CEFET-MG - Campus II
Departamento de Engenharia de Computação
Laboratório de Algoritmos e Estruturas de Dados II

TP II - Linha de Montagem

Implementação de Programação Dinâmica e Algoritmo Guloso

Alunos: Antônio Augusto Diniz Sousa - 201712040146
Rodrigo Dias Moreira
Professor orientador: Thiago de Souza Rodrigues

23 de Novembro
2018

CEFET-MG - Campus II
Departamento de Engenharia de Computação
Laboratório de Algoritmos e Estruturas de Dados II

TP II - Linha de Montagem

Implementação de Programação Dinâmica e Algoritmo Guloso

Relatório do Trabalho Prático II apresentado à Disciplina de Laboratório de Algoritmos e Estruturas de Dados II do Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, como requisito parcial para conclusão da disciplina.

Alunos: Antônio Augusto Diniz Sousa
Rodrigo Dias Moreira

Professor orientador: Thiago de Souza Rodrigues

23 de Novembro
2018

Sumário

1 Apresentação

A tarefa desse Trabalho Prático foi comparar o comportamento de um algoritmo baseado em programação dinâmica e um baseado no algoritmo guloso, ambos para solucionar um mesmo problema, o da linha de montagem.

O algoritmo guloso foi implementado utilizando a lógica do menor caminho e a solução por Porgramação Dinâmica utilizando um algoritmo recursivo sem armazenamento, reaproveitando os cálculos executados através da recursividade.

Os resultados seguem adiante nesse relatório e os códigos fontes podem ser acessados clicando aqui ou acessando em `https://gitlab.com/antonioaads/LAEDII`.

2 Descrição de atividades

Após a execução do código, que não iremos tratar especificamente nesse relatório, pois o mesmo se encontra no repositório citado na introdução devidamente comentado e organizado, o programa retorna uma mensagem, informando o nome do algoritmo utilizado, o tempo gasto para executá-lo, o peso total do percurso e o caminho que foi adotado.

Na impressão do caminho, para diferenciar um ponto da linha de montagem com um caminho de uma linha para outra, a impressão da passagem foi feita entre "!", conforme Figura 1.

O início do programa, uma mensagem conforme essa é impressa: "Entre com o número da montagem que deseja analisar:", na qual deverá informar o número da montagem que deseja analisar, que tem o nome conforme modelo: "montagem**n**.txt", na qual **n** representa qual grafo é analisado e será o valor que deverá digitar.

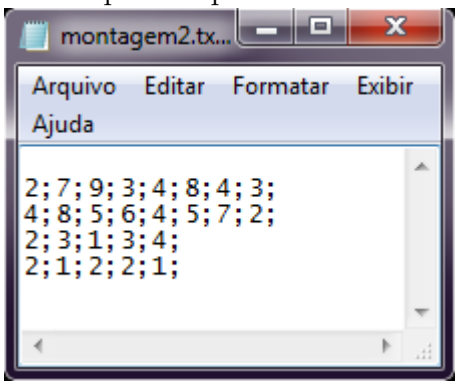
Figura 1: Saída do programa

```
run:
Entre com o número da montagem que deseja analisar:2
CÓDIGO POR PD:
Tempo Total: 3 milissegundos
Total = 38
Caminho -> 2 - 7 - !2! - 5 - !1! - 3 - !1! - 4 - 5 - !1! - 4 - 3 -  FIM

CÓDIGO POR GULOSO:
Tempo Total: 0 milissegundos
Total = 39
Caminho -> 2 - 7 - !2! - 5 - !1! - 3 - 4 - 8 - 4 - 3 -  FIM
CONSTRUÍDO COM SUCESSO (tempo total: 6 segundos)
```

As montagens para teste, deve seguir um padrão de escrita, na qual as duas primeiras linhas do aquivo representa as estações de cada linha, com seu respectivo valor e as duas últimas o tempo de transporte. Além disso, os valores devem ser separados por ";"e com uma linha em branco no início do arquivo, conforme Figura 2.

Figura 2: Modelo para arquivo de entrada do algoritmo



Além das duas montagens dadas como requisito, executou-se também a dada em sala de aula, para averiguar o correto funcionamento do algoritmo, já que esse tinha o melhor caminho conhecido.

2.1 Características

2.1.1 Guloso

O guloso implementado escolhe o caminho com uma soma total menor até uma próxima ramificação, ou seja, não observa só o próximo, mas analisa se terá alguma possível ramificação, e então escolhe a menor.

Isso é perceptível na linha de montagem, principalmente no final, pois após seguir para a última estação da linha, não a mais o que fazer, então o valor considerado deve ser o da última estação mais o valor para sair da estação, e em caso de mudança da linha, o valor da passagem também deve ser considerado.

