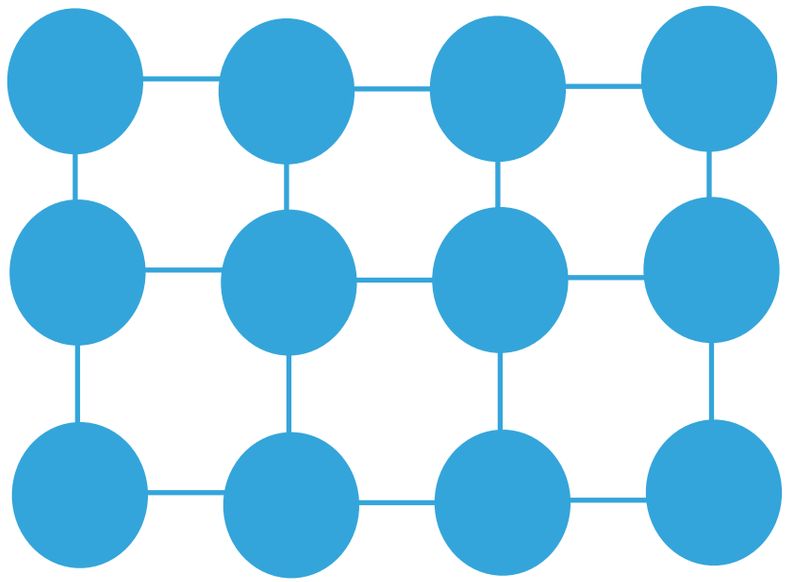
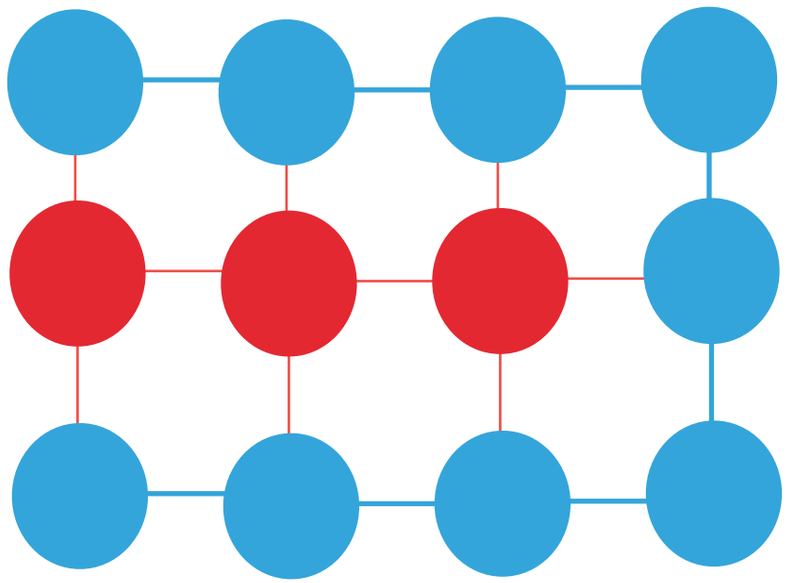
MATHEUS CONNOLYN

