PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Coração Eucarístico

Projeto e Análise de Algoritmos

Trabalho Prático 01

Adhonay Júnior Silva

27/05/2018

Como esse problema pode ser modelado para o paradigma guloso?

O problema foi modelado para maximizar o lucro de Alfred, levando apenas em consideração o maior lucro e ignorando o custo, a resolução entrada base (dado junto com problema) não seria possível, pois o custo ultrapassaria o problema, então o modelo foi baseado da seguinte forma:

-Lucro / custo assim criando uma nova variável de cada prato que chamei de peso da relação entra as duas variáveis .

-Existe um objeto Prato como atributos: lucro, custo, frequência, peso.

-Ordeno(Quicksort) esse vetor de objetos pelo peso e trabalho somente com a primeira posição do vetor que dar o maior lucro/custo no determinado momento.

-Após os devidos cálculos, e ordenado novamente o vetor com o novo peso usado na primeira posição.

-Feito isso X vezes Número de Dias, o método guloso retorna os melhores pratos de Lucro/custo do dia.

-O resultado final e a suma do lucro desses pratos, e a posição a quais ele pertence na entrada.

Seu algoritmo guloso dá a solução ótima? Por quê?

Não garante. Para alguns casos ele pode gerar e melhor solução, mas não para todos, pois o modelo utilizado para ele, baseia na variável peso (lucro/custo), relacionando a mesma indiretamente com outras relevantes no problema como numero de dias, pratos e orçamento. Segue abaixo exemplo pra 2 casos.

Caso 1	Caso 2:
Entrada: 3 5 20 2 5 18 6 1 1 3 3 2 3 0 0 0	Entrada: 5 3 21 2 5 1 4 3 4 0 0 0
Saída Guloso: 13.0 1 5 1 Saída Esperada: 13.0 1 5 1	Saída Guloso: 22.0 2 1 2 1 2 Saída Esperada: 23.0 1 2 1 2 1

Para o ambos os casos ele calcula os pesos para cada prato e trabalha com esse peso para dar o resultado, no Caso 1 ele pegou os pratos: 1(peso = 2.5), 5(peso = 1.5) e 1 novamente totalizando de lucro 13.0 sendo a solução ótima para esse caso. No Caso 2 segue a mesma linha de raciocínio pegando o prato 2(peso = 4) e prato 1(peso = 2.5), mas a melhor solução seria utilizar três pratos 1 e não somente 2 como foi utilizado no guloso, mas ele escolhe o melhor peso entre lucro/custo e isso não me garante a solução ótima pois caso o orçamento seja muito grande comparados com

o custo, o melhor caso pode ser dado pelo prato que me retorne o maior lucro sem levar em consideração o custo.

Como esse problema pode ser modelado para o paradigma de programação dinâmica?

O problema foi modelado para maximizar o lucro de Alfred, Levando em consideração todas variáveis do problema construindo uma tabela: prato x dias com o maior lucro para cada posição, diferente do método guloso que trabalha diretamente com apenas lucro e custo na programação dinâmica vai ser levado em conta também o número de dias, pratos, orçamento fatores esses que influencia diretamente a solução ótima do problema em questão. Resolvendo todos subproblemas menores, mas somente reusa as soluções ótimas.

- Aplicando programação dinâmica a uma função que leva 4 parâmetros inteiros dinamico(int ultimoPrato, int frequencia, int dia, int custoTotal)

- As devidas comparações de regras como divdir lucro por dois se prato repetido no dia seguinte, não ultrapassar orçamento, que foram propostas no problema o problema, e a chamada recursiva para a função faz a montagem da tabela com o melhor lucro possível.

- Para evitar chamar essa função cara repetição, foi armazenado os resultados em uma matriz com quatro dimensões "tabela[ultimoPrato][frequencia][dia][custoTotal] = melhorPrato"com o valor do melhor prato.

-O print do caminho do último dia até o primeiro dia, se o prato x for o mesmo lucro que o anterior eu devo considerar o menor custo, pela tabela.

