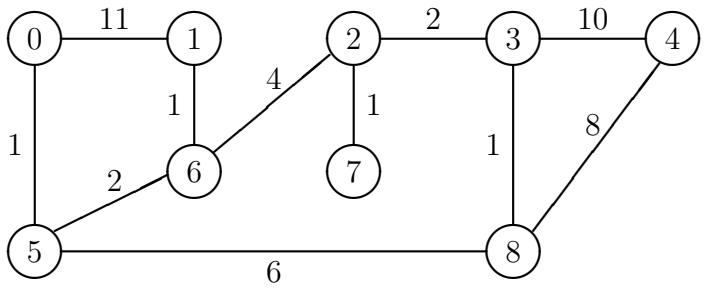
Exame de Análise e Desenho de Algoritmos

Departamento de Informática
Universidade Nova de Lisboa
14 de Junho de 2010

1. [3 valores] Suponha que se executa o algoritmo de Kruskal com o grafo esquematizado na figura.



   Indique:

   - uma ordem pela qual os arcos podem ser inseridos no resultado (i.e., na variável *mst*);
   - o número total de vezes que o método *removeMin* é executado; e
   - o custo da árvore encontrada.

2. [3 valores] Considere a seguinte função recursiva $f(i,j)$, onde i e j são números inteiros não negativos.

$$f(i,j) = \begin{cases} 0 & \text{se } i = 0 \text{ e } j \geq 0; \\ i & \text{se } i \geq 1 \text{ e } j = 0; \\ \max\limits_{0 \leq k < j} f(i,k) & \text{se } i,j \geq 1 \text{ e } i < j; \\ f(i,j-1) + f(i-1,j) & \text{se } i,j \geq 1 \text{ e } i = j; \\ \Big(\ f(i-1,j-1) \ \Big)^2 & \text{se } i,j \geq 1 \text{ e } i > j. \end{cases}$$

   Apresente um algoritmo, desenhado segundo a técnica da programação dinâmica, que, dados dois números inteiros positivos x e y, calcula o valor de $f(x,y)$. Estude (justificando) as complexidades temporal e espacial do seu algoritmo, no melhor caso, no pior caso e no caso esperado.

3. [3 valores] Considere a classe *Statistics*, que permite calcular a média e a variância de uma amostra dinâmica de inteiros. Como o cálculo da variância é mais pesado, só se pode obter a variância quando a dimensão da amostra é uma potência de dois.

```
public class Statistics
{
    private Queue<Integer> data;
    private int nextSizeToUpdateVariance;
    private double sum, variance;

    public Statistics( int item )
    {   data = new LinkedList<Integer>();
        data.add(item);
        nextSizeToUpdateVariance = 2;
        sum = item;
        variance = 0;
    }

    public void addItem( int item )
    {   data.add(item);
        sum += item;
        if ( data.size() == nextSizeToUpdateVariance )
        {   this.updateVariance();
            nextSizeToUpdateVariance *= 2;
        }
    }

    public double getMean( )
    {   return sum / data.size();  }

    public double getVariance( ) throws InvalidDataSizeException
    {   if ( data.size() > nextSizeToUpdateVariance / 2 )
            throw new InvalidDataSizeException();

        return variance;
    }
    private void updateVariance( )
    {   double theMean = this.getMean();
        double total = 0;
        for ( int item : data )
            total += Math.pow(item − theMean, 2);
        variance = total / data.size();
    }
}
```

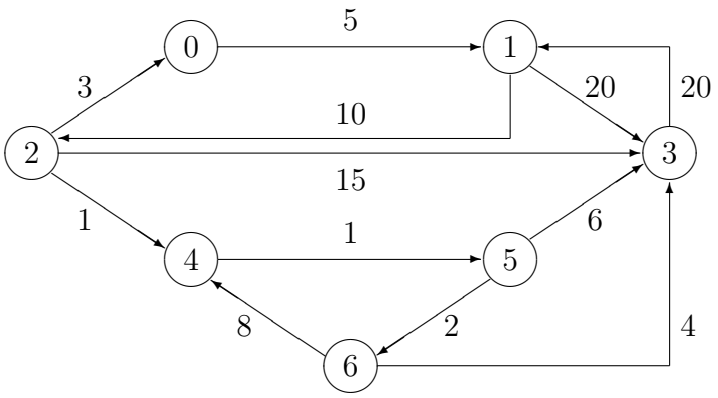
Considere também a função $\Phi(I)$, que atribui a cada instância I da classe *Statistics* o seguinte valor, onde s denota o número de elementos guardados no atributo *data* e n denota o valor do atributo *nextSizeToUpdateVariance*:

$$\Phi(I) = 2s - n\,.$$

Prove que Φ é uma função potencial válida e calcule a complexidade amortizada dos métodos *addItem*, *getMean* e *getVariance*, justificando. No estudo da complexidade amortizada do método *addItem*, analise separadamente os casos em que a condição do *if* é verdadeira e falsa. Assuma que todos os métodos efectuados sobre filas (criação, *add* e *size*) são constantes e que o custo de uma iteração completa da fila é linear no número de elementos da fila.

