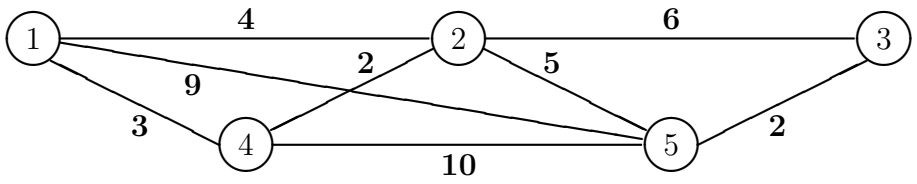
Recurso de Análise e Desenho de Algoritmos

Departamento de Informática
Universidade Nova de Lisboa
22 de Julho de 2008

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



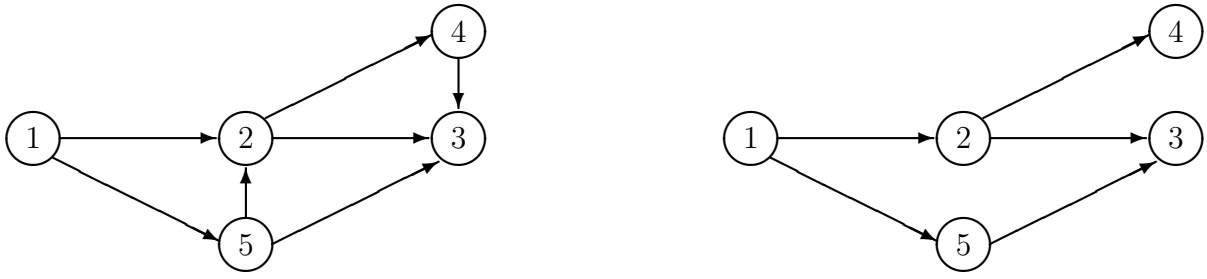
   Assumindo que a origem é o vértice 1 (ou seja, que o método *G.aVertex()* retorna 1):

   - indique a ordem pela qual os vértices são seleccionados; e
   - represente a árvore mínima de cobertura encontrada, desenhando os vértices e os arcos do grafo da forma usual.

2. [4 valores] Considere um grafo orientado e acíclico, onde cada vértice representa uma tarefa de um projecto. A existência de um arco do vértice v para o vértice w indica que a tarefa v terá de estar concluída antes de se iniciar a tarefa w. Pretende-se saber se todas as tarefas do projecto têm de ser executadas sequencialmente ou se, pelo contrário, há tarefas que podem ser executadas em paralelo.

   Para exemplificar, considere os dois grafos (independentes) esquematizados na figura.

   - No grafo da esquerda, todas as tarefas têm de ser executadas sequencialmente. A ordem de execução das tarefas tem de ser 1, 5, 2, 4, 3.
   - No grafo da direita, há tarefas que podem ser executadas em paralelo. Depois de executar a tarefa 1, as tarefas 2 e 5 podem ser executadas simultaneamente. Neste projecto, há outros exemplos de tarefas que podem ser executadas ao mesmo tempo.



   Escreva um algoritmo (em pseudo-código) que, dado um grafo orientado e acíclico, com a informação sobre as tarefas do projecto, retorna *true* se todas as tarefas têm de ser executadas sequencialmente, e retorna *false* se há tarefas que podem ser executadas em paralelo. Estude (justificando) a complexidade temporal do seu algoritmo, no pior caso.

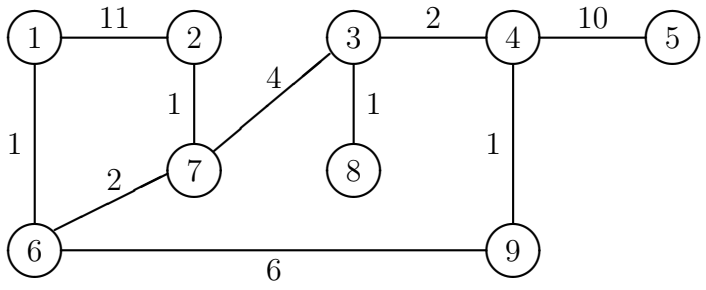
3. [3 valores] O método *numberOfMinRoots*, da classe *FibQueue*, calcula o número de entradas cuja chave é a chave mínima que se encontram na raiz de uma árvore da fila de Fibonacci.

```
class FibNode<K,V>
{
    public FibNode( K key, V value );
    public EntryClass<K,V> getEntry( );
    public K getKey( );
    public V getValue( );
    public int getDegree( );
    public FibNode<K,V> getChild( );
    public FibNode<K,V> getLeftSibling( );
    public FibNode<K,V> getRightSibling( );
    public FibNode<K,V> getParent( );
    public boolean isMarked( );
    public void setEntry( EntryClass<K,V> newEntry );
    public void setEntry( K newKey, V newValue );
    public void setValue( V newValue );
    public void setDegree( int newDegree );
    public void incrementDegree( );
    public void decrementDegree( );
    public void setChild( FibNode<K,V> newChild );
    public void setLeftSibling( FibNode<K,V> newLeftSibling );
    public void setRightSibling( FibNode<K,V> newRightSibling );
    public void makeSingleton( );
    public void setParent( FibNode<K,V> newParent );
    public void mark( );
    public void unmark( );
}

class FibQueue<K extends Comparable<K>, V>
{
    // (Pointer to) a tree with the smallest key.
    protected FibNode<K,V> min;

    // Number of entries in the priority queue.
    protected int currentSize;

    .....

    // Returns the number of entries whose key is the smallest key
    // that are in a tree root of the Fibonacci queue.
    // If the Fibonacci queue is empty, the returned value is zero.
    protected int numberOfMinRoots( );
}
```