2.1.2 PD

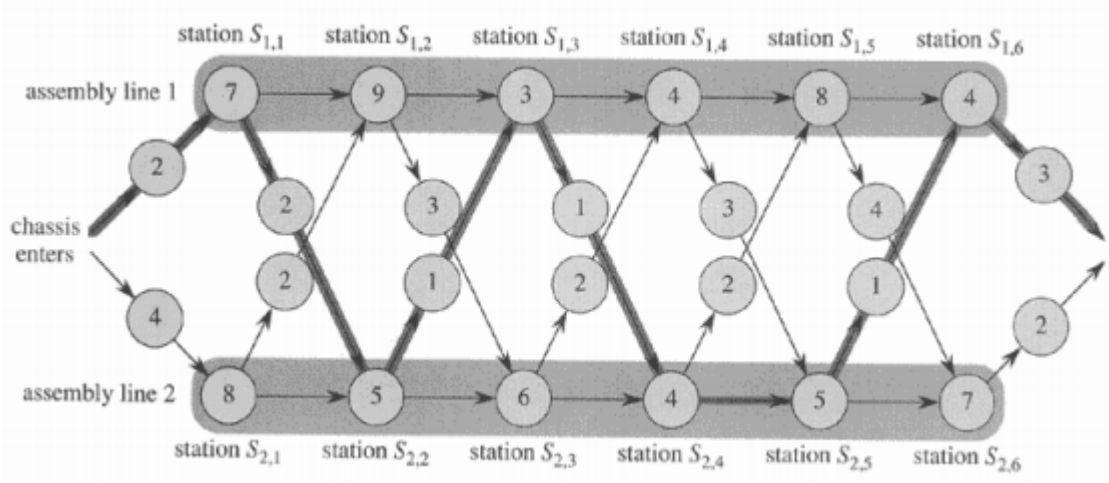
O algoritmo PD utilizado reaproveita os cálculos iniciais, utilizando uma lógica recursiva, na qual todo o percurso feito é memorizado na memória de recursão, para quando for necessário buscá-lo, ele estará disponível.

Essa metodologia diminui consideravelmente o número de cálculos necessário para execução do algoritmo, sendo possível encontrar a solução ótima de maneira eficiente, principalmente, quando comparado a algoritmos de força bruta.

3 Resultados

Para a montagem dada em sala de aula, Figura 3, obteve-se o resultado demonstrado na figura Figura 1. Se seguir o caminho dado e somar os devidos pesos, pode-se confirmar o devido funcionamento do algoritmo com o caminho em negrito da imagem, no caso da aplicação em PD, que retorna o melhor caminho.

Figura 3: Montagem dada em aula



Para as outras montagens, solicitadas no enunciado, foram convertidas para o formato de entrada compatível, conforme Figura 4, obtendo os resultados para a montagem 1, conforme Figura 5, e para a montagem 2, Figura 6.

Figura 4: Entradas das montagens solicitadas

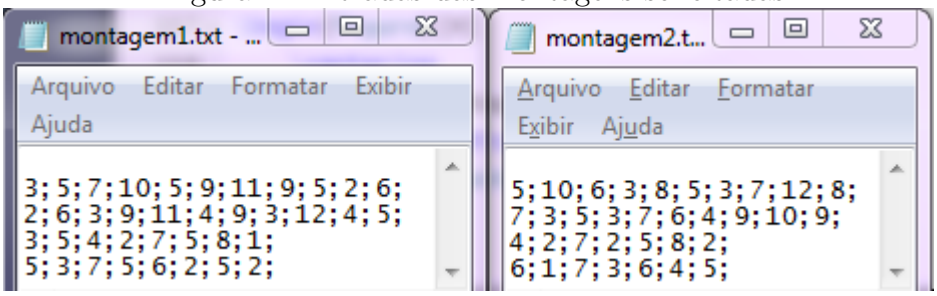


Figura 5: Saída para a montagem 1

```
run:
Entre com o número da montagem que deseja analisar:1
CÓDIGO POR PD:
Tempo Total: 4 milissegundos
Total = 65
Caminho -> 2 - 6 - 3 - 9 - 11 - 4 - 9 - 3 - !5! - 5 - 2 - 6 -  FIM

CÓDIGO POR GULOSO:
Tempo Total: 0 milissegundos
Total = 68
Caminho -> 3 - 5 - !3! - 3 - 9 - 11 - 4 - 9 - 3 - !5! - 5 - 2 - 6 -  FIM
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

Figura 6: Saída para a montagem 2

```
run:
Entre com o número da montagem que deseja analisar:2
CÓDIGO POR PD:
Tempo Total: 2 milissegundos
Total = 62
Caminho -> 7 - 3 - 5 - !1! - 3 - 8 - 5 - 3 - 7 - 12 - 8 -  FIM

CÓDIGO POR GULOSO:
Tempo Total: 0 milissegundos
Total = 63
Caminho -> 7 - 3 - 5 - 3 - 7 - 6 - 4 - 9 - 10 - 9 -  FIM
CONSTRUÍDO COM SUCESSO (tempo total: 1 segundo)
```

4 Conclusão

Pode-se perceber, rodando os mesmos problemas, que o tempo gasto pelo algoritmo feito baseando em Programação Dinâmica demora muito mais do que o feito baseado na lógica Gulosa. Uma observação é que o algoritmo Guloso é executado de maneira tão eficiente que, com a função utilizada, não foi possível captar nem 1 milissegundo.

Porém, pode-se perceber, também, que o algoritmo Guloso implementado retorna bons resultados mas em nenhum dos casos encontrou o caso ótimo, mas se aproximou bem de todos, tendo uma diferença do caso ótimo (dado pela PD) de 3 na montagem 1 e 1 na montagem 2.

Para aplicação desses algoritmos, depende muito de caso para caso. Em casos que nenhum tipo de erro é tolerado e o melhor caminho deve, obrigatoriamente, ser tomado, deve-se aplicar o algoritmo em PD ou algum outro que retorne o caso ótimo, mas em casos que essa necessidade não é crucial, e prefere-se um tempo mais rápido de resposta, a aplicação gulosa consegue ter bons resultados.