Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Engenharia de Software
Fundamentos de Projeto e Análise de Algoritmos
Semestre: 1/2022 - Prof. João Caram
Nomes: Nataniel Geraldo Mendes Peixoto, Pedro Campos Miranda e Izabella de Castro Lucas

Trabalho prático em grupo – parte 2

- **Tarefas do trabalho**

a) Modelar soluções

- **Guloso**

Para modelagem do algoritmo guloso, tipicamente utilizado em problemas de otimização, foram consideradas as seguintes situações:

- O algoritmo escolhe a solução que parece ser a mais correta naquele momento, ou seja, deverá escolher sempre o ótimo local na esperança que isso resulte na solução ótima global (critério guloso);
- Foi utilizado o critério de "custo" por "mm" para escolha de qual rolo utilizar.
  - Ex.: rolo de 3mm -> custo de 9 = 9 / 3 = 3 (custo x mm)
- No caso de empate do "custo x mm" entre as opções de rolo, foi considerado a primeira opção encontrada;
- Função objetivo: min (somatório dos custos, tal que para cada espessura de entrada deverá ser escolhido o menor custo por mm);

- ***BackTracking***

Para modelagem do algoritmo *backTracking*, foram consideradas as seguintes situações:

- O algoritmo irá percorrer cada passo, de forma gradativa, combinando as opções para encontrar o melhor custo possível. Ou seja, ele irá testar soluções parciais para encontrar a melhor solução;
- Caso alguma solução de custo encontrada em uma sequência seja maior do que a já encontrada e salva, a sequência subsequente não será nem avaliada. Com isso, garante-se um critério de "poda" satisfatório, o que tende a diminuir o número de tentativas e consequentemente de recursões;
- Será utilizado como estrutura auxiliar a pilha, tendo em vista armazenar os rolos que estão sendo utilizados naquela solução do momento. Caso a solução seja ótima, ou seja, com menor custo em relação ao apurado anteriormente, a sequência de rolos será salva em uma lista, em substituição aos possíveis valores já salvos;
- O critério de parada será a melhor solução possível, que indicará o menor custo. Para isso, serão analisadas todas as combinações possíveis (considerando, obviamente, o critério de poda citado acima).

- **Programação Dinâmica**

- Para implementação da programação dinâmica será utilizada uma tabela, tendo em vista armazenar uma memória dos resultados previamente calculados. A tabela terá a seguinte característica:
  - Linhas: foram colocadas as possibilidades de espessura das lâminas para entrada nos rolos;
  - Colunas: foram colocadas todas as possibilidades de estado da lâmina após terem passado no rolo, começando pela espessura original de entrada e terminando na espessura objetivo, que é 4mm;
- A abordagem do problema será bottom-up, sendo construída uma solução de forma gradativa, "olhando" sempre para trás, aproveitando os valores já calculados.

Questões comuns:

- O problema é de "mínimo", tendo em vista que o objetivo é minimizar o custo de produção de uma lâmina;
- Restrição: a espessura alvo é de 4 mm. Então, a última espessura de entrada nos rolos é de 5 mm e o respectivo rolo possível de 1 mm.

b) Ler dados de um arquivo

Para leitura dos arquivos de teste, foi criada uma classe específica, com o nome de "GerenciarArquivo". De maneira geral, esta classe teve as seguintes responsabilidades:

- Ler os arquivos de teste, individualmente;
- Após leitura, tratamento da string para separação da espessura inicial da barra e do custo de redução por espessuras;
- Criação de objetos do tipo “Rolo” para cada linha do arquivo;

Além das questões relacionadas a leitura do arquivo, esta classe foi responsável também por:

- Criação de arquivo e posterior escrita dos resultados dos algoritmos. Este arquivo foi salvo na pasta “files”, com o nome de “relatorio”;

c) Implementação das soluções/algoritmos modelados

Para implementação dos algoritmos, foram criadas classes específicas, sendo:

- BackTracking;
- Programação Dinâmica;
- Guloso.

Cada algoritmo representa, basicamente, a modelagem descrita na etapa “a”.

d) Implementação das soluções/algoritmos modelados

Para cada arquivo de teste, foram executados os algoritmos propostos, e os respectivos resultados salvos no arquivo. Segue print do resultado, que pode ser verificado também diretamente no arquivo “relatorio.txt” na pasta “file”:

LaminaçãoTeste	Algoritmo	Custo	Qtd. Rolos	Tempo (ms)*
1	BackTracking	24	2	6
	Guloso	24	4	0
	Dinâmico	24	3	1
2	BackTracking	51	9	6
	Guloso	51	9	0
	Dinâmico	51	9	0
3	BackTracking	57	10	13
	Guloso	58	12	0
	Dinâmico	57	8	0
4	BackTracking	95	12	82
	Guloso	95	13	0
	Dinâmico	95	9	0

* Para cálculo do tempo de execução, foi considerado o retorno original do método “System.**currentTimeMillis**()”, em milissegundos, tendo em vista os pequenos valores encontrados.

Em síntese, os resultados foram:

- Com o aumento do número de possibilidades de espessura de entrada nos rolos, foram observadas mais diferenças na “qtd. Rolos”. Tal comportamento pode ser explicado devido ao aumento do número de caminhos e possibilidades/combinações. Quanto mais caminhos, mais variações podemos ter para um mesmo resultado de custo mínimo. No Teste 4 por exemplo, embora o custo tenha sido o mesmo (95), o número de rolos do dinâmico foi 9, sendo 4 menos do que o guloso;
- Além disso, a distribuição dos valores do arquivo de teste pode interferir muito, bem como os critérios de seleção dos resultados, tais como os critérios de desempate. Ex.: se o rolo de 1mm e 3mm tiverem o mesmo custo por mm, qual utilizar? Nós optamos por escolher o primeiro como critério puramente guloso, mas se o critério fosse guloso e com restrição de maior espessura de redução (3mm), poderíamos ter encontrado outro resultado;
- Podemos considerar que o guloso foi uma boa solução para minimizar o custo. Em comparação com o backTracking e dinâmica, que encontram a solução ótima, o guloso "errou" apenas uma vez (teste 3);
- No cenário em que a quantidade de rolos não é uma restrição, podemos encontrar mais de uma combinação para o mesmo custo;
- O tempo de execução dos algoritmos que mais se destaca, em todos os testes, é o backTracking. Este comportamento se justifica pois é o algoritmo que faz o maior número de combinações. Esse comportamento não é observado no guloso, tendo em vista que ele pega a melhor opção local, sem voltar atrás nas decisões, e nem na dinâmica, uma vez que aproveita os valores já calculados anteriormente e que estão salvos em tabela.