Pergunta 1

Considere o seguinte algoritmo:
(consider the following function)

```
int what (int A[], int N)
{ int i, j;
  i = -1;
 for (j=0; j<N; j++)
    if (A[j]%2 == 0) {
      i++;
      swap(A, i, j);
    }
  return i+1;
}
```

Escreva uma pós-condição que descreva o que este algoritmo faz (pode experimentar esta função no seguinte permalink do pythontutor: https://tinyurl.com/4ob3zwuq). Tente que a pós-condição seja o mais informativa possível sobre o comportamento do algoritmo.

Esta função coloca os números pares que pertencem ao array no seu início. Uma pós-condição que descreve este resultado é:

forall (0 <=k <= i) (A[k]%2 == 0) && **forall(i < k < j) (A[k]%2 != 0)** && **i < j** && **j == N**

Pergunta 2

Escreva agora um invariante de ciclo que permita provar a correcção parcial do algoritmo, tendo em conta a pós-condição que especificou.

(Considering the condition provided above, write down an invariant of the loop in order to prove the partial correction of the function)

Um invariante poderá ser:

forall (0 <=k <= i) (A[k]%2 == 0) && **forall(i < k < j) (A[k]%2 != 0)** && **i < j** && **j <= N**

Pergunta 3

Seja T_swap(N) o número de operações swap executadas; analise o seu pior caso, identificando-o.

(Let T_swap(N) denote the number of swap operations performed by this function. Identify its worst case and compute it)

O pior caso corresponde a todos os elementos do array serem pares. Neste caso, há um swap por cada iteração do ciclo. Traduz-se no seguinte somatório:

sum (0 <= j <= N - 1) 1 = (N-1-0+1) * 1 = N swaps

Pergunta 4

Calcule o valor médio do número de operações swap executadas.

(Perform a average case analisys on the the number of swap operations executed by the function what)

Ora bem ardi nesta pergunta. Não percebo nada de casos médios mas resolvi assim:
Possibilidade do 1º elemento ser par: 0.5
Possibilidade do 2º elemento ser par:

- 1º elemento ímpar + 2º elemento par => 0.5*0.5 = 0.25
- 1º elemento par + 2º elemento ímpar => 0.5*0.5 = 0.25

Logo, 0.5.

Possibilidade do N elemento ser par: 0.5
Caso médio: T(N) = 0.5 * sum (0 <= i <= N-1) = 0.5 * N

Pergunta 5

Relembre o tipo BTree usado para implementar árvores binárias de inteiros:

(Recall the definition ot the type BTree, used to implement binary trees of integers)

typedef struct btree { int node; struct btree *left, *right) *BTree;

Defina uma função int test (BTree a) que, em tempo linear no número de elementos da árvore argumento, testa se a árvore satisfaz a seguinte propriedade:

Em **todos** os nós, os números de nós da árvore da esquerda e da direita diferem no máximo numa unidade

(Define a function int test (BTree a) that tests, in linear time, that all nodes of a tree satisfy the following property:

The differnece between the number of nodes of the left and rigth sub-trees is at most one)

```
int isValid (BTree a, int * size) {

   if (!a) {
       *size = 0;
       return 1;
   }

   int size_l, size_r, valid_l, valid_r;
   valid_l = isValid(a->left, &size_l);
   valid_r = isValid(a->right, &size_r);
   if (!valid_l || !valid_r || (abs(size_l - size_r) < 1))
       return 0;
   *size = size_l + size_r;
   return 1;
}
```

```c
int test (BTree a) {
   int size;
   return isValid(a, &size);
}
```

Pergunta 6

Justifique que a sua implementação da função test é de tempo linear, apresentado e resolvendo a respectiva recorrência.

(Prove that your implementation of the function test runs indeed in linear time, by writing down and solving a recursive deifition of its complexity function)

A recorrência é traduzida por T(N) = 1 + 2*T((N-1)/2) que é limitada superiormente pela recorrência T(N) = 1 + 2*T(N/2), tendo-se ainda que T(0) = 0.

A árvore terá o aspeto seguinte:

```
                T(N)                          ---     1 ( = 2^0 * 1)
              /      \
          T(N/2)      T(N/2)                  ---     2 * 1 (= 2^1 * 1)
         /    \       /    \
    T(N/4) T(N/4) T(N/4) T(N/4)               ---     4 * 1 (= 2^2 * 1)
                  …
```

T(N/2^i) = ??

T(1) será o último T(...) que retornará um valor diferente de zero. Logo, a última iteração da recorrência será quando N/2^i = 1 ⇔ N = 2^i ⇔ i = log2(N)

Então, traduz-se no seguinte somatório:

sum (0 <= i <= log2(N)) 2^i = 2 * N - 1 (resolvido no wolfram alpha idk how to solve this sem ser aí). Logo, trata-se de tempo linear.

Pergunta 7

Considere o grafo orientado não-pesado com o seguinte conjunto de vértices e arestas:

(Consider the directed graph with the following set V of nodes and E of edges)

V = {0,1,2,3,4}

E = {(0,3),(0,4),(2,3),(2,4),(3,1),(3,2),(4,1),(4,2)}

Mostre como poderia inicializar a variável g com este grafo no seguinte programa.

