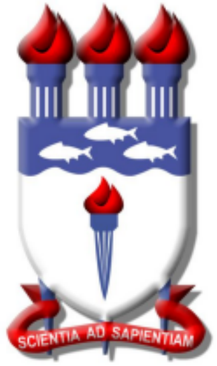


Estrutura de Dados / Programação 2
A*

Davi Celestino

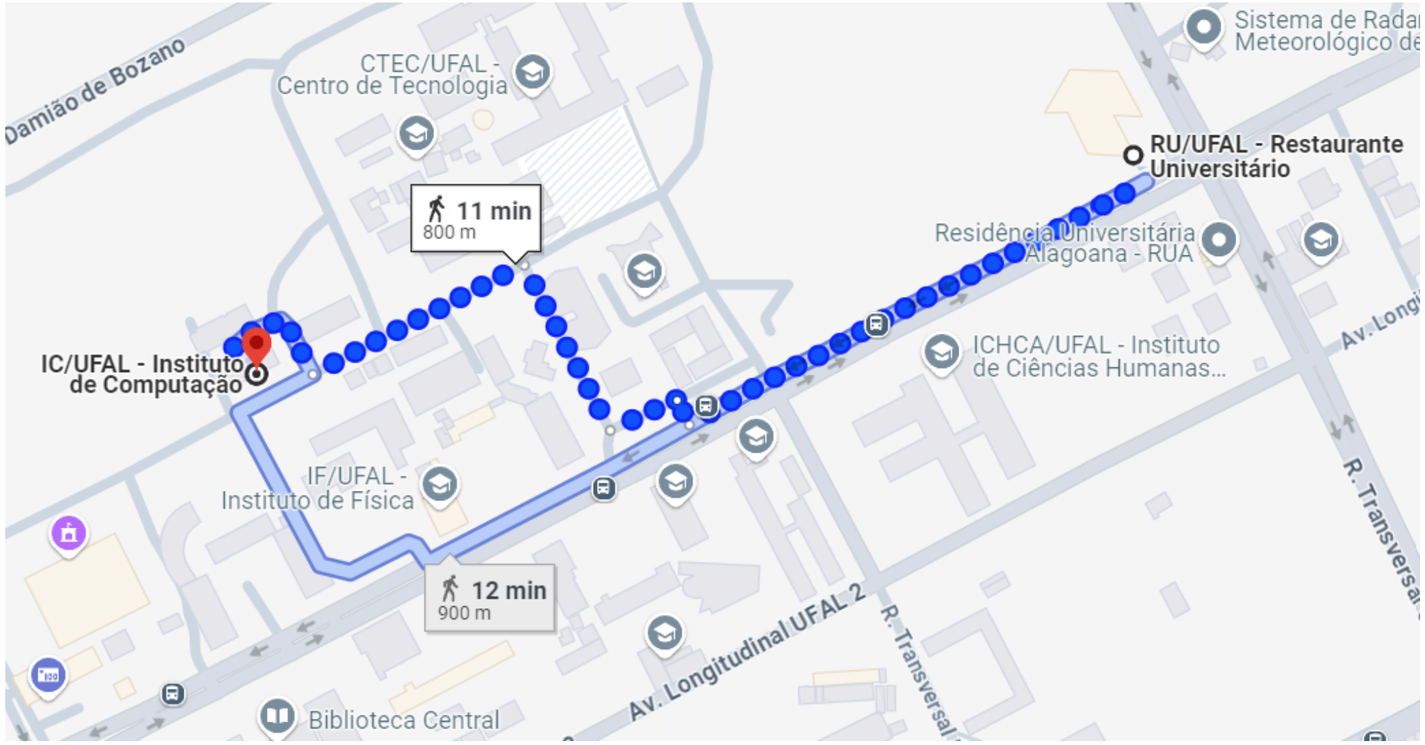
Humberto Barros

João Tenório

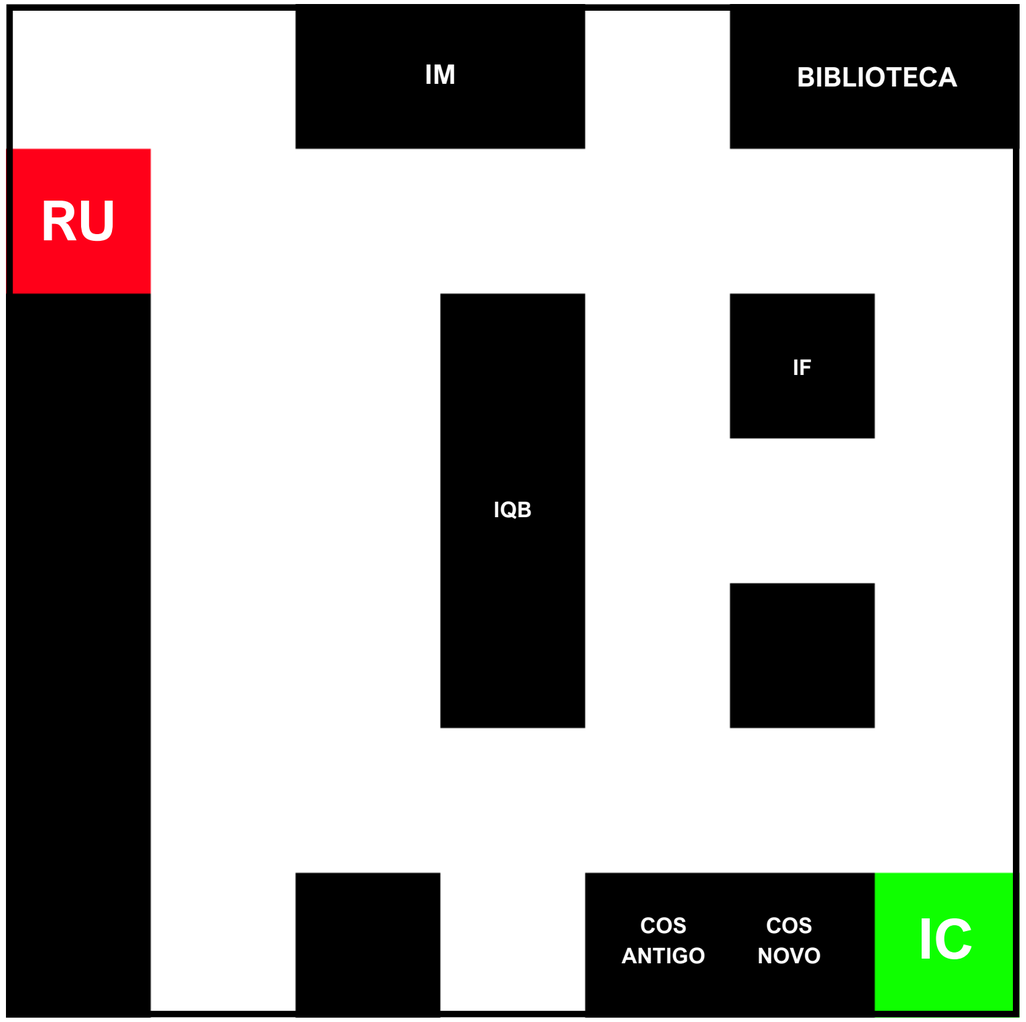
Lihttps://github.com/Davicsb/Estrutura-de-Dadosnk

