Tema 4

Deslocação do herói no labirinto

Relatório Final do Primeiro Projecto

Concepção e Análise de Algoritmos

2º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

José Taveira – 201005306 – ei10157@fe.up.pt

Pedro Faria – 201001822 – ei11167@fe.up.pt

Rui Figueira – 201005406 – ei11021@fe.up.pt

Turma 4 – Grupo D

25/04/14

Descrição do Problema

O trabalho desenvolvido consiste em encontrar o caminho mais curto que deve ser seguido por um héroi que se encontra num labirinto, com vários dragões, até á saída do mesmo. Para o héroi poder sair terá primeiro de encontrar a espada que se encontra no labirinto sem encontrar nenhum dragão e depois matar todos os dragões existentes e dirigir-se á saida.

O labirinto é representado por uma matriz escrita num ficheiro de texto em que :

- H – héroi;
- E – espada;
- D – dragão;
- S – saída;
- X – parede;
- _ – casa vazia;

Numa primeira versão do projecto, os dragões existentes são imóveis e numa segunda, podem movimentar-se aleatoriamente. Para a resolução deste problema, o labirinto é representado por um grafo planar em que os vértices representam posições no labirinto e as arestas representam movimentações unitárias possíveis. Como algoritmo de resolução do problema usamos o disjktra com pesos unitários, algoritmo que iremos explicar mais detalhadamente na próxima secção.

O nosso programa começa por ler o ficheiro de texto com a informação do labirinto e processa essa informação para a criação de um grafo com os respectives vértices e arestas. Em ambas as versões do projecto o problema pode ser divido em três principais fases:

1. Encontrar o caminho mais curto até á espada;
2. Encontrar o caminho mais curto até cada dragão no labirinto;
3. Encontrar o caminho mais curto até á saida.

No passo 1, o nosso programa ao criar o grafo ignora todas as posições com "X" ou com "D", pois são casas que não serão utilizadas no cálculo do caminho mais curto. Depois do grafo estar criado é aplicado o algoritmo de disjktra para encontrar o caminho mais curto da posição atual do herói até à espada. Depois do héroi ter a espada, recalculamos o grafo, desta vez considerando as posições com "D" e estando o herói na nova posiçao. Neste segundo passo, voltamos a usar o disjktra, mas desta vez para calcular o caminho mais curto até ao dragão mais próximo, repetindo este processo até não haver mais dragões no labirinto. Quando não houver mais dragões, passamos para o terceiro e último passo em que é calculado o caminho mais curto da posição atual do herói até à saída. A segunda versão do projecto mantem os mesmos passos principais, mas em cada iteração de movimento do héroi o grafo tem de ser recalculado. No primeiro passo, uma vez que os dragões se podem movimentar, o caminho que pode ser considerado até à espada, pode mudar. Já no segundo passo, aplica-se o mesmo raciocionio, pois os dragões nem sempre estão na mesma posição. E o terceiro mantem-se igual.

Para visualização dos grafos e do cálculo do caminho mais curto, utilizamos o graphviewer usado nas aulas práticas.

Formalização do Problema

Dados de Entrada: nome do ficheiro de texto a ser processado.

Limites de aplicação: labirintos quadrados até 100x100.

Situações de contorno: o parser do ficheiro apenas aceita ficheiros cujas matrizes sejam quadradas e no caso de a matriz não apresentar um caminho possível de ser calculado, é mostrada uma mensagem de impossibilidade de cálculo e terminado o programa.

Resultados esperados: caminho mais curto em cada etapa, com respectivo peso total (distância).

