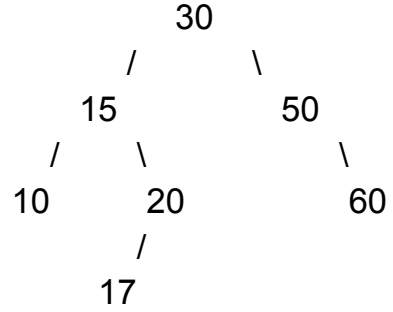
Algoritmos e Complexidade — Universidade do Minho
Exame 23 de Janeiro de 2024

Número: _____ Nome: _____

1. Especificação e Correção (4 valores)

Recorde que as estruturas de dados conhecidas como *heaps* são árvores binárias implementadas fisicamente sobre arrays. É possível implementar quaisquer outras árvores binárias sobre arrays, bastando para isso utilizar um valor especial para representar as lacunas, i.e. posições da árvore em que não se encontra qualquer nó. Por exemplo a seguinte árvore de números inteiros



pode ser representada por arrays preenchidos desta forma:

| 30 | 15 | 50 | 10 | 20 | H | 60 | H | H | 17 | ... H ... |

Considere agora a seguinte função de procura numa árvore binária (BST) representada desta forma:

```
int bsearch (int t [], int x, int N) {
     // Pre: ...
     int i = 0 ;
     while (i<N && a[i]!=H && a[i]!=x)
          if (a[i]>x) i = 2*i+1 ; else i = 2*i+2 ;
     if (i==N || a[i]==H) i = -1 ;
     // Pos: ...
     return i ;
}
```

a) Escreva uma **pré-condição** e uma **pós-condição** que descrevam o seu comportamento.
b) Escreva um invariante e um variante para o ciclo, que lhe permitam provar a correção total da função. No caso do invariante se não souber especificá-lo formalmente explique o mesmo “por palavras”.

Algoritmos e Complexidade — Universidade do Minho
Exame 23 de Janeiro de 2024

Número: _____ Nome: _____

2. Complexidade algoritmos iterativos e recursivos (6 valores)

a) Observe agora a seguinte versão recursiva da função da Questão 1:

```
int bsearch (int t [], int i, int x, int N) {
     if (i>=N || a[i]==H) return -1 ;
     if (a[i]>x) return bsearch (t, 2*i+1, x, N) ;
     if (a[i]<x) return bsearch (t, 2*i+2, x, N) ;
     return i ;
}
```

Identifique o **melhor** e o **pior caso** do seu tempo de execução em função de `N` (o tamanho do array).
Escreva, e resolva, recorrências para esses dois casos, contando as operações de comparação `(a[i]>x)` e `(a[i]<x)`.

b) Considerando o caso particular das árvores de procura representadas por arrays sem ocorrências de H (i.e., árvores completas, em que o comprimento do array coincide com o número de elementos da árvore) e assumindo que x se encontra na árvore, analise o tempo de execução da função de procura no **caso médio**.

N.B. Poderá basear a sua análise em qualquer uma das duas versões da função.

Algoritmos e Complexidade — Universidade do Minho
Exame 23 de Janeiro de 2024

Número: _____ Nome: _____

3. *Heaps* (3 valores)

Considere uma *min-heap* inicialmente vazia, implementada sobre um *array dinâmico* com comprimento inicial igual a 1 (e que duplica de tamanho quando está cheio), e a seguinte sequência de operações sobre ela:

Insert 8 ;
Insert 18 ;
Insert 7 ;
Insert 10 ;
ExtractMin ;
Insert 29 ;
Insert 3 ;
ExtractMin ;
Insert 20 ;

Apresente os arrays correspondentes aos estados da heap em todos os passos desta sequência.

Algoritmos e Complexidade — Universidade do Minho
Exame 23 de Janeiro de 2024

Número: _____ Nome: _____

4. Grafos (3 valores)

Considere o grafo não orientado, com pesos, representado pela seguinte matriz de adjacências (peso 0 significa que não há aresta entre os nós respectivos):

```
0 5 0 3 7 0
5 0 4 0 0 0
0 4 0 1 0 2
3 0 1 0 3 0
7 0 0 3 0 2
0 0 2 0 2 0
```

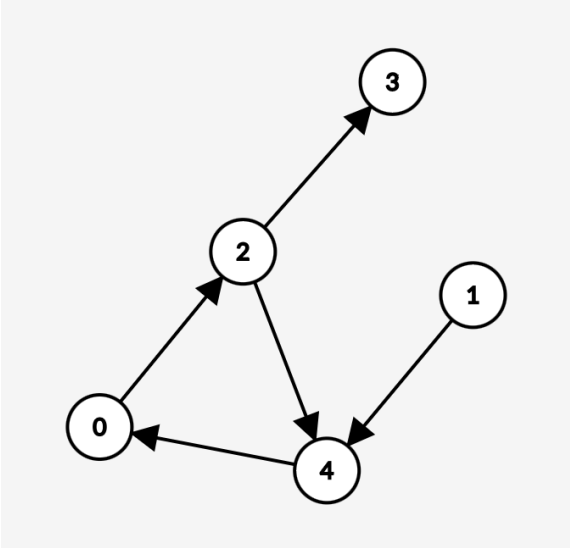
a) Desenhe o grafo correspondente a esta matriz de adjacências.
b) Apresentando todos os passos, aplique o algoritmo de Prim para construir uma árvore geradora mínima do grafo, começando no nó 0.

Algoritmos e Complexidade — Universidade do Minho
Exame 23 de Janeiro de 2024

Número: _____ Nome: _____

5. Grafos (4 valores)

Escreva uma função que detecta se um vértice x de um grafo orientado faz parte de um ciclo. Por exemplo, para o grafo seguinte a função deverá devolver 0 (falso) para os vértices 1 e 3, e 1 para todos os restantes vértices.




```
struct edge { int dest; struct edge *next; };
typedef struct edge *GrafoL[MAX];

int in_cycle (GrafoL g, int x) {
    ...
}
```

Analise o tempo de execução da função que escreveu, identificando o melhor e o pior caso.

Serão valorizadas as soluções mais eficientes.