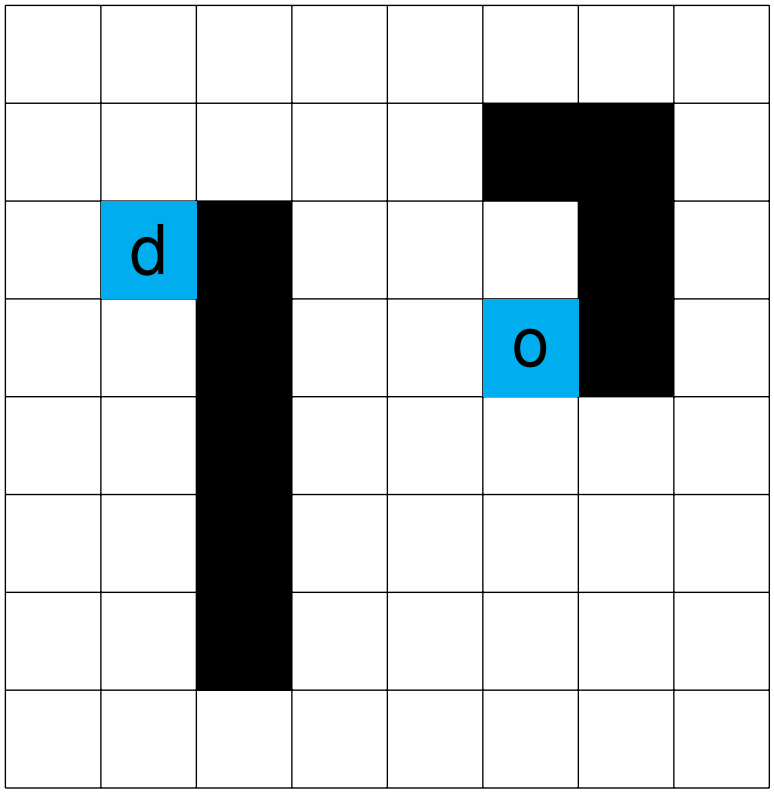
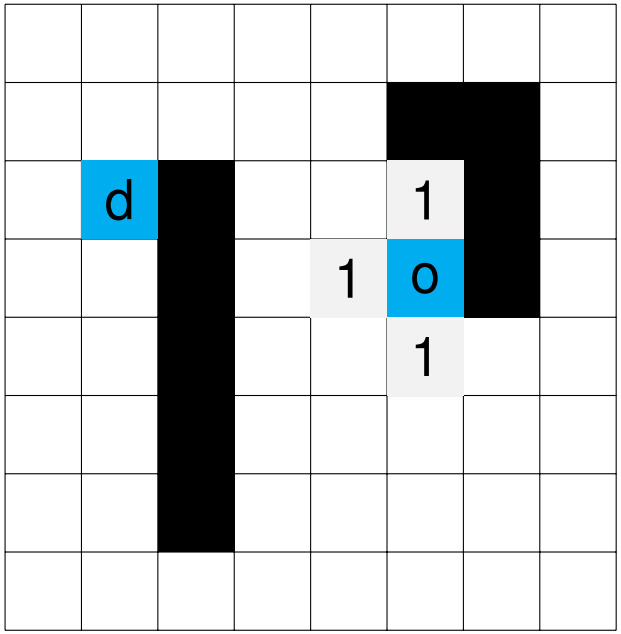
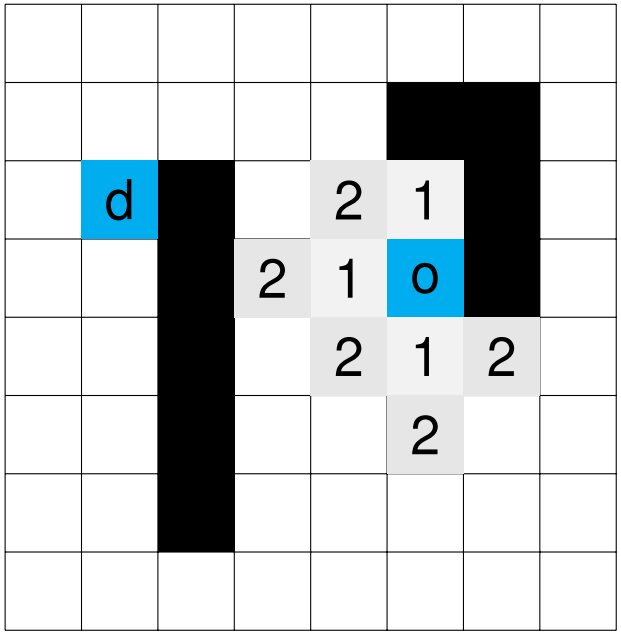
Trabalho 1: Programação Paralela para Processador Multicore com Memória Compartilhada Usando OpenMP

