Algoritmos e Complexidade
Época Especial
Duração: 2h

19 de Julho de 2018

1. Considere a definição de uma função que faz procura binária num array ordenado.

   (a) Apresente (informalmente, se preferir) a especificação desta função, i.e., apresente a pré e pós condições desta função.

   (b) Determine as condições de verificação necessárias para provar que esta função termina. Para isso deverá identificar um variante do ciclo em causa e apresentar as condições de verificação necessárias para mostrar que se trata efectivamente de um variante.

```
int binsearch (int a[], int N, int x){
  // PRE
  int high, low, middle, r;
  high = N; low = 0; r=-1;
  while (low < high) {
     middle = low + ((high - low)/2);
     if (a[middle] > x)
        high = middle;
     else if (a [middle] < x)
        low = middle + 1;
    else r=high=low=middle;
  }
  // POS
  return r;
}
```

   Considere a definição ao lado da função `inorder` que constrói uma lista ligada a partir da travessia inorder de uma árvore binária. Assuma que a função `concat` tem uma complexidade linear no tamanho do primeiro argumento (i.e., $T_{\texttt{concat}}(A, B) = A$ com A e B os comprimentos das listas argumento), e que a função `cons` tem complexidade constante ($T_{\texttt{cons}}(A) = 1$).

```
LInt inorder (ABin a) {
   LInt e,d,r;
   if (a==NULL) r=NULL;
   else {
      e=inorder (a->esq);
      d=inorder (a->dir);
      r = concat (e, cons(a->valor,d));
   }
   return r;
}
```

   Analise a complexidade desta função.

   Para isso identifique o melhor e pior casos e escreva e resolva recorrências que traduzam essa complexidade em cada um dos casos identificados.

2. Considere a função ao lado que determina o índice onde um determinado elemento ocorre num array não ordenado (`-1` se não existir). Calcule a complexidade média desta função (medida em termos do número de elementos do array que são consultados) **no caso de sucesso**, assumindo que o elemento pode ser encontrado com igual probabilidade em qualquer uma das posições.

```
int search(int v[], int N, int x){
   int i;
   for (i=0; i<N; i++)
      if (v[i]==x) return i;
   return -1;
}
```

3. Considere que numa tabela de Hash, implementada num array dinâmico se decide **triplicar** o tamanho da tabela sempre que o factor de carga atinge 0.75. Admitindo que uma inserção que não obriga ao redimensionamento da tabela executa em tempo constante (1) e que o redimensionamento da tabela tem um custo adicional de N (sendo N o tamanho do array), mostre que o custo amortizado da operação de inserção é constante ($\Theta(1)$).

4. Defina uma função `int quantos (int v[], int N)` que determina quantos elementos distintos tem um array. Defina e use as estruturas necessárias para que a sua solução tenha uma complexidade (esperada) linear no tamanho do array `v`.

5. Apresente a evolução de uma árvore AVL de inteiros, inicialmente vazia e onde foram inseridos (por esta ordem) os elementos:

   30, 18, 43, 25, 60, 35, 40, 20 e 41

6. Dado um grafo não orientado G com N vértices, uma coloração c com k cores (i.e., uma função que atribui a cada um dos vértices do grafo uma de k cores) diz-se uma coloração fraca do grafo sse, para cada vértice não isolado do grafo **existe pelo menos um** vértice adjacente que tem uma cor diferente da desse vértice.

   Defina uma função `int weakColor (Grafo g, int cores[], int k)` que testa se uma coloração é uma coloração fraca válida.

7. Relembre o problema de determinar os caminhos mais curtos entre todos os pares de vértices (*all-pairs shortest path*), i.e., de dado um grafo G com N vértices e pesos inteiros, preencher uma matriz $T[N][N]$ de tal forma que $T[i][j] == x$ se o caminho mais curto em G com origem i e destino j tem custo x ($T[i][j] == -1$ caso não exista nenhum caminho entre i e j).

   Suponha que tem disponível uma função `int dijkstraSP (Grapfo g, int o, int pais[], int pesos[])` que implementa o algoritmo de Dijkstra de caminho mais curto.

   Use essa função para definir a função `void AllPairs(Grafo g, int T[N][N])` que preenche a matriz `T` com os custos dos caminhos mais curtos no grafo `g`.

   Admitindo que a função `dijkstraSP` tem um custo assimptótico de $\mathcal{O}(N * (N + A))$ qual o custo assimptótico da função que definiu?