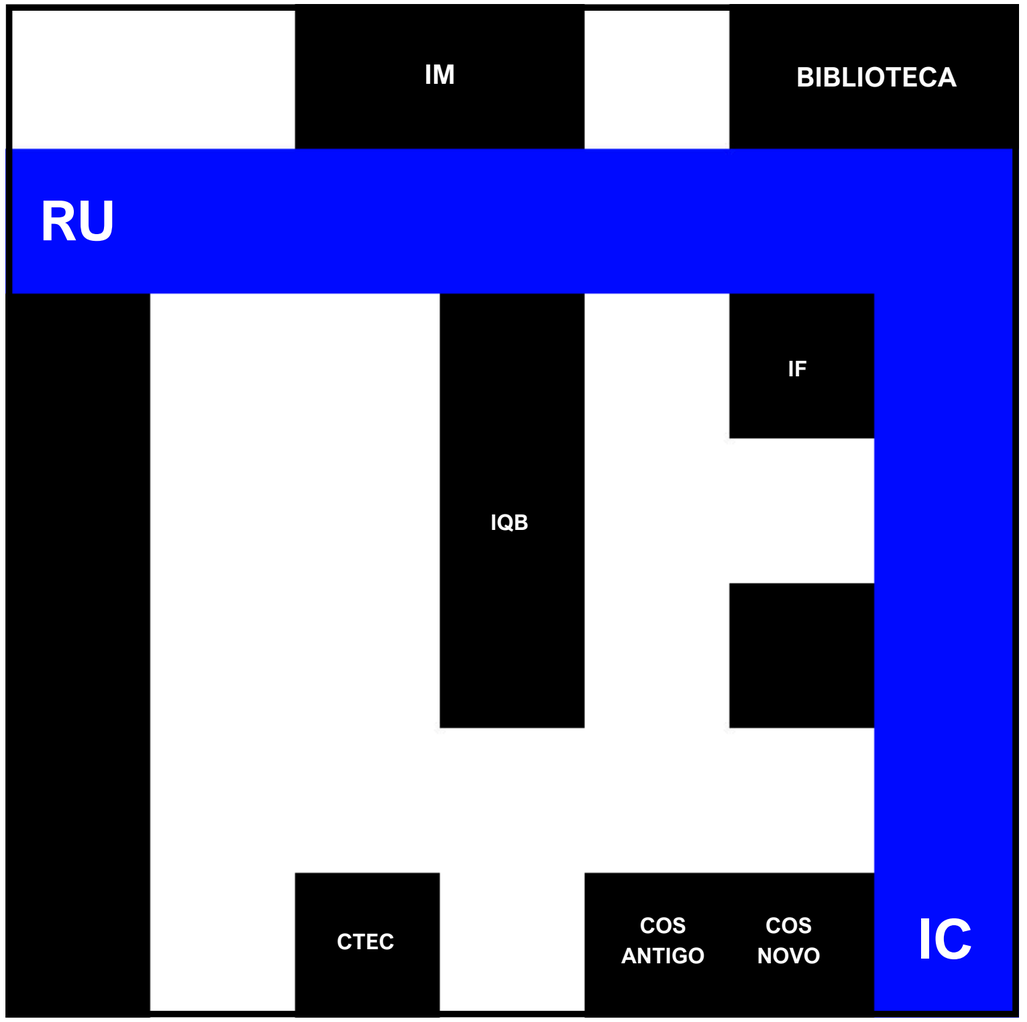
Introdução

- Como podemos achar o **menor** caminho de um ponto a outro no mapa?



