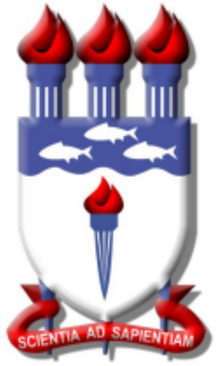


Estrutura de Dados / Programação 2
A*

Davi Celestino

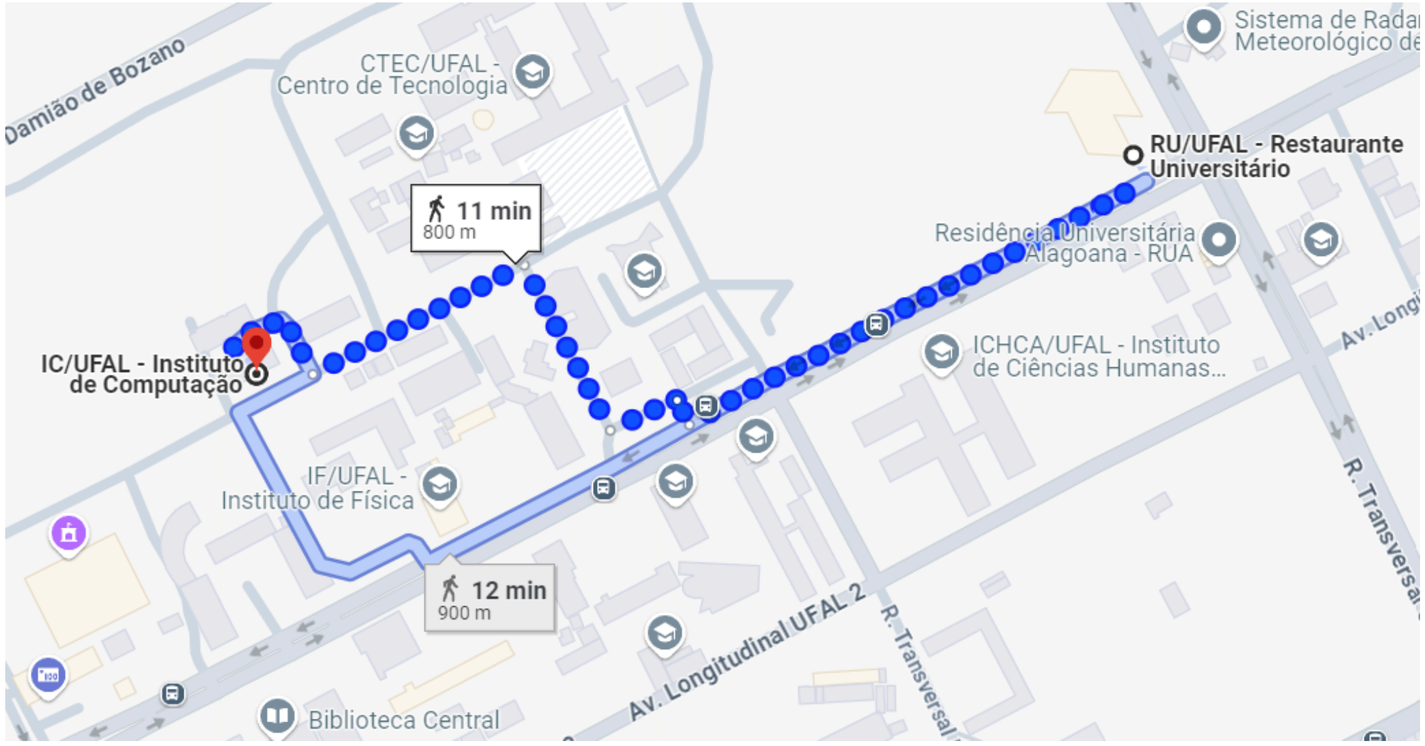
Humberto Barros

João Tenório

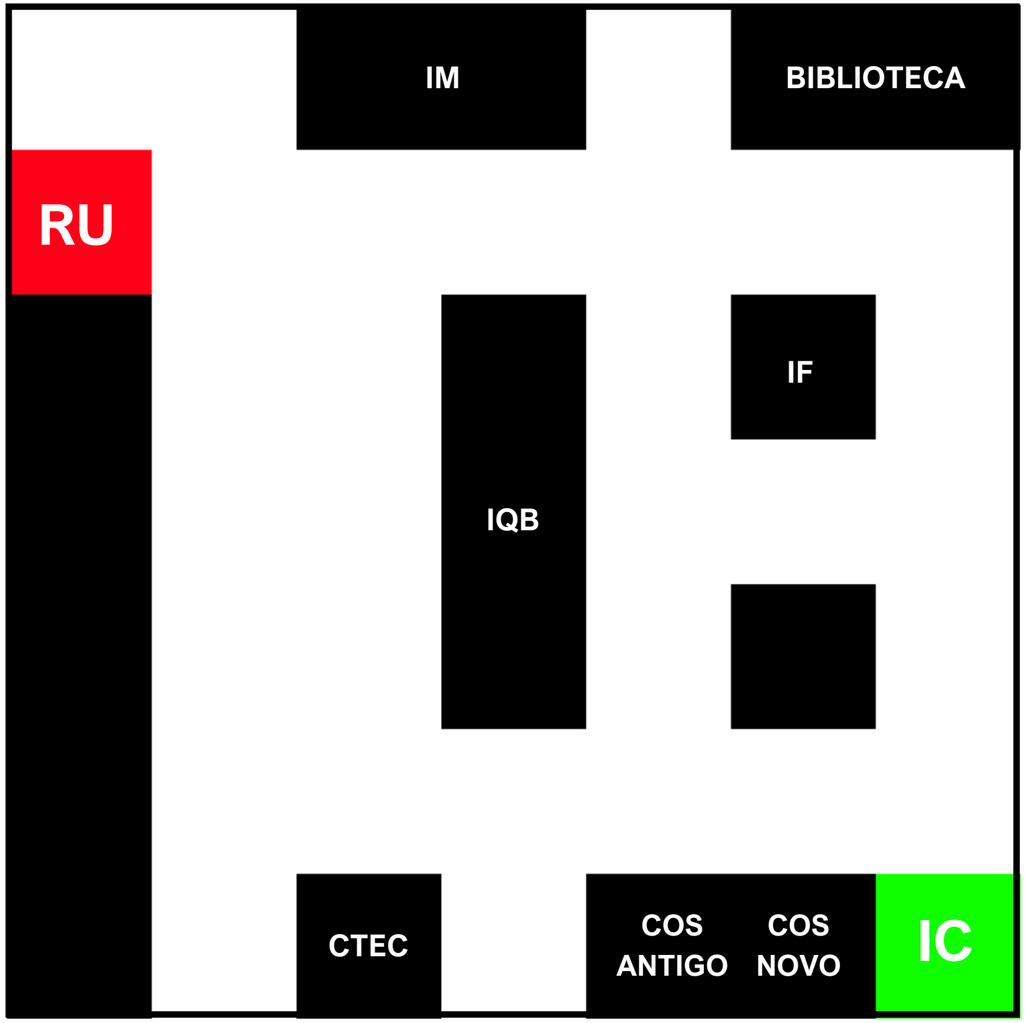
Lihttps://github.com/Davicsb/Estrutura-de-Dadosnk