DUENDE PERDIDO

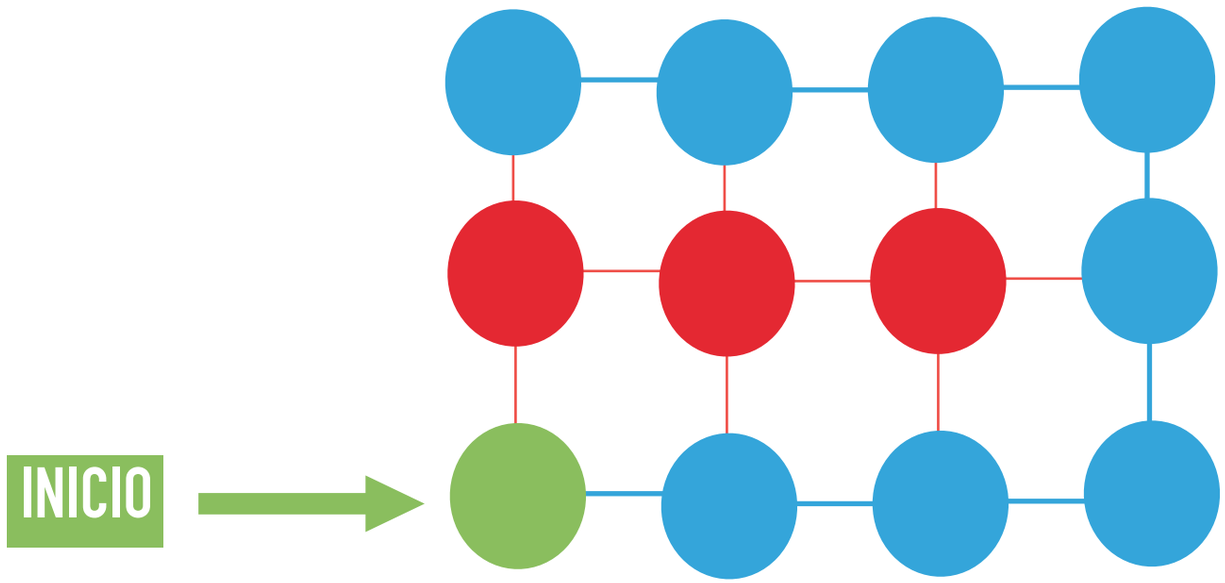
ESTRUTURA DO LABIRINTO



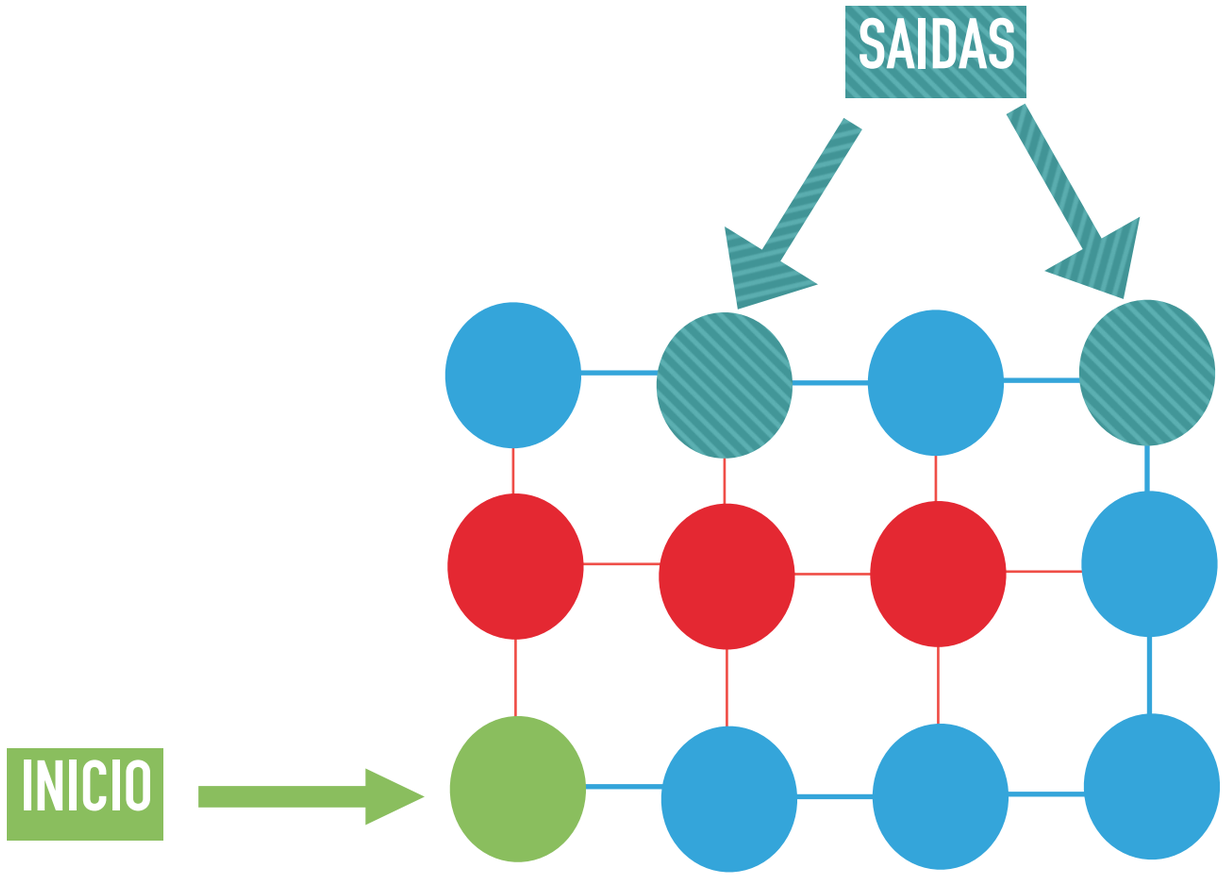
