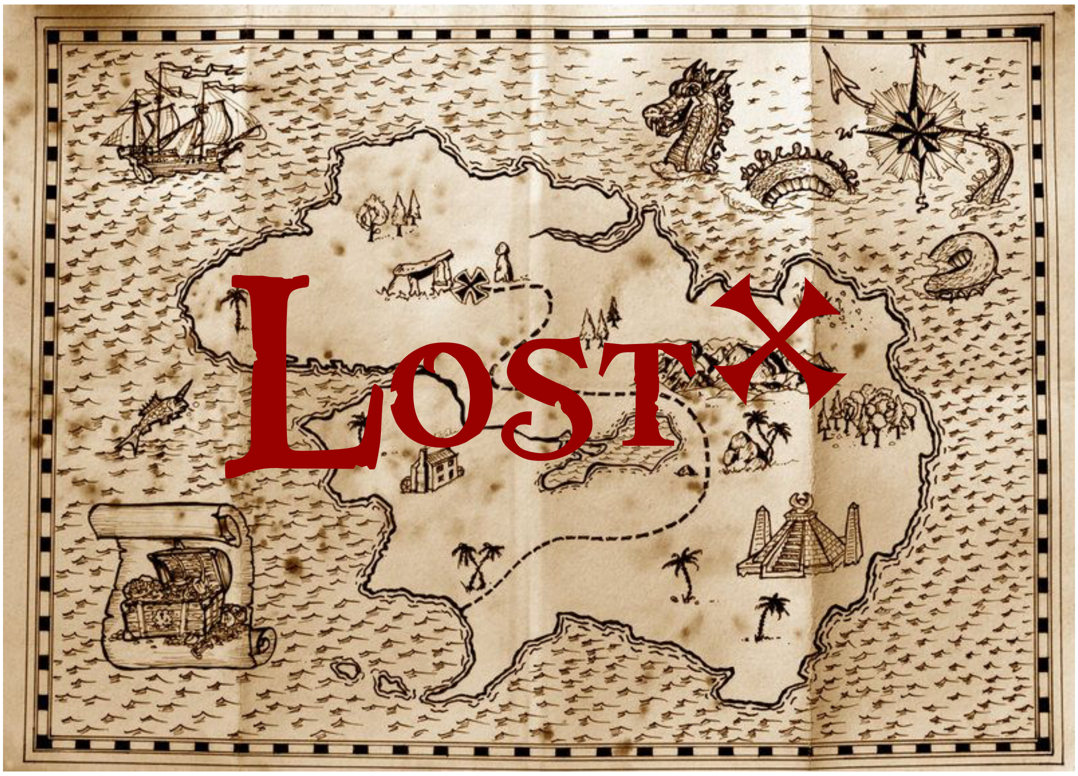
Problem:



Relatório do Trabalho Prático

Universidade de Évora

Curso: Engenharia Informática

Disciplina: Estrutura de Dados 2

Docentes: Vasco Pedro

Entregue Maio 2022

Alunos:

Joana Carrasqueira nº48566

João Condeço nº48976

g301

Índice

1. Pseudo-código do Bellman-Ford

```
BF(INT S, INT X)

v <- g.nodes
let d[0... g.nodes] be an array
for i <- 0 to v do
    d[i] <- INFINITY
d[s] <- 0
for j <- 0 to v-1 do
     cont <- False
     for each Edge up in g.adjacents[i]
         if d[i] != INFINITY and
         d[i] + up.weight() < d[up.dest()] then
             cont <- True
if cont <- False
    break

for i <- 0 to v-1 do       // v -> nº de vértices
  for each Edge up  g.adjacents[i] do
     if d[i] != INFINITY &&
     d[i] + up.weight() < d[up.dest()] then
         Print "Lost in Time"
           return False;
if d[x] != INFINITY then
     Print d[x]
else
     Print "Unreachable"
return True
```

2. Descrição do Algoritmo

De modo, a calcular o tempo menor para o John e a Kate alcançarem a saída foi concebido um algoritmo que obtém o valor pretendido com uma complexidade constante. Para tal foi feita a representação dos grafos construídos para aplicar este algoritmo no problema em causa.

O programa começa por recolher as informações relativas às dimensões do mapa e ao número de Magic Wheels. Com base nisto são gerados dois grafos, um para o John e outro para a Kate, respetivamente, um com as dimensões indicadas e outro com duas colunas e duas linhas a menos (de forma a ignorar as células de água no caso do John). De seguida são adicionados os valores aos grafos, onde, são percorridas uma a uma cada célula do mapa e são feitos os seguintes procedimentos:

- Se for um bloqueio a célula é ignorada.
- Ligação à esquerda se não se tratar da primeira coluna e se não tiver um bloqueio à esquerda.
- Ligação acima se não se tratar da primeira linha e não tiver um bloqueio acima.
- Ligação à direita se não se tratar da última coluna e se não tiver um bloqueio à direita.
- Ligação abaixo se não se tratar da última linha e não tiver um bloqueio abaixo.
- Em qualquer um dos casos anteriores apenas é feita a ligação no grafo do John se não se tratar de uma célula com água e o destino não for água.
- Se for destino este é armazenado para ser usado no algoritmo Bellman-Ford.
- Se for Magic Wheel é armazenada a sua posição para posteriormente ser feita a ligação quando recolhidas as coordenadas destino.

