Algoritmos e Complexidade — Universidade do Minho
Teste 10 de Janeiro de 2023

Número: _____ Nome: _____

1. Árvores AVL (5 valores)

Defina uma função `AVLTree build (int a[], int N)` que constrói uma árvore AVL a partir de um array ordenado. Garanta que os factores de balanço ficam bem preenchidos e que a função executa em tempo linear.

(Define a function `AVLTree build (int a[], int N)` that builds an AVL tree from an ordered array. Make sure that the balance factors are correctly calculated and ensure that your function executes in linear time)

```
typedef struct avlnode {
   int valor; int bal;
   struct avlnode *esq, *dir;
} *AVLTree;
```

2. Análise de caso médio (5 valores)

Faça a análise de caso médio da função `f`, assumindo que `a` é um array de `0`s e `1`s e que `g` é uma função da classe $\Theta(2\text{^}N)$. Comece por identificar o melhor e o pior casos.

(Compute the average time taken by the function f below assuming that g is a function that runs in $\Theta(2\text{^}N)$. *Start by identifying the best and worst cases*)

```
int f(int a[], int N) {
   int b = 0;
   for (int i = 0; i < N; i++)
       if (a[i] == 1) b++;
   if (b == 1) return g(a,N);
   else return 0;
}
```

```
AVLTree build(int a[], int N){
    return buildAVL(a, 0, N-1);
}
AVLTree buildAVL(int a[], int start, int end){
    if(start > end) return NULL;
    int mid = (start + end)/2;
    AVLTree node = malloc(sizeof(struct avltree));
    node->valor = a[mid];
    node->esq = buildAVL(a, start, mid-1);
    node->dir = buildAVL(a, mid+1, end);
    int leftB = (node->esq != NULL)?
                node->esq->bal + 1 : 0;
    int rightB = (node->dir != NULL)?
                 node->dir->bal + 1 : 0;
    node->bal = rightB - leftB;
    return node;
}
```

3. Min-heaps (5 valores)

Implemente uma função `int decrease(int x, int y, int h[], int N)` que dada uma min-heap `h` com `N` elementos e um valor `x` que está na min-heap, troque esse valor pelo valor `y`, assumindo que `y < x`. Analise o pior e melhor caso da função implementada.

(Define a function `int decrease(int x, int y, int h[], int N)`that, given a min-heap h with N elements and a value x which occurs in h, substitute that value by a smaller value y. Identify and compute the best and worst case of that function).

4. Grafos (5 valores)

Implemente a função `int maior(int image[N][N])` que dado um bitmap com uma imagem de dimensão `N` por `N`, devolva o tamanho da maior figura (pixeis contíguos horizontal ou verticalmente) nela contida. Por exemplo, no seguinte bitmap a maior figura tem tamanho 7. Pode assumir que os píxeis a `1` nunca têm coordenadas `0` ou `N-1`.

```
{{0,0,0,0,0,0},
 {0,1,0,1,1,0},
 {0,1,1,1,0,0},
 {0,1,0,0,1,0},
 {0,0,1,1,1,0},
 {0,0,0,0,0,0}}
```

(Define the function `int maior(int image[N][N])` that, given a bitmap of an NxN image, computes the size (number of pixels) of the biggest figure (adjacent pixels) of that image. For instance, in the bitmap above, it should return 7. You may assume that the borders of the image are all `0`s.)

```
int decrease(int x, int y, int h[], int N){
  int i;
  for(i=0; i<NV; i++)
    if(h[i]==x) break;
  if(i==N) return -1;
  h[i]=y;
  while(i>0 && h[(i-1)/2]>h[i])
    swap(&h[i], &h[i/2-1]);
    i=(i-1)/2
  }
void swap(int *a, int *b){
   int aux= *a;
   *a = *b;
   *b = aux;
}
```

```
int maior(int image[N][N]){
```