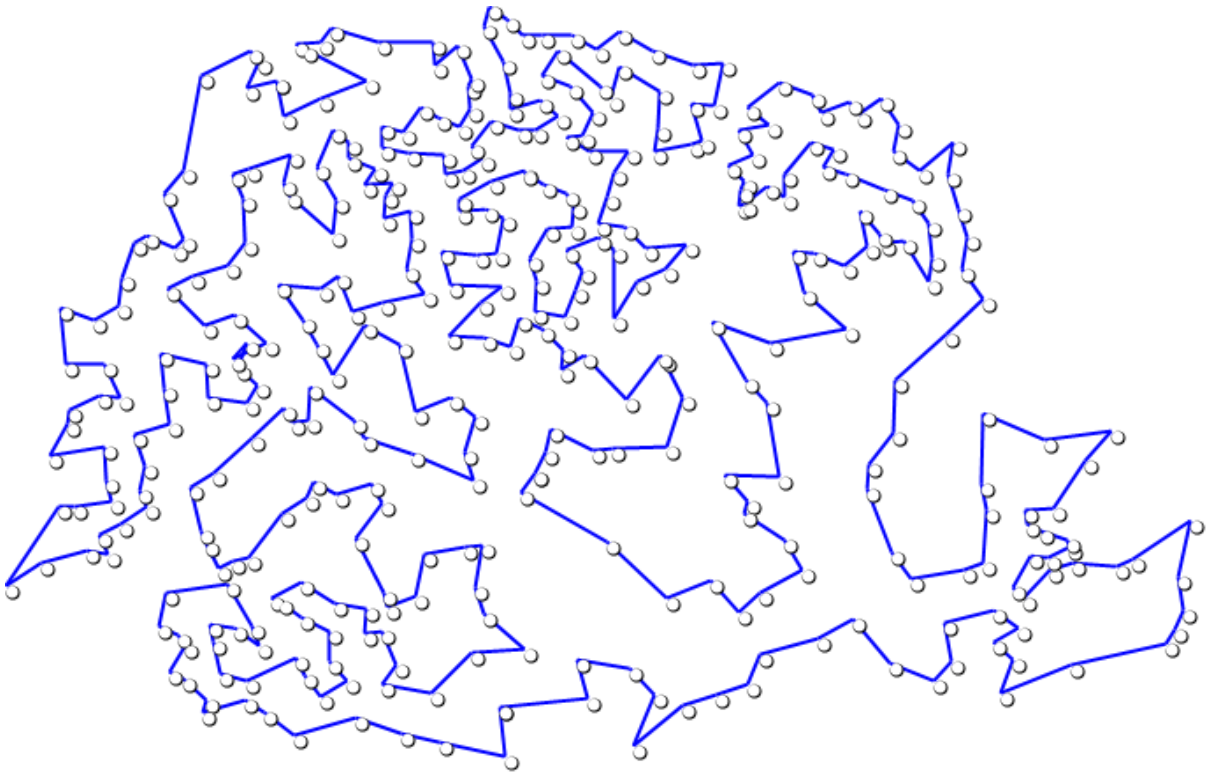
# Trabalho Prático 1



## Heurística do Problema do Caixeiro Viajante

Documentação

Nomes: Alec Mizote Garcia 2016026140
Jesimon Barreto Santos 2016070093

### Representação de dados:

Para construir um algoritmo com esse objetivo, é crucial a forma de representar os dados, nesse caso, a distância entre as cidades, pois, sempre é necessário procurar essa informação. De frente a esse problema, foi decidido optar por usar uma representação baseada em matriz de adjacência [1]. Porém, algumas alterações foram feitas, foram elas: na geração da matriz, os ‘1’s, que devem representar a aresta entre os vértices, foram substituídos pelas distâncias entre cada cidade, porque o problema pede que considere que todas as cidades são ligadas entre si. Além disso, a matriz é zerada da diagonal principal para baixo, pois dar-lhe valores seria duplicar a atribuição sem necessidade. Abaixo segue um exemplo de matriz gerada a partir de um arquivo hipotético.

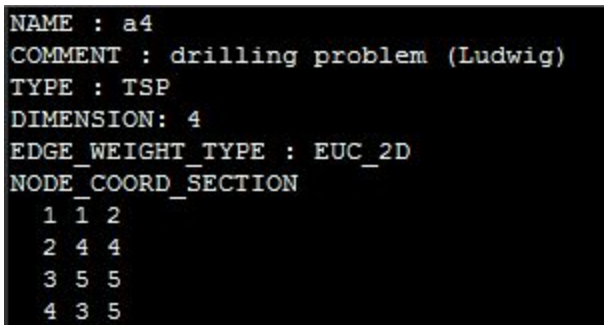



Figura 1. Imagem que demonstra um arquivo no formato padrão, com 4 cidades, gerado com base em um dos arquivos passados como amostra de teste.


Figura 2. Matriz gerada a partir das distâncias calculadas do arquivo da figura 1.