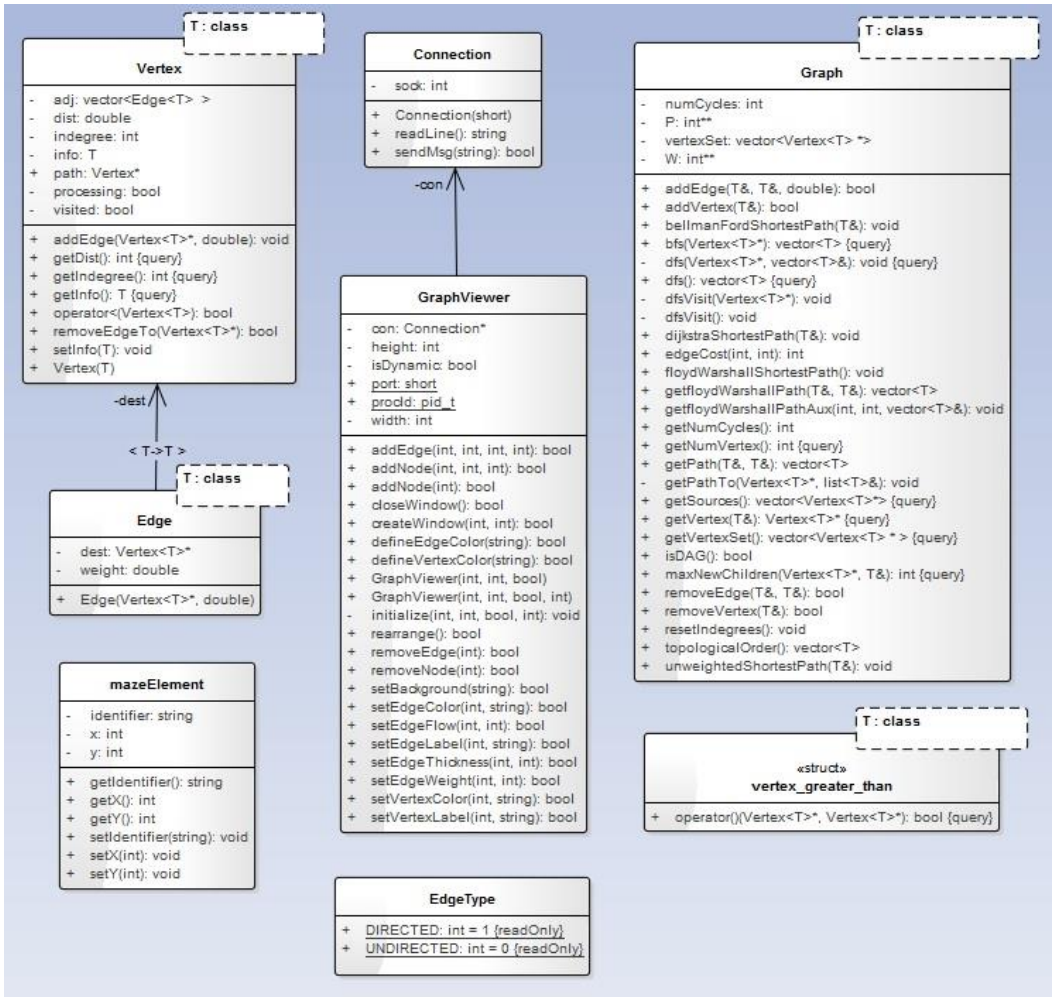
Algoritmo de Disjktra

Tal como foi referido na secção anterior, para resolver o problema do caminho mais curto, recorremos ao algoritmo de disjktra, apresentado nas aulas da cadeira. Em termos de análise de complexidade espacial e temporal, este algoritmo apresenta:

Espaço – O(| V| ^2) | V| - número de vértices

Tempo – O(| E| +| V| log| V|) | E| - número de arestas

Diagrama UML



Casos de Utilização

Temos três casos de utilização distintos para o nosso projecto. O primeiro utiliza um labirinto pequeno que foi utilizado para questões de teste do algoritmo. O segundo utiliza um labirinto de grande tamanho para testar limites temporais e o terceiro utiliza um labirinto em que não existe caminho mais curto na primeira parte sem passar por um dragão. Depois de correr o programa, este pede um ficheiro de texto que irá utilizar para construir o grafo e calcular o caminho mais curto e qual o modo no qual queremos os dragões (1- Dragões estáticos/ 2- Dragões com movimento aleatório). Os nomes dos ficheiros são os seguintes :

Primeiro caso de utilização – smallMaze.txt;

Segundo case de utilização – bigMaze.txt;

Terceiro caso e utilização – impossibleMaze.txt.

Em termos gráficos, tal como referido em cima, o nosso projecto utiliza o *graphviewer*. O caminho mais curto encontra-se pintado a azul, a posição inicial a verde, a posição final a vermelho e os dragões do labirinto a preto. As imagems mostradas de seguida demonstram as 3 prinicpais fases da nossa resolução aplicada ao primeiro caso de utilização (smallMaze.txt) :

