UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS	**Universidade Federal de Minas Gerais** **Turma:** Ciência da Computação **Prof.:** Vinícius **Nome:** Rubia Alice Moreira de Souza **Matrícula:** 2022043507

1. Introdução

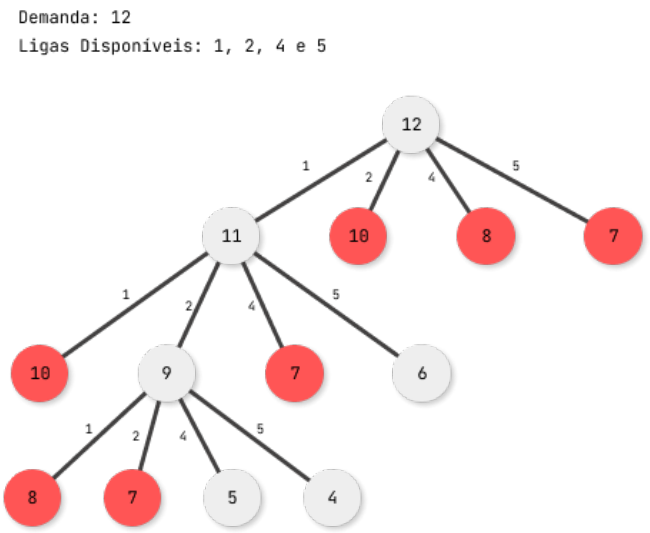
O trabalho consiste no desenvolvimento de um algoritmo para conseguir determinar a menor quantidade de ligas metálicas dentre diferentes tamanhos que somam uma certa quantidade solicitada por um cliente. Os tamanhos de ligas metálicas que podem ser entregues devem ser selecionadas de um grupo específico disponível no estoque.

Esse problema se assemelha muito ao problema da mochila. Contudo, neste caso, temos que a quantidade de itens que podem ser selecionados é ilimitada. Ou seja, para atender uma demanda, podemos selecionar qualquer quantidade dos diferentes tamanhos de ligas metálicas a fim de atender o tamanho solicitado.

Todo o histórico de desenvolvimento, pode ser acompanhado pelos commits neste repositório: UFMG-ALG-Centro-De-Distribuicao. Ele apenas se tornará público após a data final de entrega deste trabalho.

2. Modelagem

Assim como no problema da mochila, encontramos diversas repetições ao dividir o problema original em subproblemas. Por exemplo: neste caso com uma demanda de 12 unidades de ligas metálicas para ser atendida com ligas de tamanhos 1, 2, 4 e 5, teríamos as seguintes possíveis divisões:



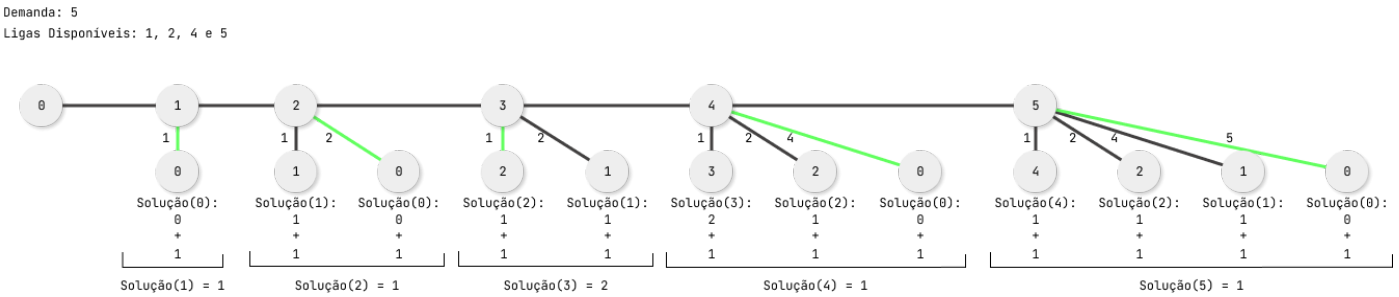
É possível notar que assim que resolvemos o subproblema 11, todos subproblemas restantes do problema original 12 já foram computados, que são os vértices marcados em vermelho. Uma vez que os subproblemas se sobrepõem, podemos utilizar Programação Dinâmica para evitar repetições de soluções.

Podemos usar um vetor para armazenar as soluções dos problemas que já foram computados anteriormente. Além disso, podemos identificar um subproblema apenas observando o valor da demanda informada, uma vez que as ligas disponíveis para resolver cada subproblema sempre serão as mesmas. Assim, o index de uma posição no vetor de soluções será equivalente à demanda do problema definido por ela.

Podemos resolver problemas de duas formas: Top Down e Bottom Up. A única diferença entre as duas soluções consiste na ordem que os subproblemas serão solucionados. No entanto, a solução escolhida foi a Bottom Up, pois a Top Down tem o risco de ultrapassar o limite da Call Stack definida pelo sistema operacional, gerando um erro de StackOverflow.