- Jayme estava no RU e está atrasado para sua apresentação do Huffman


IM
BIBLIOTECA
RU
IF
IQB
COS
ANTIGO
COS
NOVO
IC

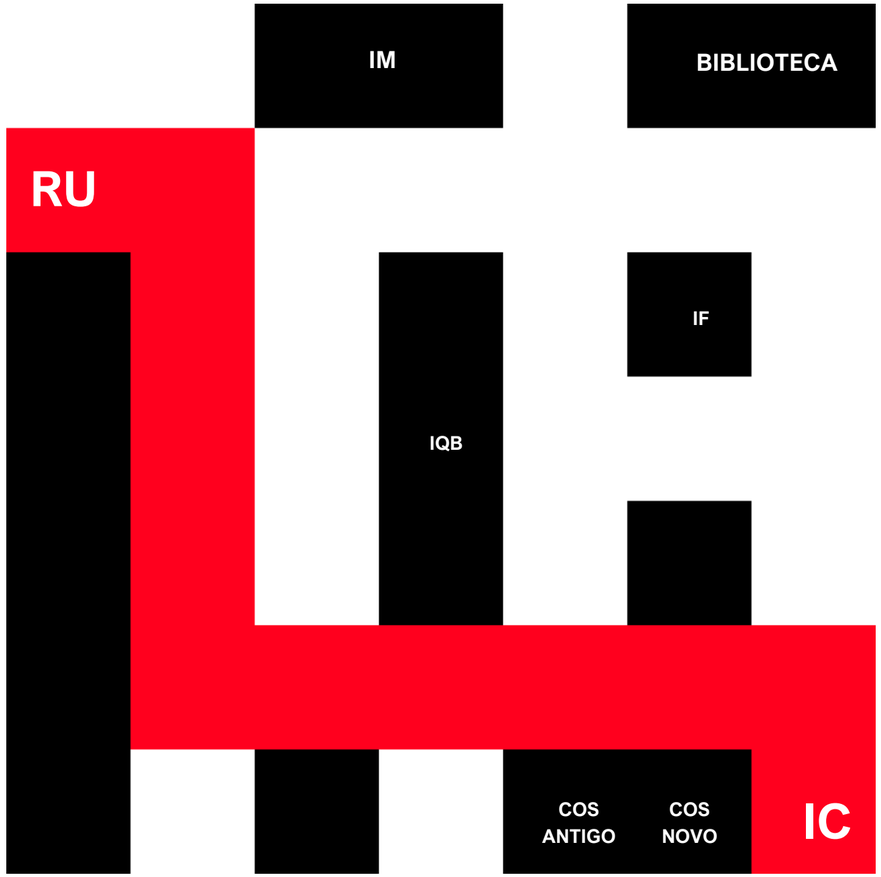

IM
BIBLIOTECA
RU
IF
IQB
CTEC
COS
ANTIGO
COS
NOVO
IC

Nada eficiente...

A* !!

A*

- Um método que busca o caminho **mais curto** entre dois pontos.

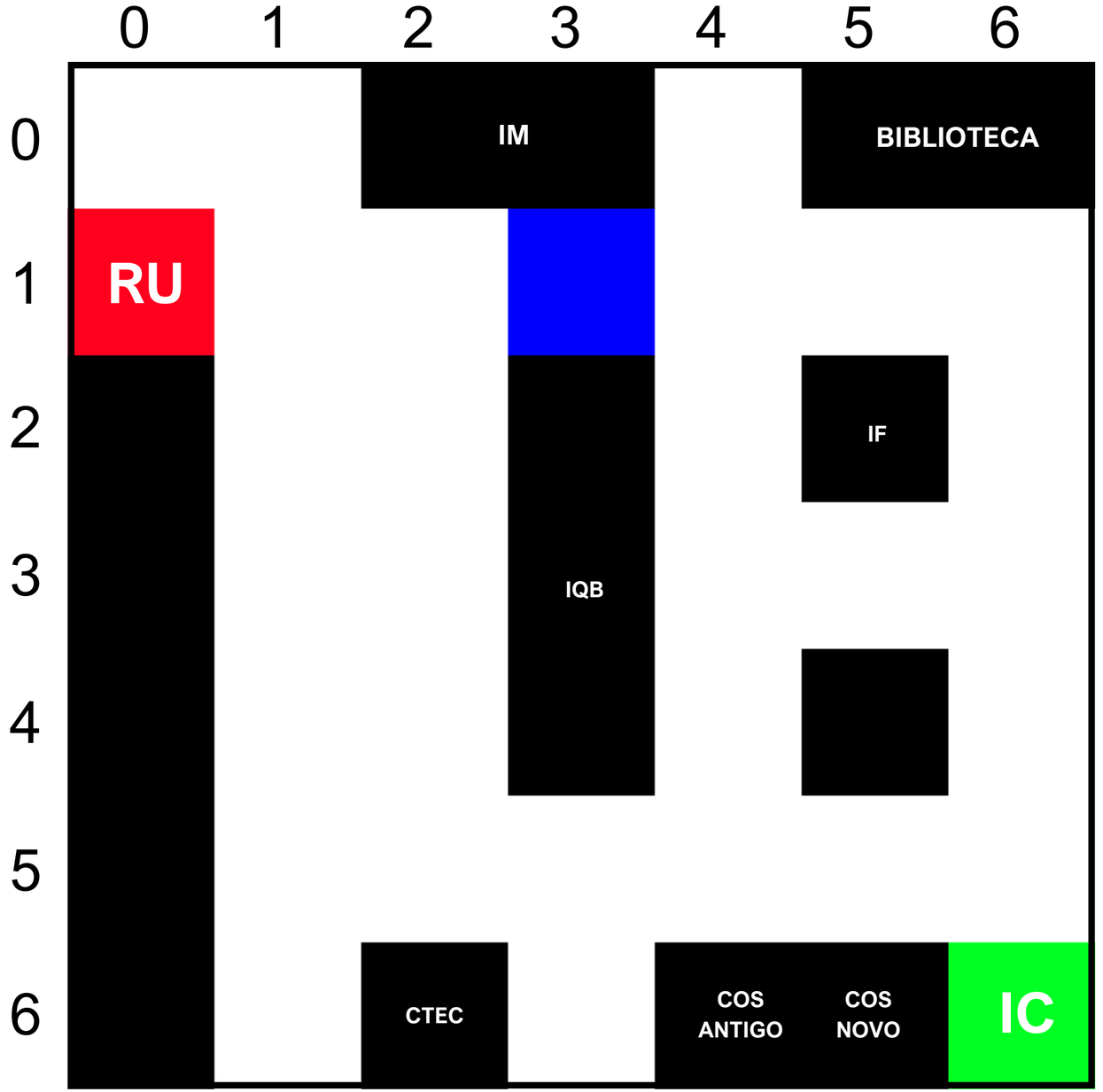


Definições

- **Partida:** célula de inicio do algoritmo.
- **Destino:** célula objetivo do algoritmo.
- **Custo G:** custo de “passos” da partida até a célula analisada acumulados.
- **Custo H:** custo de “passos restantes” da célula até o destino.
- **Custo F:** soma dos custos H e G.
- **Lista aberta:** a lista aberta contém pontos (coordenadas) que ainda não tiveram o seu custo calculado.
- **Lista fechada:** a lista fechada contém objetos que já tiveram seu custo calculados.