ESTRUTURA DO LABIRINTO



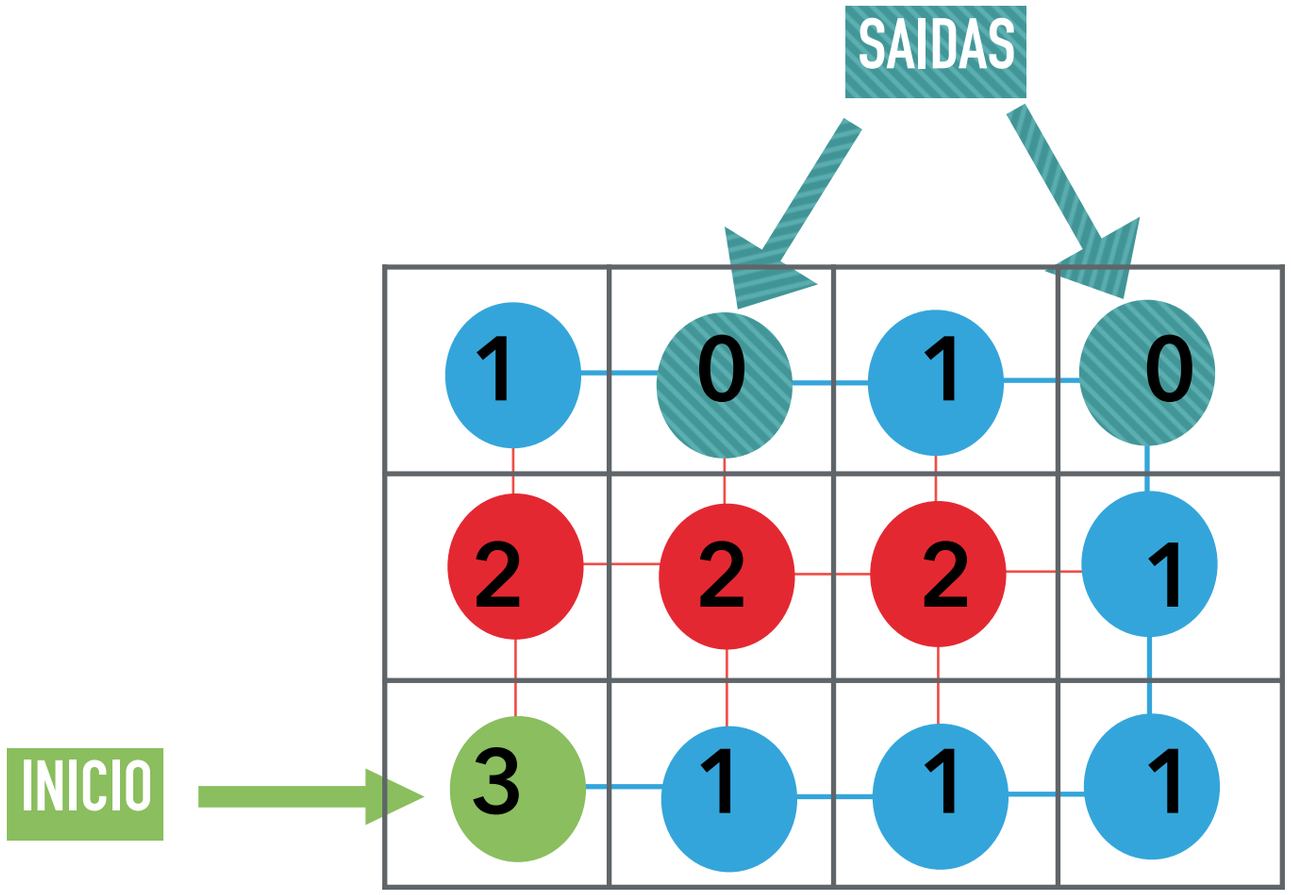
ESTRUTURA DO LABIRINTO



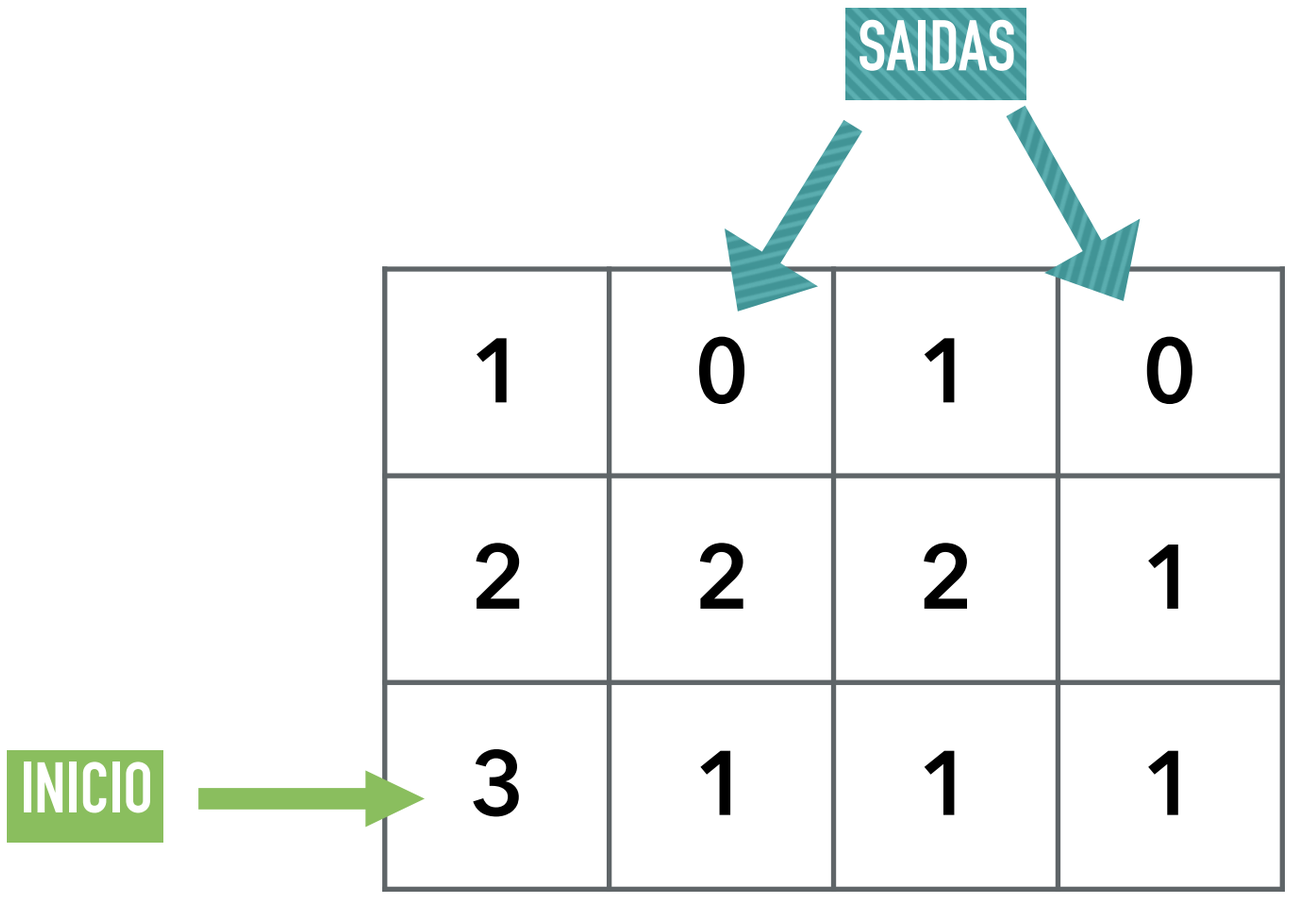