(Fill in the ... in the following program in order to initialize the variable g with that graph)

```
#define N 5
typedef struct edge {
 int dest;
 struct edge *prox;
} Edge;
typedef Edge *Graph[N];

void travessia(Graph g, int v, int pais[N]);

int main() {
 Graph g;
 int p[N];
 // inicializar g
 ...
 travessia(g,0,p);
 return 0;
}
```

```
int acrescenta (Graph g, int o, int d) {
   int r = 1;

   if (0 <= o && o < N && 0 <= d && d < N) {
       Edge * temp = malloc(sizeof(struct edge));
       if (temp) {
           r = 0;
           temp->dest = d;
           temp->prox = g[o];
           g[o] = temp;
       }

   }

   return r;
}
```

```
int main() {
   Graph g;
   int p[N];
   int r;

   for (int i = 0; i < N; i++) g[i] = NULL;
   r = acrescenta(g, 4, 2);
   r = acrescenta(g, 4, 1);
   r = acrescenta(g, 3, 2);
   r = acrescenta(g, 3, 1);
   r = acrescenta(g, 2, 4);
   r = acrescenta(g, 2, 3);
   r = acrescenta(g, 0, 4);
   r = acrescenta(g, 0, 3);

   travessia(g,0,p);
   return 0;
}
```

Por ter usado inserções à cabeça (custo constante), decidi acrescentar pela ordem inversa de forma a garantir a mesma ordem que a apresentada.

Pergunta 8

Considerando que a função travessia da pergunta anterior implementa uma travessia em profundidade, qual o valor do array p após a invocação da mesma na função main? Apresente também a ordem pela qual os vértices são visitados e tenha em atenção a forma como incializou o grafo.

(Assuming that the function travessia refered above implements a depth first traversal of the graph starting at a particular node, which is the final value of the array p at the end of its call? Present also the order in which the nodes are traversed and take into consideration the way you initialized the graph)

Antes de tudo, p[5] = {-2, -2, -2, -2, -2}. Ao decidir-se iniciar a travessia pelo vértice 0, tem-se que p[5] = {-1, -2, -2, -2, -2}. Acedendo a g[0], temos a lista ligada 3 -> 4- > NULL. Pela travessia depth first, marcamos o 3 como acedido a partir de 0 alterando o array p para p[5] = {-1, -2, -2, 0, -2} e acedemos a g[3] onde temos a lista 1 -> 2. Marcamos o vértice 1 como acedido a partir do vértice 3 alterando o array p para p[5] = {-1, 3, -2, 0, -2} e acedemos a g[1] onde temos uma lista vazia. Então acede-se agora a g[2] marcando-se o vértice 2 como visitado a partir do vértice 3, p[5] = {-1, 3, 3, 0, -2} onde temos a lista ligada 3 -> 4. Como o vértice 3 já foi visitado, visita-se agora o vértice 4 a partir do vértice 2 alterando o array p para p[5] = {-1, 3, 3, 0, 2}. Todos os vértices foram visitados.

A ordem da travessia foi 0 - 3 - 1 - 2 - 4.

Pergunta 9

Considere o seguinte problema: seja P um conjunto de pessoas, e A um conjunto de pares (x,y) com x,y ∈ P, exprimindo o facto de x e y serem amigos.
Dadas duas pessoas a,b ∈ P, pretende-se agora dizer se existe ou não uma cadeia de amizades através da qual a pessoa **a** possa ser apresentada à pessoa **b**.

(Consider the following problem: let P be a set of persons, and A a set of pairs (x,y) with x,y ∈ P meaning that x is friend of y.
*Given two persons **a** and **b**, find out if there is a chain of friendship connections through whitch person **a** can be introduced to person **b**)*
Considere agora uma instância particular deste problema em que o conjunto de pares A é fornecido como um array de N pares de Strings enquanto que o conjunto P pode ser calculado como o conjunto das strings que aparecem nesse array.
(Suppose that the set A is represented by an array of pairs of names and the set V is the set of names that occurr in A)

```
typedef struct {
 char *p1, *p2;
} Par;
```

Descreva uma estrutura de dados que lhe permita representar a informação relevante para resolver o problema, e descreva por palavras, sem escrever código, uma função `int existe (Par amigos[], int N, char *a1, char *a2)` que o resolva.
Analise o seu tempo de execução no pior caso.

(Describe the necessary data types needed to solve this problem by a function `int existe (Par amigos[], int N, char *a1, char *a2)`.

Without writing down C-code, describe all the necessary steps of the function existe.

Analize its complexity in the worst case)

Ardeu outra vez. Don’t trust.

Uma estrutura possível seria um grafo não orientado e não pesado, utilizando listas de adjacência. Cada par de amigos (x,y) seria inserido 2 vezes nesse grafo (uma vez com x como destino e y como origem, outra vez com x como origem e y como destino). Se fosse possível estabelecer um caminho entre uma pessoa a e b nesse grafo, ou seja, se b fosse alcançável a partir de a ou a fosse alcançável a partir de b, então poderíamos assumir que a pode ser apresentada à pessoa B.

O custo desta operação iria envolver a criação do grafo e o custo de percorrer o dito grafo. Para N pares de amigos, criam-se 2*N arestas no grafo. Logo, trata-se de custo linear. Para percorrer, no pior caso todo o grafo é percorrido logo 2*N + Número de amigos. Assim, o custo da função “*existe*” seria linear.