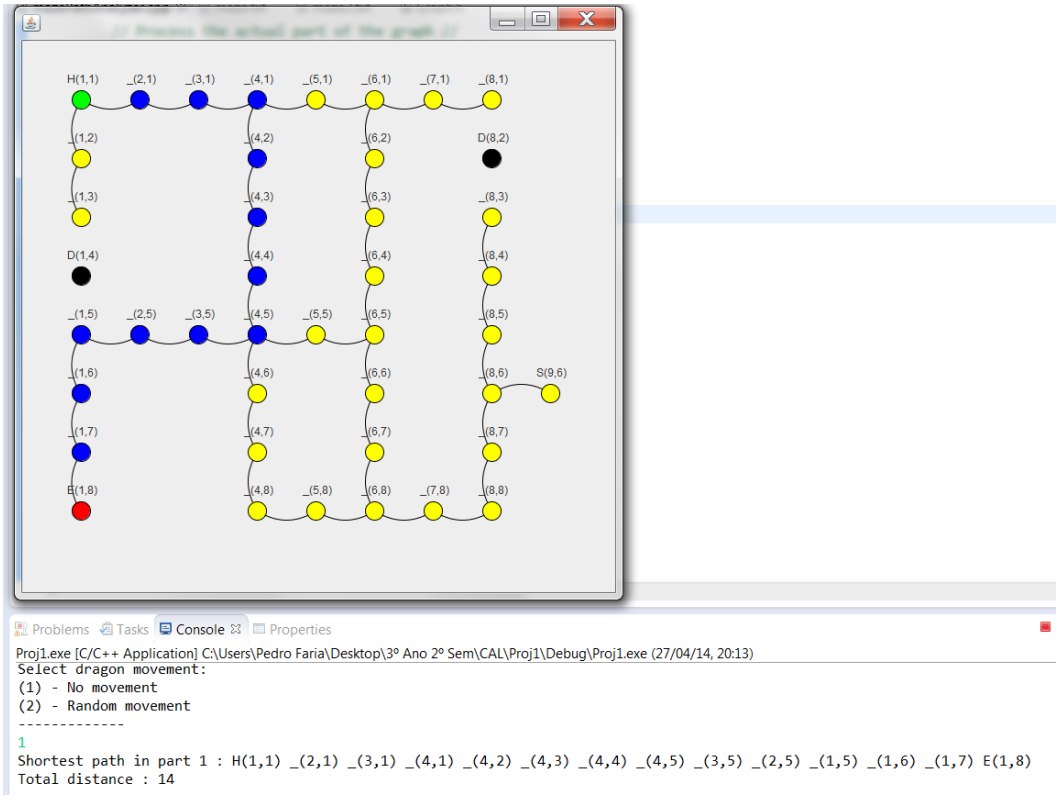


Imagem 1 – Caminho mais curto do herói até à espada. (Fase 1)

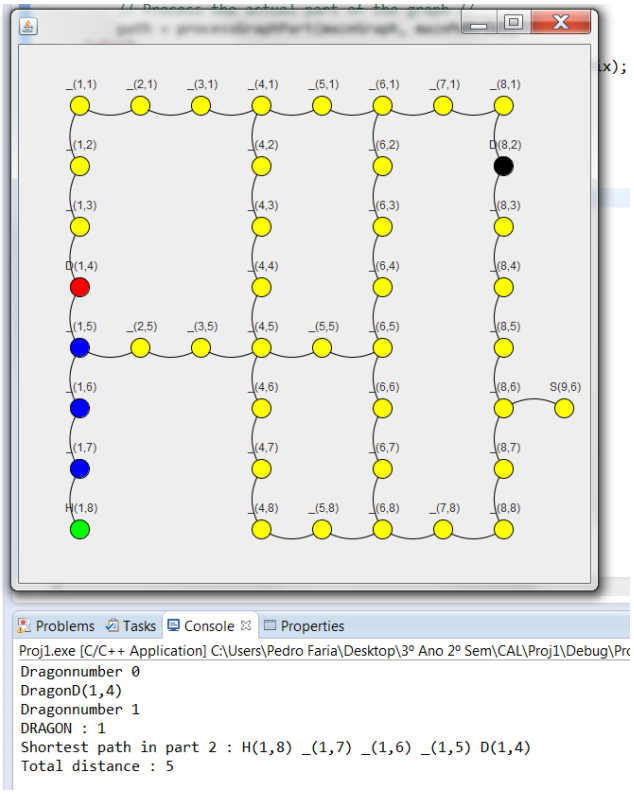


Imagem 2 – Caminho mais curto do herói com a espada até ao dragão mais próximo. (Fase 2)