ESTRUTURA DO LABIRINTO



ESTRUTURA DO LABIRINTO

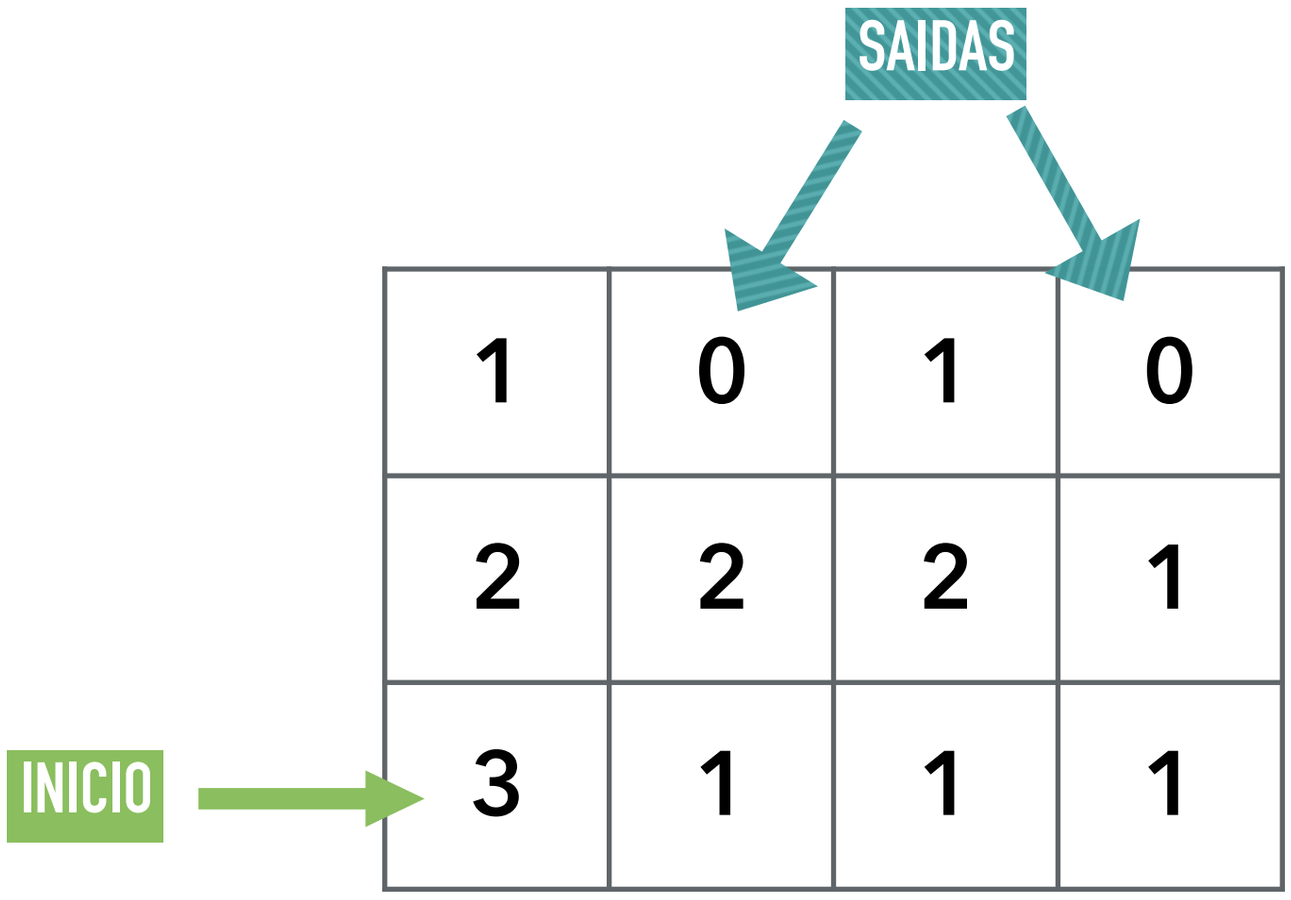


ESTRUTURA DO LABIRINTO



ESTRUTURA DO LABIRINTO

Dimensões: 3x4



ESTRUTURA DO LABIRINTO

Entrada:

```
4 5
0 1 1 1 1
0 2 2 2 1
2 1 1 1 1
1 1 1 3 1
```

TAMANHO DO MENOR CAMINHO

Saída:

