11ª aula prática – Conjuntos Disjuntos

Faça download do ficheiro ***aeda1920_fp11.zip*** da página da disciplina e descomprima-o o (contém a pasta ***lib***, a pasta ***Tests*** com os ficheiros ***disjointSet.h, disjointSet.cpp***, ***maze.h, maze.cpp*** e ***tests.cpp***, e os ficheiros ***CMakeLists*** e ***main.cpp***)

Deverá realizar a ficha respeitando a ordem das alíneas.

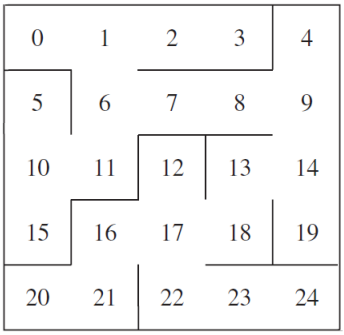
Enunciado

Considere a implementação de conjuntos disjuntos (de valores inteiros 0…n-1), de acordo com a classe **DisjointSets**, apresentada nas aulas, sendo a operação de pesquisa (**find**) realizada pelo método simples e a união de conjuntos (**unionSets**) pelo método fraco.
Considere um labirinto (classe **Maze**) representado como um tabuleiro de *n* linhas e *m* colunas, *nxm* células. É possível navegar de uma célula para outra adjacente, desde que não exista uma parede.
Pretende-se construir configurações de labirinto em que seja possível navegar de qualquer célula para qualquer célula. O procedimento pode ser o seguinte:

1. Considerar, inicialmente, o labirinto com todas as paredes (exemplo na figura ao lado)
   Assim, cada célula pertence a um conjunto distinto.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

2. Escolher aleatoriamente uma célula **x** e uma célula **y** adjacente a **x**. Se existir parede entre **x** e **y**, remover essa parede (equivale a unir os dois conjuntos). Repetir o processo até existir um único conjunto.
   A figura ao lado representa uma configuração possível.



As classes **DisjointSets** e **Maze** estão parcialmente definidas como indicado:

```
class DisjointSets
{
    vector<int> set;
public:
    DisjointSets();
    DisjointSets(int n);
    int find(int v) const;
    void unionSets(int root1, int root2);
    int getNumberOfSets() const;
};
```

```
class Maze {
    DisjointSets maze;
    int nrows=0;
    int ncols=0;
    vector<pair<int,int>> noWalls;
    vector<int> getNeighbours(int x) const;
public:
    Maze(int rows, int cols);
    void buildRandomMaze();
    DisjointSets getSets() const;
    void printMaze() const;
};
```

a) Implemente, na classe **DisjointSets**, o membro-função:

```
int DisjointSets::getNumberOfSets() const
```

Esta função retorna o número de conjuntos disjuntos existentes.

b) Implemente o construtor da classe **Maze**:

```
Maze::Maze(int rows, int cols)
```

Esta função inicializa o labirinto, considerando a existência de todas as paredes. Inicialize convenientemente os membros-dado ***nrows*** (número de linhas do labirinto), ***ncols*** (número de colunas do tabuleiro) e ***noWalls*** (vetor com identificação das paredes que vão sendo retiradas do labirinto, sendo uma parede representada por pair<x,y>, x e y é a numeração de uma célula)

c) Implemente, na classe **Maze**, o membro-função:

```
void Maze::buildRandomMaze()
```

Esta função gera uma possível configuração para a labirinto, de modo que seja possível navegar de qualquer célula para qualquer célula.

Nota: Este teste não falha. Verifique, pela visualização do labirinto, se a configuração está correta.