Trabalho 3

Lost

g314(Carlos Palma_46520 e Manuel Cunha_48482)

- **Algoritmo**

1. Pedimos ao utilizador o input que guardamos numa matriz.
2. Percorremos a matriz e verificamos qual o carácter que se encontra em cada posição, cada posição é associada a um vértice (sendo esse vértice identificado por um inteiro obtido por número da linha * número de colunas + número da coluna, desta forma obtemos números que correspondem à matriz lida da esquerda para a direita e de cima para baixo, o que é mais intuitivo).
3. Se o carácter da posição atual for ‘G’ e as posições adjacentes não forem obstáculos é criada uma aresta orientada entre o vértice correspondentes à posição atual e o vértice correspondente à adjacente com peso 1 e carácter ‘A’ associado (A corresponde a: todos podem percorrer a aresta), se o carácter for ‘W’ é criada uma aresta com peso 2 e carácter ‘K’ (K corresponde a: só a Kate pode percorrer a aresta), se ‘X’ é guardada a posição numa variável (corresponde à posição final) e se for um número é guardado num array, na posição do array correspondente ao número que se encontrava nessa posição (posteriormente é criada uma aresta orientada entre este vértice e um vértice escolhido pelo utilizador, com peso também escolhido pelo utilizador, é também associada a esta aresta o carácter ‘J’ que corresponde a: apenas o John pode passar), nesta mesma são também criadas arestas como se se trata-se do carácter ‘G’ pois as magical whells podem ser usadas como grass cells por ambos.

4. Por fim, após termos o grafo construido para o John, optamos por utilizar o algoritmo de Bellman-Ford pois trata-se de um algoritmo que além de calcular o caminho com menor peso verifica também se existem ciclos no grafo e trabalha com pesos negativos, as alterações que realizamos ao algoritmo fornecido pelo professor são que apenas são percorridas as arestas com ‘A’ ou ‘J’ e o ciclo que calcula as distâncias termina quando após percorremos todos os nós não é encontrada nenhuma alteração no array de distâncias. Se forem encontrados ciclos, consideramos que o John se encontra “Lost in Time”, se não existir caminho para o vértice consideramos que o mesmo é “Unreachable” e caso exista retornamos o valor do caminho com peso mínimo.
5. A nível da Kate, resolvemos optar por um algoritmo mais rápido pois no caso da mesma não temos de lidar com pesos negativos, como tal escolhemos o algoritmo de Dijkstra, trata-se de uma implementação em tudo semelhante à fornecida pelo professor nas aulas teóricas com a utilização de uma Queue, a única alteração realizada é a nível do percurso no grafo em que apenas são percorridos os caminhos com os caracteres ‘A’ ou ‘K’. Retornamos “Unreachable” se não existir caminho ou o valor do caminho com peso mínimo caso exista.

- **Grafo construido**

Foi utilizado um grafo que associa uma lista de vértices a uma lista com as arestas adjacentes a esse vértice.

Para representar as arestas foi utilizado um objeto “Edge” que contém o peso da aresta, o destino da aresta e quem pode percorrer essa mesma aresta, dentro desse mesmo objeto foi declarado um construtor para atribuir os valores a cada aresta.

- **Análise de complexidade**

1. Na função main percorremos toda uma matriz para avaliar de que modo deve ser construido o grafo. Como tal apresenta uma complexidade temporal e espacial de Θ (RC) sendo R o número de linhas da matriz e C o número de colunas da matriz.
2. De seguida percorremos um array onde estão guardadas as magic whells logo a complexidade tanto temporal como espacial é Θ (M) sendo M o número de magic whells que o utilizador pretende inserir.
3. No algoritmo de Bellman_Ford são percorridos todos os vértices e arestas, o que resulta numa complexidade temporal e espacial de *O (VE) sendo V o número de vértices e E o número de arestas*
4. *No algoritmo de Dijkstra percorremos todos os vértices e arestas o que resulta numa complexidade espacial e temporal de O (E log V) sendo E o número de arestas e V o número de vértices.*