Universidade Federal de Pernambuco
Centro de Informática
Bach. em Engenharia da Computação

IF672ec Algoritmos e Estr. de Dados 2016.1
Prof. Paulo Fonseca

Prova Final — 14 de Julho de 2016 (Dia da Bastilha)

QUESTÃO 1 (2,0pt)

Ilustre a tabela de dispersão de tamanho $m = 11$ após a inserção das chaves

$$63,\ 125,\ 85,\ 112, 58,$$

nesta ordem. A tabela armazena no máximo um elemento por posição, e emprega a técnica de *double hashing* com as funções de dispersão

$$h_0(k) = k \bmod m \quad \text{e} \quad h_1(k) = 7 - (k \bmod 7).$$

QUESTÃO 2 (2,0pt)

Seja T a árvore AVL cuja enumeração dos elementos em *pós-ordem* é $10,\ 40,\ 30,\ 20$. Ilustre a inserção das chaves 50 e 35 em T, nesta ordem.

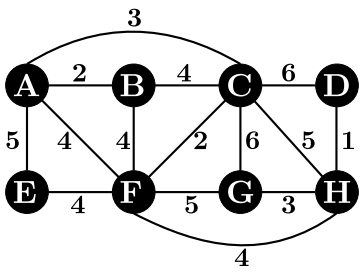
QUESTÃO 3 (2,0pt)

O Algoritmo de Kruskal é um algoritmo guloso para encontrar uma árvore geradora de custo mínimo (MST) de um grafo ponderado que faz uso da estrutura de dados *union-find*.

Algoritmo *Kruskal*
Entrada $G = (V, E)$ grafo ponderado.
Saída T uma MST de G

```
1 C ← union-find vazio
2 para cada u ∈ V faça
3     make-set(C, u)
4 T ← (V, E' = ∅) ▷ MST inicialmente sem arestas
5 para cada aresta (u, v, w) de E em ordem crescente de peso faça
6     se find(C, u) ≠ find(C, v) então
7         E' ← E' ∪ {(u, v, w)} ▷ adiciona a aresta à MST
8         union(C, u, v)
9 devolva T
```

Considerando a execução do Algoritmo Kruskal sobre o grafo



pede-se:

a) Represente graficamente a MST T encontrada. *Obs.:* Em caso de arestas com o mesmo peso, deve-se considerar a ordem alfabética das suas extremidades. Por exemplo, a aresta $(\texttt{A}, \texttt{B}, 2)$ deve ser considerada antes de $(\texttt{C}, \texttt{F}, 2)$.

b) Represente graficamente a estrutura *union-find* C ao final da execução do algoritmo, considerando o emprego das heurísticas de *união ponderada* e *compressão de caminhos*.

■ QUESTÃO 4 (2,0pt)

A matriz abaixo corresponde à tabela de programação dinâmica do algoritmo Floyd-Warshall após a execução da penúltima iteração sobre o grafo da questão 3.

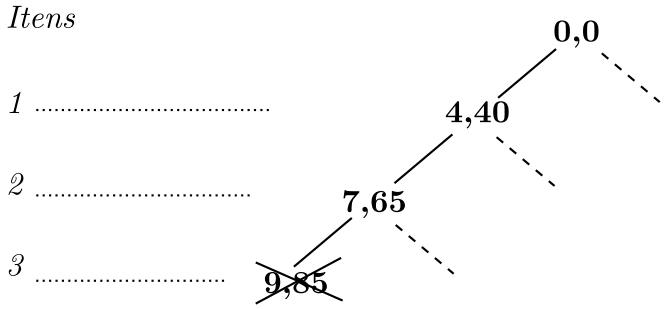
	A	B	C	D	E	F	G	H
A	0	2	3	9	5	4	9	8
B	2	0	4	10	7	4	9	8
C	3	4	0	6	6	2	7	5
D	9	10	6	0	12	8	13	1
E	5	7	6	12	0	4	9	8
F	4	4	2	8	4	0	5	4
G	9	9	7	13	9	5	0	3
H	8	8	5	1	8	4	3	0

Exiba a matriz após a execução da próxima (e última) iteração.

■ QUESTÃO 5 (2,0pt)

O algoritmo de busca exaustiva para o problema da mochila sem repetição pode ser estendido para um algoritmo *branch&bound* da seguinte maneira. Quando estiver num nó de profundidade i da árvore de busca, no qual se considera incluir ou não o item i, além do peso e do valor acumulado já conhecidos, uma cota superior para o valor máximo transportado pode ser obtida adicionando-se o valor de todos os itens restantes $v_{i+1}, \ldots, v_n$ ao valor acumulado correspondente ao nó corrente. Se o peso acumulado excede a capacidade da mochila ou se a cota superior não superar a melhor solução já conhecida, o algoritmo retrocede e abandona o ramo. Nesse algoritmo, a cada passo deve-se tentar primeiro estender a solução parcial do nó corrente incluindo o próximo item, e depois não incluindo. Complete a árvore abaixo correspondente à execução desse algoritmo sobre a entrada

Item	1	2	3	4	5
Peso (w)	4	3	2	1	2
Valor (v)	40	25	20	10	15



Em cada nó da árvore estão representados o peso e o valor acumulados correspondentes à solução parcial representada por ele.

■ ■ ■