Algoritmos e Complexidade – Exame da Época Especial

10 de Setembro de 2014 – Duração: 150 min

Parte A

1. Complete a anotação do seguinte código de forma a provar que está parcialmente correcto. Apresente ainda as condições de verificação associadas a essa prova.

```
// N >= 0
i=0; j=N;
//......
while (i<N) {
  //......
  i=i+1; j=j-1;
}
// j==0
```

2. Apresente uma definição recursiva da função `int bsearch (int v[], int N, int x)` que, dado um array ordenado de `N` inteiros e um inteiro `x` calcula a posição onde esse inteiro ocorre no array (a função devolve `-1` caso o elemento não ocorra no array).

   Certifique-se que a função que definiu executa em tempo logarítmico no pior caso e apresente uma relação de recorrência que traduza a complexidade da função nesse caso.

3. Complete a seguinte definição da função `void heapify (int h[], int N)` que transforma um array de inteiros numa min-heap.

```
void heapify (int h[], int N) {
  int i = N/2;
  while (i>=0){
     ....
     i=i-1;
  }
}
```

   Sugestão: escreva o código do ciclo por forma a que tenha como invariante a seguinte propriedade:

   As sub-árvores que se iniciam em posições a partir de `i` *são min-heaps.*

4. Considere o seguinte tipo para representar as arestas de um grafo orientado e pesado (com `N` vértices):

```
typedef struct aresta {
  int destino;
  int peso;
  struct aresta *prox;
} *Grafo [N];
```

   (a) Apresente uma definição da função `int pesoC (Grafo g, int p[], int l)` que calcula o peso de um caminho `p` de comprimento `l` no grafo `g`. Assuma que a sequência de vértices dada em `p` corresponde a um caminho válido no grafo.

   (b) Relativamente à função definida na alínea anterior calcule a sua complexidade assimptótica.

   Diga justificando como essa complexidade seria alterada se o grafo estivesse armazenado numa matriz de adjacência.

Parte B

1. Relativamente à função referida na questão 2 da Parte A, defina uma versão iterativa dessa mesma função e calcule o número médio de posições consultadas no caso em que o inteiro `x` existe no array. Assuma que o elemento pode ocorrer em qualquer posição do array com igual probabilidade.

2. Considere que para implementar tabelas de hash se usa o seguinte tipo:

```
#define HSIZE 1001
typedef struct celula {
  int valor;
  int estado; // 0:Livre, 1:Ocupado, 2:Apagado
  } THash [HSIZE];
```

   Defina uma função `int addKey (THash t, int k)` de inserção de um novo valor na tabela assumindo que as colisões são resolvidas usando open-addressing com linear probing.

   A função apresentada retorna `0` em caso de sucesso e deve garantir que a tabela não tem elementos repetidos e reaproveitar células apagadas sempre que possível.

   Assuma que existe implementada uma função `int hash (int key, int hsize)` que, dado um inteiro `key` e um tamanho `hsize` devolve um inteiro na gama `0..hsize-1`.