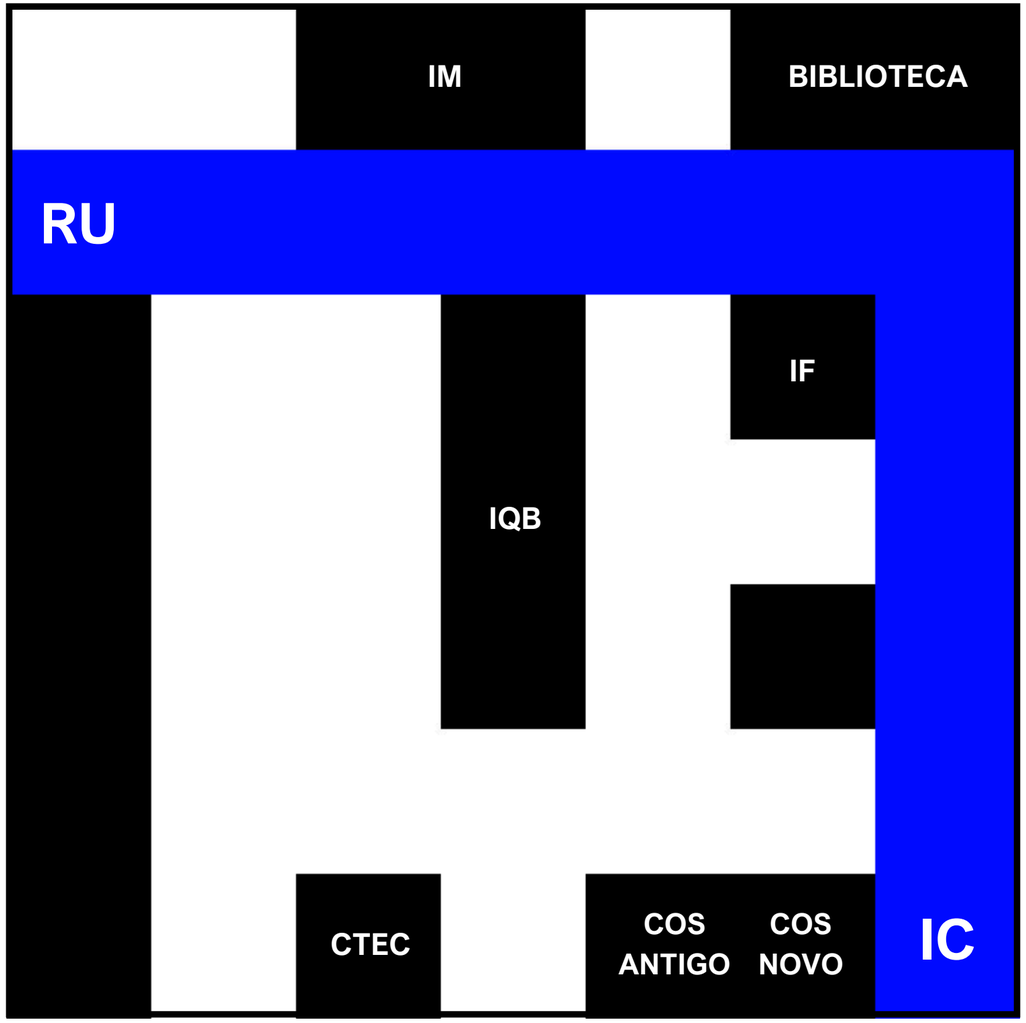
Introdução

- 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
CTEC
COS
ANTIGO
COS
NOVO
IC