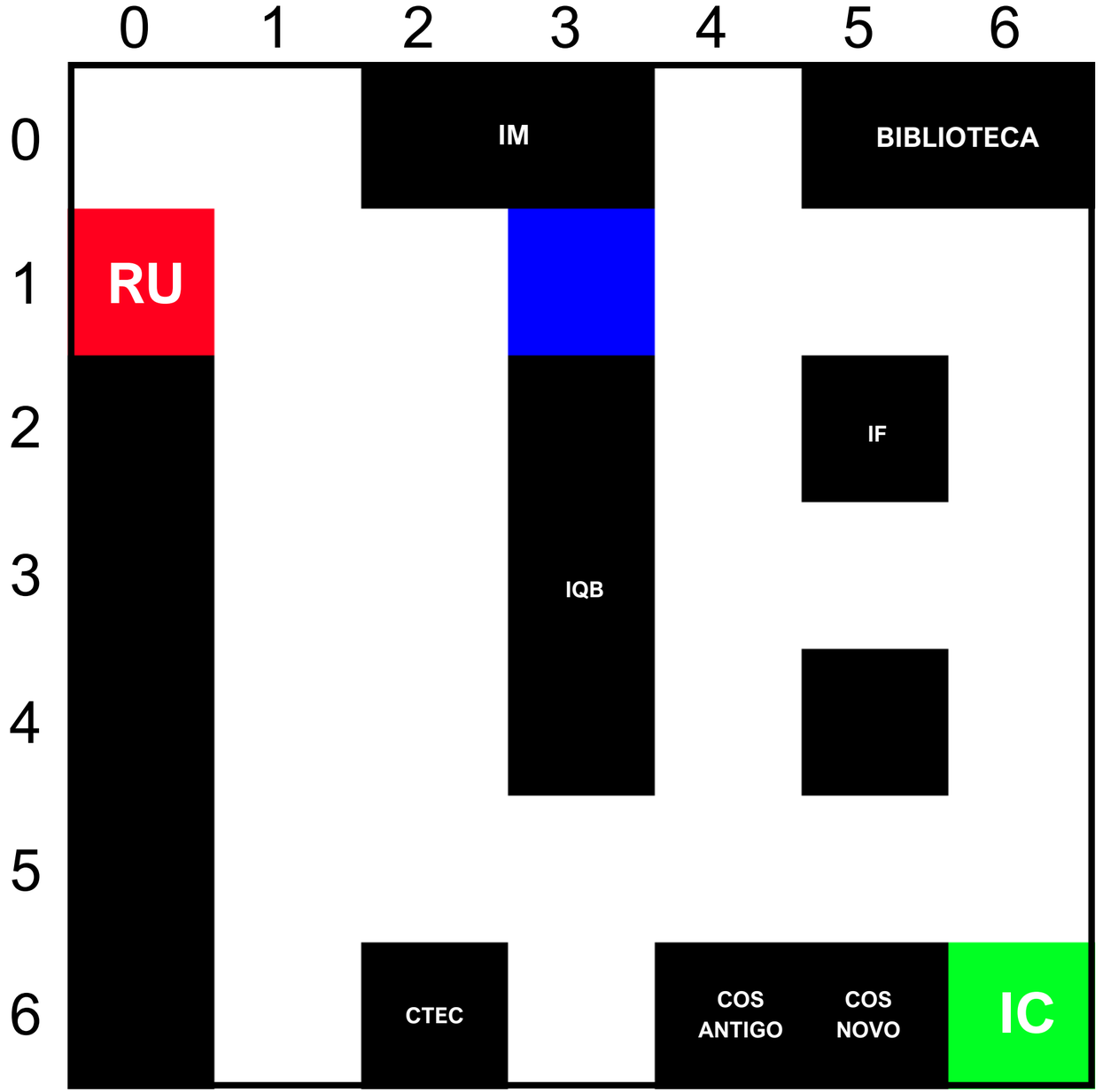
Definições

- Qual a **partida**?
- Qual o **destino**?
- Qual o custo **G**, **H** e **F** da célula em **azul**?