Uma vez construído o grafo, são recolhidas as coordenadas destino das Magic Wheels e adicionadas essas ligações ao mesmo. De seguida são retiradas as coordenadas iniciais da Kate e do John e calculados os vértices em que se encontram no grafo.

Por fim é aplicado o algoritmo Bellman-Ford a cada um deles. Sendo a única diferença a existência de um ciclo que procura ciclos negativos de forma a identificar a situação "Lost in Time".

```
for i <- 0 to v-1 do      // v -> nº de vértices
    for each Edge up  g.adjacents[i] do
        if d[i] != INFINITY &&
        d[i] + up.weight() < d[up.dest()] then
             Print "Lost in Time"
             return False;
if d[x] != INFINITY then
    Print d[x]
else
    Print "Unreachable"
return True
```

3. Descrição dos grafos utilizados

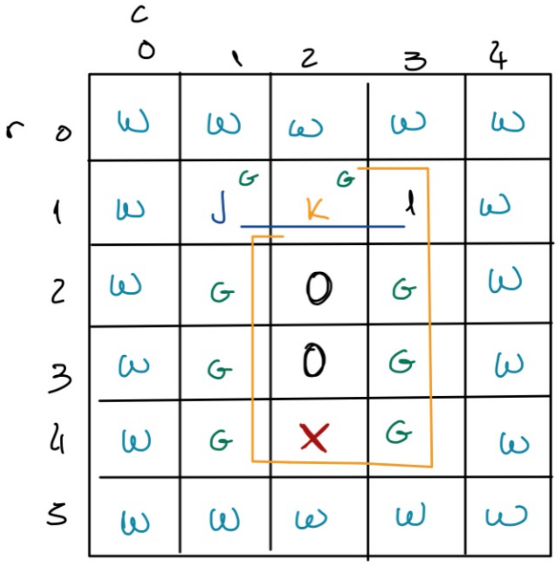


Fig. 1 – Mapa referente ao exemplo 1 do enunciado

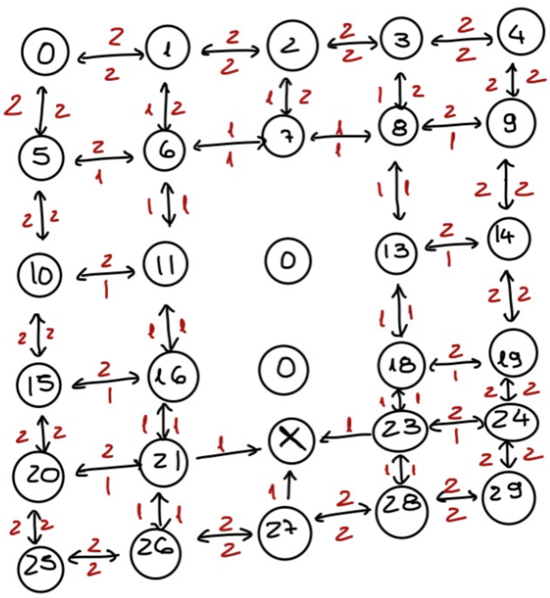


Fig. 2 – Grafo do percurso da Kate do exemplo 1

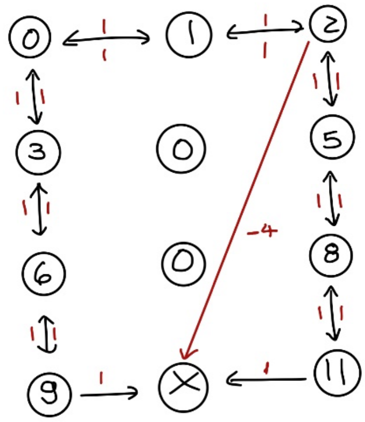


Fig. 3 – Grafo do percurso do John do exemplo 1

4. Análise da complexidade

- Complexidade espacial

  Sendo a complexidade espacial o espaço exigido pelo algoritmo para executar até ao fim em função do input, a mesma corresponderá ao tamanho do array criado para armazenar as distâncias dos vértices em relação à origem, ou seja, será o número de vértices do grafo, assim O(V).

- Complexidade temporal

  Complexidade temporal consiste na porção de tempo que o algoritmo demora a ser executado em função do input. Assim, tendo em conta que efetuamos os cálculos das distâncias dos vértices à origem V-1 vezes (sendo V o número de vértices) para os Edges de cada vértice. Logo é possível concluir que, tendo três ciclos for para este efeito, a complexidade temporal será de $O(V^2 \cdot E)$.

5. Decisões e dificuldades encontradas

Na elaboração deste trabalho surgiram alguns obstáculos, dos quais se destacam o facto de, no momento do cálculo do peso dos portais, quando interpretados como Grass, era atribuído o valor do número referente ao portal e não 1 devido a um erro na função responsável por calcular o peso de cada célula (cellCost).

Outra situação que surgiu foi o programa apresentar que o John não consegue chegar à saída, em situações em que fica Lost in Time devido a falhas no algoritmo.