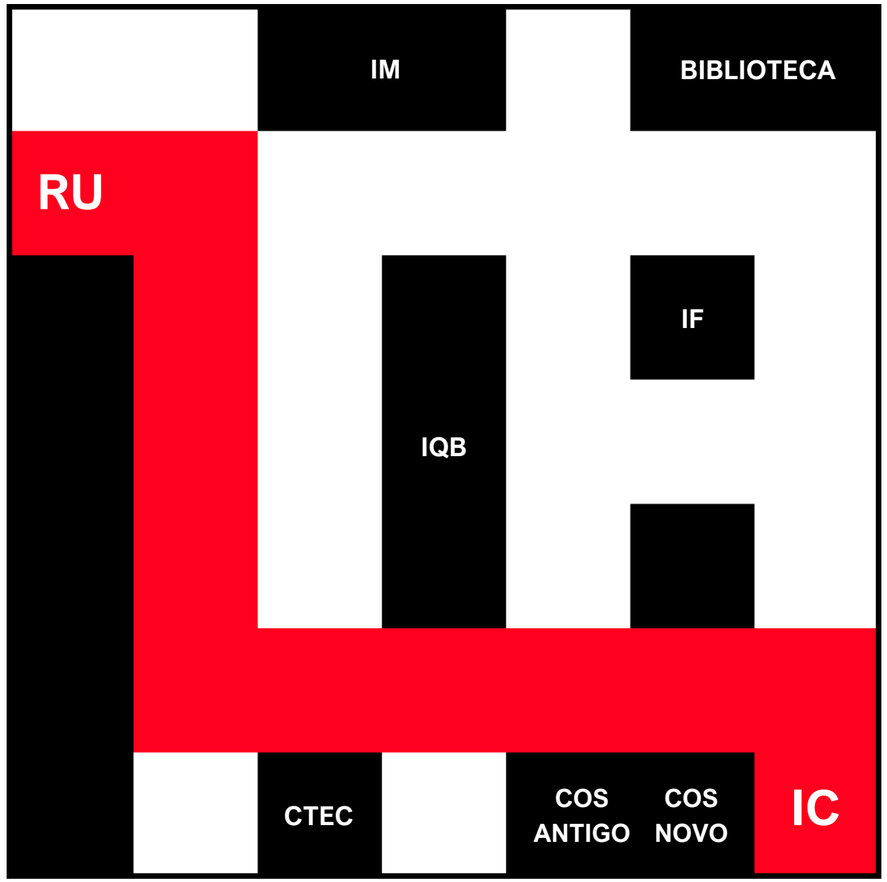

IM
BIBLIOTECA
RU
IF
IQB
CTEC
COS
ANTIGO
COS
NOVO
IC

Nada eficiente...

A* !!

A*

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

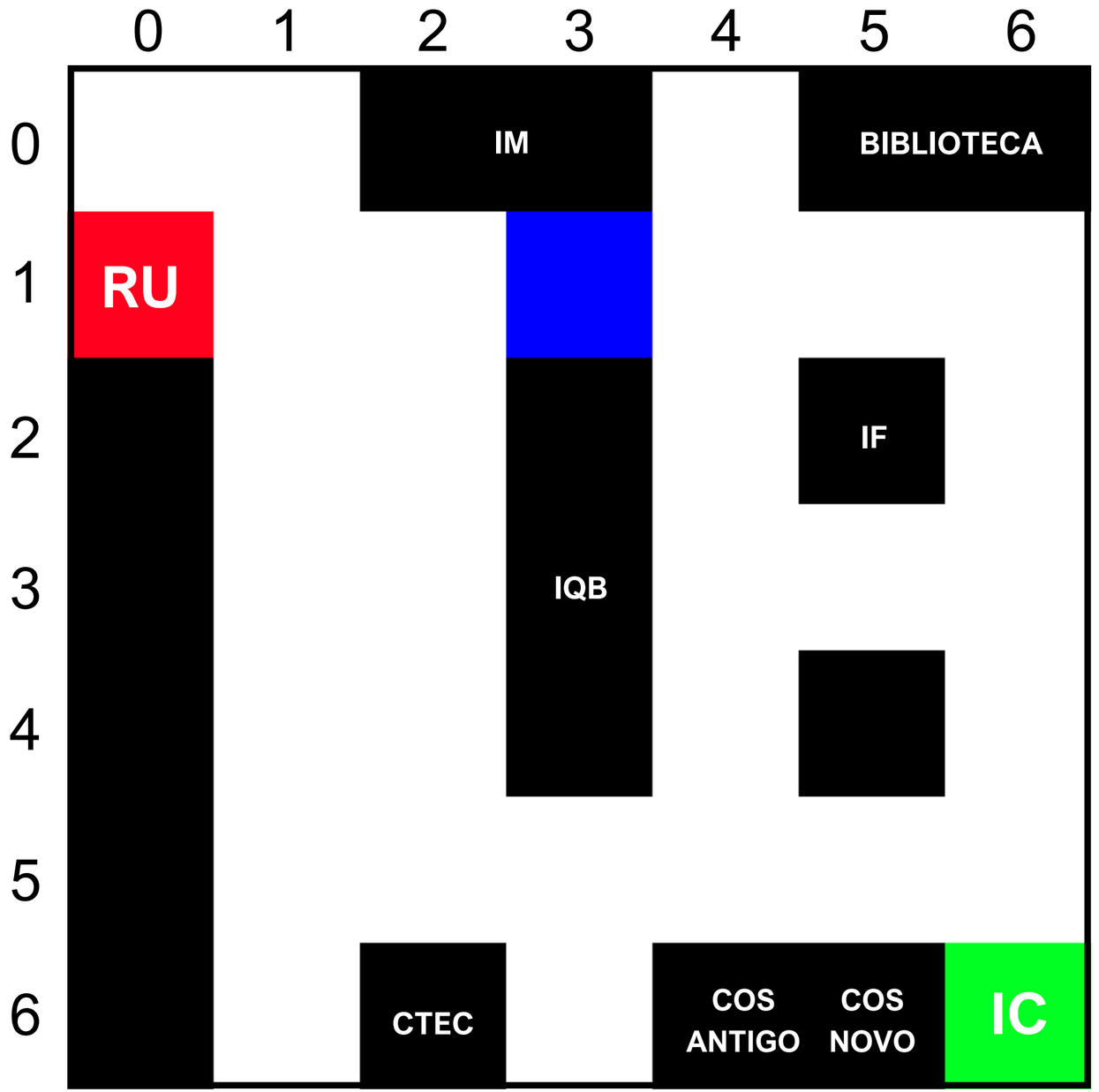


Definições

- **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.