8

ESTRATÉGIA

- Busca em largura
- Identificar pontos de saída
- Armazenar “custo" do caminho

CÓDIGO

```
// classe que representa um grafo direcionado
class Grafo
{
 int V;          // numero de vertices
 list<int> *adj;  // ponteiro para um vetor com as listas de adjacencias

 public:
  // construtor
  Grafo(int V);
  // metodo para adicionar uma aresta direcionada ao grafo
  void adicionaArestaDirecionada(int v, int w);
  // metodo para adicionar uma aresta nao direcionada ao grafo
  void adicionaArestaNaoDirecionada(int v, int w);
  // imprime os vertices visitados pela BFS
  void BFS(int s, bool *saidas);
};
```

CÓDIGO

```
void Grafo::BFS(int s, bool *saidas)
{
 // marca todos os vertices como nao visitados
 bool *visitado = new bool[V];
 int *distancia = new int[V];

 for ( int i = 0; i < V; i++ )
  visitado[i] = false;

// cria uma fila para a BFS
list<int> fila;

// marca o vertice atual como visitado e enfileira
visitado[s] = true;
distancia[s] = 0;
fila.push_back(s);
```

CÓDIGO

```
// se um vizinho ainda nao foi visitado, entao marca como visitado
// e enfileira
for( i = adj[s].begin(); i != adj[s].end(); ++i )
{
 if ( !visitado[*i] )
 {
  visitado[*i] = true;
  distancia[*i] = distancia[s] + 1;

  if(saidas[*i]){

    cout<<distancia[*i];
    return;
  }
  fila.push_back(*i);
 }
}
```

CÓDIGO

```
void Grafo::BFS(int s, bool *saidas)
{
 // marca todos os vertices como nao visitados
 bool *visitado = new bool[V];
 int *distancia = new int[V];

 for ( int i = 0; i < V; i++ )
  visitado[i] = false;

// cria uma fila para a BFS
list<int> fila;

// marca o vertice atual como visitado e enfileira
visitado[s] = true;
distancia[s] = 0;
fila.push_back(s);
```

CÓDIGO

```
// obtem todos os vizinhos do vertice s
// se um vizinho ainda nao foi visitado, entao marca como visitado
// e enfileira
for( i = adj[s].begin(); i != adj[s].end(); ++i )
{
 if ( !visitado[*i] )
 {
  visitado[*i] = true;
  distancia[*i] = distancia[s] + 1;

  if(saidas[*i]){

    cout<<distancia[*i];
    return;
  }
  fila.push_back(*i);
 }
}
}
```

CÓDIGO

```
int main()
{

  int n, m, s=0;

  cin>>n >> m;

  //labirinto em forma de matriz
  int lab[n][m];
  //Saidas em forma de  vetor
  bool saidas [n*m];

for (int i = 0; i < n; ++i)
{
  for (int j = 0; j < m; ++j)
  {
    cin>> lab[i][j];

    //localiza o inicio
    if(lab[i][j] == 3) s = converte(i, j, m);

    //localiza as saidas
    saidas[converte(i,j,m)] = lab[i][j] == 0 ;

  }
}
```