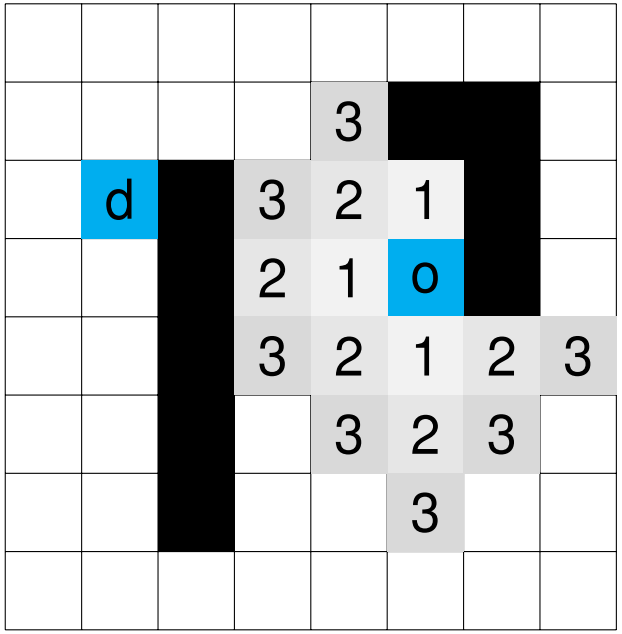
- **Problema de roteamento usando algoritmo de Lee**
  - Grid de $n \times m$ células
  - Célula origem e célula destino
  - Obstáculos (ocupam células que não podem ser usadas no roteamento)
  - Encontrar menor caminho entre células origem e destino