Implemente o método *numberOfMinRoots* (na classe interna *FibQueue*) e calcule a complexidade temporal do seu algoritmo, no melhor caso e no pior caso, justificando.

4. [4 valores] Sejam G um grafo não orientado e pesado, cujos arcos têm custo positivo, e k um inteiro positivo. Diz-se que G *tem um circuito mais longo que* k se, e só se, existir um circuito simples em G cujo comprimento pesado é superior a k.

   Por exemplo, o grafo esquematizado na figura tem um circuito mais longo que 24.



   Circuito simples cujo comprimento pesado é 26:

   `1 2 7 3 4 9 6 1`

   O **Problema do Circuito Mais Longo** formula-se da seguinte forma.
   Dados um grafo G não orientado e pesado, cujos arcos têm custo positivo, e um inteiro positivo k, G tem um circuito mais longo que k?

   Prove que o Problema do Circuito Mais Longo é NP-completo.

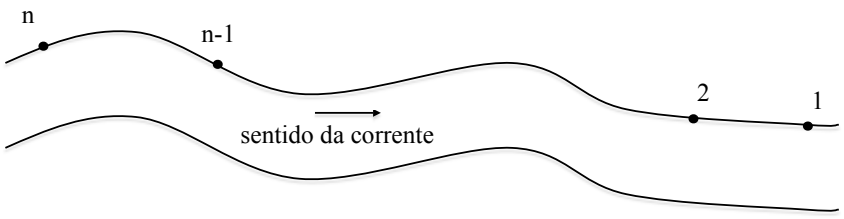
5. [3 valores] Considere a seguinte função recursiva $F_X(i, v)$, onde $X = (e_1, e_2, \ldots, e_n)$ é uma sequência não vazia de números inteiros positivos, i é um inteiro entre 1 e n, e v é um inteiro não negativo. (A expressão $\lfloor x/y \rfloor$ denota a divisão inteira de x por y.)

$$F_X(i,v) = \begin{cases} v, & \text{se } i = 1 \text{ ou } v = 0; \\ F_X(i-1, v), & \text{se } i > 1 \text{ e } e_i > v; \\ \min\limits_{0 \le k \le \lfloor v/e_i \rfloor} \Big(\; k + F_X(i-1, v - k\,e_i) \; \Big), & \text{se } i > 1 \text{ e } e_i \le v; \end{cases}$$

   Apresente um algoritmo, desenhado segundo a técnica da programação dinâmica, que, dados uma sequência não vazia $X = (e_1, e_2, \ldots, e_n)$ de inteiros positivos e um inteiro positivo m, calcula o valor da função $F_X(n, m)$. Estude (justificando) a complexidade temporal e espacial do seu algoritmo, no pior caso.

(Continue, porque o exame tem mais uma pergunta.)

6. [3 valores] Suponha que existem n cais de aluguer de canoas ao longo de um rio, identificados pelos números n, $n-1$, ..., 2, 1, como se ilustra na figura. Pode-se alugar uma canoa em qualquer um dos cais (excepto no número 1), devolvendo-a num dos cais rio abaixo. É impossível remar rio acima porque a corrente é muito forte.



Considere que existe uma tabela P com os preços do aluguer de uma canoa entre qualquer cais de partida x e qualquer cais de chegada y (para $n \geq x > y \geq 1$). Ao contrário do que seria de esperar, podem existir cais x e y para os quais o custo da viagem directa entre x e y (envolvendo um só aluguer) é superior ao custo da viagem entre x e y com escalas (envolvendo uma série de alugueres):

$$P[x, y] > P[x, c_1] + P[c_1, c_2] + \cdots + P[c_{k-1}, c_k] + P[c_k, y],$$

onde $k \geq 1$ e $n \geq x > c_1 > c_2 > \cdots > c_{k-1} > c_k > y \geq 1$.

Apresente **uma função recursiva** que, dados o número n de cais existentes ao longo do rio (com $n \geq 2$) e a tabela P com os preços de aluguer de uma canoa entre os vários cais, calcula o custo mínimo de uma viagem de canoa desde o cais n até ao cais 1.