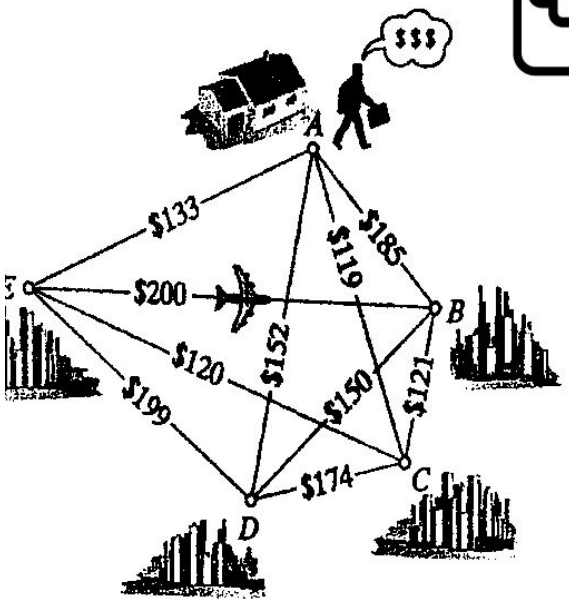
Universidade Federal de Viçosa - *Campus* Florestal

Ciência da Computação

Documentação do Trabalho Prático 2 de Algoritmos e estruturas de dados



Integrantes:

Aryel Penido - 3500

Samuel Sena – 3494

Yuri Dimitre – 3485

OBJETIVO DO TRABALHO:

O objetivo do trabalho foi, além de exercitar os conceitos de estruturas de dados e programação, entrar em contato de forma mais direta com o desempenho de um algoritmo. Para isso foi implementado uma resolução para o problema do caixeiro viajante, recorrente na computação.

O ALGORITMO ESCOLHIDO:

O algoritmo de permutação escolhido foi encontrado em pesquisas no GitHub como uma Gist publica. Link→https://gist.github.com/marcoscastro/60f8f82298212e267021.

O Algoritmo funciona de forma simples, a primeira função *void troca* funciona trocando a posição de números em um vetor de inteiros.

```
void troca(int vetor[ ], int i, int j)
{
        int aux = vetor[i];
        vetor[i] = vetor[j];
         vetor[j] = aux;
}
```

Foi adicionada a função “void VerificaRota" que procura o melhor caminho. Recebe como parâmetro um vetor “permutado”, a matriz de custos, um vetor de rotas, a cidade inicial e um ponteiro para o melhor caminho. A função faz a soma da distancia percorrendo a matriz de custos

```
for (size_t i = 1; i < QuantidadeCidades - 1 ; i++) {
        Soma_Distancia+=MatrizCusto[vetor[i-1]][vetor[i]];
}
```

e no final compara o melhor caminho armazenado (*MelhorCaminho) com essa soma(Soma_Distancia), caso a soma seja menor, *MelhorCaminho recebe um novo valor.

```
 if (Soma_Distancia < *MelhorCaminho) {
        *MelhorCaminho = Soma_Distancia;
}
```

TEMPOS DE EXECUÇÃO:

Abaixo estão os tempos de execução do algoritmo que foi testado em uma maquina com as seguintes configurações:

