

Conceção e Análise de Algoritmos  
*Sistema de Evacuação*

*2MIEICo2 – Grupo 3 – Tema 3*

Luís Borges **up201605859@fe.up.pt**

Patrícia Janeiro **up201605946@fe.up.pt**

Sandro Campos **up201605947@fe.up.pt**

# *Índice*

1. Introdução .....	3
2. Descrição do Problema .....	4
2.1. Input .....	4
2.2. Introdução de dados .....	4
2.3. Output .....	4
2.4. Objetivo .....	4
2.5. Restrições .....	4
3. Algoritmos implementados .....	5
4. Casos de Utilização .....	7
5. Diagrama de Classes .....	11
6. Principais Dificuldades .....	12
7. Contribuição no Projeto .....	12
8. Conclusão .....	13

## *1. Introdução*

Na segunda parte do trabalho prático da unidade curricular de Conceção e Análise de Algoritmos foi-nos proposta a implementação, na primeira parte do trabalho prático de um Sistema de linha de emergência, na qual se insere o nome de uma rua e o Sistema retorna uma rota de evacuação a partir desta.

Procedeu-se assim à implementação de algoritmos de pesquisa exata e aproximada, de forma a procurar a localização do utilizador no grafo.

## *2. Descrição do Problema*

### *2.1 Input*

Construção de um grafo  $G = (V, E)$ , em que:

- Vértices ( $V$ ) – representam os vários pontos num mapa de estradas.
- Arestas ( $E$ ) – representam as ligações entre os vários pontos do mapa, o nome da rua, a capacidade máxima e o tráfego atual da via.
- Nós de início e de destino.

### *2.2 Dados de entrada*

Ficheiros com o tráfego a percorrer as estradas.

### *2.3 Output*

O percurso que deve ser realizado de um ponto inicial para um ponto final, ou todos os percursos possíveis a partir de um ponto tendo em conta a existência de algum troço fechado, apresentado o percurso realizado no menor tempo possível.

### *2.4 Objetivo*

Apresentar o percurso que deve ser realizado para evitar um troço fechado e chegar ao destino em segurança, no menor tempo possível, processando todo o tráfego. Neste caso, na inexistência de informação relativa à velocidade média de cada troço da autoestrada, assume-se que o caminho mais rápido a ser percorrido é aquele que corresponde ao caminho de menor distância. Apresentar todos os percursos disponíveis a partir de uma rua cujo nome é introduzido.

### *2.5 Restrições*

Evitar um troço fechado e impedir que os veículos não se dirijam para troços lotados das autoestradas, uma vez que cada um destes possui uma capacidade limitada, isto é, um número máximo de automóveis que suporta.

### 3. Algoritmos e estruturas de dados

Com o objetivo de pesquisar o nome exato da rua, optou-se pela implementação do **algoritmo de Knuth-Morris-Pratt**. Este algoritmo efetua um pré-processamento do padrão em tempo  $O(|P|)$  e, ao contrário do **algoritmo baseado em autômato finito**, não chega a gerar explicitamente um autômato. Depois do pré-processamento, segue-se o processamento do texto em  $O(|T|)$ . Somando-se o tempo das duas etapas, totaliza-se  $O(|T|+|P|)$ , muito inferior ao tempo do **algoritmo naïve**,  $O(|P|*|T|)$  e do **algoritmo baseado em autômato finito** que depende do tamanho do alfabeto( $\Sigma$ ),  $O(|P|*|\Sigma|)$ . Por esta razão considerou-se tanto o **algoritmo naïve** como o **baseado em autômato finito** ineficiente. No **algoritmo de Knuth-Morris-Pratt** desloca-se o padrão para a direita de uma forma que permite continuar a comparação na mesma posição de texto, evitando comparações inúteis, sendo este deslocamento determinado pela função prefixo ( $\pi[q]$ ).

$$\pi[q] = \max \{k: 0 \leq k < q \text{ e } P[1..k] = P[(q-k+1)..q] \}$$

- $q = 1, \dots, |P|$
- $P[i..j]$  - substring entre índices  $i$  e  $j$
- Índices a começar em 1
- $\pi[q]$  é o comprimento do maior prefixo de  $P$  que é um sufixo próprio do prefixo de  $P$  de comprimento  $q$

Figura 1 Definição da função prefixo

```

KMP-MATCHER( $T, P$ )
1   $n \leftarrow \text{length}[T]$ 
2   $m \leftarrow \text{length}[P]$ 
3   $\pi \leftarrow \text{COMPUTE-PREFIX-FUNCTION}(P)$ 
4   $q \leftarrow 0$  ▷ Number of characters matched.
5  for  $i \leftarrow 1$  to  $n$  ▷ Scan the text from left to right.
6      do while  $q > 0$  and  $P[q + 1] \neq T[i]$ 
7          do  $q \leftarrow \pi[q]$  ▷ Next character does not match.
8          if  $P[q + 1] = T[i]$ 
9              then  $q \leftarrow q + 1$  ▷ Next character matches.
10         if  $q = m$  ▷ Is all of  $P$  matched?
11             then print "Pattern occurs with shift"  $i - m$ 
12          $q \leftarrow \pi[q]$  ▷ Look for the next match.