CÓDIGO

```
int converte(int i, int j, int m)
{
    return i*(m) + j;
}
```

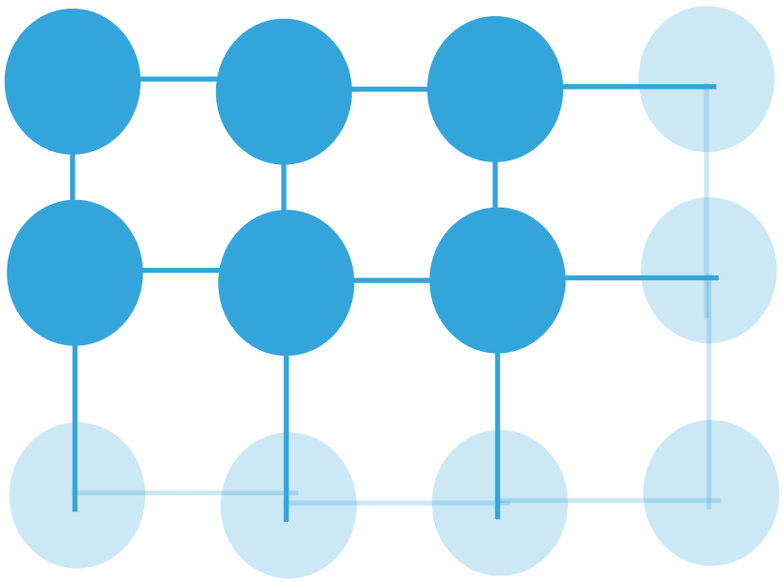
CÓDIGO

```
// cria um grafo
 Grafo g(n*m);

 //Adiciona as aretas
 for (int i = 0; i < n-1; ++i)
 {
   for (int j = 0; j < m-1; ++j)
   {

     //Se esse no nao for dois
     if(lab[i][j]!=2)
     {

       //se o no abaixo nao for dois
       if(lab[i+1][j] !=2)
       {
         g.adicionaArestaNaoDirecionada(
           converte(i,j,m),
           converte(i+1,j,m)
           );

       }
```

CÓDIGO

```
            //se o no ao lado nao for dois
            if(lab[i][j+1] !=2)
            {

              g.adicionaArestaNaoDirecionada(
                converte(i,j,m),
                converte(i,j+1,m)
                );
            }
          }
        }
      }
```

CÓDIGO



CÓDIGO

```
//vertices do lado direito
for (int i = 0; i < n-1; ++i)
{
   if(lab[i][m-1] !=2 && lab[i+1][m-1] !=2 )
      g.adicionaArestaNaoDirecionada(
         converte(i,m-1,m),
         converte(i+1,m-1,m)
      );
}

//vertices do lado inferior
for (int j = 0; j < m-1; ++j)
{
   if(lab[n-1][j] !=2 && lab[n-1][j+1] !=2 )
      g.adicionaArestaNaoDirecionada(
         converte(n-1,j,m),
         converte(n-1,j+1,m)
      );
}
```

CÓDIGO

```
g.BFS(s, saidas);
```

RESULTADO

: Submissões
Duende Perdido

ID		DATA	PROBLEM	RESULT	TIME	MEM	LING
20789272	☐	2017-12-14 00:42:23	Duende Perdido	accepted edit run	0.00	2.8M	C++ 4.3.2

Invert
Selected submissions: - Execute

RESULTADO

RANK	DATA	Usuário:	RESULT	TIME	MEM	LING
1	2009-12-18 22:30:15	Kleber da Silva Santos[USJT]	accepted	0.00	2.7M	C++ 4.3.2
2	2010-01-11 20:14:47	marcelo	accepted	0.00	2.7M	C++ 4.3.2
3	2010-01-17 18:30:42	André	accepted	0.00	2.8M	C++ 4.3.2
4	2010-01-24 19:40:17	selrahc	accepted	0.00	2.7M	C++ 4.3.2
5	2010-02-17 17:06:49	Chandlli	accepted	0.00	2.6M	C++ 4.3.2
6	2010-03-26 16:58:38	Vitor Castro Veloso Soares [INF-UFG]	accepted	0.00	2.8M	C++ 4.3.2
7	2010-04-06 19:19:07	Rômulo	accepted	0.00	2.7M	C++ 4.3.2
8	2010-04-09 14:27:59	[UNICAMP] Davi Costa	accepted	0.00	2.8M	C++ 4.3.2
9	2010-04-09 19:47:34	Henrique Gaspar Nogueira	accepted	0.00	2.8M	C++ 4.3.2
10	2010-04-09 20:33:59	Thiago Caetano	accepted	0.00	2.8M	C++ 4.3.2

DÚVIDAS?