OS: Ubuntu 18.04 LTS

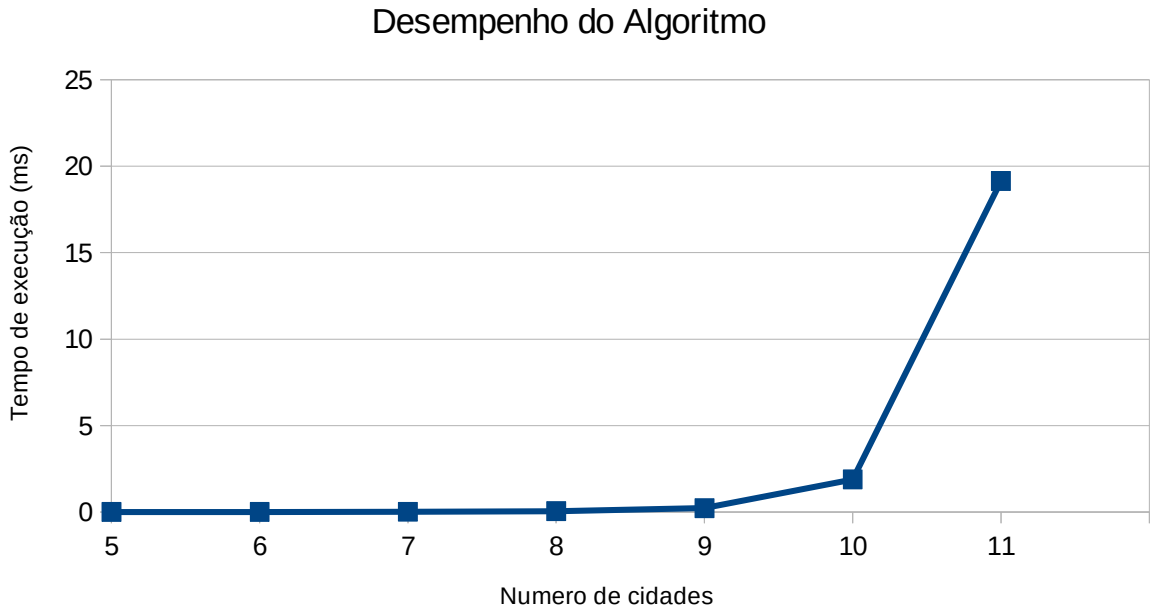
Kernel: x86_64 Linux 4.15.0-38-generic

Interface: KDE 5.44.0 / Plasma 5.12.6

CPU: Intel Core i5-7200u

RAM: 8192mb

Numero de cidades	Tempo de execução
5	0.001 ms
6	0.002 ms
7	0.013 ms
8	0.049 ms
9	0.220 ms
10	1.884 ms
11	18.609 ms
12	3.52 min
13	43.8 min



VOCÊ USARIA ESSE ALGORITMO?

Caso você fosse contratado por uma transportadora para desenvolver um sistema que encontrasse a menor rota a ser percorrida por seus caminhões, saindo e chegando do galpão de estoque e percorrendo uma única vez um conjunto de N localidades, você utilizaria a solução desenvolvida neste trabalho? Justifique a resposta.

Não, pois o algoritmo funciona bem apenas para entradas muito pequenas, a partir de 13 cidades se torna inviável sua utilização.

EXECUÇÃO:

Para realizar a compilação do programa pelo terminal usando o GCC:

(Primeiramente certifique-se que o terminal encontra-se navegado até a pasta

principal do programa)

->Execute os seguintes comandos:

$ make all

(O makefile irá executar a compilação utilizando o GCC)

$ make run

(O programa irá começar a rodar)

OBS: O programa já se encontra compilado, então caso não ache necessário re-compilar,

apenas chame-o executando apenas o último comando.

Para realizar a compilação do programa pelo terminal usando o CLANG:

(Primeiramente certifique-se que o terminal encontra-se navegado até a pasta

principal do programa)

->Execute os seguintes comandos:

$ make clang

(O makefile irá executar a compilação utilizando o CLANG)

$ ./EXEC_CLANG

(O programa irá começar a rodar)

OBS: O programa já se encontra compilado, então caso não ache necessário re-compilar,

apenas chame-o executando apenas o último comando.

UTILIZAÇÃO DO PROGRAMA

A execução do programa é uma atividade bastante intuitiva, o programa inicialmente apresenta uma tela de boas vindas e informa 3 opções, sendo duas delas para a escolha da forma de entrada preferida e a última a porta de saída do programa(finalização do programa), tendo esta o número 9 escolhido propositalmente para evitar finalizações acidentais por parte do usuário.

Caso o usuário opte pela opção de entrada interativa(Opção 1), o programa pedira que o usuário entre com o numero de cidades e depois com as matriculas.

Caso o usuário opte pela opção de entrada por arquivo(Opção 2), uma requisição pelo nome do arquivo de entrada será realizada, então o usuário necessitará preencher o campo com o nome do mesmo que deverá estar previamente localizado na mesma pasta em que o programa.(OBS: O nome do arquivo deverá ser escrito obrigatoriamente com a extensão “.txt” no final. Ex: Nome_do_Arquivo.txt)

O programa ao fim de uma execução de alguma função independente de gênero ou tipo, deverá sempre retornar a tela de boas vindas inicial, a fim de prosseguir com a próxima tarefa ou então a finalização da execução do mesmo.

NOTAS FINAIS E AGRADECIMENTOS

Todo o desenvolvimento do programa e disponibilização do mesmo se encontra licenciado pela licença GPL-v3.0 e hospedado em um repositório do GitHub.

-->Link do repositório: https://github.com/Globson/Tp-do-Caixeiro-Viajante-Assimetrico

Integrantes do grupo de trabalho prático:

Yuri Dimitre – 3485

Samuel Pedro – 3494

Aryel Penido - 3500