2.1. Solução Bottom Up

A solução Bottom Up será construída gradualmente a partir de zero até o valor da demanda:



Para cada valor de zero até a demanda final, iremos testar todos os tamanhos de ligas metálicas disponíveis para atendê-lo. Cada teste será a liga metálica atual mais a menor quantidade necessária para atender a demanda restante.

Por exemplo: Se estamos no valor 3, teremos dois casos. O primeiro será a liga metálica de tamanho 1, que terá uma quantidade mínima de ligas igual a 1 mais a solução do subproblema em que a demanda era 3 - 1 = 2. Já o segundo será a liga metálica de tamanho 2, que terá uma quantidade mínima de ligas igual a 1 mais a solução do subproblema em que a demanda era 3 - 2 = 1.

Ou seja, além da liga metálica que estamos considerando atualmente, iremos precisar da menor quantidade de ligas para atender a demanda restante, que já foi armazenada no vetor de soluções da programação dinâmica.

Além disso, é certo que já teremos calculado o valor ideal dos subproblemas resultantes da subtração, pois estamos construindo todas as soluções de baixo para cima até a demanda original.

Por fim, dentre todos os resultados encontrados em cada caso de teste, basta selecionar o menor e armazená-lo no vetor de soluções para que possa ser utilizado em soluções futuras.

3. Complexidade

A solução consiste em 2 loops aninhados. O externo itera de 1 até a demanda do problema original. Já o interno percorre cada um dos tamanhos de ligas metálicas disponíveis. Assim, se temos uma demanda “d” e uma quantidade “n” de ligas metálicas, a complexidade será: O(n * d).

4. NP-Completude

4.1. Prova que está em NP

No entanto, a complexidade é diferente ao avaliarmos o tamanho da entrada em sua representação em bits. Se temos uma demanda com valor “d”, a sua representação será: $rd = \log_2(d)$. Escrevendo “rd” em função de “d”, temos: $2^{rd} = 2^{\log(d)} \rightarrow d = 2^{rd}$. Então, colocando a complexidade em função da representação em bits, temos: $O(n * 2^{rd})$, sendo “n” a quantidade de ligas metálicas disponíveis e “rd” a representação em bits de “d”.

Ou seja, o tempo de execução do algoritmo, O(n * d), é polinomial para o valor numérico de “d”. No entanto, para a sua representação em bits, em que $d = 2^{rd}$, ele é exponencial, $O(n * 2^{rd})$. Além disso, é possível notar que O(n * d) jamais será um limite superior para $O(n * 2^{rd})$, independentemente de qual constante você multiplicá-lo. Portanto, isso indica que este problema não está em P.

Por outro lado, sabemos que existe um algoritmo para validar uma possível solução para o problema. Se temos um conjunto de ligas metálicas e os seus respectivos tamanhos, podemos iterar sobre ele e ver se a soma delas é igual a demanda. Além disso, se desejamos que a quantidade de ligas seja menor ou igual a certo valor, basta compará-lo a quantidade de elementos no conjunto.

Portanto, se temos “n” ligas metálicas e o maior tamanho entre elas é “t”, a representação em bits de “t” será: $rt = \log_2(t)$. O custo para iterar sobre o conjunto de ligas e obter a soma delas será: O(n * rt). Como as comparações são constantes, o algoritmo para validar uma possível solução, tem custo polinomial a representação da entrada. Por esse motivo, temos que este problema está em NP.

4.2. Prova que está em NP-Difícil

Além disso, podemos provar que este problema está em NP-Difícil por meio da seguinte redução: 3-SAT $\leq_P$ SUBSET SUM $\leq_P$ Ligas Metálica. Nós podemos definir os três problemas como problemas de decisão da seguinte forma:

3SAT

Entrada: Uma fórmula booleana na CNF

Pergunta: A fórmula é satisfatível?

SUBSET SUM

Entrada:

- Um vetor “u”, com inteiros positivos
- Um valor inteiro positivo “w”

Pergunta: Existe um subconjunto dentre os elementos de “u” de modo que somem ao valor “w”?

Ligas Metálicas

Entrada:

- Um vetor “v” de inteiros positivos com o tamanho das ligas metálicas
- Um inteiro positivos “d”, que é a demanda do cliente
- Um inteiro positivos “k”, que é a quantidade máxima de barras utilizadas

Pergunta: Existe um subconjunto dentre os elementos de “v” de modo que somem ao valor “d” e possui no máximo “k” elementos?

4.2.1. 3SAT $\leq_P$ SUBSET SUM

Primeiro devemos fazer a prova de 3SAT $\leq_P$ SUBSET SUM. Para isso, podemos mapear cada literal e cláusula a um número decimal.

Para isso, cada número terá uma quantidade de dígitos igual a quantidade de cláusulas mais a quantidade de variáveis. Os dígitos associados a cada variável será 1 para o seu respectivo literal e a sua negação e 0 para os outros. Já os números das cláusulas será 1, caso o literal associado a ele satisfaz a respectiva cláusula, ou 0 caso contrário. Além disso, iremos adicionar valores que garantem que ao selecionar exatamente os literais necessários para satisfazer a fórmula SAT resultem na soma “w”. Um exemplo de construção seria:

$$\underset{C_1}{(X \vee Y \vee Z)} \wedge \underset{C_2}{(\neg X \vee Y \vee \neg Z)} \wedge \underset{C_3}{(X \vee \neg Y \vee \neg Z)} \wedge \underset{C_4}{(\neg X \vee \neg Y \vee Z)}$$

	X	Y	Z	C_1	C_2	C_3	C_4	Número
X	1	0	0	1	0	1	0	1001010
¬X	1	0	0	0	1	0	1	1000101
Y	0	1	0	1	1	0	0	101100
¬Y	0	1	0	0	0	1	1	100011
Z	0	0	1	1	0	0	1	11001
¬Z	0	0	1	0	1	1	0	10110
C_1	0	0	0	1	0	0	0	1000
C_1	0	0	0	3	0	0	0	3000

C_2	0	0	0	0	1	0	0	100
C_2	0	0	0	0	3	0	0	300
C_3	0	0	0	0	0	1	0	10
C_3	0	0	0	0	0	3	0	30
C_4	0	0	0	0	0	0	1	1
C_4	0	0	0	0	0	0	3	3
Valor alvo (w)	1	1	1	4	4	4	4	1114444

Neste exemplo, vemos que ao satisfazer a fórmula SAT selecionando X = 1, Y = 0 e Z = 0 implica diretamente que a soma dos valores respectivos a eles mais os valores das cláusulas satisfeitas somará o valor “w”. Ou seja, dentre os elementos do vetor “u” com os números { 1001010, 1000101, 101100, 11001, 10110, 1000, 3000, 100, 300, 10, 30, 1, 3 }, os números { 1001010, 100011, 10110, 3000, 300, 10, 3 } foram selecionados para somar o valor “w”, que é 1114444.

Por outro lado, a prova de volta também ocorre por construção da tabela, uma vez que ela associa cada literal a números que os identificam. Dessa forma, é possível saber quais literais que são utilizados para tornar a fórmula verdadeira a partir dos números do SUBSET SUM.

Por fim, é possível notar que essa transformação leva tempo polinomial, pois a tabela sempre terá “2n + 2m” linhas e “n + m” colunas, sendo “n” a quantidade de literais e “m” a quantidade de cláusulas. O tempo para construção também será polinomial de acordo com esses valores.

4.2.2. SUBSET SUM $\leq_p$ Ligas Metálicas

Ambos os problemas se assemelham muito, pois nos dois desejamos selecionar um subconjunto de valores de um vetor, em que pode haver repetição. No entanto, para as Ligas Metálicas, temos um limite “k” de ligas que podemos selecionar.

A fim de transformar o problema do SUBSET SUM em Ligas Metálicas, podemos considerar que o vetor “u” do SUBSET SUM equivale ao vetor “v”, que contém os tamanhos das ligas. Além disso, o valor “w” que desejamos atingir no primeiro problema é igual ao valor da demanda do cliente “d”. Por fim, o valor “k” pode ser definido como a quantidade de elementos do vetor “u”. Isso é uma transformação polinomial, pois são apenas atribuições.

Toda solução que havia no SUBSET SUM também é uma solução no problema das Ligas Metálicas, uma vez que estamos considerando um “k” de tamanho igual a quantidade de elementos do vetor. Por outro lado, as soluções

Assim, por transitividade, temos que o problema das Ligas Metálicas é um problema NP-Difícil. Contudo, também sabemos que é um problema em NP. Logo, ele é NP-Completo.

5. Referências

Algoritmos 1 - Problemas NP e NP-Completo - Redução polinomial (parte 6). Aulas - Jussara Almeida/DCC/UFMG. Youtube 12/10/2023. Disponível em: <https://www.youtube.com/watch?v=krLHt7vh-Zs>. Acesso em: 30/07/2023.

Computer Science Theory Explained. SubsetSum. Youtube, 25/02/2021. Disponível em: <https://www.youtube.com/watch?v=B-Tp8Twr5gw>. Acesso em: 30/07/2023.

Informal-CS. Reduction: 3-CNF SAT to Subset Sum. Youtube, 20/11/2018. Disponível em: <https://www.youtube.com/watch?v=k8RkYp5KhhU>. Acesso em: 30/07/2023.

Learn IT easy with Mehbooba. NP Completeness of Knapsack Problem. Youtube, 27/10/2020. Disponível em: <https://www.youtube.com/watch?v=XQajKPzRh2A>. Acesso em: 30/07/2023.

StackExchange. Reduction from SUBSET SUM to COIN CHANGING. Disponível em: <https://cs.stackexchange.com/questions/155565/reduction-from-subset-sum-to-coin-changing>. Acesso em: 30/07/2023.

WAYNE, Kevin. Lecture slides - INTRACTABILITY I. Slides 71 - 76. 30/04/2018. Tipo de apresentação. Disponível em: <https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/08IntractabilityI.pdf>. Acesso em: 30/07/2023.