Algoritmos e Complexidade — Universidade do Minho
Exame 4 de Fevereiro de 2023

Número: _____ Nome: _____

1. Correcção (4 valores)

A função `prefSum` calcula o comprimento do **maior prefixo** de um array cuja soma não excede um determinado valor.

1. Complete a especificação abaixo fornecendo uma pós-condição apropriada
2. Defina um invariante do ciclo que permita provar a correção parcial desta função.

(The function `prefSum` computes the length of the longest prefix of an array whose sum does not exceed a given value.

1. *Complete its specification by providing a suitable post-condition*
2. *Define a loop invariant in order to prove the partial correction of this function*

)

```
int prefSum (int v[], int N, int lim){
    // pre: N >= 0 && lim >= 0
      int sum=0, i=0;
      while (i<N && sum+v[i] <= lim){
          // inv: ??
          sum = sum + v[i]; i = i+1;
      }
      // pos: ??
      return i;
}
```

2. Complexidade (4 valores)

Considere a função `fromArray` que constrói uma árvore binária equilibrada a partir de um array.
Apresente e resolva uma recorrência para calcular o número de acessos ao array efectuados por esta função.
(The function `fromArray` builds a balanced tree from an array. Express the number of accesses to the array as a recurrence relation and solve it.)

```
ABin fromArray (int v[], int N){
      ABin r=NULL; int m;
      if (N>0) {
```

```
        r = malloc (sizeof (struct abin));
        m = N/2;
        r->esq   = fromArray (v,m);
        r->valor = v[m];
        r->dir   = fromArray (v+m+1,N-m-1);
     }
     return r;
}
```

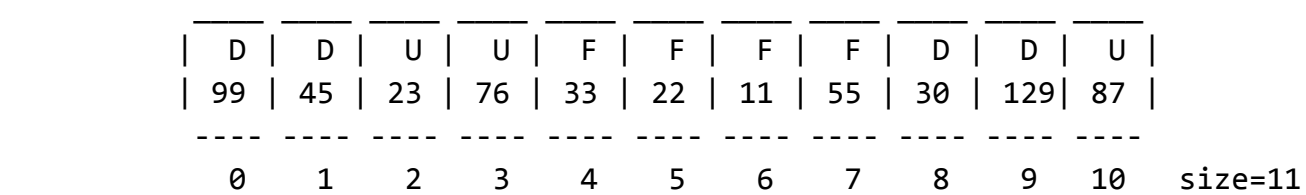
3. Tabelas de Hash (4 valores)

Considere uma tabela de Hash de inteiros implementada com um array dinâmico com linear probing e onde as remoções são feitas marcando as respectivas células como apagadas (`D`); `U` e `F` são usados para marcar células ocupadas e livres

Considere ainda que quando o rácio de células removidas atinge um determinado valor se faz uma *garbage collection* que consiste em (sem qualquer redimensionamento) recalcular a tabela sem as células removidas (e sem redimensionar a tabela).

Assumindo que a função de hash é definida apenas como `hash(x) = x % size`, diga qual o estado da seguinte tabela após a referida operação de *garbage collection*.
(Consider a hash table of integers implemented with a dynamic array and linear probing and where deletions are made by labeling the deleted cell with D (U and F are used to label cells as Used and Free).
Assume that when the ratio of deleted cells has a certain value, a garbage collection is executed. This corresponds to recalculating the hash table in order to eliminate all deleted cells.

Assuming that the hash function is defined as `hash(x) = x % size`, show the result of the garbage collection procedure applied to the following table.
)



4. Análise amortizada (4 valores)

Relativamente à tabela referida na questão anterior, assuma que a operação de *garbage collection* é realizada quando pelo menos 1/4 das células estão apagadas.

Assuma ainda que o custo real de inserção e remoção de um elemento são feitos em tempo constante (1),

1. Qual o custo real da operação de *garbage collection* definida acima
2. Mostre que os custos amortizados das operações de remoção e *garbage collection* são constantes.

(With respect to the table referred in the previous question, assume that the garbage collection operation is performed when at least 1/4 of the cells are deleted. Assume that the real cost of inserting and removing an element is done in constant time (1),

1. *What is the actual cost of the garbage collection operation defined above*
2. *Show that the amortized costs of the remove and garbage collection operations are constant.*

)

5. Grafos (4 valores)

Considere uma matriz de inteiros que representa as altitudes dos pontos de um mapa. Considere ainda um robot que se pode deslocar nesse mapa segundo os habituais movimentos Norte, Sul, Este e Oeste. Defina uma função `int maior_descida (int mapa [N][N])` que calcula o comprimento do maior caminho no mapa que só implica descidas (todos os movimentos correspondem a passar para um ponto com menor altitude). Comece por descrever sucintamente a estratégia que irá usar para resolver este problema.
Por exemplo, para a matriz abaixo, a função deve retornar 10, correspondendo às posições marcadas.
(Consider a matrix of integers representing the altitudes of points on a map. Consider also a robot that can move on that map according to the usual North, South, East and West movements. Define a function `int maior_descida (int mapa [N][N])` that computes the length of the longest path in the map where all moves correspond to moving to a point with lower altitude). Start by briefly describing the strategy you will use to solve this problem.
For example, for the matrix below, the function should return 10, corresponding to the marked positions.)

```
int alturas[10][10] = {{10, 4, 3,10, 1, 7, 5, 5, 4,1}
                      ,{ 1, 2, 5, 9, 2, 5, 4, 6, 6,1}
                      ,{ 9, 3,10, 8, 9, 3, 4,10, 2,1}
                      ,{ 6, 5, 6, 7, 5, 6, 6, 4, 6,1}
                      ,{ 7, 8, 5,10, 1,12, 1, 1, 7,1}
                      ,{ 4,15, 4, 3, 2, 8, 5, 6,11,1}
                      ,{ 2, 3,10, 8, 1, 4, 3, 3, 9,1}
                      ,{10,12, 4,10, 0, 7, 2, 4, 2,1}
```

```
,{ 2, 1,10, 3, 1, 6, 8,13, 8,1}
,{ 5,10, 5, 8,13, 4, 5,15, 1,1}
};
```