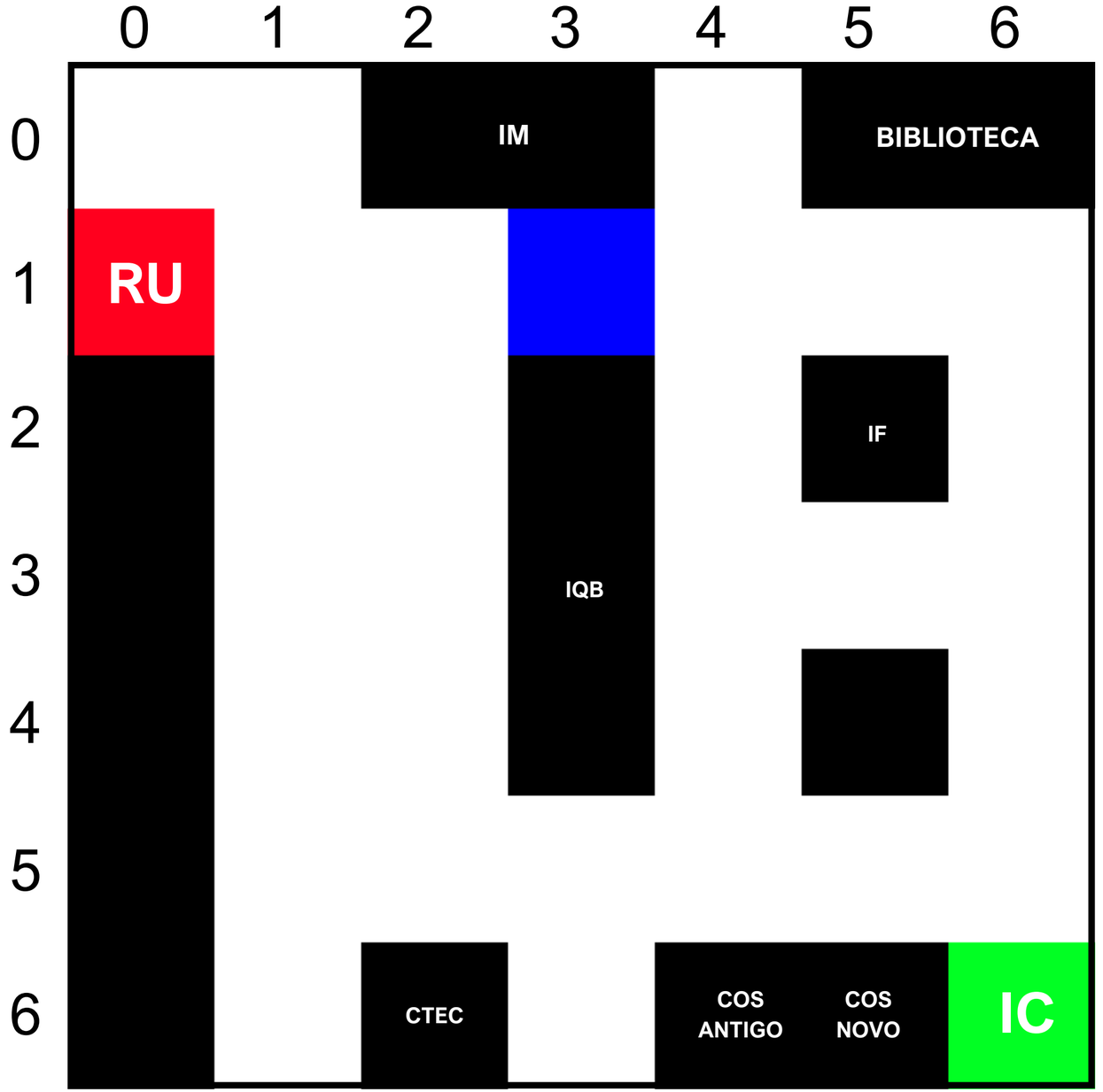
Definições

- Qual a **partida**?
  - Célula (1,0)
- Qual o **destino**?
- Qual o custo **G**, **H** e **F** da célula em **azul**?



