Algoritmos e Complexidade – 7 de Janeiro de 2016 – Duração: 90 min

Parte A

1. Relembre a definição de uma *min-heap* armazenada num array e a função `void bubbledown (int heap[], int N, int i)` que recebe um array `heap` com `N` elementos e faz o *bubble-down* do elemento que está no índice `i`. Considere agora a função que transforma um array arbitrário numa *min-heap*.

```
void heapify (int v[], int N){
  int i;
  for (i=(N-2)/2; i>=0; i--)
    bubbledown (v,N,i);
}
```

   (a) Diga qual o resultado da função `heapify` quando invocada com um array `v` com os seguintes 7 elementos por esta ordem: {`50, 20, 42, 13, 2, 12, 36`}. Justifique a sua resposta apresentando o estado do array no final de cada iteração do ciclo.

   (b) Note que a função `bubbledown` só é aplicada a metade dos elementos do array. Além disso,

   - para 1/4 dos elementos (a metade superior dessa metade) a função `bubbledown` faz no máximo 1 iteração.
   - para 1/8 dos elementos a função `bubbledown` faz no máximo 2 iterações, e assim sucessivamente.

   Com base nestas observações, analise a complexidade da função `heapify` apresentada.

2. Um grafo $G = (V, E)$ diz-se **bi-partido** sse é possível determinar um sub-conjunto V_1 tal que para cada aresta $(a, b) \in E$, **exactamente uma** das extremidades (a ou b) pertence a V_1.

   Defina uma função `int bipartido (Graph g, int V1[])` que testa se um grafo é bipartido. A função deverá retornar 1 se tal acontecer e 0 no outro caso. Em caso afirmativo, a função deverá ainda preencher o array `V1` com 1 ou 0 de forma a que o conjunto representado por esse array demonstre que o grafo é bi-partido (`V1[i]=1` significará que `i` pertence ao conjunto representado por `V1`).
   Sugestão: Faça uma ou mais travessias de forma a visitar a totalidade dos vértices. Para cada vértice visitado teste se é possível inclui-lo no conjunto `V1`.

```
#define NV 1000
typedef struct edge {
  int dest;
  int cost;
  struct edge *next;
} *Graph [NV];
```

3. Considere que ao executar o algoritmo de Warshall (*all-pairs shortest paths*) sobre um grafo G com 5 vértices, as matrizes de pesos (p) e caminhos (c) produzidas são as que se apresentam ao lado. Responda, justifiicando, às seguintes questões sobre o grafo G.

   (a) Qual o caminho mais curto (i.e., a sequência de vértices) que liga o vértice 0 ao vértice 3?

   (b) Qual o caminho mais curto (i.e., a sequência de vértices) entre os vértices mais distantes entre si?

   (c) Será o grafo cíclico?

$c =$

	0	1	2	3	4
0	–	4	–	4	–
1	–	–	–	–	–
2	–	4	–	4	0
3	–	–	–	–	–
4	–	–	–	1	–

$p =$

	0	1	2	3	4
0	∞	3	∞	4	1
1	∞	∞	∞	1	∞
2	2	5	∞	6	3
3	∞	∞	∞	∞	∞
4	∞	2	∞	3	∞

Parte B

O problema da partição de um conjunto de inteiros consiste em, dado um conjunto X de números inteiros (positivos e negativos) determinar se existe um sub-conjunto Y de X cuja soma dos elementos seja exactamente metade da soma dos elementos de X.

Este problema é da classe NP. Demostre tal facto descrevendo um algoritmo não determinístico polinomial que o resolva (esta descrição deve indicar (1) como são codificadas as soluções propostas (resultado da fase não determinística) (2) incluir uma definição em C da fase determinística e (3) a argumentação de que cada uma destas componentes é de facto polinomial).