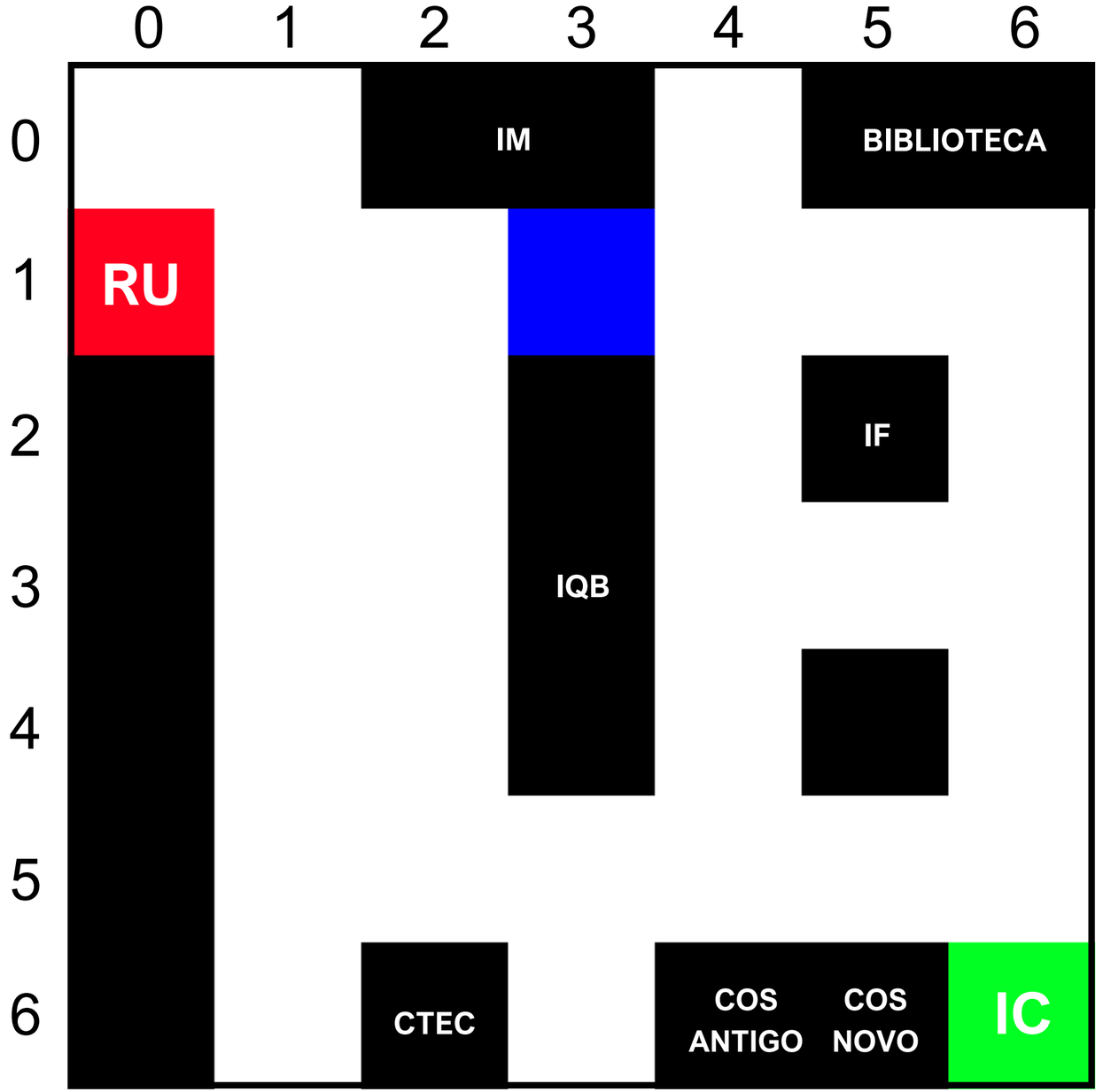
Definições

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



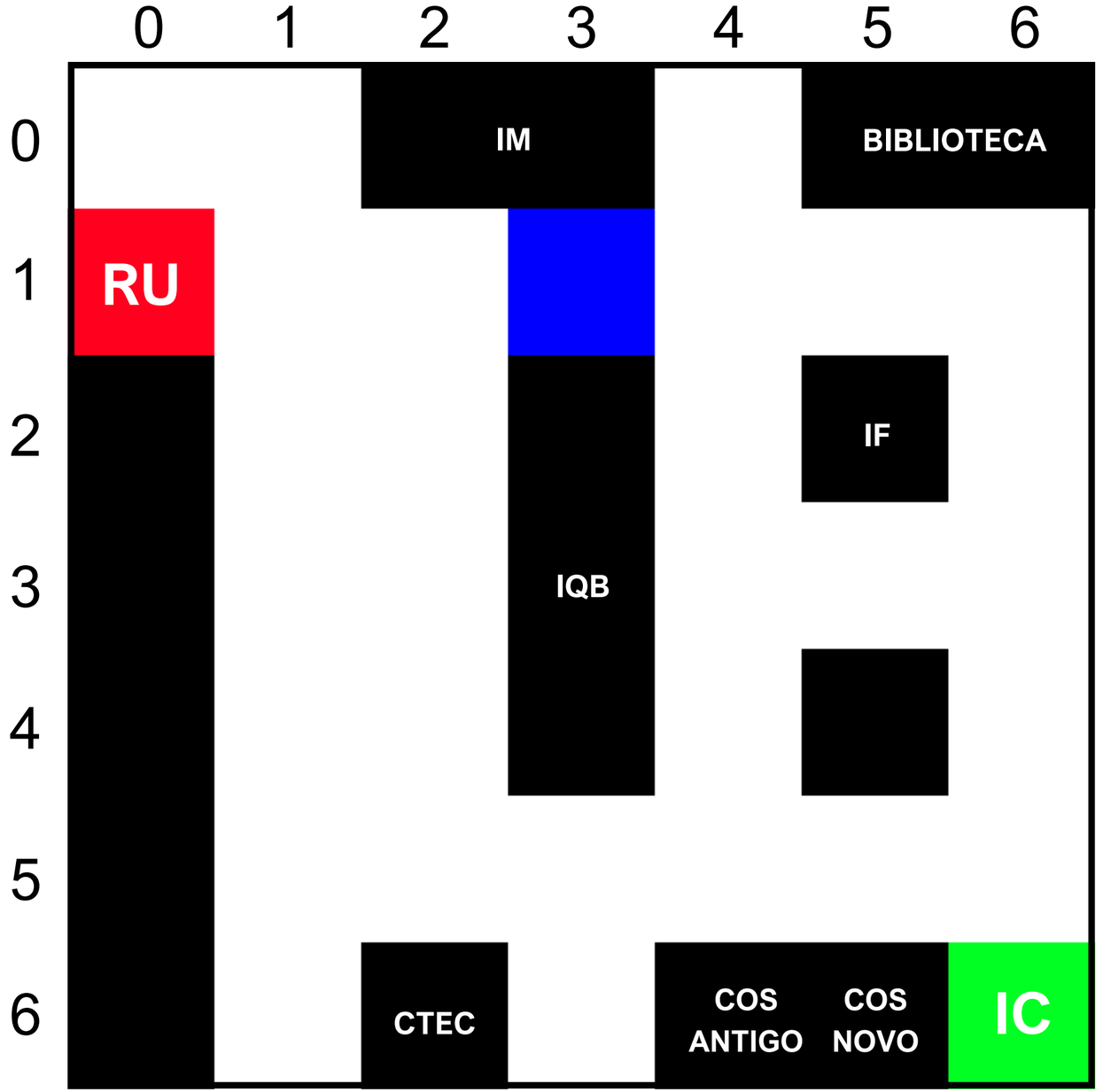
Definições

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