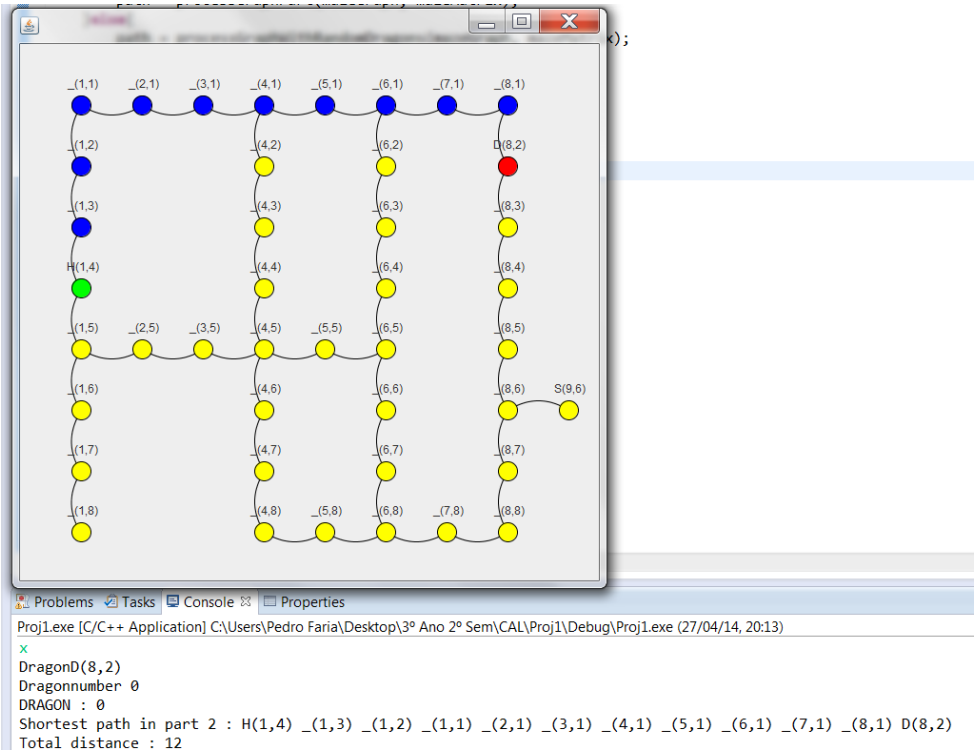


Imagem 3 – Caminho mais curto do herói após matar o primeiro dragão até ao segundo dragão. (Fase 2)

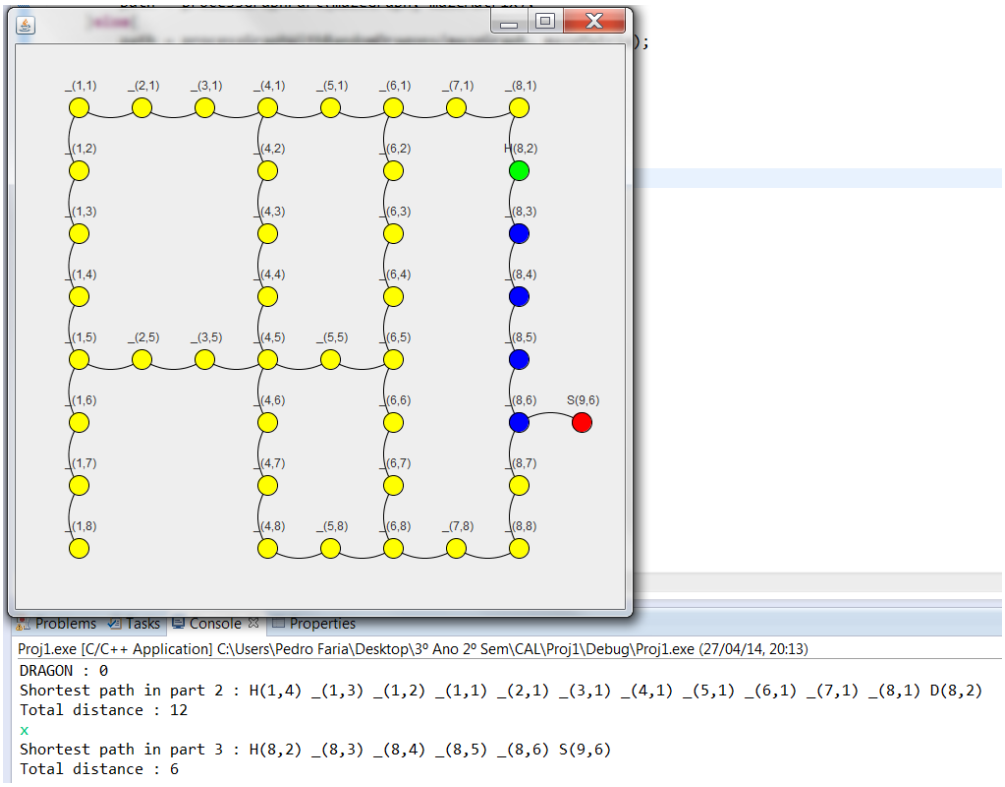


Imagem 4 – Caminho mais curto do herói depois de ter morto todos os dragões até à saída. (Fase 3)

Dificuldades

Ao longo do desenvolvimento do projecto, não sentimos grandes dificuldades. O mais complicado terá sido colocar o *graphviwer* como desejevamos com o nosso projecto. O algoritmo de disjktra utilizado foi o mesmo que foi usado nas aulas teórico-práticas e o desenvolvimento do processamento do grafo demorou algum tempo, sendo que a sua maior dificuldade foi o modo de processamento do grafo em cada iteração para o cálculo completo do caminho mais curto.

Divisão do trabalho

José Taveira – Melhoria do relatório (5%);

Pedro Faria – Parser da informação do ficheiro de texto para grafo e desenvolvimento do código de processamento do grafo (47.5%);

Rui Figueira - Relatório do projecto e desenvolvimento do código de processamento do grafo (47.5%).