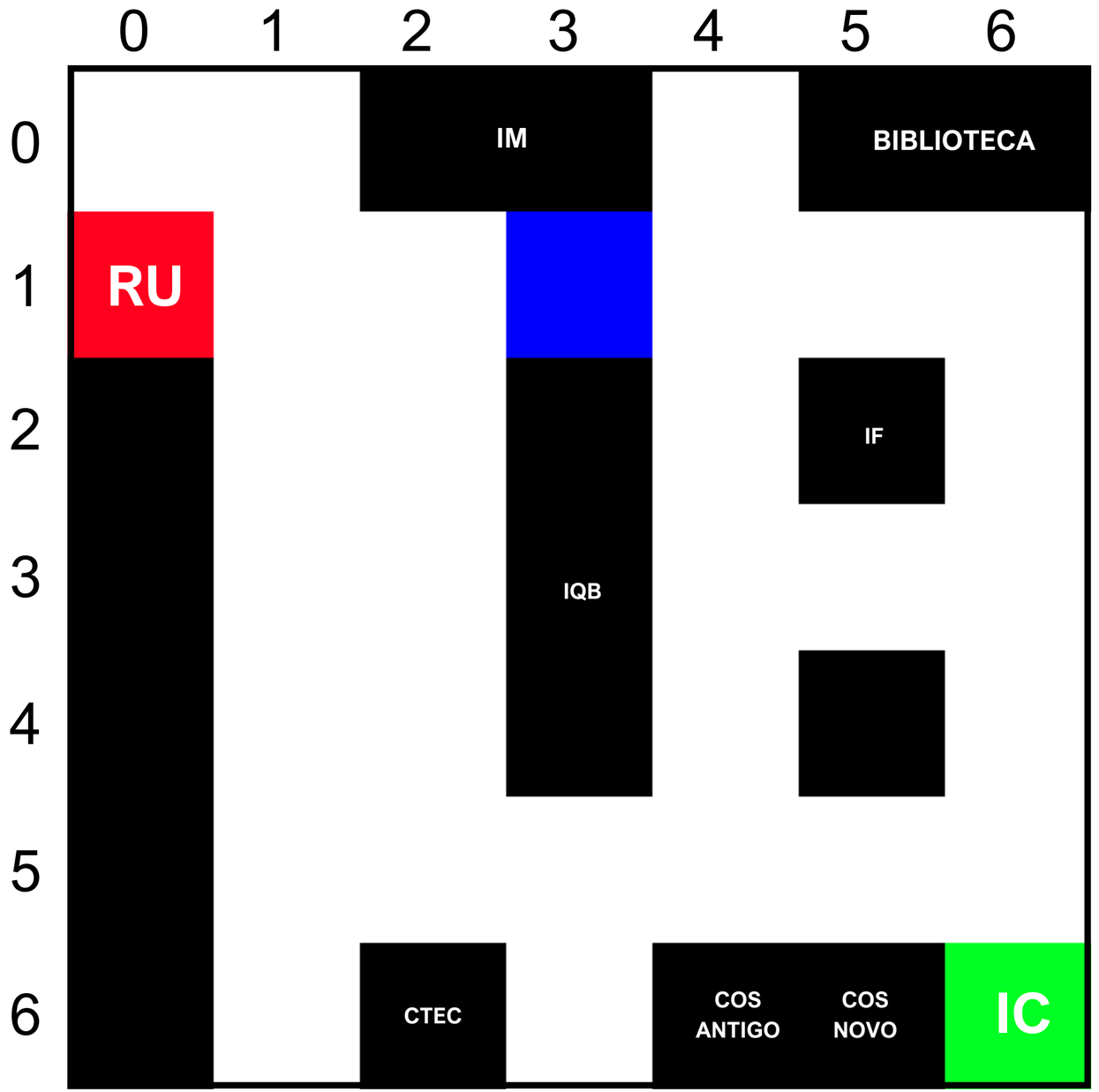
Definições

- Qual a **partida**?
  - Célula (1,0)
- Qual o **destino**?
  - Célula (6,6)
- Qual o custo **G**, **H** e **F** da célula em **azul**?



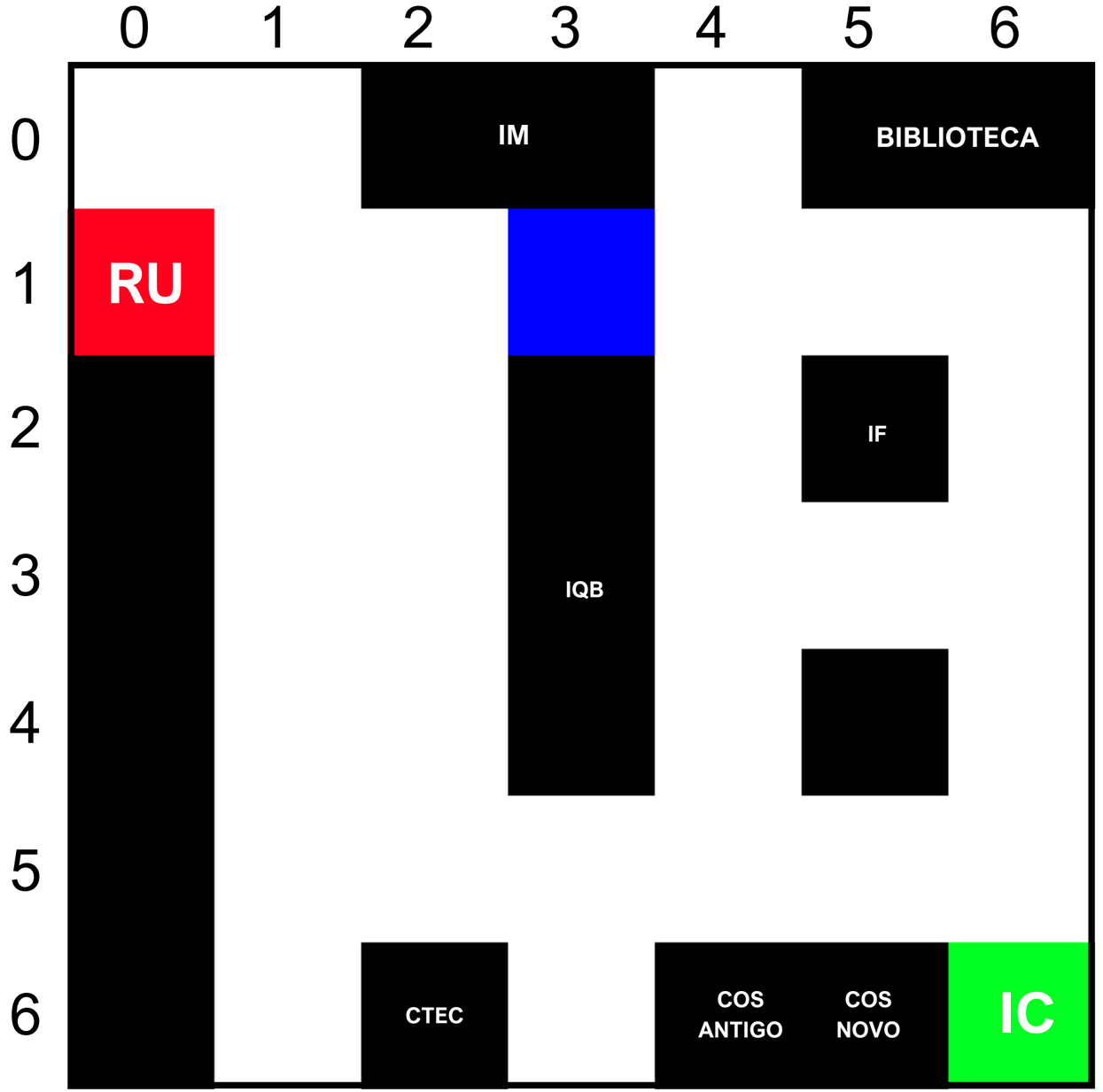
Definições

- Qual a **partida**?
  - Célula (1,0)
- Qual o **destino**?
  - Célula (6,6)
- Qual o custo **G**, **H** e **F** da célula em **azul**?
  - G = 3



