

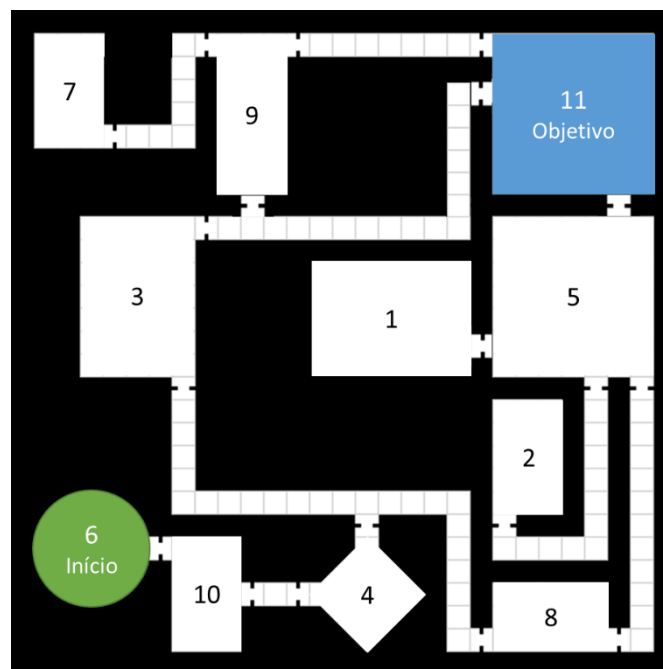
A resolução deste projeto tem de ser realizada em java. A entrega do mesmo deve ser a partir da extração direta do eclipse (ou IDE equivalente) para uma pasta comprimida cujo nome deve possuir o número de aluno e o respetivo primeiro e último nome (exemplo: "12345_JoaoSilva.zip").


A entrega é realizada através do e-learning até 23:59 de 20/03/2022.

Caminhos Entre Masmorras

Dungeons & Dragons inspired

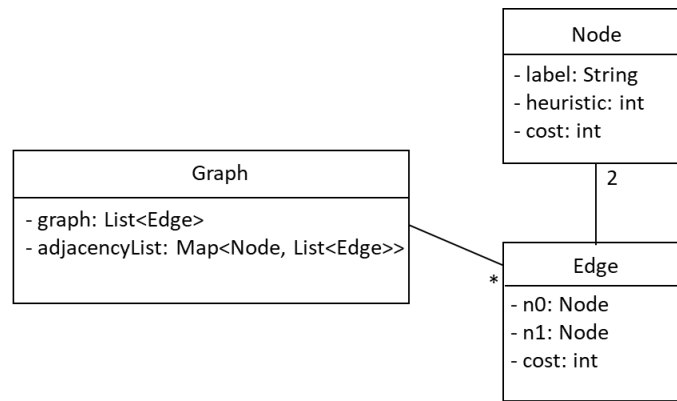
Este problema é baseado no jogo de fantasia *Dungeons & Dragons*. Nesta versão apenas existe 1 jogador, e o jogo termina assim que este chegue a uma masmorra específica. Para isso, o jogador terá de percorrer os caminhos que se encontram apresentados no mapa em baixo.



O mapa apresenta corredores, masmorras (numeradas de 1 a 11) e portas . (A base do mapa foi gerada aleatoriamente através do seguinte link: <https://donjon.bin.sh/fantasy/dungeon/>).

O jogador encontra-se na masmorra 6 e o objetivo é chegar à masmorra 11. Neste trabalho deverá implementar algoritmos de procura que indiquem o caminho que o jogador tem de percorrer até chegar ao objetivo.

Na resolução deste projeto pode fazer alterações ao código disponibilizado, adicionar métodos e atributos, com a condição de indicar as mesmas em forma de comentários no código, com as devidas justificações. No entanto, **não pode alterar** a estrutura do código referente às classes *Graph.java*, *Node.java* e *Edge.java*, representada na imagem seguinte.



- 1) Transforme o mapa num grafo representativo do espaço de procura do problema. Apresente uma imagem (pode ser uma fotografia) com a representação do mesmo e **adicione-a à pasta do projeto java antes de exportar para a entrega**. Crie uma instância do grafo usando as classes java fornecidas (*Graph.java*, *Node.java*, *Edge.java*), nas classes respetivas à resolução (pacote *tests*).