Executando o programa criado com o arquivo exibido na figura 1, é gerada uma matriz quadrada de tamanho 4x4. A matriz gerada é demonstrada na figura 2, com as características já citadas, e suas respectivas distâncias.

## Lógica do Algoritmo Heurístico implementado:

O algoritmo implementado foi baseado na ideia de grafos e matriz para organizar as distâncias entre os vértices (função ProcessingData), e da “Inserção mais barata” (IMB) como heurística do problema [2], o qual pode ser resumido por 4 passos:

1º - Adiciona a cidade 0 à rota (ou o vetor Path no algoritmo);
2º - Criar uma rota 0 -> X (onde x é a cidade mais próxima);
3º - Encontra e insere Y entre 0 e X, contanto que DistYe0 + DistYeX - Dist0eX seja mínimo;
4º - Repete o 3º passo até acabar os vértices e formar um ciclo.

Equação de complexidade de atribuições e comparações (ignorando alocações de memória):

Criação da matriz → $n+n^2$
1º e 2º passos → $n - 1$
3º e 4º passos → $3n^2/2 - 2n - 2$
addPath e verifyPath → $(3n^2 - 3n - 2)/2$

Equação Final → $4n^2 - 3n/2 - 4$

## Tabela de testes:

| Arquivo | Ótimo | Calculado | Dist. Perc. (%) | Tempo (seg) |
|---|---|---|---|---|
| a280 | 2579 | 3967 | 46.18 | 0.0170 |
| berlin52 | 7542 | 11749 | 44.22 | 0.0210 |
| bier127 | 118282 | 191825 | 37.82 | 0.0460 |
| brd14051 | 469385 | 721290 | 46.33 | 14.669 |
| ch130 | 6110 | 9685 | 41.49 | 0.1690 |
| ch150 | 6528 | 12760 | 4.53 | 0.0170 |
| d15112 | 1573084 | 2879142 | 16.97 | 15.988 |
| d18512 | 645238 | 988419 | 46.81 | 29.539 |
| eil8 | ~ | 164 | ~ | 0.0150 |
| eil51 | 426 | 614 | 55.87 | 0.0530 |
| eil76 | 538 | 778 | 55.39 | 0.0600 |
| eil101 | 629 | 807 | 71.70 | 0.0390 |
| fnl4461 | 182566 | 280409 | 46.41 | 0.8480 |
| gil262 | 2378 | 4289 | 19.64 | 0.0300 |
| kroA100 | 21282 | 40332 | 10.49 | 0.0430 |
| kroA150 | 26524 | 48792 | 16.04 | 0.0360 |
| kroA200 | 29368 | 53123 | 19.11 | 0.0350 |
| kroB100 | 22141 | 36498 | 35.16 | 0.0220 |
| kroB150 | 26130 | 45429 | 26.14 | 0.0410 |
| kroB200 | 29437 | 51618 | 24.65 | 0.0210 |
| kroC100 | 20749 | 36550 | 23.85 | 0.0270 |
| kroD100 | 21294 | 38447 | 19.45 | 0.0330 |
| kroE100 | 22068 | 38505 | 25.52 | 0.0650 |
| lin105 | 14379 | 24225 | 31.52 | 0.0570 |
| lin318 | 42029 | 70946 | 31.20 | 0.0400 |
| linhp318 | 41345 | 70946 | 28.40 | 0.0030 |
| nrw1379 | 56638 | 85992 | 48.17 | 0.0980 |
| pr76 | 108159 | 157588 | 54.30 | 0.0390 |
| pr107 | 44303 | 63233 | 57.27 | 0.0250 |

| | | | | |
|---|---|---|---|---|
| pr124 | 59030 | 102581 | 26.22 | 0.0380 |
| pr136 | 96772 | 150869 | 44.10 | 0.0460 |
| pr144 | 58537 | 111305 | 9.85 | 0.0330 |
| pr152 | 73682 | 121571 | 35.00 | 0.1010 |
| pr226 | 80369 | 125860 | 43.40 | 0.0700 |
| pr264 | 49135 | 83317 | 30.43 | 0.0530 |
| pr299 | 48191 | 75137 | 44.08 | 0.0670 |
| pr439 | 107217 | 171018 | 40.49 | 0.0340 |
| pr1002 | 259045 | 401335 | 45.07 | 0.0900 |
| rat99 | 1211 | 1828 | 49.05 | 0.0420 |
| rat195 | 2323 | 3641 | 43.26 | 0.0690 |
| rat575 | 6773 | 10405 | 46.37 | 0.0400 |
| rat783 | 8806 | 14135 | 39.48 | 0.0430 |
| st70 | 675 | 1095 | 37.78 | 0.0540 |
| ts225 | 126643 | 225162 | 22.21 | 0.0300 |
| tsp225 | 3916 | 6059 | 45.27 | 0.0200 |

## Referências:

1-https://pt.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs. Khan Academy.

2-http://www-usr.inf.ufsm.br/~andrezc/ia/heuristicas_construtivas_transparencias.pdf. Souza, M. J. F.