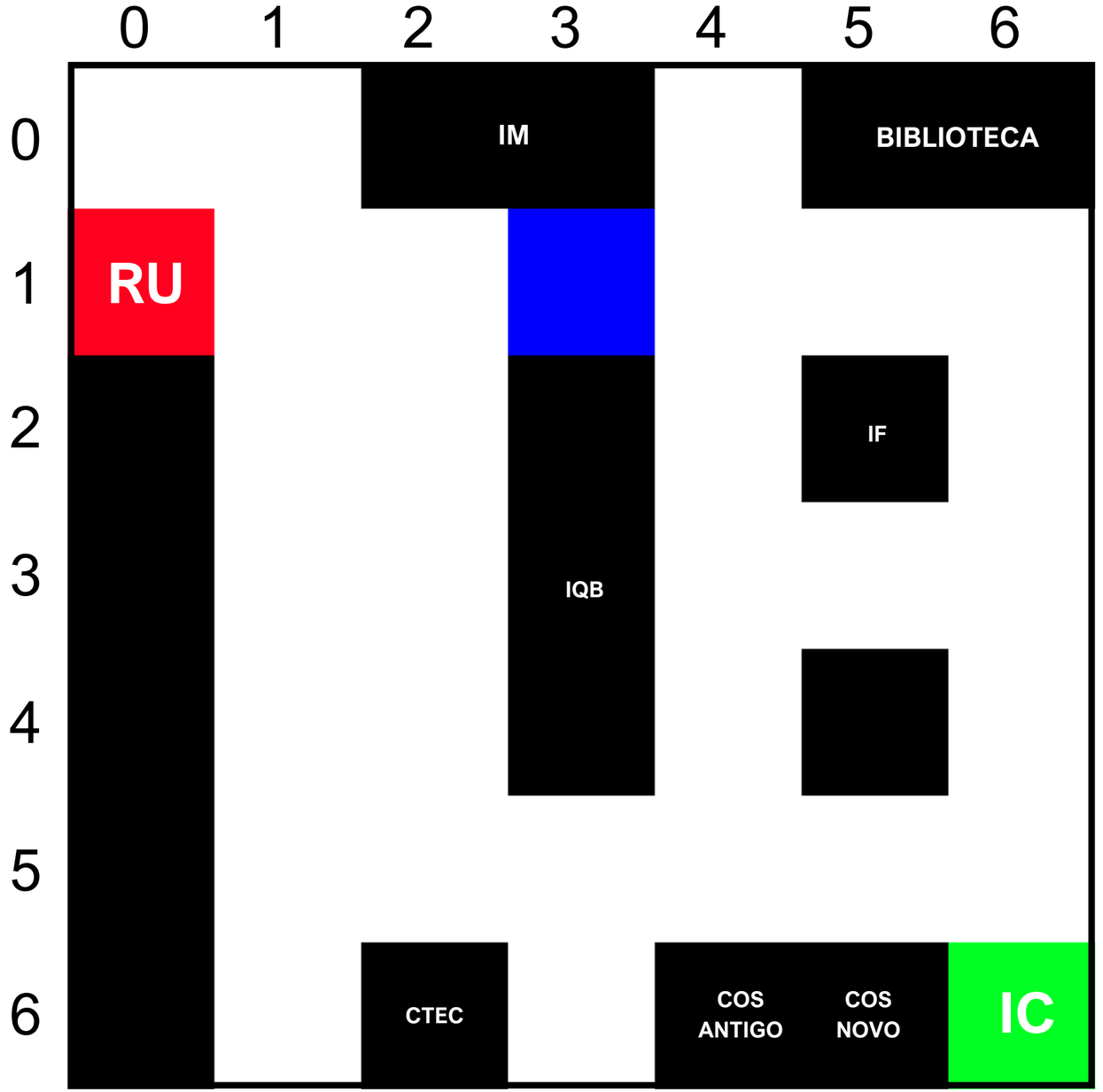
Definições

- Qual a **partida**?
  - Célula (1,0)
- Qual o **destino**?
  - Célula (6,6)
- Qual o custo **G**, **H** e **F** da célula em **azul**?
  - G = 3
  - H = 8



Definições

- Qual a **partida**?
  - Célula (1,0)
- Qual o **destino**?
  - Célula (6,6)
- Qual o custo **G**, **H** e **F** da célula em **azul**?
  - G = 3
  - H = 8
  - F = 11



Código

```
typedef struct {
    int linha;
    int coluna;
} Cell;

typedef struct {
    Cell cell;
    int f;
    int g;
    int h;
} Node;
```