Ver exemplo seguinte:

```

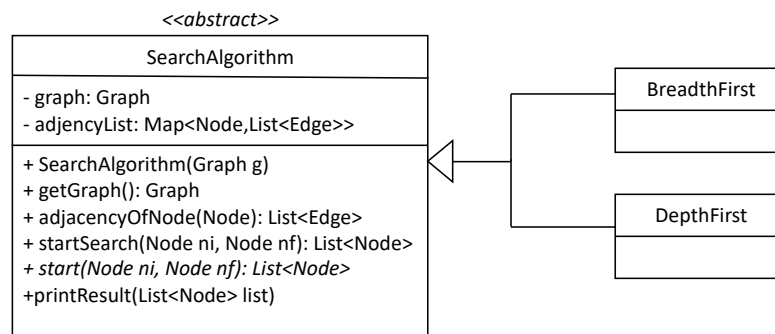
public class Exemplo {
    public static void main(String[] args) {
        /*
         * (1)---->(2)---->(3)
         *      \      /
         *       V    V
         *      (4)---->(5)
         *
         * Initial -> (1) Final -> (5)
         */

        Graph graph = new Graph();
        // creating the nodes
        Node n1 = new Node("1");
        Node n2 = new Node("2");
        Node n3 = new Node("3");
        Node n4 = new Node("4");
        Node n5 = new Node("5");

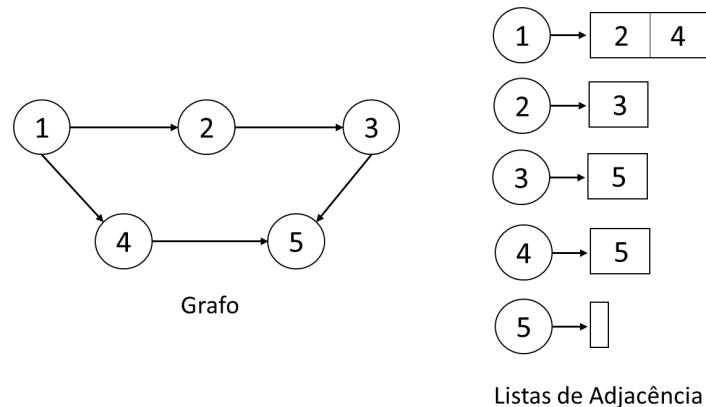
        // creating and adding edges to graph
        graph.addEdge(n1, n2);
        graph.addEdge(n2, n3);
        graph.addEdge(n3, n5);
        graph.addEdge(n1, n4);
        graph.addEdge(n4, n5);

        System.out.println(graph.toString());
    }
}
  
```

- 2) Implemente os algoritmos de procura **largura primeiro** e **profundidade primeiro**. Estes deverão ser duas classes que estendem a classe abstrata *SearchAlgorithm.java*, como demonstrado no seguinte diagrama. Adicione estas classes ao pacote *search_algorithm*.



A classe abstrata possui como atributo um mapa que representa a lista de adjacência dos nós presentes no grafo. Este mapa liga cada nó a uma lista que possui os seus nós vizinhos (através de objetos do tipo *Edge*), como exemplificado na seguinte imagem:



Adicione o código necessário para testar ambos os algoritmos (na classe *SolutionBreadthDepth.java*), de modo a encontrar um caminho entre a masmorra 6 e a masmorra 11.

Note que na implementação dos algoritmos terá de implementar o método `start(Node nInitial, Node nFinal)`, método abstrato herdado. Ao testar use o método `startSearch(Node nInitial, Node nFinal)`, método incompleto, como exemplifica o seguinte excerto de código:

```
System.out.println("initial node:" + n6.getLabel());
System.out.println("final node:" + n11.getLabel());

System.out.println("-----DFS-----");
Search_Algorithm dfsAlg = new DepthFirst(graph);
dfsAlg.printResult(dfsAlg.startSearch(n6, n11));
```

- 3) Implemente agora o algoritmo informado **A-star** (que deverá também estender a classe abstrata *SearchAlgorithm.java* e pertencer ao pacote *search_algorithms*).

Para isso terá de criar uma função heurística e atribuir esse valor a cada nó:

- através do método da classe *Node* `setHeuristic(int heuristic)`, ou
- através do construtor `Node(String label, int heuristic)`.

E associar um custo a cada arco entre dois nós (dica: pode usar o número de quadrados do mapa ou criar outra forma de custo):

- através do construtor `Edge(Node n0, Node n1, int cost)` ou
- do método da classe *Graph* `addEdges(Node n0, Node n1, int cost)`.

Deve explicitar em forma de comentário qual a função heurística utilizada e os custos associados aos arcos.

Note que a classe *Node.java* já se encontra preparada para guardar informação de heurística e de custo, bem como o método para calcular o valor de F ($f(n) = h(n) + g(n)$), através do método `getF()`. Para obter o correto valor de F, terá de primeiro alterar o custo do nó com o método `setCost(int cost)`.