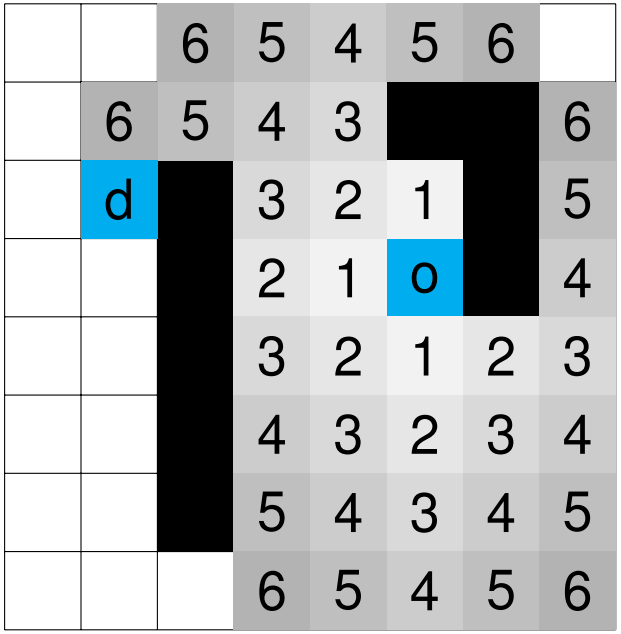
Algoritmo Sequencial: Fase de Expansão



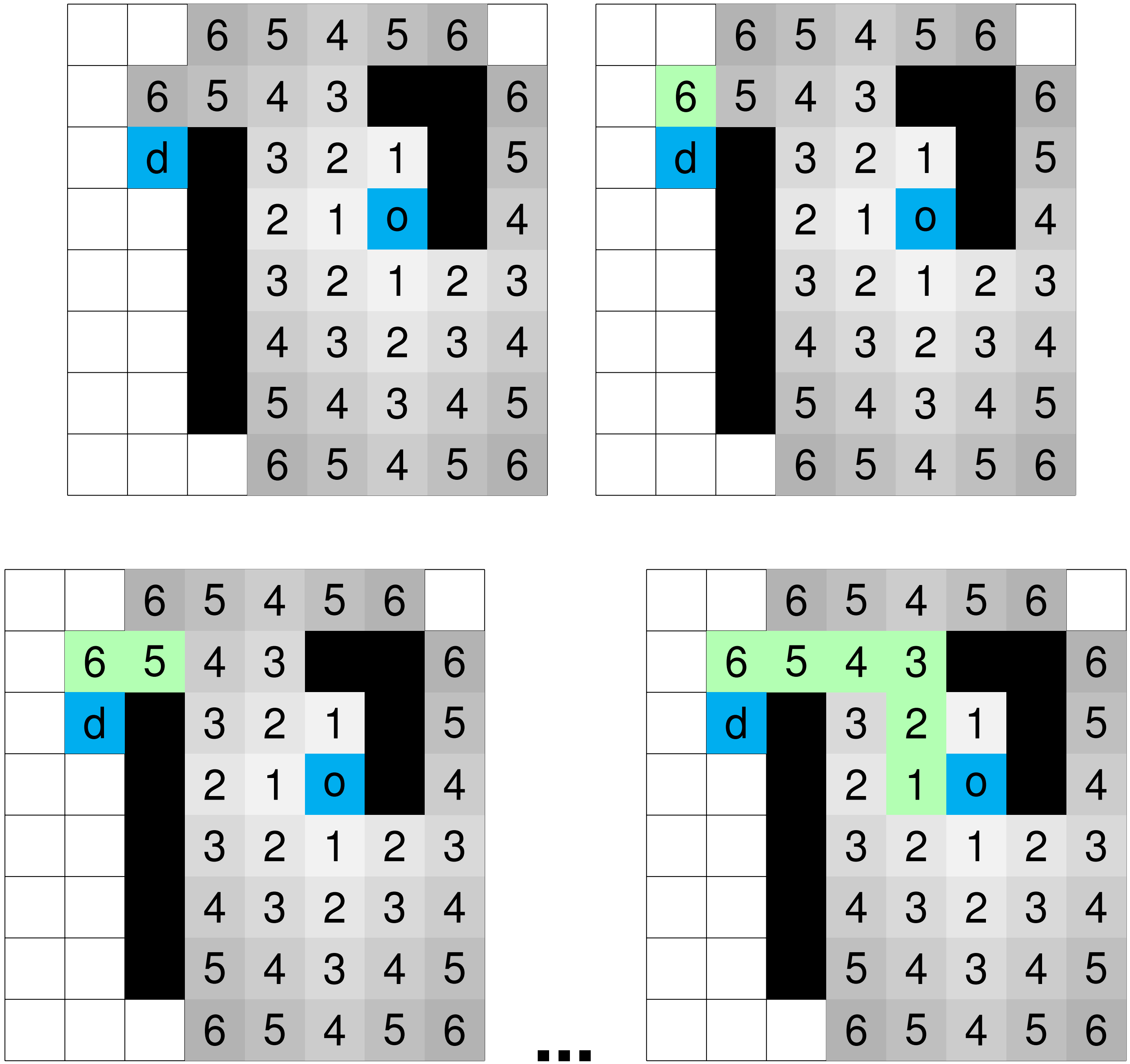




...



Algoritmo Sequencial: Fase de Backtracking



Algoritmo Sequencial: Fase de Expansão

```
// Inicialização: Células obstáculos do grid: inicializadas com -1
// Demais células do grid: inicializadas com infinito
Fila = vazia // Fila FIFO de células a serem expandidas
Grid[origem.i][origem.j] = 0 // Distância da origem a ela mesma é 0
achou = false // Destino foi encontrado?

insere(Fila, origem) // Insere origem no fim da fila
while (Fila != vazia AND NOT achou)
{
   cel = remove(Fila) // Remove célula do início da fila
   if (cel.i == destino.i AND cel.j == destino.j) // cel é o destino
      achou = true
   else
   {
      for (cada célula viz vizinha de cel) // No máximo 4 vizinhos
         if (Grid[viz.i][viz.j] == infinito)
         {
            Grid[viz.i][viz.j] = Grid[cel.i][cel.j] + 1
            insere(Fila, viz) // Insere célula viz no fim da fila
         }
   }
}
```

Fila FIFO garante que expansão ocorre por níveis: primeiro encontra células com distância 1 da origem, depois células com distância 2, ...