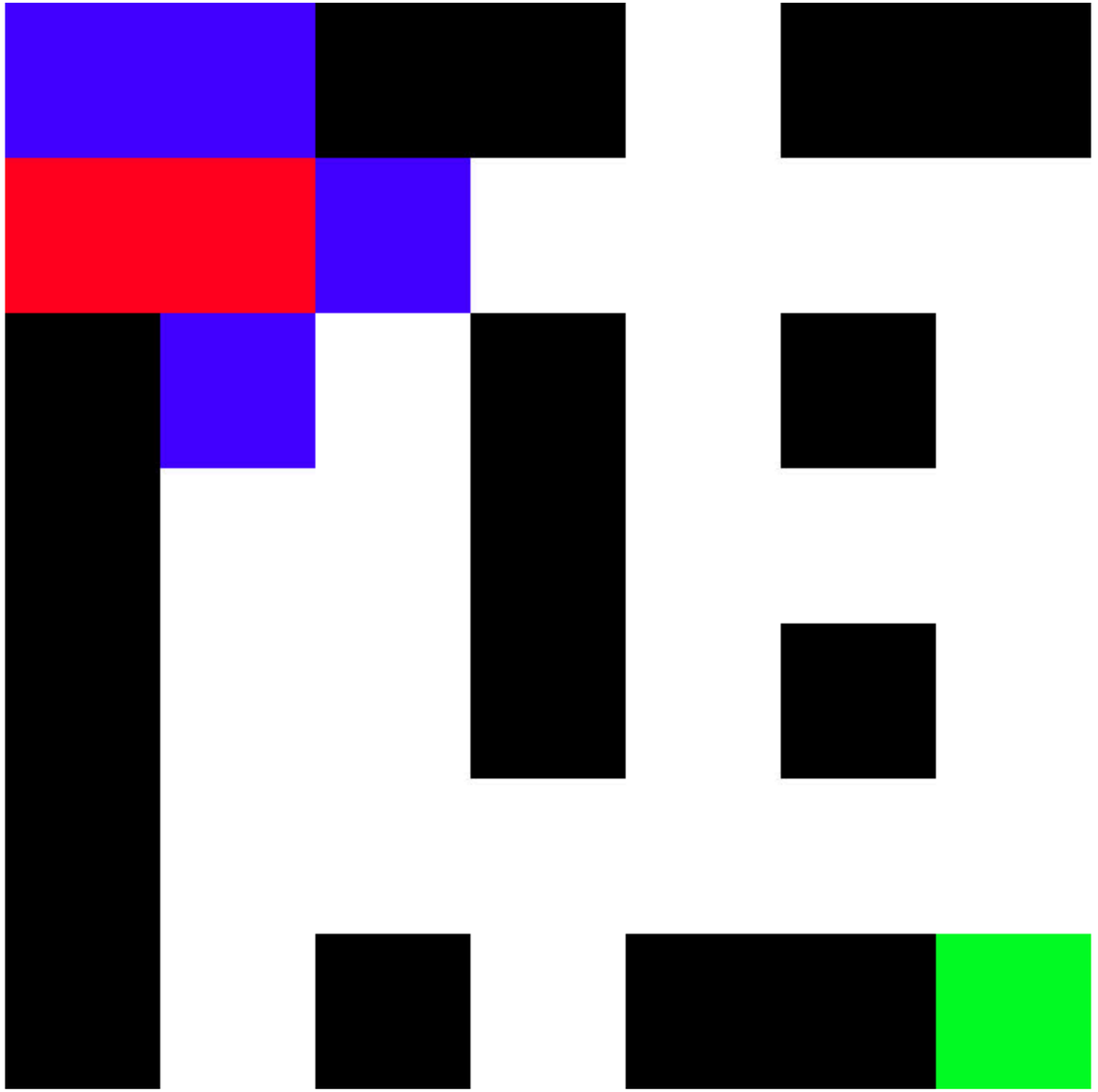
```
int maze[LINHAS][COLUNAS] = {
    {1, 1, 0, 0, 1, 0, 0},
    {1, 1, 1, 1, 1, 1, 1},
    {0, 1, 1, 0, 1, 0, 1},
    {0, 1, 1, 0, 1, 1, 1},
    {0, 1, 1, 0, 1, 0, 1},
    {0, 1, 1, 1, 1, 1, 1},
    {0, 1, 0, 1, 0, 0, 1}
};

Cell start = {1, 0};
Cell destination = {6, 6};
```

```
void aestrela(int maze[LINHAS][COLUNAS], Cell start, Cell destination, Cell
caminho[MAX_CELLS], int* caminho_length) {

  int f_score[LINHAS][COLUNAS];
  int g_score[LINHAS][COLUNAS];
  PriorityQueue openList;
  initQueue(&openList);

  int closedList[LINHAS][COLUNAS] = {0};

  for (int i = 0; i < LINHAS; i++) {
    for (int j = 0; j < COLUNAS; j++) {
      f_score[i][j] = INT_MAX;
      g_score[i][j] = INT_MAX;
    }
  }
```

```
  g_score[start.linha][start.coluna] = 0;
  f_score[start.linha][start.coluna] = g_score[start.linha][start.coluna] +
h_score(start, destination);

  enqueue(&openList, (Node){start, f_score[start.linha][start.coluna],
g_score[start.linha][start.coluna], h_score(start, destination)});

  Cell caminho_map[LINHAS][COLUNAS];
  for (int i = 0; i < LINHAS; i++) {
    for (int j = 0; j < COLUNAS; j++) {
      caminho_map[i][j] = (Cell){-1, -1};
    }
  }
```

```
while (!isEmpty(&openList)) {
    Node current = dequeue(&openList);

    closedList[current.cell.linha][current.cell.coluna] = 1;

    if (current.cell.linha == destination.linha && current.cell.coluna == destination.coluna) {
      Cell c = destination;
      *caminho_length = 0;
      while (c.linha != start.linha || c.coluna != start.coluna) {
        caminho[(*caminho_length)++] = c;
        c = caminho_map[c.linha][c.coluna];
      }
      caminho[(*caminho_length)++] = start;
      return;
    }

    Cell directions[] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
    for (int i = 0; i < 4; i++) {
      Cell neighbor = {current.cell.linha + directions[i].linha, current.cell.coluna + directions[i].coluna};

      if (isValid(maze, neighbor) && closedList[neighbor.linha][neighbor.coluna] == 0) {
        int novo_g_score = g_score[current.cell.linha][current.cell.coluna] + 1;
        int novo_f_score = novo_g_score + h_score(neighbor, destination);

        if (novo_f_score < f_score[neighbor.linha][neighbor.coluna]) {
          f_score[neighbor.linha][neighbor.coluna] = novo_f_score;
          g_score[neighbor.linha][neighbor.coluna] = novo_g_score;
          caminho_map[neighbor.linha][neighbor.coluna] = current.cell;

          enqueue(&openList, (Node){neighbor, novo_f_score, novo_g_score, h_score(neighbor, destination)});
        }
      }
    }
  }

  *caminho_length = 0; // Nenhum caminho encontrado
}
```

Animação



De volta à Motivação

- Pegando o caminho mais rápido, graças ao algoritmo, Jayme chegou a tempo de apresentar e começará a **perder pontos pelo professor Márcio**.