Trabalho prático em grupo – parte 2

Problema da laminação de barras de ferro

Grupo: Erick Vinicius Oliveira de Paiva, Maria Clara Sales Jabali Maruch, Pedro Vítor Felix da Costa

Tarefas do trabalho:

1) **Modelar soluções**

- **Guloso**

O algoritmo guloso realiza a escolha que parece ser a melhor naquele momento, tomando decisões com base nas informações disponíveis. Ele possui a característica de não se arrepender de uma decisão, logo ele não volta à sua solução. Para fazer a escolha foi utilizado um critério guloso: custo por redução de espessura. Por se tratar de um problema de minimização, se o resultado obtido for menor que o disponível, a decisão do algoritmo será ele. Exemplo: (custo) 6 / 2mm (redução de espessura) = 3.

- **Backtracking**

A princípio, foi preciso ter em vista como a ideia do backtracking funciona: buscar todas as soluções, como um força bruta, em um formato de árvore, no qual será dado um passo atrás (backtrack) toda vez que encontrar o resultado, que é a lâmina em 4mm de espessura.

Assim, a melhor modelagem de estrutura de algoritmo para esse algoritmo é através da recursividade, uma vez que a recursão faz operações sucessivas até chegar à base pré definida, e quando chega a essa base ele retorna à última chamada de recursão solicitada e continua o algoritmo a partir daquele ponto. Desse modo, a recursividade se encaixa perfeitamente nesse tipo de solução de problemas que é o backtracking.

- **Programação dinâmica.**

A programação dinâmica foi construída utilizando a seguinte função:

MT[i][j] = min(T[i-1][j], S[i][j-i], T[i][j-1]+C[1][j-1])

- **MT[i][j]** = Melhor total para a célula na posição i,j;
- **T[i-1][j]** = Custo total da célula acima;

- **S[i][j-i]** = O custo do passo para chegar na espessura da célula i,j somado ao melhor total na espessura antes desse passo;
- **T[i][j-1]+C[1][j-1]** = O custo do passo de 1mm somado ao melhor total anterior (um milímetro a menos).

Construiu-se uma matriz em que na primeira coluna ficaram os passos de espessura possíveis (1, 2 e 3mm) e na primeira linha as espessuras da máxima colocada no arquivo até o objetivo de 4mm, reduzindo de 1 em 1mm.

Na tabela do programa foi necessário armazenar nas células da primeira linha o total de MAX_INT, já que é necessário minimizar o total da célula acima. Nas células centrais, junto com o custo total se armazenou o custo para sair daquela espessura de acordo com o passo, para que fosse possível calcular os totais sem consultar os custos na matriz inicial.

Para chegar na sequência de cilindros é necessário percorrer a última linha da tabela. Compara-se o valor o custo do passo com a diferença dos totais de acordo com o passo correspondente, por exemplo:

SE (C[3][j-1] == MT[3][j] - MT[3][j-1]) - caso isso seja verdadeiro, quer dizer que o último passo para chegar naquela espessura da célula MT[3][j] foi de 1mm.

2) Ler dados de um arquivo de configuração

Para a leitura dos dados de um arquivo foi utilizado o Data Access Object. Foi implementado uma classe com o nome “MatrizDAO” que implementa a interface “DAO”.

3) Implementar as soluções/algoritmos

As três soluções propostas foram implementadas e cada uma possui uma pasta no projeto com suas respectivas classes. O vídeo é uma referência da explicação de tais algoritmos.

4) Resultados

Testes	Soluções	Sequência	Qnt. rolos	Custo	Tempo (ms)
1	Backtracking	-	-	24	11
	Prog. Dinâmica	3, 1, 1, 1	4	24	4
	Guloso	1, 1, 1, 1, 1, 1	6	28	<1
2	Backtracking	-	-	51	13
	Prog. Dinâmica	3, 1, 1, 1, 1, 1, 1, 1, 1	9	51	5
	Guloso	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	11	55	0
3	Backtracking	-	-	57	15
	Prog. Dinâmica	3, 3, 3, 1, 1, 1, 1, 1, 1, 1	10	57	4
	Guloso	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	16	65	1
4	Backtracking	-	-	95	14
	Prog. Dinâmica	1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3	13	95	3
	Guloso	1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	21	127	1

Visto a tabela acima, pode-se perceber que:

- Neste problema o algoritmo guloso não apresentou boas soluções para minimizar o custo.
- Em relação ao tempo de execução dos algoritmos, o guloso apresentou o menor tempo entre eles, visto que faz o menor número de operações para tentar encontrar o melhor resultado. O tempo da programação dinâmica pode ser explicado pelo fato de aproveitar os valores já calculados anteriormente e o backtracking apresentou o maior tempo, pois com o aumento da entrada, ele calcula um número maior de caminhos.

- Foi observado que com o tamanho da espessura de entrada pode-se perceber que a quantidade de rolos aumentou também. No primeiro e segundo teste, a diferença do guloso para a dinâmica estava em mais ou menos 2 rolos, todavia, já no teste 4 a diferença foi de 7.
- O backtracking, apesar de não ter sido possível mostrar a sequência de rolos, possui as mesmas respostas que a programação dinâmica.