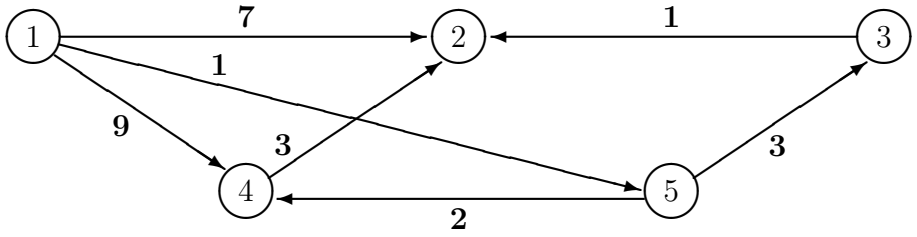
Exame de Análise e Desenho de Algoritmos

Departamento de Informática
Universidade Nova de Lisboa
20 de Junho de 2009

1. [3 valores] Suponha que se executa o algoritmo de Dijkstra com o grafo esquematizado na figura e o vértice origem 1.



   Indique:

   - a ordem pela qual os vértices são seleccionados;
   - o número total de vezes que o método *decreaseKey* é executado; e
   - o conteúdo dos vectores *length* e *via*, quando o algoritmo termina.

2. [4 valores] Considere um grafo não orientado, pesado e conexo, que modela uma rede de saneamento básico num concelho. Cada ligação entre dois nós distintos da rede é efectuada através de um cano, cuja capacidade se mede em metros cúbicos por segundo.

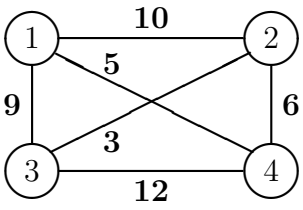
   Embora quase todos os canos estejam a precisar de ser substituídos, numa primeira fase, a Câmara pretende intervir em apenas alguns. O conjunto de canos a substituir (por canos novos, com a mesma capacidade dos existentes) deve satisfazer os três seguintes requisitos:

   1. permitir que quaisquer dois nós da rede fiquem inter-ligados pelos novos canos, ainda que essa "inter-ligação" obrigue a passar por outros nós;
   2. ter o menor número de elementos possível
      (quanto menos canos a substituir, menor o custo da obra);
   3. maximizar a soma das capacidades dos novos canos
      (ou seja, de entre todos os conjuntos de canos que satisfazem as duas restrições anteriores, pretende-se um cuja soma das capacidades dos canos seja a maior possível).

   Escreva um algoritmo (em pseudo-código) que, dado um grafo não orientado, pesado e conexo, que modela uma rede de saneamento básico, indique a soma das capacidades de um conjunto de canos que satisfaz os requisitos da Câmara. Estude (justificando) a complexidade temporal do seu algoritmo, no pior caso.

3. [4 valores] Sejam G um grafo não orientado, pesado e completo, cujos arcos têm custo positivo, e M um inteiro positivo. Um *circuito de Hamilton limitado por M* em G é um circuito de Hamilton em G que passa apenas por arcos de custo inferior ou igual a M.

   Por exemplo, 1 4 2 3 1 é um circuito de Hamilton limitado por 9 no grafo esquematizado na figura.



   O **Problema do Caixeiro Viajante Limitado** formula-se da seguinte forma.
   Dados um grafo G não orientado, pesado e completo, cujos arcos têm custo positivo, e um inteiro positivo M, existe um circuito de Hamilton limitado por M em G?

   Prove que o Problema do Caixeiro Viajante Limitado é NP-completo.