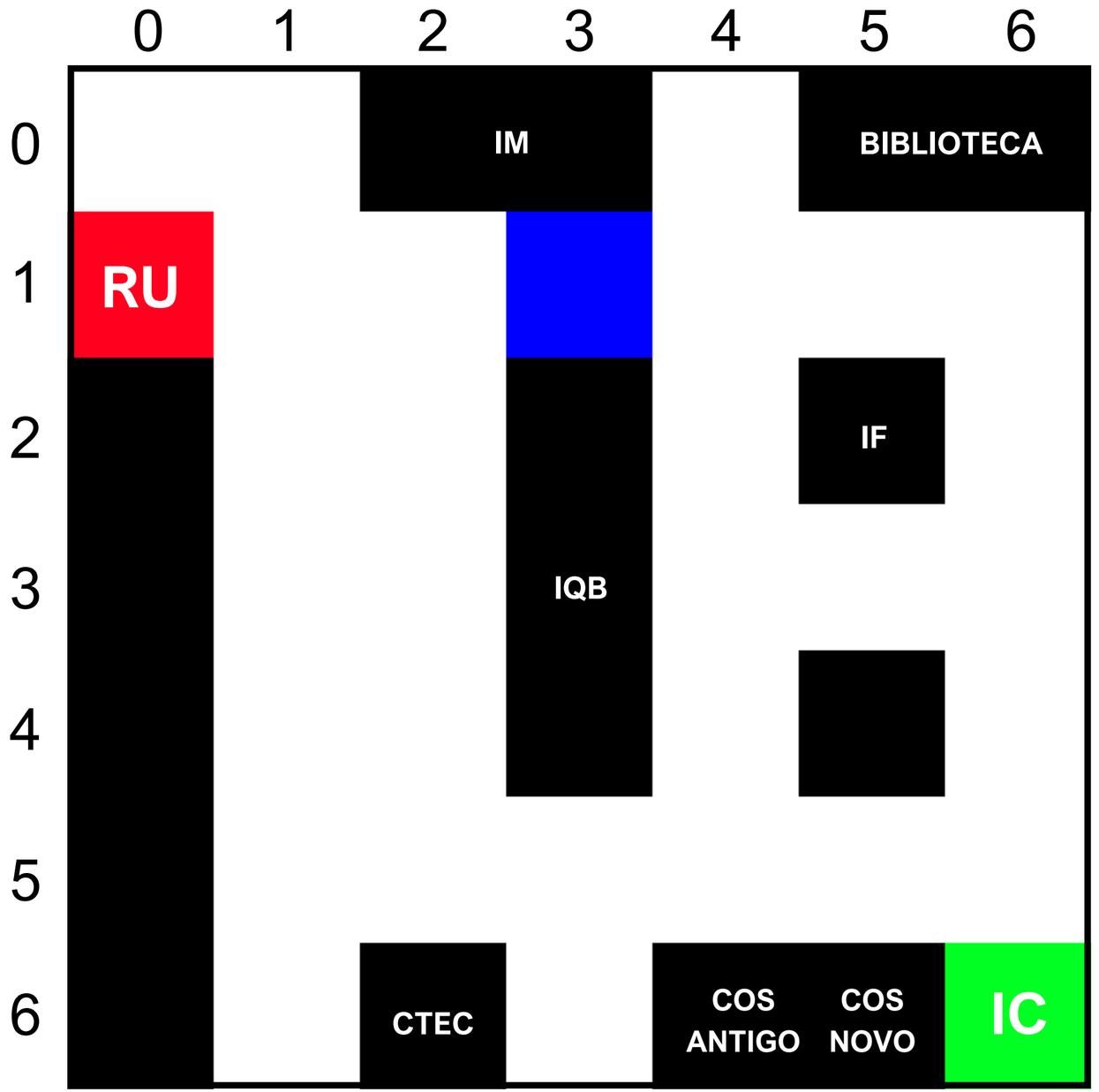
Definições

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



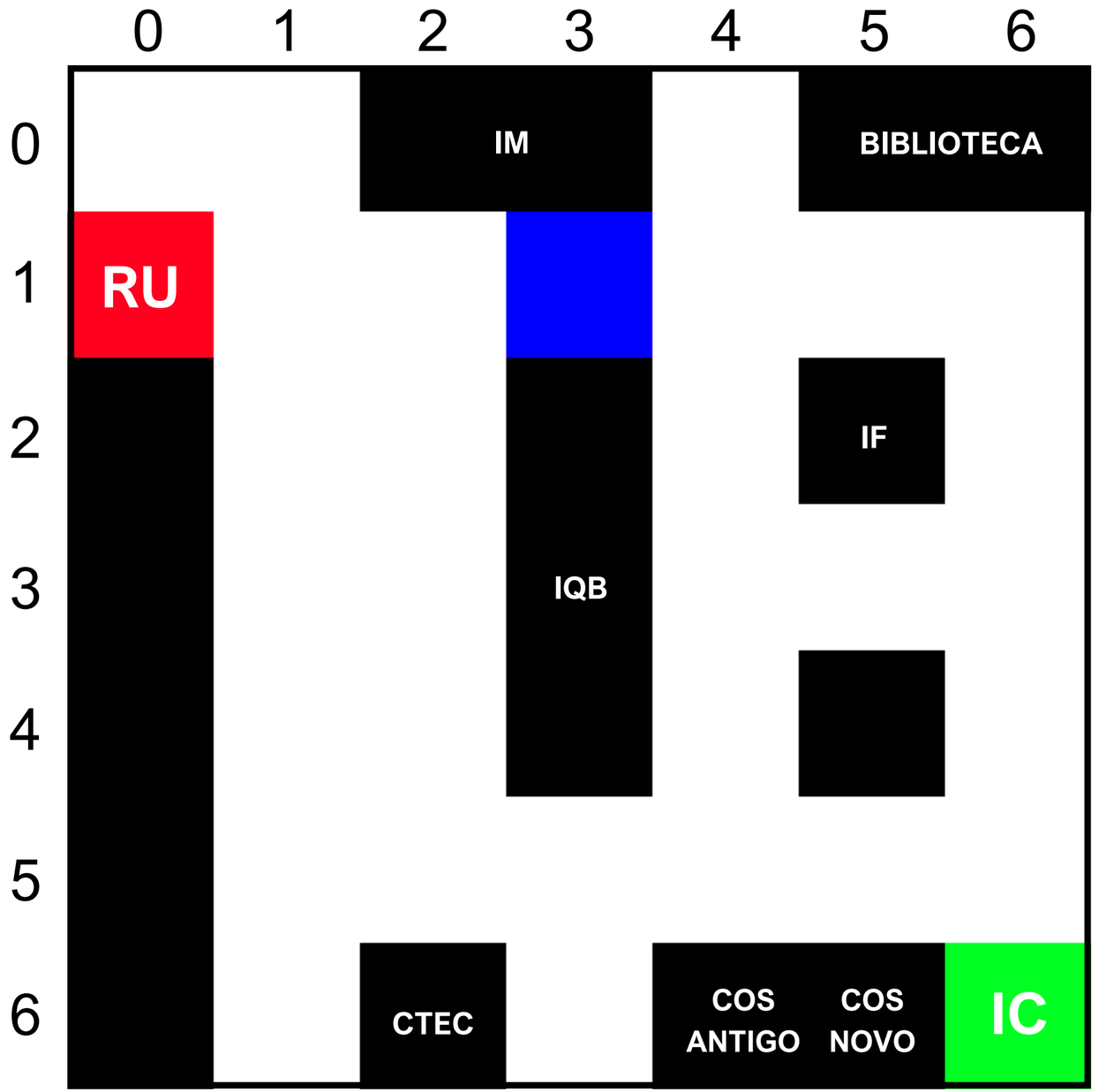
Definições

- 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



