2015-16 Recurso

1

- a)

```
int preco[n];

int findMaxDiff(int begin, int end){
    if (end - begin == 1)
        return abs(preco[end] - preco[begin]);
    int medium = (end + begin) / 2;
    int m1 = findMaxDiff(begin, medium);
    int m2 = findMaxDiff(medium, end);
    int best = max_value(medium, right) - min_value(begin, end);
    return max(best, m1, m2);
}
```

O(nlog(n))

- b) Source

```
/* The function assumes that there are
   at least two elements in precoay. The
   function returns a negative value if the
   precoay is sorted in decreasing order and
   returns 0 if elements are equal */
int maxDiff(int preco[], int n)
{
    // Create a diff precoay of size n-1.
    // The precoay will hold the difference
    // of adjacent elements
    int diff[n-1];
    for (int i=0; i < n-1; i++)
        diff[i] = preco[i+1] - preco[i];

    // Now find the maximum sum
    // subprecoay in diff precoay
    int max_diff = diff[0];
    for (int i=1; i<n-1; i++) {
        if (diff[i-1] > 0)
            diff[i] += diff[i-1];
        if (max_diff < diff[i])
            max_diff = diff[i];
    }
    return max_diff;
}
```

O(n)

2

- a)

	A	B	C	D	E	F
A	0	7		2		
D	0	5	10	2	6	
B	0	5	7	2	6	
C	0	5	7	2	6	12
E	0	5	7	2	6	12
F	0	5	7	2	6	12

- b)

	A	B	C	D	E	F
A	0	10		10		
B	0	10	30	10		
D	0	10	30	10	32	
C	0	10	30	10	32	62
E	0	10	30	10	32	57

- c)

Ordenação topológica não tem em conta distâncias entre vértices mas sim o número de arestas (baseado em DFS). Caso as arestas tivessem tamanho unitário seria possível considerar distâncias para a ordenação topológica.

3

- a) Árvore de exapansão mínima resulta nas seguintes ligações: F-A; A-B; F-C; A-D; B-E;
- b) A árvore de expansão mínima garante que todos os vértices se encontram conectados de forma mínima. Para garantir que o vértice C está conectado tem que se optar ou por C-F ou C-D. Uma vez que C-F tem menor peso é a aresta escolhida.
- c) Falso. Entre B e J o caminho mais curto na árvore original teria um peso de 17. Na árvore de expansão mínima resultaria B-E-G-J com maior peso.

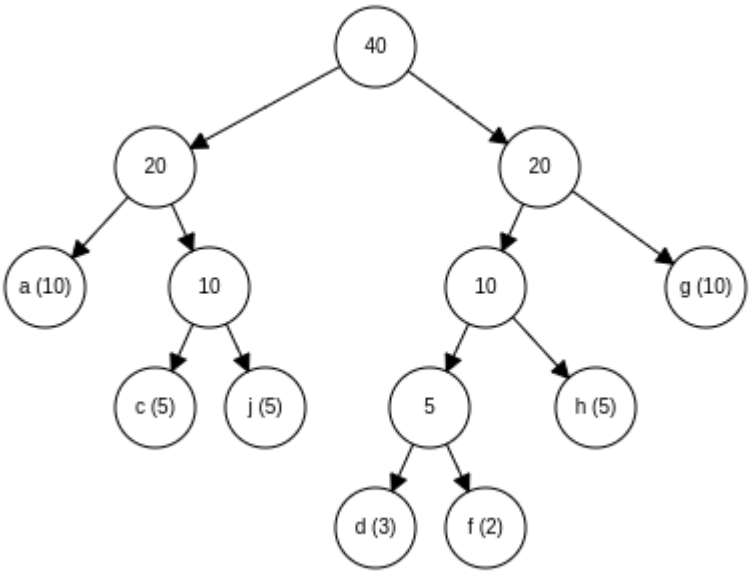
4

- a)

- b) [s1, a, b, c, d], [t1, t2, t3]
- c) Fluxo alteráva-se de 13 para 11, uma vez que já só podem chegar 4 unidades a t2.

5

- a) Possui nove elementos distintos, pelo que cada elemento terá que ser representado no mínimo com 4 bits. Freqência total = 40, resultando 4*40 = 160 bits.
- b) Seguindo ordem alfabética para representar cada elemento:



- c) 1000 1001 00 011 01

(1000) -> d = f[3] (1001) -> f = f[5] (00) -> a = f[0] (011) -> j = f[9] (01) -> Não tem elemento.

De acordo com a árvore apresentada não é possível encontrar uma sequência compatível.

Caso se troque a(10) com 10: (1000) -> d = f[3] (1001) -> f = f[5] (000) -> c = f[2] (11) -> g = f[6] (01) -> a = f[0].

6

- a) É possível monotorizar todas as ligações com um número de routers = k?
- b) O problema é NP-Completo (logo não resolúvel em tempo polinomial), pois:
    - É NP, pois é possível verificar se todas as ligações são monitorizadas em tempo polinomial.
    - É NP-difícil, pois o problema é redutível em tempo polinomial ao problema da Cobertura de Vértices:
        - Dado um grafo G=(V, E) tendo V como routers e E como ligações pretende-se encontrar um subconjunto de routers(W) contido em V tal que para toda a ligação/aresta {i,j} pertencente ao conjunto de arestas, ou o router i ou o router j pertencem ao conjunto W