```

Figura 2 Pseudo-código do algoritmo de Knuth-Morris-Pratt

```

COMPUTE-PREFIX-FUNCTION( $P$ )
1   $m \leftarrow \text{length}[P]$ 
2   $\pi[1] \leftarrow 0$ 
3   $k \leftarrow 0$ 
4  for  $q \leftarrow 2$  to  $m$ 
5      do while  $k > 0$  and  $P[k + 1] \neq P[q]$ 
6          do  $k \leftarrow \pi[k]$ 
7          if  $P[k + 1] = P[q]$ 
8              then  $k \leftarrow k + 1$ 
9           $\pi[q] \leftarrow k$ 
10 return  $\pi$ 

```

Figura 3 Pseudo-código do algoritmo da função prefixo

Para a pesquisa aproximada do nome da rua, utilizou-se o algoritmo dado nas aulas da unidade curricular. Neste algoritmo a **distância de edição** entre  $P$  (string do padrão a procurar) e  $T$  (string das ruas) é o menor número de alterações necessárias para transformar  $T$  em  $P$ , recorrendo à substituição de um carácter por outro e à inserção ou remoção de um outro. Utilizou-se o algoritmo com otimização de espaço, ocupando  $O(|T|)$ .

- $D[i,j] = \text{EditDistance}(P[1..i], T[1..j]), 0 \leq i \leq |P|, 0 \leq j \leq |T|$
- Condições fronteira:
  - $D[0, j] = j, D[i, 0] = i$
- Caso recursivo ( $i > 0$  e  $j > 0$ ):
  - Se  $P[i] = T[j]$ , então  $D[i, j] = D[i-1, j-1]$
  - Senão, escolhe-se a operação de edição que sai mais barata; isto é,  $D[i,j]$  é o mínimo de:
    - $1 + D[i-1, j-1]$  (substituição de  $T[j]$  por  $P[i]$ )
    - $1 + D[i-1, j]$  (inserção de  $P[i]$  a seguir a  $T[j]$ )
    - $1 + D[i, j-1]$  (eliminação de  $T[j]$ )

Figura 4 Distância de edição entre duas strings

```

EditDistance(P,T) {
  // inicialização
  for j = 0 to |T| do D[j] = j  // D[0,j]
  // recorrência
  for i = 1 to |P| do
    old = D[0] // guarda D[i-1,0]
    D[0] = i   // inicializa D[i, 0]
    for j = 1 to |T| do
      if P[i] == T[j] then new = old
      else new = 1 + min(old,
                        D[j],
                        D[j-1])

      old = D[j]
      D[j] = new
  // finalização
  return D[|T|]
}

```

*Ainda tem valor anterior  $D[i-1,j]$*  →  $D[j]$

*Tem  $D[i-1,j-1]$*  →  $old$

*Já tem valor da iteração corrente, i.e.,  $D[i, j-1]$*  →  $D[j-1]$

Figura 5 Pseudo-código da função de distância de edição

## 4. Casos de Utilização

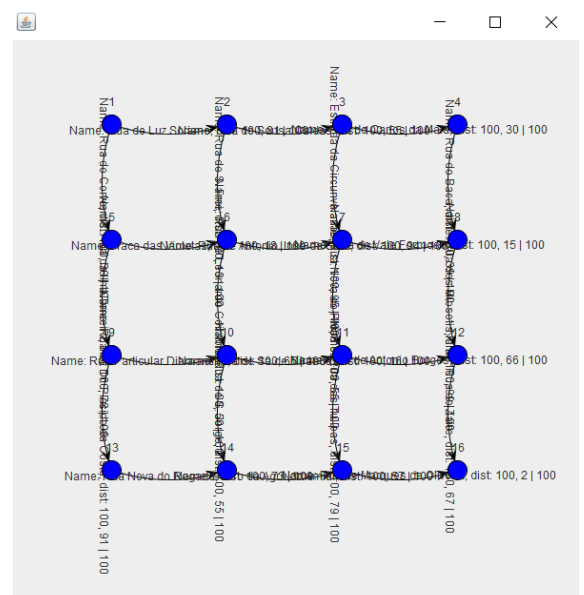
A interface do programa foi desenvolvida para a linha de comandos, sendo que todas as instruções são inseridas através desta. Todos os menus, incluindo inputs e outputs estão definidos na classe “Utils.cpp”.

