Algoritmos e Complexidade – 16 de Janeiro 2014 – Duração: 90 min

Parte A

1. A função `heap2ALV` apresentada ao lado transfere um conjunto de números inteiros armazenado numa *minheap* para uma árvore AVL. Considere que a função `extractMin` remove um elemento da *minheap* e retorna zero se a *heap* é vazia. Considere ainda que a função `insertTree` faz a inserção balanceada numa árvore AVL.

```
Node* heap2AVL (Heap *h) {
  int x;
  Node *t = NULL;
  while (extractMin(h, &x))
    t = insertTree (t, x);
  return t;
}
```

   (a) Considere que a função é invocada com a *minheap* $[10, 20, 30, 40, 60, 50, 70]$. Apresente visualmente o conteúdo da *heap* `h` e da árvore AVL `t` em todas as iterações do ciclo.

   (b) Analise a complexidade assimptótica desta função em função do tamanho (número de elementos) da heap argumento.

2. Considere os seguintes tipos de dados para a representação de grafos não-orientados *coloridos*, ou seja em que a cada vértice está associada uma *cor* (inteiro). Considera-se que os grafos contêm sempre `MAXV` vértices. Diz-se que um grafo está *bem colorido* se nenhum arco liga dois vértices com a mesma cor, i.e., todos os pares de vértices adjacentes são pintados com cores diferentes.

```
typedef int COLOR;
struct edge {
  int dest;
  struct edge *next;
}
typedef struct edge *Graph[MAXV];
typedef COLOR Colors[MAXV];
```

   Escreva uma função `int check_coloring(Graph g, Colors c)` tão eficiente quanto possível, que verifica se um grafo satisfaz esta definição, devolvendo um valor booleano. Efectue depois a análise do seu tempo de execução no melhor e no pior caso, dizendo quando ocorre cada caso.

3. Considere que se usa a representação acima para implementar um grafo não orientado em que cada vértice corresponde a um país e as arestas correspondem a fronteiras: existe uma aresta de a para b (e logo de b para a) sse existe uma fronteira terrestre entre os países a e b.

   Defina uma função `int Maior_cont (Graph)` que calcula o número de países do maior continente. Considere que um continente é um conjunto de países que têm fronteira terrestre e que o tamanho de um continente é o número de países desse continente.

Parte B

1. Numa implementação de tabelas de hash com tratamento de colisões por open addressing, na operação de remoção de um elemento da tabela, sempre que o número de chaves apagadas passa a ser maior ou igual a metade do tamanho do array é feita uma operação de *garbage collection* que elimina (passa a livres) todas as células apagadas. Assumindo que as operações de inserção e remoção sem garbage collection executam em tempo constante (1), e que a operação de garbage collection é linear no tamanho do array (N), mostre que o custo amortizado dessas duas operações é constante.

   Para isso calcule o custo amortizado das operações de inserção e remoção, pelo método do potencial usando como função de potencial

$$\Phi = \texttt{ocupados} + 2 * \texttt{apagados}$$