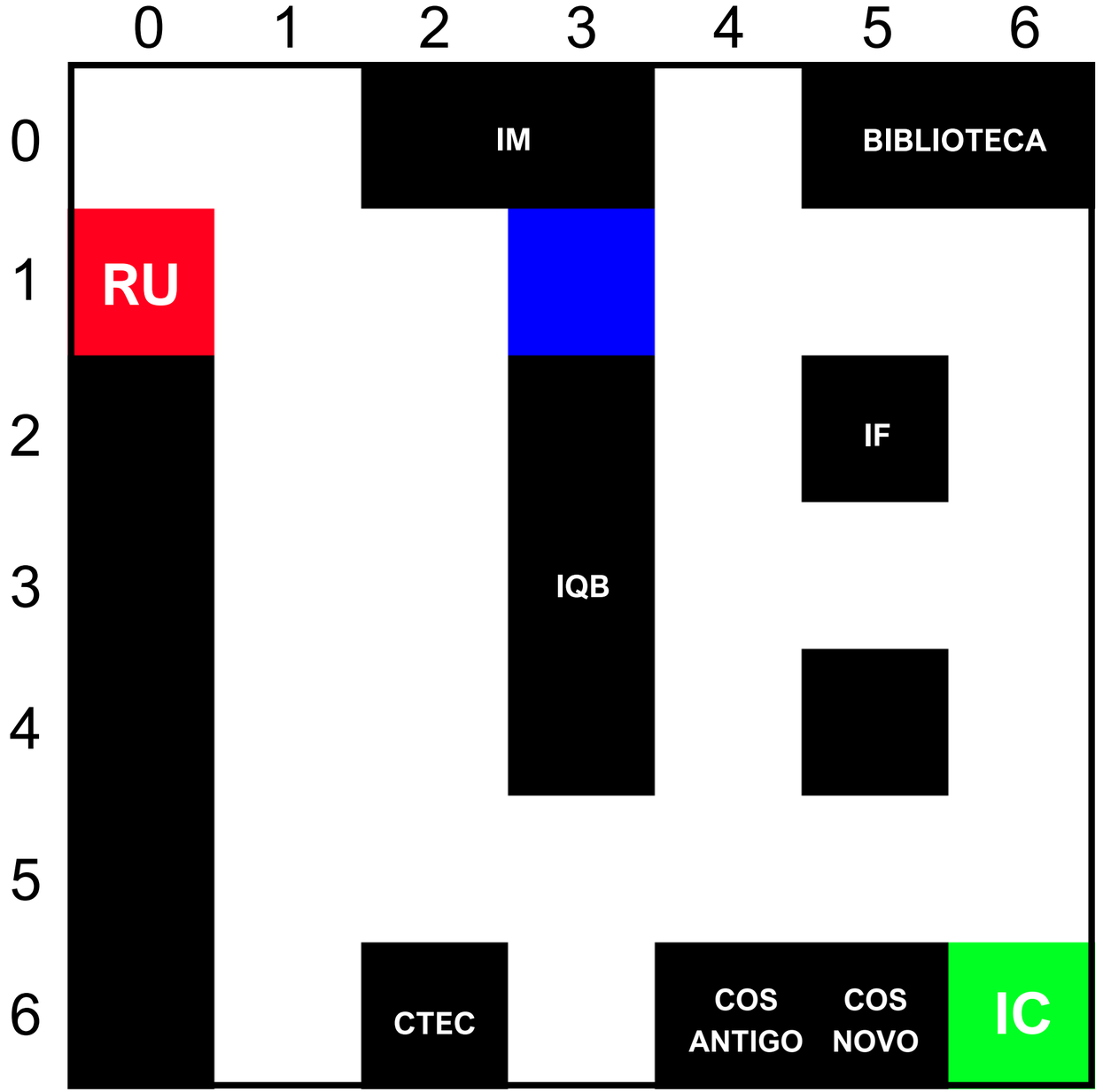
Definições

- 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

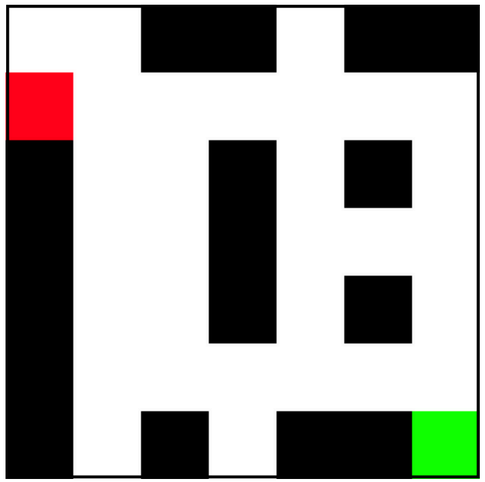


Código

```
#define MAX_CELULAS 100
#define LINHAS 7
#define COLUNAS 7
```

```
typedef struct {
  int linha;
  int coluna;
  int caminhavel;
  int f;
  int g;
  int h;
} No;

typedef struct {
  No* nos[MAX_CELULAS];
  int tamanho;
} FilaPrioridade;
```

