Estruturas de Dados e Algoritmos II
2ª Frequência e Exame

Departamento de Informática
Universidade de Évora

8 de Junho de 2018

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 8 letras maiúsculas $\{A, B, C, D, E, F, G, H\}$, desenhe uma *trie* cujo conteúdo sejam as cinco palavras:

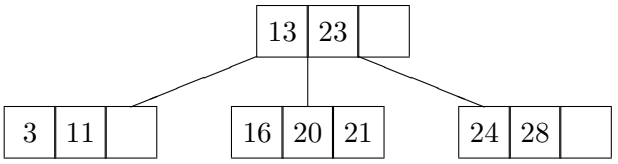
ADAGA BEBA BEBE DADA DA

Qual seria a memória ocupada por uma implementação em C da *trie* que desenhou, numa máquina com endereços de 64 bits e palavras de 32 bits? (Não precisa de calcular o valor, mas apresente e justifique todos os cálculos efectuados ou a efectuar.)

♣ **2.** [2,5 valores] A *B-tree* da figura tem grau de ramificação mínimo 2. Apresente o seu estado depois da execução de *cada uma* das operações seguintes, *em sequência*, pela ordem apresentada:

i 25 **r** 16 **r** 13 **i** 26 **i** 18 **i** 19 **r** 24 **r** 28

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



♣ **3.** [2,5 valores] Considere a função recursiva $A_{XY}[i, j]$, onde:

- $X = (x_1\ x_2\ \ldots\ x_m)$ é uma sequência não vazia;
- $Y = (y_1\ y_2\ \ldots\ y_n)$ é uma sequência não vazia;
- $0 \leq i \leq m$; e
- $0 \leq j \leq n$.

$$A_{XY}[i,j] = \begin{cases} 2j & \text{se } i = 0 \wedge j \geq 0 \\ 2i & \text{se } i > 0 \wedge j = 0 \\ A_{XY}[i-1, j-1] - 1 & \text{se } i > 0 \wedge j > 0 \wedge x_i = y_j \\ \min\left\{\ A_{XY}[i-1, j-1] + 1,\ A_{XY}[i, j-1] + 2\ \right\} & \text{se } i > 0 \wedge j > 0 \wedge x_i \neq y_j \end{cases}$$

Apresente o pseudo-código de um algoritmo iterativo que, dadas duas sequências não vazias $X = (x_1\ x_2\ \ldots\ x_m)$ e $Y = (y_1\ y_2\ \ldots\ y_n)$, calcula e devolve o valor de $A_{XY}[m, n]$.

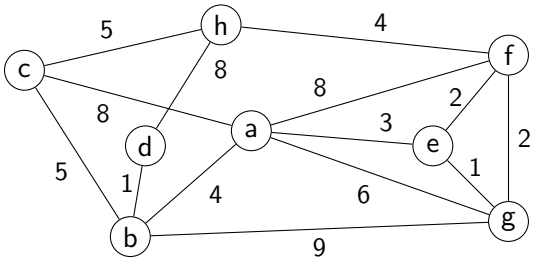
♣ **4.** [2,5 valores] Um palíndromo é uma sequência de símbolos que se lê da mesma forma da esquerda para a direita e da direita para a esquerda. Uma *subsequência palíndroma* é uma subsequência de uma sequência de símbolos que é um palíndromo. Por exemplo, arara é um palíndromo e contém várias subsequências palíndromas, como ara, aaa, rr, a e arara (a maior). Por outro lado, elefante não é um palíndromo mas contém várias subsequências palíndromas, como eee, ee, efe e t. (Neste caso, as maiores subsequências palíndromas têm comprimento 3, não havendo uma que seja maior do que qualquer outra.)

Apresente *uma função recursiva* que, dada uma sequência não vazia $S = (s_1\ s_2\ \ldots\ s_n)$, calcula o comprimento de uma maior subsequência palíndroma. Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial. (Note que não é pedido que escreva código.)

♢ **5.** [3 valores] Apresente *um* grafo orientado, não pesado, com as seguintes características:

i. Contém um caminho simples com comprimento 4;
ii. Contém um ciclo simples com comprimento 3; e
iii. É constituído por duas componentes fortemente conexas.

♣ ♢ **6.** Seja G_6 o grafo seguinte:



(a) [1,5 valores] Apresente uma árvore de cobertura mínima para G_6.

(b) [2 valores] Apresente uma ordem pela qual os arcos poderiam ser acrescentados a uma árvore de cobertura mínima construída através uma aplicação do algoritmo de Prim, a partir do vértice a.

(c) [1,5 valores] Diga, justificando, que algoritmos poderia aplicar para calcular os caminhos (pesados) mais curtos neste grafo, a partir de um vértice dado.

(d) [2 valores] Apresente a ordem pela qual os vértices do grafo poderiam ser retirados da fila com prioridade, durante uma aplicação do algoritmo de Dijkstra, a partir do vértice b.

♢ **7.** O algoritmo REACHABLE, cujo pseudo-código é mostrado na figura, calcula o conjunto dos nós do grafo $G = (V, E)$ a que é possível chegar a partir do nó s.

```
REACHABLE(G, s)
 1.  for each nó u ∈ G.V do
 2.      u.colour ← WHITE
 3.  S ← ∅                              // conjunto
 4.  P ← ∅                              // pilha
 5.  s.colour ← GREY
 6.  PUSH(P, s)                         // empilha s
 7.  while P ≠ ∅ do
 8.      u ← POP(P)                     // desempilha um nó
 9.      S ← S ∪ { u }
10.      for each nó v ∈ G.adj[u] do    // explora o arco (u, v)
11.          if v.colour = WHITE then
12.              v.colour ← GREY
13.              PUSH(P, v)             // empilha
14.      u.colour ← BLACK
15.  return S
```

(a) [2,5 valores] Identifique o melhor e o pior casos do comportamento do algoritmo, dizendo que características poderão ter G e s para que cada um deles se dê, e explicando brevemente.

(b) [2,5 valores] Analise a complexidade temporal do algoritmo no pior caso, assumindo que o grafo está implementado através de uma matriz de adjacências e que as operações que envolvem o conjunto S e a pilha P têm custo constante.

♢ **8.** [2 valores] Durante a aplicação do algoritmo de Floyd-Warshall, podem aparecer valores diferentes de 0 na diagonal principal da uma matriz com os pesos dos caminhos mais curtos? Qual a razão porque não podem ou, podendo, que outros valores lá podem aparecer e qual o seu significado?

♣ ♢ **9.** [3 valores] O responsável de uma rede informática, de grande dimensão, quer garantir que um pacote, que saia de qualquer uma das máquinas da rede, não precisa de percorrer mais do que 8 *hops* para chegar a qualquer outra máquina da rede. (Um *hop* corresponde a um *router* por onde um pacote passa. Por exemplo, se um pacote que saia da máquina A precisar de passar por 2 *routers* para chegar à máquina B, então o pacote percorre dois *hops* no caminho de A para B.)

Para verificar se a situação é a pretendida, é necessário calcular o número máximo de *hops* que um pacote precisa de percorrer para chegar de alguma máquina da rede a uma outra. A informação disponível consiste nas ligações existentes entre máquinas e *routers*, e entre *routers*. Estas ligações são ponto a ponto e bidireccionais. Sabe-se, também, que qualquer máquina da rede consegue comunicar com todas as outras máquinas da rede.

Como poderia calcular aquele valor? Descreva detalhadamente como modelaria os dados e como os representaria, e diga que algoritmo(s) utilizaria, porquê, e como obteria o resultado pretendido.