-Tabela apenas para ilustração dinâmica

X =Lucro por prato (cada objeto possui esse valor atualizado implicitamente), linha na tabela apenas para ilustrar.
Y= Soma do lucro por Dias, existe um somador que vai gerar esse resultado, mostrado apenas para ilustrar o resultado final.

O caso abaixo serve somente para ilustrar as partes da melhor solução de um problema x em questão, mas na tabela vai existir todas possíveis combinações geradas a partir de outros pratos, resolvendo esses subproblemas até chegar na resolução dos pratos mais lucrativos.

Modelo

Ultimo prato	Frequência
Dia	Custo Total

X/Y	5		9		14		18		23	
5	1	0	1	1	1	0	1	1	1	0
	1	2	2	4	3	5	4	7	5	8
4	2	0	2	0	2	1	2	0	2	1
	1	1	2	3	3	4	4	6	5	7
4	3	0	3	0	3	0	3	0	3	0
	1	3	2	5	3	6	4	8	5	9

Para cada situação através das condições dadas, e retornado o melhor prato para cada dia e no final somado os lucros deles e a ordem em qual foi selecionado.

Discuta a sub-estrutura ótima e a sobreposição dos problemas.

No algoritmo de programação dinâmica a escolha pode depender da solução dos subproblemas, enquanto um algoritmo guloso vai tentar escolher a melhor solução naquele momento, ou seja, a solução local.

A solução dos problemas na programação dinâmica parte de baixo para cima, enquanto um algoritmo guloso vai de cima para baixo, ou seja, na programação dinâmica, as soluções para todos os subproblemas são calculadas partindo dos menores subproblemas para os maiores.

Os resultados dos subproblemas na programação dinâmica são salvos em tabelas por exemplo, facilitando a prova de correção para determinar qual melhor resultado para o problema tento como função minimizar ou maximizar algum ganho por exemplo

Se algum algoritmo clássico foi adaptado para resolver o problema, qual foi ele?

Foi abordado a lógica da mochila dinâmica mas com algumas variações , com intuito de maximizar o lucro, onde meu orçamento seria o tamanho da mochila, o custo seria o valor subtraído a esse orçamento, o lucro pode ser ilimitado ou seja quanto maior melhor pois a ideia e maximizar o mesmo, existe também o numero de pratos que seria o número de itens que temos para escolher e o numero de dias que seria o numero de itens que posso levar na minha mochila ou pratos que posso fazer em casa dia.

Qual o custo assintótico (em Big-O) do tempo de execução e espaço em memória utilizado? Não se esqueça de formular a equação de recorrência da abordagem baseada em programação dinâmica (não é necessário resolver a equação explicitamente).

O custo assintótico do tempo de execução dos algoritmos guloso e dinâmico são representados da seguinte forma: O guloso considera duas estruturas principais são elas uma repetição (for) que é executado "i" vezes sendo representado pelo numero de dias e um laço mais interno que seta as posições com seus respectivos valores de entrada e ordena novamente a primeira posição do vetor cujo o valor foi alterado "k" vezes, assim a complexidade do algoritmo guloso e O(ik). Entretanto o Dinâmico apresenta apenas uma estrutura/laço de repetição principal, mais nela e tratada outras subestruturas importantes para o problema, onde e feita a recursão, caracterizada pelo número do dia(u), frequência(m), ultimo prato(p) e custo total(c), "dinamico(p, freqx, dia+1, custoTotal-custo[i])" para todo valor inteiro sendo essa chamada interna em relação a repetição externa para o numero de dias(n), temos que O (p *m *(u+1)* (c - p[u])).