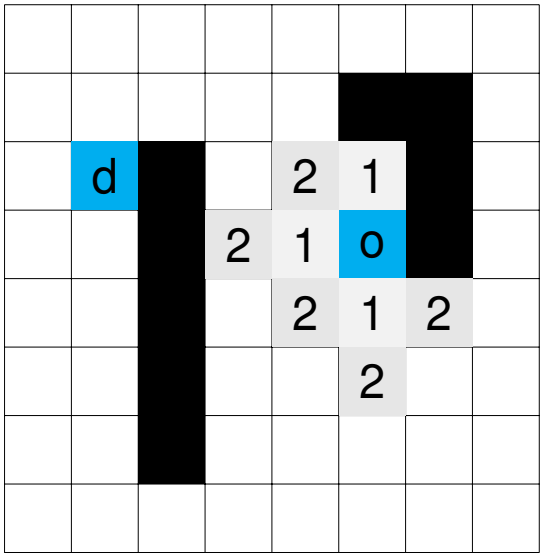
Algoritmo Sequencial: Fase de Backtracking

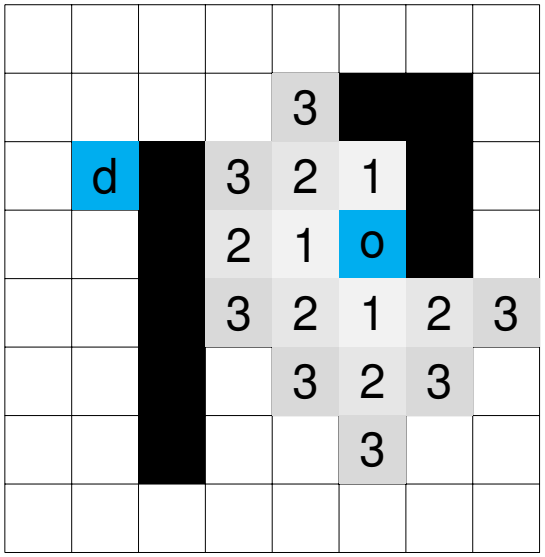
```
Caminho = vazio // Caminho da origem até o destino
if (achou)
{
   cel = destino
   insere(caminho, destino) // Insere destino no início do caminho

   // Enquanto não chegar na origem
   while (cel.i != origem.i OR cel.j != origem.j)
   {
      Investiga 4 células vizinhas de cel e seleciona viz,
         tal que Grid[viz.i][viz.j] == Grid[cel.i][cel.j] - 1
      cel = viz
      insere(Caminho, viz) // Insere célula viz no início do caminho
   }
}
```

Ideias sobre o Algoritmo e a sua Paralelização

- **Na fase de expansão**:
  - Fila FIFO impõe ordem sequencial de tratamento das células
  - Essa restrição é mais “forte” do que o necessário
    para o correto funcionamento do algoritmo
  - Células em um mesmo nível de expansão podem ser tratadas em qualquer ordem
  - Se tratar células em diferentes níveis de expansão em ordem ”errada”,
    o que acontece?
    Algoritmo pode ser modificado para funcionar? Vale a pena?





Programas a serem Desenvolvido

- **Programas devem ser em C ou C++**
- **Desenvolver programa sequencial**:
  - Baseado no algoritmo apresentado
- **Desenvolver programa paralelo, usando OpenMP**:
  - Explorar paralelismo na fase de expansão
  - Se possível, explorar paralelismo na fase de backtracking
- **Interface de execução dos programas**:
  - Por linha de comando com argumentos:

```
rotseq entrada.txt saida.txt
rotpar entrada.txt saida.txt
```

- **Submissão**: um único arquivo .zip com programas fonte sequencial e paralelo
  - **Programas devem ter no cabeçalho**:
    - Nome dos alunos do grupo (máximo 2 alunos)
    - Comando de compilação por linha de comando

Entradas e Saídas do Programa

- **Entradas**: Em um único arquivo texto
  - Nº de linhas do grid
  - Nº de colunas do grid
  - Índices i e j da célula origem
  - Índices i e j da célula destino
  - Nº de obstáculos
  - Para cada obstáculo:
    - Índices i e j da célula inicial do obstáculo (superior esquerda)
    - Nº de linhas e colunas do obstáculo

      (Obs.: Obstáculos são sempre retangulares)
- **Saídas**: Em um único arquivo texto
  - Comprimento do maior caminho da origem até o destino

    (valor da célula do destino ao final da expansão)
  - Menor caminho: sequência de células (índices i e j) da origem até o destino
- **Arquivos com exemplos de entrada e saída fornecidos**