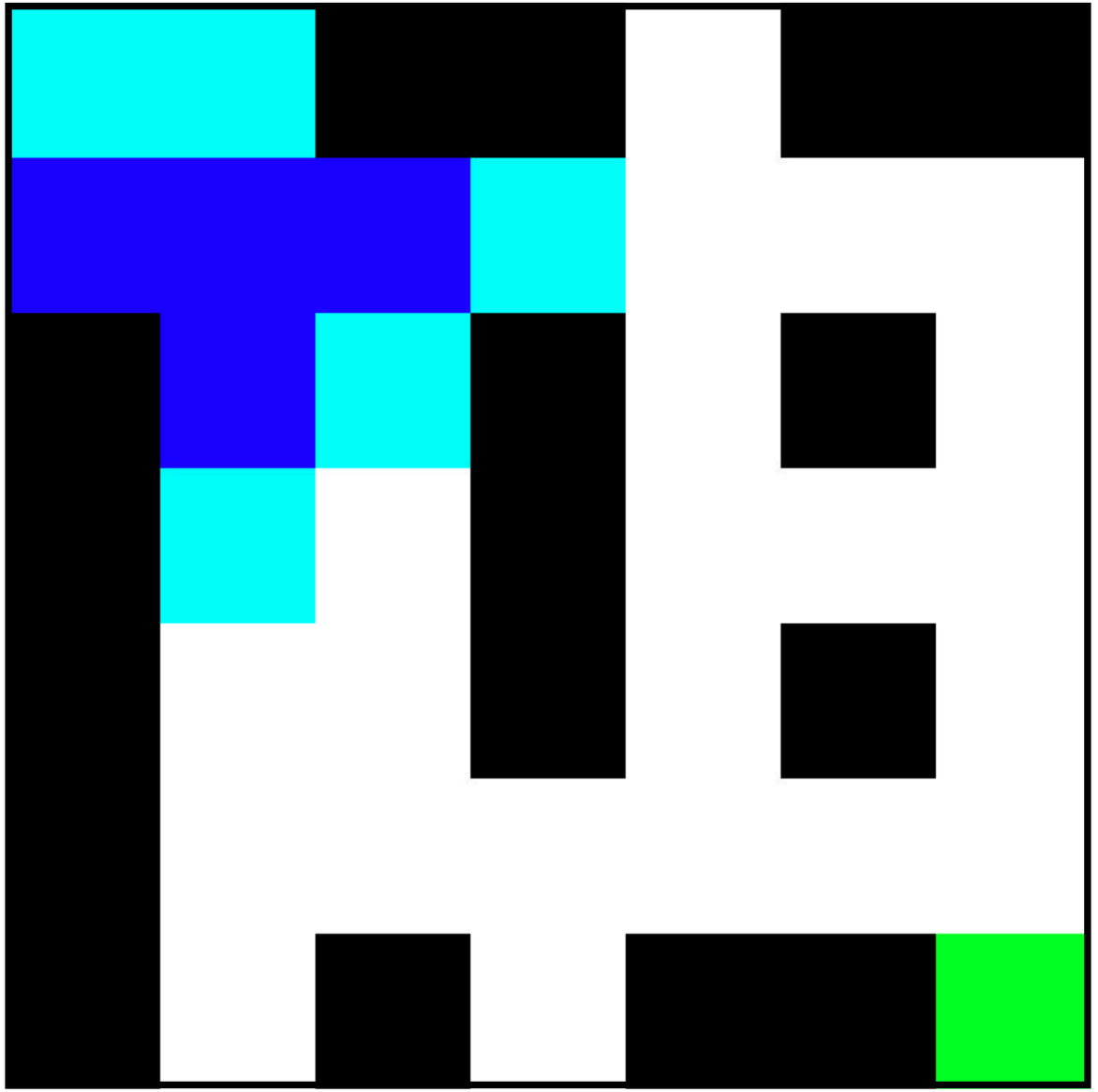
```
No labirinto[LINHAS][COLUNAS] = {
    {{0, 0, 1}, {0, 1, 1}, {0, 2, 0}, {0, 3, 0}, {0, 4, 1}, {0, 5, 0}, {0, 6, 0}},
    {{1, 0, 1}, {1, 1, 1}, {1, 2, 1}, {1, 3, 1}, {1, 4, 1}, {1, 5, 1}, {1, 6, 1}},
    {{2, 0, 0}, {2, 1, 1}, {2, 2, 1}, {2, 3, 0}, {2, 4, 1}, {2, 5, 0}, {2, 6, 1}},
    {{3, 0, 0}, {3, 1, 1}, {3, 2, 1}, {3, 3, 0}, {3, 4, 1}, {3, 5, 1}, {3, 6, 1}},
    {{4, 0, 0}, {4, 1, 1}, {4, 2, 1}, {4, 3, 0}, {4, 4, 1}, {4, 5, 0}, {4, 6, 1}},
    {{5, 0, 0}, {5, 1, 1}, {5, 2, 1}, {5, 3, 1}, {5, 4, 1}, {5, 5, 1}, {5, 6, 1}},
    {{6, 0, 0}, {6, 1, 1}, {6, 2, 0}, {6, 3, 1}, {6, 4, 0}, {6, 5, 0}, {6, 6, 1}}
  };
```



```
void aEstrela (No labirinto[LINHAS][COLUNAS], No* inicio, No* destino, No*
caminho[MAX_CELULAS], int* caminho_comprimento) {

  FilaPrioridade filaAberta;
  inicializarFila(&filaAberta);

  int listaFechada[LINHAS][COLUNAS] = {0};
  No* mapaCaminho[LINHAS][COLUNAS];

  for (int i = 0; i < LINHAS; i++) {
    for (int j = 0; j < COLUNAS; j++) {
      labirinto[i][j].g = INT_MAX;
      labirinto[i][j].f = INT_MAX;
      mapaCaminho[i][j] = NULL;
    }
  }

  inicio->g = 0;
  inicio->h = calcularH(inicio, destino);
  inicio->f = inicio->g + inicio->h;

  enfileirar(&filaAberta, inicio);
```

```
while (!estaVazia(&filaAberta)) {
    No* atual = desenfileirar(&filaAberta);
    listaFechada[atual->linha][atual->coluna] = 1;

    if (atual->linha == destino->linha && atual->coluna == destino->coluna) {
      No* c = destino;
      *caminho_comprimento = 0;
      while (c != NULL) {
        caminho[(*caminho_comprimento)++] = c;
        c = mapaCaminho[c->linha][c->coluna];
      }
      return;
    }

    int direcoes[4][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};

    for (int i = 0; i < 4; i++) {
      int linha_vizinha = atual->linha + direcoes[i][0];
      int coluna_vizinha = atual->coluna + direcoes[i][1];

      if (linha_vizinha >= 0 && linha_vizinha < LINHAS && coluna_vizinha >= 0 && coluna_vizinha < COLUNAS && labirinto[linha_vizinha]
[coluna_vizinha].caminhavel && !listaFechada[linha_vizinha][coluna_vizinha]) {

        No* vizinho = &labirinto[linha_vizinha][coluna_vizinha];
        int novo_g = atual->g + 1;

        if (novo_g < vizinho->g) {
          vizinho->g = novo_g;
          vizinho->h = calcularH(vizinho, destino);
          vizinho->f = vizinho->g + vizinho->h;
          mapaCaminho[linha_vizinha][coluna_vizinha] = atual;

          enfileirar(&filaAberta, vizinho);
        }}}
   *caminho_comprimento = 0;
  }
```

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**.