Em relação ao custo de memória utilizado e assumindo que o tipo double ocupa 8 bytes e o tipo int ocupa 4 bytes. Para o algoritmo guloso existe um vetor inteiro e um do tipo "Prato" constituído por 7 atributos sendo 4 do tipo "int" e 3 do tipo "double", considerando que o vetor prato e alocado em relação ao número de dias(k), e o vetor de inteiros e alocado em relação ao numero de pratos(i) temos o custo de memória aproximado pela seguinte equação
k *((4 bytes * 4) + 8 bytes * 3) + i * 4 bytes, enquanto para o algoritmo dinâmico tenho um vetor de quatro dimensões do tipo "Prato" que tem 3 atributos, 2 do tipo "int" e 1 do tipo "double" dado que o tamanho da tabela e descrito por número do dia(n), frequência(m), último prato(p) e custo total(c) então a equação em relação ao custo de memória do dinâmico baseia-se na proporção da tabela alocada na memoria que poderia ser um calculo aproximado de F = n*m*n*p , para ((4 bytes * (2 * (F)) + 8 bytes * (F)) * 4) * 4 bytes. Sendo todas 4 variáveis de alocação inteiras.

Em qual máquina o tp foi testado?

Especificações do Windows

Edição	Windows 10 Home
Versão	1709
Compilação do SO	16299.371

Especificações do dispositivo

Nome do dispositivo	DESKTOP-GE7INMC (Adhonay-júnior após reiniciar)
Processador	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz
RAM instalada	8,00 GB

Como sua implementação se comporta em termos de tempo de execução e memória alocada? Apresente uma variação experimental da solução, com entradas de tamanho diferente tentando, pelo menos, discutir também o melhor e o pior caso.

Ambos algoritmos foram implementados em Java, para analise de tempo de execução e memoria alocada foi usado respectivamente:
- System.currentTimeMillis(); // para analise do tempo em determinado momento
- (Runtime.getRuntime().totalMemory() - Runtime.getRuntime().freeMemory()); // para calculary a memória usada.

Ambas funções foram chamadas antes da inicialização do algoritmo e depois do seu termino e subtraindo respectivamente seu valor para obter o resultado final, as medidas foram calculadas a partir do processo java criado após execução do código no NetBeans.
Segue a tabela com resultados obtidos:
Entrada pequena: Exemplo proposto no próprio exercício.
Entrada Media: Gerada manualmente com 7 subproblemas aleatórios.
Entrada Grande: Gerada a partir da pequena e media entrada, mais 12 subproblemas aleatórios.
Memória e tempo: Medidas em kbytes é milissegundos respectivamente, do lado esquerdo da linha de cada entrada sendo o algoritmo dinâmico e do lado direito do guloso.

Entrada/Custo	Memoria (kb)		Tempo(ms)	
Pequena	665	33	329	198
Media	16640	198	606	472
Grande	18913	554	1279	776

A partir da tabela notamos que o melhor e pior caso para ambos algoritmos está correlacionado com o tamanho da entrada, sendo o melhor quando possuo apenas um prato/dia para haver somente uma escolha a ser feita para resolução do caso ou não. E o pior podendo variar o número de dias/pratos, variáveis principais para ditar o quão custoso vai ser o algoritmo tanto para o guloso em relação a escolha local na hora da comparação quanto ao dinâmico para construção da tabela de soluções.

REFERÊNCIAS

PROJETO E ANÁLISE DE ALGORITMOS PUC FELIPE DOMINGOS. **Técnicas de projeto (parte 4) projeto e análise de algoritmo**. Disponível em: <https://drive.google.com/file/d/1ypwsdnurcbaa7regcqch-4tquqk7pgvi>. Acesso em: 21 mai. 2018.

PROJETO E ANÁLISE DE ALGORITMOS PUC-FELIPE DOMINGOS. **Técnicas de projeto (parte 4) projeto e análise de algoritmos**. Disponível em: <https://drive.google.com/file/d/1nvzpuwu4ldbeishjwlznxuvrzrrobrud>. Acesso em: 21 mai. 2018
.
WWW.IME.USP.BR. **Algoritmos gulosos**. Disponível em: <https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/guloso.html>. Acesso em: 21 mai. 2018.

WWW.IME.USP.BR. **Programação dinâmica**. Disponível em: <https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/dynamic-programming.html>. Acesso em: 21 mai. 2018.