4. [3 valores] Considere a seguinte função recursiva $f(i,j)$, onde i e j são números inteiros que verificam $i \geq 1$ e $j \geq 3$.

$$f(i,j) = \begin{cases} 1, & \text{se } i = 1 \text{ e } j \geq 3; \\ 2\,f(i-1,j) + 1, & \text{se } i \geq 2 \text{ e } j = 3; \\ \min\limits_{1 \leq k < i} \Big(\; 2\,f(k,j) + f(i-k,j-1) \; \Big), & \text{se } i \geq 2 \text{ e } j \geq 4. \end{cases}$$

   Apresente um algoritmo, desenhado segundo a técnica da programação dinâmica, que, dados dois números inteiros positivos D e E, com $E \geq 3$, calcula o valor da função $f(D,E)$. Estude (justificando) a complexidade temporal e espacial do seu algoritmo, no melhor caso, no pior caso e no caso esperado.

(Continue, porque o exame tem mais duas perguntas.)

5. [3 valores] Considere a classe *BinNode*, dos nós das árvores binomiais de chaves do tipo K e valores do tipo V, e a classe *BinQueue*, das filas binomiais de chaves do tipo K e valores do tipo V, ambas internas ao pacote *dataStructures*.

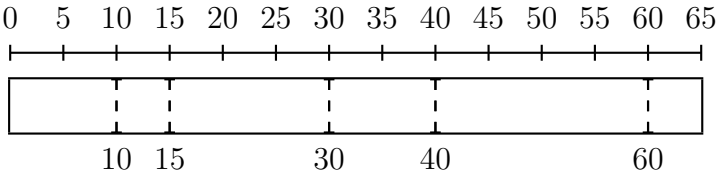
```
class BinNode<K,V>
{
    ..........................................
    public BinNode( K key, V value );
    public Entry<K,V> getEntry( );
    public K getKey( );
    public V getValue( );
    public int getDegree( );
    public BinNode<K,V> getChild( );
    public BinNode<K,V> getLeftSibling( );
    public BinNode<K,V> getRightSibling( );
    public BinNode<K,V> getParent( );
    public void setEntry( Entry<K,V> newEntry );
    public void setEntry( K newKey, V newValue );
    public void setKey( K newKey );
    public void setValue( V newValue );
    public void setDegree( int newDegree );
    public void incrementDegree( );
    public void setChild( BinNode<K,V> newChild );
    public void setLeftSibling( BinNode<K,V> newLeftSibling );
    public void setRightSibling( BinNode<K,V> newRightSibling );
    public void setParent( BinNode<K,V> newParent );
}

class BinQueue<K extends Comparable<K>, V>
{
    // (Pointer to) the first binomial tree.
    protected BinNode<K,V> head;

    ..........................................
    // Returns the number of entries in the binomial queue.
    public int size( );
}
```

Implemente o método *size* (na classe *BinQueue*), que retorna o número de entradas na fila binomial. Note que *head* é o único atributo. Calcule a complexidade temporal do seu algoritmo, no melhor caso e no pior caso, justificando.

6. [3 valores] Um carpinteiro tem uma tábua de madeira, com um determinado comprimento, que tem de ser cortada nas posições p_1, p_2, ..., p_n (com $n \geq 1$), onde p_i é a distância ao extremo esquerdo da tábua (como se ilustra na figura).



Note que, depois de fazer o primeiro corte, o carpinteiro fica com duas tábuas de madeira; depois do segundo corte, o carpinteiro fica com três tábuas de madeira, etc.

Assuma que o custo de efectuar um corte numa tábua de madeira de comprimento c é igual a c, independentemente da posição onde o corte é efectuado. Por exemplo, o custo de efectuar um corte qualquer na tábua da figura é 65.

Apresente **uma função recursiva** que, dados o comprimento C da tábua de madeira e a sequência p_1, p_2, ..., p_n das posições onde devem ser efectuados os cortes (com $n \geq 1$ e $0 < p_1 < p_2 < \cdots < p_n < C$), calcula o custo mínimo de efectuar todos os n cortes. Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial. **Sugestão:** Considere a sequência p_0, p_1, p_2, ..., p_n, p_{n+1}, onde $p_0 = 0$ e $p_{n+1} = C$.