4. [4 valores] Considere um grafo orientado e pesado. Dados dois caminhos c_1 e c_2, com a mesma origem e o mesmo destino, diz-se que c_1 é *mais equilibrado* do que c_2 se o maior peso dos arcos de c_1 é menor que o maior peso dos arcos de c_2.

   Dados um grafo orientado e pesado, cujos arcos têm custo positivo, e dois vértices v e w, pretende-se encontrar um caminho entre v e w que tenha o menor comprimento pesado e que, de entre os caminhos que satisfazem a restrição anterior, seja dos mais equilibrados.



   Por exemplo, no grafo representado na figura, há dois caminhos de menor comprimento pesado entre os vértices 2 e 3.

Caminho	**Comprimento Pesado**	**Maior Peso dos Arcos**
2 4 5 3	8	$6 = \max\{1,1,6\}$
2 4 5 6 3	8	$4 = \max\{1,1,2,4\}$

   Neste caso, o caminho seleccionado teria de ser 2 4 5 6 3, porque é mais equilibrado do que o caminho 2 4 5 3.

   Escreva um algoritmo (em pseudo-código) que, dados um grafo orientado e pesado, cujos arcos têm custo positivo, e dois vértices v e w, determina um caminho entre v e w nas condições referidas acima. Para simplificar, pode assumir que há, pelo menos, um caminho entre v e w. Estude (justificando) a complexidade temporal do seu algoritmo, no pior caso.

5. [4 valores] Considere que:

   - U é um conjunto finito;
   - $\mathcal{C}$ é um conjunto de subconjuntos de U;
   - para cada elemento $X \in \mathcal{C}$, w_X é um real positivo que indica o peso do conjunto X;
   - p é um real positivo.

   Uma *cobertura de conjuntos de $\mathcal{C}$ para U com peso p* é um subconjunto $\mathcal{C}' \subseteq \mathcal{C}$ tal que:

$$\bigcup_{X \in \mathcal{C}'} X = U \quad \text{e} \quad \sum_{X \in \mathcal{C}'} w_X = p\,.$$

   Por exemplo, se:

   - $\overline{U} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$;
   - $\overline{\mathcal{C}} = \{\ \{1, 3, 5, 7, 9\}\ ,\ \{2, 4, 6, 8\}\ ,\ \{1\}\ ,\ \{2\}\ ,\ \{3\}\ ,\ \{4\}\ ,\ \{5\}\ ,\ \{6, 7, 8, 9\}\ \}$; e
   - o peso de cada $X \in \overline{\mathcal{C}}$ for o menor elemento de X,

   os conjuntos

$$\{\ \{1, 3, 5, 7, 9\}\ ,\ \{2, 4, 6, 8\}\ \} \qquad \text{e} \qquad \{\ \{1\}\ ,\ \{2\}\ ,\ \{3\}\ ,\ \{4\}\ ,\ \{5\}\ ,\ \{6, 7, 8, 9\}\ \}$$

   são ambos coberturas de $\overline{\mathcal{C}}$ para $\overline{U}$, cujos pesos são, respectivamente, 3 e 21.

   O **Problema da Cobertura Pesada de Conjuntos** formula-se da seguinte forma.
   Dados um conjunto finito U, um conjunto $\mathcal{C}$ de subconjuntos de U, um real positivo w_X para cada elemento $X \in \mathcal{C}$, e um real positivo p, existe uma cobertura de conjuntos de $\mathcal{C}$ para U com peso p?

   Prove que o Problema da Cobertura Pesada de Conjuntos é NP-completo.

6. [3 valores] Sejam $X = (x_1, x_2, \ldots, x_m)$ e $Y = (y_1, y_2, \ldots, y_n)$ duas sequências não vazias. Uma *intercalação* de X e Y é uma sequência com $m + n$ elementos que pode ser obtida intercalando os elementos de X e de Y, sem nunca alterar a ordem relativa dos elementos de X nem a ordem relativa dos elementos de Y.

   Para exemplificar, sejam $X' = (1, 2, 3, 4, 5)$ e $Y' = (-1, -2, -3)$.

   - $(1, 2, 3, -1, 4, -2, -3, 5)$ é uma intercalação de X' e de Y', mas
   - $(1, 2, 3, -1, 4, -3, 5, -2)$ não é uma intercalação de X' e de Y', porque o terceiro elemento de Y' ocorre antes do segundo.

   Apresente **uma função booleana recursiva** que, dadas três sequências não vazias de **bits**, $X = (x_1, x_2, \ldots, x_m)$, $Y = (y_1, y_2, \ldots, y_n)$ e $Z = (z_1, z_2, \ldots, z_{m+n})$, determina se Z é uma intercalação de X e Y. Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial.