Documentação Trabalho Prático 3 da Disciplina de Algoritmos 1

Daniel Oliveira Barbosa
Matrícula: 2020006450

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

1. Introdução

Esse trabalho lida com o problema de encontrar o número mínimo de ligas metálicas necessárias para atender a demanda de um cliente, dado tamanhos de liga pré-determinados. Dado isso, usamos o paradigma de programação dinâmica para resolver esse problema. Aqui nessa documentação mostraremos a modelagem do problema, analisaremos a complexidade polinomial que tivemos e também discutiremos sobre a NP-Completude do problema levando em consideração a entrada em bits.

2. Modelagem

Modelamos esse problema usando o paradigma de programação dinâmica usando uma implementação bottom-up. Assim, a solução resolve o problema para o caso trivial (demanda de zero demanda zero ligas para se satisfazer) e vai encontrando a solução ótima para subproblemas cada vez maiores até chegar à solução ótima de N (a demanda inicial que recebemos como entrada).

Para cada demanda que não tenha sido encontrada a o número mínimo de ligas, calculamos o número mínimo de ligas para essa demanda e armazenamos na tabela dinâmica. Já para as demandas que já tivermos resolvido, ao invés de re-calcularmos o número mínimo de ligas, utilizamos o valor armazenado na tabela dinâmica.

Pseudo-código:

- Criamos uma tabela dinâmica chamada **demands** onde armazenaremos a solução ótima para cada demanda e inicializamos ela com infinitos.
- Tratamos o caso base de demanda zero preenchendo a primeira posição de **demands** para zero.
- Para cada tamanho de liga, iteramos sobre todas as possíveis demandas que são maiores ou iguais ao tamanho da liga atual.
- Verificamos se a demanda atual menos o tamanho da liga atual pode ser atendida.
- Se for possível atender a demanda atual usando um número menor de ligas, atualizamos o valor correspondente em **demands**.

- Depois de percorrer todos os tamanhos de ligas e todas as possíveis demandas, o valor da última posição da tabela dinâmica **demands** representa o número mínimo de ligas necessário para atender a demanda do cliente.

A estrutura de dados que utilizamos para armazenar a tabela dinâmica foi um vetor alocado dinamicamente. Selecionamos essa estrutura por sua simplicidade e eficiência de acesso (feita em O(1) para qualquer posição do vetor).

Além de uma implementação iterativa (bottom up), também fizemos uma implementação recursiva (top down) chamada getMinNumAlloysRec(), apesar de não ser utilizada na main.cpp. Escolhemos por continuar com a versão iterativa pois a recursiva, para algumas instâncias de entrada, dava um erro de stack overflow, dado que a quantidade de dados que precisava ser armazenada na pilha estourava o limite de memória de pilha padrão de diversas máquinas.

3. Análise de Complexidade

Esse algoritmo itera:

- Por todos os tipos de ligas (loop externo)
- Por todas as demandas para cada tipo de liga (loop interno)

Dessa maneira, a complexidade do nosso algoritmo é determinado pelo número de tipos de liga (que chamaremos de n) e pela demanda do cliente (que chamaremos de m). Portanto nosso algoritmo possui complexidade O(**n * m**). Porém isso é a análise do tamanho da entrada em termos do número de inteiros.

Quando analisamos o tamanho da entrada em termos do número de bits, percebemos que o problema não é polinomial como analisamos no item 3. de análise de complexidade. Isso acontece pois dado uma entrada de tamanho N, a sua representação em bits B é $B = \log_2 N$ que é equivalente a $n = 2^b$. Dessa maneira a complexidade passa de O(**n * m**) para O($\mathbf{2^n * 2^m}$), que claramente não é polinomial. Assim, chegamos à conclusão de que esse algoritmo é pseudo-polinomial.

4. NP-Completude

Agora vamos provar a NP-Completude do nosso problema.

Dado que o problema de decisão do subset sum é NP-Difícil, se conseguirmos reduzir esse problema para para o problema de decisão das ligas metálicas, poderemos provar que o problema das ligas metálicas também é NP-Difícil.

Abreviando o problema de decisão de subset sum para dSS, temos que esse problema é determinado por:

- Entrada:

    - Um conjunto S de inteiros (valores que podem ou não ser somados)
    - Um inteiro n (corresponde ao número de elementos em S)
    - Um inteiro W (soma desejada)
    - Um inteiro k
- Pergunta:
    - Existe um subconjunto S’ de S de tamanho no máximo k cuja soma dos seus elementos é igual a W
- Obs.: cada elemento do conjunto S pode ser utilizado apenas uma vez

Abreviando o problema de decisão das ligas metálicas para dLM, temos que esse problema é determinado por:

- Entrada:
    - Um conjunto M de inteiros (ligas que podem ou não satisfazer a demanda)
    - Um inteiro n’ (corresponde ao número de elementos em M)
    - Um inteiro D (demanda a ser satisfeita)
    - Um inteiro k’
- Pergunta:
    - Existe uma combinação de ligas que satisfaz a demanda D (ou seja, a soma dos elementos de M é igual a D) cujo número de ligas é no máximo k’.
- Obs.: cada elemento do conjunto L pode ser usada quantas vezes for necessária, mas será utilizada no máximo floor(D/min(M))

Dado que no problema de dSS, os elementos do conjunto de entrada podem ser usados apenas uma vez e no problema dLM, os elementos do conjunto de entrada podem ser usados quantas vezes for necessário, precisamos, para cada elemento de S de dSS (que chamaremos de s_i), fazer um mapeamento para dois elementos de M, que chamaremos de m_i1 e m_i2.

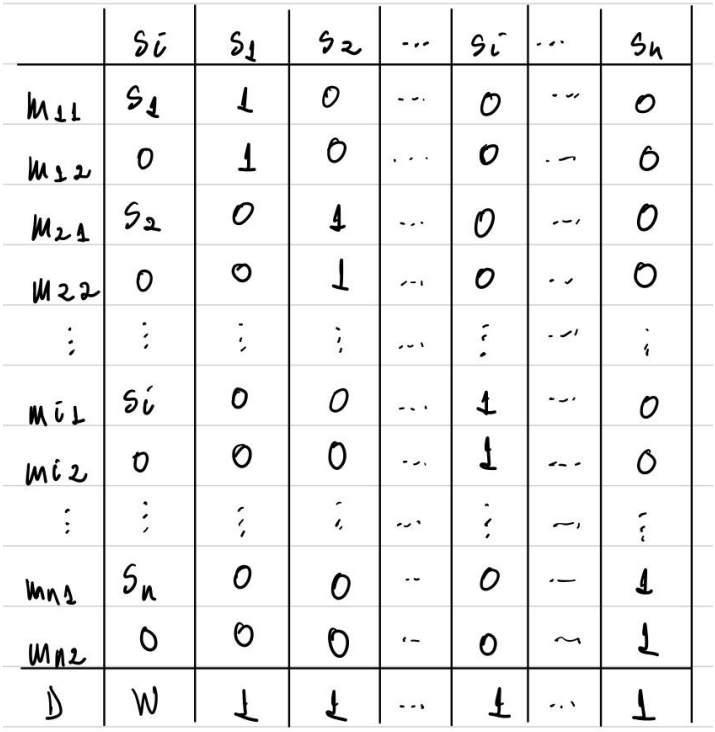
Ou seja, $(s_i \rightarrow m_{i1})$ e $(s_i \rightarrow m_{i2})$, onde

- m_i1 corresponderá ao caso em que s_i pertence a S’ (subconjunto de solução do dSS) e
- m_i2 corresponderá ao caso em que s_i não pertence a S’ (subconjunto de solução do dSS).

Assim, o valor de n’ será 2*n dado que são dois elementos de M para cada elemento de S. Para o valor de D, será feito a transformação de forma que cada elemento só possa ser escolhido uma vez. Já para o valor de k, será feita a transformação k = k’.

Dessa maneira, o primeiro dígito (da esquerda pra direita) dos elementos m_i1 (correspondente a s_i pertencente a S’) serão exatamente o valor de s_i e os de m_i2 serão 0 (correspondente a s_i não pertencente a S’). O elemento i ao qual m_i1 e m_i2 correspondem será indicado pelos próximos dígitos de m_i1 e m_i2. A demanda D, por sua vez, terá valor W em seu dígito mais

significativo e depois será seguido por n dígitos 1, impondo a restrição de que no dSS, cada s_i pode ser selecionado apenas uma vez. Toda essa transformação será descrita na tabela abaixo.



	s_i	s_1	s_2	...	s_i	...	s_n
m_{11}	s_1	1	0	...	0	...	0
m_{12}	0	1	0	...	0	...	0
m_{21}	s_2	0	1	...	0	...	0
m_{22}	0	0	1	...	0	...	0
⋮	⋮	⋮	⋮	...	⋮	...	⋮
m_{i1}	s_i	0	0	...	1	...	0
m_{i2}	0	0	0	...	1	...	0
⋮	⋮	⋮	⋮	...	⋮	...	⋮
m_{n1}	s_n	0	0	...	0	...	1
m_{n2}	0	0	0	...	0	...	1
D	W	1	1	...	1	...	1

Ou seja fizemos o mapeamento dos elementos de S para M, pois cada linha dessa tabela (com excessão da linha D) corresponderá a um elemento de M. Todos os números contidos nas linhas dessa tabela estão veja que o valor B, correspondente ao maior valor entre o maior valor do conjunto S e o valor de W sempre limita superiormente k, logo escolhemos B como a base tornando possível representar todos os elementos da tabela sejam representados em um dígito.

Fizemos também o mapeamento de W para D que está armazenado na última linha D da tabela e seus dígitos são a soma das colunas. Como fizemos o mapeamento de k = k' e k é menor ou igual a n (tamanho do conjunto S), garantimos que nunca haverá o carry na soma da coluna e interferindo com o valor de D. Fizemos o mapeamento n para n' anteriormente (n' = 2*n) então vamos omitir sua repetição. Assim fizemos o mapeamentos completo da entrada de dSS para dLM. Além disso, podemos garantir que foi feito em tempo polinomial no tamanho da entrada em bits dado que apenas dobramos o número de elementos do conjunto S e adicionamos n dígitos em m_i1 e m_i2.

Dado essa tranformação e dado que ela é feita em tempo polinomial, nos resta provar que dSS (para uma entrada I) retorna sim se, e somente se, (para a entrada I transformada da maneira descrita acima) dLM também retorna sim. Podemos provar isso pois os dígitos 1 de D garante qualquer solução do problema das ligas metálicas escolherá, para cada elemento s_i de S, um valor único valor de m_i1 ou um único valor m_i2 exclusivamente (nunca os dois), pois o dígito

i de D pode ser no máximo 1. Caso contrário (caso fosse escolhido m_i1 e m_i2 ao mesmo tempo ou um dos valores fosse escolhido mais de uma vez), a soma daquela coluna correspondente a s_i ultrapassaria o valor de 1, imposto pelo dígito 1 de D. Juntamente a isso, o dígito mais significativo de D (W), garante que a soma dos elementos seja exatamente W. Assim, provamos que as exigências do problema dSS (cada elemento de S pode ser usado uma única vez e a soma deve ser igual a W) são satisfeitas e que se dLM retorna sim, dSS também retornará sim.

Veja também que podemos fazer a construção da entrada de dSS partindo de uma entrada dLM usando a inversa da transformação utilizada na tabela. Ou seja, dado um elemento P_j do conjunto M da entrada de dLM, P_j será correspondente ao elemento m_i1 e consequentemente correspondente ao elemento s_i de S se o dígito i de P_j for 1 e se o seu dígito mais significativo for igual a s_i. O mesmo serve para a correspondência a m_i2, porém o dígito mais significativo será 0. Veja que as condições se mantêm satisfeitas para dLM. Assim, provamos que se dSS retorna sim, dLM também retornará sim.

Dado que foi provado que se dLM retorna sim, dSS também retornará sim e se dSS retorna sim, dLM também retornará sim, concluímos que o problema dLM é NP-Difícil.

Agora vamos provar que dLM pertence a NP provando que dLM é verificável em tempo polinomial no tamanho da entrada em bits dado um certificado. O certificado é uma combinação de ligas que satisfaz a demanda D (ou seja, a soma dos elementos de M é igual a D) cujo número de ligas é no máximo k'. Podemos verificar esse certificado somando as ligas e verificando se equivale a D e verificando que o número de ligas na combinação é no máximo k. Como ambas essas verificações são feitas em tempo polinomial no tamanho da entrada em bits. Assim concluímos que dLM pertence a NP.

Como foi provado que dLM é NP-Difícil e que dLM pertence a NP, temos que dLM é NP-Completo.

5. Conclusão

Em suma, neste trabalho pudemos resolver o problema de encontrar o número mínimo de ligas metálicas necessárias para atender a demanda de um cliente, dado tamanhos de liga pré-determinados. Para isso, utilizamos o paradigma de programação dinâmica para resolver o problema no que parecia ser uma complexidade polinomial. Porém, ao analisarmos a complexidade do problema em relação ao tamanho da entrada em bits, percebemos que a complexidade era não polinomial, tornando o problema um problema pseudo-polinomial. Por fim, provamos a NP-Completude desse problema provando que ele pertence a NP e depois reduzindo ele a partir de um problema NP-Difícil visto em aula. Dessa forma, concluímos o trabalho com sucesso e muito aprendizado sobre programação dinâmica e NP-Completude.

Referência: SUBSET-SUM is NP-Complete. Disponível em: community.wvu.edu/~krsubramani