Estruturas de Dados e Algoritmos II
1ª Frequência

Departamento de Informática
Universidade de Évora

13 de Abril de 2018

1. [4 valores] Assumindo que o alfabeto consiste nas 26 letras minúsculas, desenhe uma *trie* cujo conteúdo sejam as seis palavras

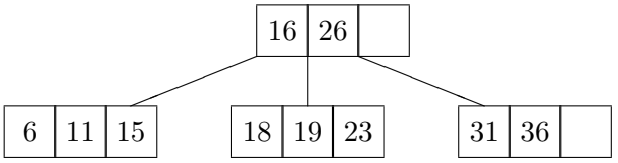
zero uma um dois duas doze

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

2. [4 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 34 **r** 23 **r** 16 **i** 27 **i** 33 **i** 28 **r** 11

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



3. [3 valores] Apresente o pseudo-código da função B-Tree-Elements(x), que calcula e devolve o número de elementos na *B-tree* cuja raiz é o nó x (que já está em memória).

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

4. [2 valores] Uma alternativa para a remoção de um elemento de um nó interno de uma *B-tree* seria associar uma *flag* a cada elemento de um nó, que indicaria se esse elemento pertencia ou não à árvore. Para remover um elemento, bastaria atribuir à *flag* o valor que indicava que o elemento já não se encontrava na árvore, deixando de ser necessário descer até a uma folha para encontrar um elemento para substituir o elemento removido.

Posteriormente, durante a inserção de um elemento que ainda não se encontrasse na *B-tree*, se fosse necessário descer para o filho à esquerda ou o filho à direita de um elemento removido, o novo elemento poderia ocupar o seu lugar, deixando, mais uma vez, de ser necessário descer até uma folha da árvore.

Diga, justificando, se esta seria uma alternativa viável para o remoção de um elemento que se encontra num nó interno de uma *B-tree*, analisando, nomeadamente, o funcionamento da pesquisa numa árvore com estas características.

(continua...)

5. [3,5 valores] Considere a função recursiva $P[i]$, onde i é um inteiro positivo:

$$P[i] = \begin{cases} 1 & \text{se } i = 1 \\ \sum_{k=1}^{i-1} P[k]\, P[i-k] & \text{se } i \geq 2 \end{cases}$$

Apresente o pseudo-código de um algoritmo iterativo que, dado $n \geq 1$, calcula o valor de $P[n]$. Estude (justificando) a complexidade temporal do seu algoritmo.

6. [3,5 valores] Um empresário tem um determinado orçamento para abastecer a sua loja.

Os produtos a comercializar são escolhidos de um catálogo onde, para cada produto, é indicado o custo de cada embalagem e o lucro que uma embalagem permite obter. Como é natural, o empresário pretende escolher os produtos que, com o orçamento disponível, maximizarão o seu lucro.

Para exemplificar, suponha que o catálogo contém os produtos indicados na tabela abaixo. Se o orçamento do empresário for 375, uma escolha de produtos que lhe proporciona o maior lucro consiste em uma embalagem de cada um dos produtos 2, 4 e 5, que leva a um lucro de 190. Se o orçamento for 500, o maior lucro que ele pode obter é 260, adquirindo duas embalagens do produto 2.

Produto	1	2	3	4	5	6
Preço (P)	60	250	50	100	25	150
Lucro (L)	24	130	22	50	10	78

Apresente *uma função recursiva* que, dados o valor inteiro (positivo) v do orçamento e duas sequências de inteiros positivos,

$$P = p_1, p_2, \ldots, p_n \;\text{ e }\; L = l_1, l_2, \ldots, l_n, \;\text{ para algum } n \geq 1,$$

com, respectivamente, os preços de cada embalagem de cada produto e os lucros por embalagem, calcula o maior lucro que o empresário pode obter com aquele orçamento.

Indique claramente o que representa cada uma das variáveis que utilizar e explicite a chamada inicial. (Note que não é pedido que escreva código.)