operations
    private Pair<Character,Integer>[] expr; // Non-evaluated operations
    private int size;               // Number of non-evaluated operations

    @SuppressWarnings("unchecked")
    public LazyExpression( int capacity, int firstNum ) {
        value = firstNum;
        expr = new Pair[capacity];
        size = 0;
    }

    public int getValue( ) {
        if ( size > 0 )
            this.evaluate();
        return value;
    }

    public void addOper( char op, int num ) {
        if ( size == expr.length )
            this.evaluate();
        expr[ size++ ] = new Pair<>(op, num);
    }

    private void evaluate( ) {
        for ( int i = 0; i < size; i++ ) {
            Pair<Character,Integer> pair = expr[i];
            char op = pair.getFirst();
            int num = pair.getSecond();
            switch ( op ) {
                case '+': value += num;
                          break;
                case '*': value *= num;
                          break;
                default:  value -= num;
                          break;
            }
        }
        size = 0;
    }

}
```

Considere a função $\Phi(E)$, que atribui a cada objeto E da classe *LazyExpression* o número de operações ainda não efetuadas:

$$\Phi(E) = E.\texttt{size}\,.$$

Número: _____ **Nome:** _____

(a) [1 valor] Prove que Φ é uma função potencial válida.

(b) [4.5 valores] Calcule as complexidades amortizadas dos métodos *getValue* e *addOper*, justificando. No estudo da complexidade amortizada dos dois métodos, analise separadamente os casos em que a condição do **if** é: falsa; verdadeira. Assuma que o construtor e os métodos da classe *Pair* têm complexidade constante.

Pergunta 4 Na cidade da Juventude só há praças (como o Rossio ou a Praça do Comércio) e ruas pedonais (com dois sentidos) que ligam duas praças distintas. Para a juventude poder andar à vontade, algumas ruas são só para jovens (com menos de 30 anos); as restantes são só para cotas (30 ou mais anos). Portanto, os encontros entre jovens e cotas têm de ocorrer numa praça. Dadas as localizações do Jaime (um jovem) e da Carla (uma cota), pretende-se descobrir quanto tempo é necessário para se encontrarem.

Praças		Restrição	Tempo (min)
0	1	J	7
0	2	J	15
0	3	C	8
0	4	C	5
1	2	C	8
1	3	C	11
1	4	J	10
1	5	J	30
2	3	J	5
2	4	C	9
3	4	J	20
3	5	C	25

Suponhamos que a cidade tem seis praças (identificadas pelos números $0, 1, \ldots, 5$) e que a tabela acima descreve as ruas existentes, as suas restrições (só para jovens (J) ou só para cotas (C)) e o tempo que demora a percorrer cada uma (em minutos). Se, num dado instante, o Jaime estiver na Praça 1 e a Carla na Praça 5, só conseguem encontrar-se decorridos **27** minutos (na Praça 3):

- Da Praça 1, o Jaime terá de ir para a Praça 0 (7 min), depois para a Praça 2 (15 min) e daí para a Praça 3 (5 min). Portanto, chega à Praça 3 ao fim de 27 minutos.
- A Carla chega à Praça 3 em 25 minutos, percorrendo a rua que liga as praças 5 e 3.

Note que há outras alternativas, mas nenhuma é melhor do que a descrita. Por exemplo, o Jaime poderia chegar à Praça 3 através da 4, mas por aí demoraria meia hora. Se o ponto de encontro fosse na Praça 5 (bastando à Carla esperar pelo Jaime), encontrar-se-iam ao fim de 30 minutos.

Neste problema, pretende-se que implemente uma função, *minTime*(P, R, M, J, C), que recebe:

- Dois inteiros positivos, P e R, que representam o número de praças e o número de ruas. As praças são identificadas pelos inteiros $0, 1, \ldots, P-1$.
- Uma matriz M de inteiros, com R linhas e 4 colunas. Cada linha tem informação sobre uma rua: as duas praças que a rua liga, o número 0 ou 1 (consoante a rua seja só para jovens ou só para cotas) e o tempo que demora a percorrer (um número positivo, em minutos). Garante-se que, entre duas quaisquer praças diferentes, há no máximo uma rua.
- Dois inteiros, J e C (ambos entre 0 e $P-1$), que representam as praças em que o Jaime e a Carla estão (no instante zero).

A função deve retornar o número mínimo de minutos que têm de passar até que o Jaime e a Carla se encontrem (em alguma praça). **O corpo da função *minTime* deve chamar:**

- uma ou várias funções que constroem grafos;
- 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).

(a) [1.5 valores] Que grafo(s) construiria para resolver este problema com o exemplo dado? Ou seja, se T for a matriz que corresponde à tabela da página anterior, **desenhe apenas o(s) grafo(s)** que seria(m) construído(s) durante a execução de *minTime*(6, 12, T, 1, 5). Desenhar um grafo é representá-lo da forma habitual (como na Pergunta 1).

(b) [3.5 valores] Implemente a função *minTime* (em pseudo-código). Chame a ou as várias funções que constroem grafos, mas não a(s) implemente (porque já ilustrou o(s) grafo(s) criado(s) na alínea anterior). Não implemente nenhum algoritmo de grafos; recorra aos algoritmos estudados, chamando as respetivas funções, sem as alterar.

```
int minTime( int P, int R, int[][] M, int J, int C ) {
```