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

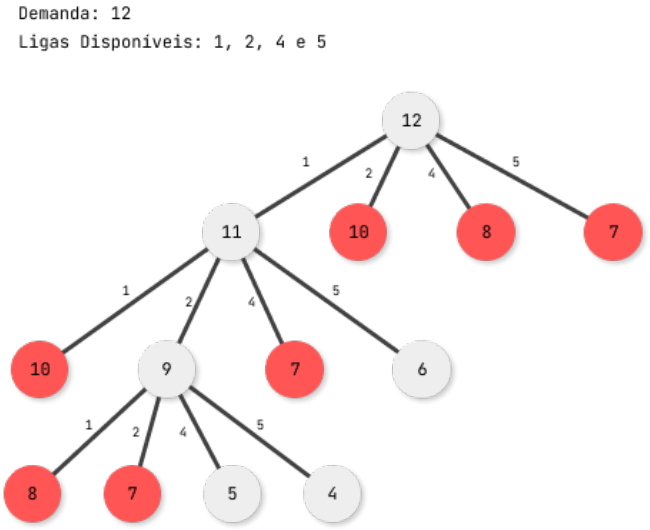
1. Introdução

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.

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:



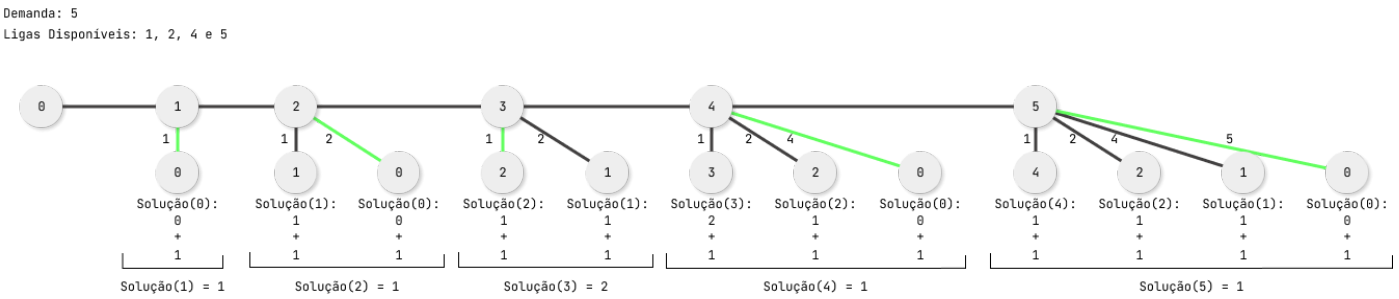
É possível notar que assim que resolvemos o subproblema 11, todos subproblemas restantes do problema original 12 já foram computados, que são os vértices marcados em vermelho. Uma vez que os subproblemas se sobrepõem, podemos utilizar Programação Dinâmica para evitar repetições de soluções.

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 sub problemas futuros.

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: SUBSET SUM $\leq_P$ Ligas Metálicas. Uma vez que o SUBSET SUM é NP-Difícil, conseguiriamos mostrar que o problema das Ligas Metálicas também será.

SUBSET SUM

Entrada:

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

Ligas Metálicas

Entrada:

- Um vetor “v” de inteiros positivos com o tamanho das ligas metálicas
- Um inteiro positivos “d”, que é a

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

demanda do cliente
- Um inteiro positivo “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?

A transformação de um problema no outro pode ser feita por meio do mapeamento de uma função que irá associar os elementos do vetor “u” às ligas metálicas. Cada elemento terá duas ligas, uma ligaGrande e uma ligaPequena.

A ligaGrande será dada pela seguinte fórmula:

$$ligaGrande_i = elemento_i * X + (i + 1)$$

Em que “i” representa a posição do elemento no vetor e “X” é o máximo entre o valor alvo que desejamos atingir “w” + 1 e a quantidade de elementos do vetor “u” + 1:

$$X = max(w + 1, u.size() + 1) + 1$$

Já a ligaPequena seria apenas a posição do elemento no vetor mais um:

$$ligaPequena_i = i + 1$$

Por exemplo, se temos um vetor “u” com elementos {1, 2, 3 e 5} e o valor “w” sendo 8 teríamos o seguinte mapeamento:

Valores vetor U	Fórmula Liga Grande	Liga Grande	Fórmula Liga Pequena	Liga Pequena
1	1 * 10 + (1 + 1)	12	1 + 1	2
2	2 * 10 + (2 + 1)	23	2 + 1	3
3	3 * 10 + (3 + 1)	34	3 + 1	4
5	5 * 10 + (4 + 1)	55	4 + 1	5

Além disso, o valor da demanda “d” também será mapeado em “w” da seguinte forma:

$$d = w * X + \sum_{i=1}^{n} i + 1$$

Sendo “n” a quantidade de elementos no vetor do SUBSET SUM. Ao mapear as ligas dessa forma, nós conseguimos garantir que um dos dois tipos serão selecionadas para compor a resposta. Também garantimos que sempre iremos selecionar uma mesma quantidade de ligas entre o SUBSET SUM e o problema das Ligas ao definir que “k” é igual a “n”.

Outra característica desse mapeamento é que quando a ligaGrande é selecionada o respectivo número associado a ela no vetor do SUBSET SUM será usado para atingir o valor “w”. Por outro lado, se a ligaPequena foi selecionada, aquele número não foi escolhido para compor a soma até “w”.

Então, para o “w” do exemplo anterior, teríamos que d = 94 e para somar esse valor, iremos selecionar as ligas {55, 34, 3, 2}. O que significa que os valores do SUBSET SUM escolhidos foram {5, 3}, mas os valores {2, 1} não foram incluídos, uma vez que as ligas pequenas deles foram selecionadas.

Por outro lado, considerando apenas as ligas {55, 34, 3, 2}, conseguimos convertê-las de volta para os valores de “u” por meio da função inversa das fórmulas definidas. Assim, os valores da ligas se tornam: 55 → 5; 34 → 3; 3 → 2; 2 → 1. Já a demanda passaria a ser: w = 8. Como sabemos que as ligas pequenas de 2 e 1 foram selecionadas, a resposta para a soma “w” não incluirá elas, mas apenas os valores 3 e 5.

Assim, é possível claramente ver por meio do mapeamento que se o SUBSET SUM tem uma solução o problema das Ligas também terá e vice-versa. Ou seja, se há uma instância “sim” para um problema, também haverá uma instância sim para o outro. Além disso, essa construção segue um tempo polinomial, pois apenas é necessário aplicar a fórmula sobre cada elemento do SUBSET SUM.

Por fim, ao converter SUBSET SUM $\leq_P$ Ligas Metálica, 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.

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.

Krsubramani Courses. Coin-Changing is NP-Hard Disponível em: <https://community.wvu.edu/~krsubramani/courses/backupcourses/CS520Fa2013/lecprep/NPCProofs/SubsetSum.pdf>. Acesso em: 02/07/2023

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.