Inicialmente, tal como na primeira parte do trabalho, opta-se por carregar um grafo já existente ou gerar um aleatoriamente.

```
C:\WINDOWS\system32\cmd.exe
EvacuationSystem - CAL 1718
-> 1. Load graph from files
-> 2. Generate random graph
-> 0. Leave
. Option:
```

De seguida é possível seleccionar uma das seguintes opções: calcular o caminho mais curto, divergir e observar o tráfego atual e reportar um novo acidente. É também apresentado o esquema do grafo, com o nome de cada rua, assim como a distância entre nós, o número de veículos atuais num troço e a sua capacidade máxima. De referir que os nomes atribuídos às arestas são selecionados aleatoriamente de um ficheiro (roads.txt).

```
C:\WINDOWS\system32\cmd.exe
EvacuationSystem - main menu:
-> 1. Calculate shortest routes
-> 2. Divert current traffic
-> 3. Watch current traffic
-> 4. Report accidents
-> 0. Leave
. Option:
```





Todo os menus seguintes e as respectivas funcionalidades são semelhantes à primeira parte do trabalho prático. Por esta razão apresentam-se apenas as funcionalidades implementadas nesta parte do trabalho. Desta forma, é possível agora seleccionar o nome da rua inicial e da rua de destino, ou os seus IDs respetivos. O percurso apresentado inicia-se no final da rua inicial (vértice que é ligado) até ao final da rua de destino. É possível pesquisar pelo nome exato da rua, ou pelo nome aproximado da mesma. Neste último caso é apresentada uma mensagem ao utilizador com todos os nomes de ruas similares.

```
C:\WINDOWS\system32\cmd.exe
. Option: 1

EvacuationSystem - path searching:
-> 1. Dijkstra algorithm
-> 2. A* search algorithm
-> 0. Return
. Option: 1

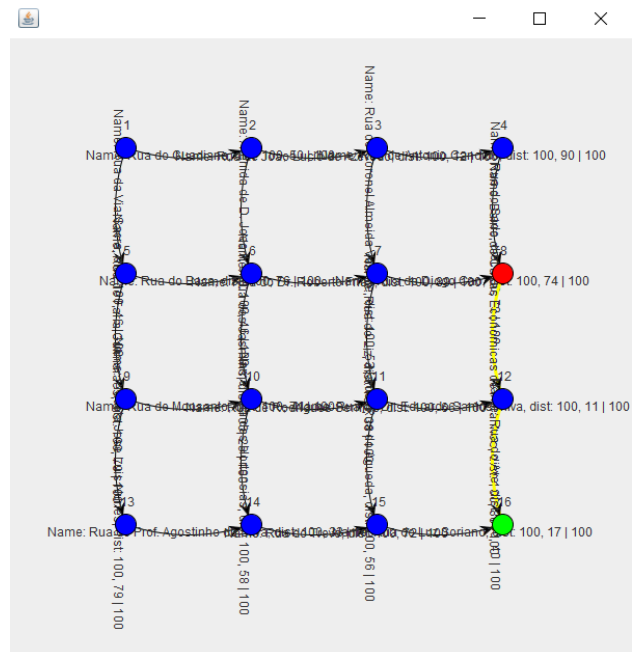
EvacuationSystem - type of search:
-> 1. Search by node id
-> 2. Search by road name
-> 0. Return
. Option: 2

-> Insert start road name: Rua de Diogo
. Did you mean "Rua de Diogo Cao" (Y/N)? y

-> Insert destiny road name: Rua de Luz Soriano

. Path from road Rua de Diogo Cao to road Rua de Luz Soriano:
Rua de Diogo Cao->Bairro de Casas Economicas de Paranhos->Rua de Luz Soriano

EvacuationSystem - type of search:
-> 1. Search by node id
-> 2. Search by road name
-> 0. Return
. Option:
```



Caso o nome da rua não seja encontrado, é apresentada uma lista ao utilizador com o todos os nomes de ruas similares.

```
C:\WINDOWS\system32\cmd.exe
EvacuationSystem - main menu:
-> 1. Calculate shortest routes
-> 2. Divert current traffic
-> 3. Watch current traffic
-> 4. Report accidents
-> 0. Leave
. Option: 1

EvacuationSystem - path searching:
-> 1. Dijkstra algorithm
-> 2. A* search algortihm
-> 0. Return
. Option: 1

EvacuationSystem - type of search:
-> 1. Search by node id
-> 2. Search by road name
-> 0. Return
. Option: 2

-> Insert start road name: Rua Cuna

. Road not found. Similar roads:
[1]: Rua do Cunha
[2]: Rua do Tamega
[3]: Rua da Arroteia
[0]: Try inserting another name
-> Option:
```

## 5. Diagrama de Classes



## *6. Principais Dificuldades*

Na realização da segunda parte do trabalho, e ao contrário da primeira parte, não encontramos grandes dificuldades. Os algoritmos contruídos eram, em geral, de fácil implementação.

## *7. Contribuição no Projeto*

Consideramos que a divisão de tarefas foi a seguinte: Luís Borges - 40%, Patrícia Janeiro – 20% e Sandro Campos - 40%.

## *8. Conclusão*

Com a realização deste projeto foi possível uma continuação no contacto com algoritmos de pesquisa do caminho mais curto em grafos e um contacto com novos algoritmos e conceitos, como no caso da pesquisa exata e aproximada de strings.