Estruturas de Dados e Algoritmos II
2ª Frequência e Exame

Departamento de Informática
Universidade de Évora

12 de Junho de 2015

Os símbolos à esquerda de cada pergunta identificam a prova ou provas a que ela pertence:
♣ assinala as perguntas do exame; ♢ assinala as perguntas da 2ª frequência.

♣ **1.** [2,5 valores] Assumindo que o alfabeto consiste nas 26 letras minúsculas, desenhe uma *trie* cujo conteúdo sejam as cinco palavras

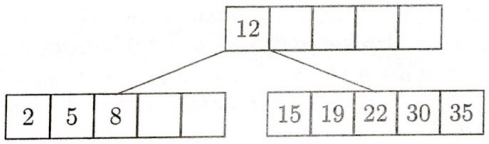
marco parco meta metade meia

Qual seria a memória ocupada por uma implementação em C da *trie* que desenhou, numa máquina com endereços e palavras de 32 bits? (Não precisa de calcular o valor, mas apresente e justifique todos os cálculos efectuados ou a efectuar.)

♣ **2.** [3 valores] A *B-tree* da figura tem grau de ramificação mínimo 3. Apresente o seu estado depois da execução de *cada uma* das operações da sequência

i 20 r 12 r 2 i 38 r 15 i 15 r 30 r 35

pela ordem apresentada. As letras i e r indicam, respectivamente, a inserção e a remoção do elemento que se lhes segue.

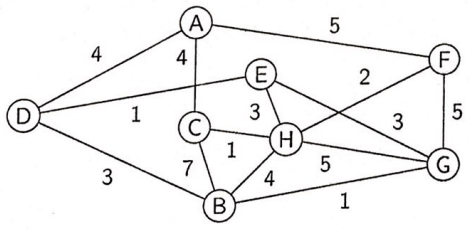


♣ **3.** [2 valores] Escreva o (pseudo-)código da função B-Tree-Max-At-Depth(x, d), que devolve o maior elemento, à profundidade d, na *B-tree* não vazia cuja raiz é o nó x. Analise a complexidade temporal da sua função no pior caso.

Considere que a profundidade da raiz é zero. Se *B-tree* não tiver nós à profundidade d, a função deverá terminar executando a instrução throw INVALID-DEPTH.

Considere que os nós de uma *B-tree* têm os campos n (ocupação), c (filhos), key (elementos) e leaf (é folha?).

♣ **4.** Seja G_4 o grafo seguinte:



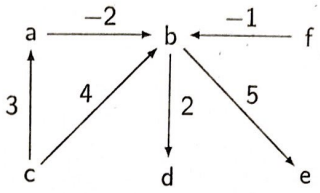
(a) [1 valor] Diga, justificando se G_4 é conexo.

(b) [1,5 valores] Apresente uma árvore de cobertura mínima para G_4. Qual o peso dessa árvore?

♣ ◇ **5.** Seja DAG o algoritmo seguinte, onde $G = (V, E)$ é um grafo pesado orientado, w é a função de peso do grafo e S é uma ordenação topológica dos nós de G, e seja G_5 o grafo da figura.

```
DAG(G, w, S)
1.  n ← |G.V|
2.  let x be uma matriz com dimensões
    [1..n,1..n]
3.  foreach nó u ∈ G.V do
4.      x[u, u] ← 0
5.      foreach nó v ∈ G.V − {u} do
6.          x[u, v] ← ∞
7.  foreach nó u ∈ G.V pela ordem S do
8.      foreach arco (u,v) ∈ G.E do
9.          x[u,v] ← w(u,v)
10.         foreach nó t < u na ordem S do
11.             if x[t,u] + x[u,v] < x[t,v] then
12.                 x[t,v] ← x[t,u] + x[u,v]
13. return x
```

V
E



Grafo G_5

(a) [1 valor] Apresente uma ordenação topológica para os nós de G_5.

(b) [1,5 valores] Sendo o grafo representado através de listas de adjacências, calcule a complexidade temporal do algoritmo DAG no pior caso.

(Assuma que a selecção dos nós feita nas linhas 7 e 10 tem custo constante.)

♣ ◇ **6.** [2,5 valores] Nas redes de computadores do tipo da Internet, cada rede local (por exemplo, a rede do CLV ou a rede do CES) inclui um *router*, responsável por encaminhar os pacotes enviados por uma máquina local para o exterior e os dirigidos do exterior a uma máquina local. Cada *router* está ligado a um ou mais *routers* e, quando recebe um pacote destinado ao exterior, envia-o para um desses *routers*. O *router* escolhido, por sua vez, envia-o para um dos *routers* seus vizinhos, que o enviará para um vizinho, e assim sucessivamente até o pacote chegar ao seu destino. Cada *router* sabe qual o tempo que um pacote demora a chegar a qualquer dos seus vizinhos e, dado o destino de um pacote, para qual (ou quais) dos seus vizinhos o pode encaminhar.

Uma maneira de decidir qual o vizinho a usar, quando há vários possíveis, é escolher aquele a que o pacote demora menos tempo a chegar. Esta técnica não garante, no entanto, que o pacote siga o caminho mais rápido até ao destino final.

Se um *router* possuísse a informação completa sobre quais os *routers* existentes na rede, as ligações entre eles e o tempo que demora um pacote a percorrer cada uma das ligações existentes, ele poderia decidir qual o vizinho a quem enviar um pacote, de modo a garantir que ele seguia o caminho mais rápido.

Como é que poderia ser implementar isso? Descreva detalhadamente como modelaria os dados e como os representaria, e diga que algoritmo(s) utilizaria, porquê, e como obteria o resultado pretendido.

Note que o tempo de um pacote entre dois *routers* vizinhos pode não ser o mesmo nos dois sentidos e que pode haver ligações em que os pacotes só transitam num dos sentidos.