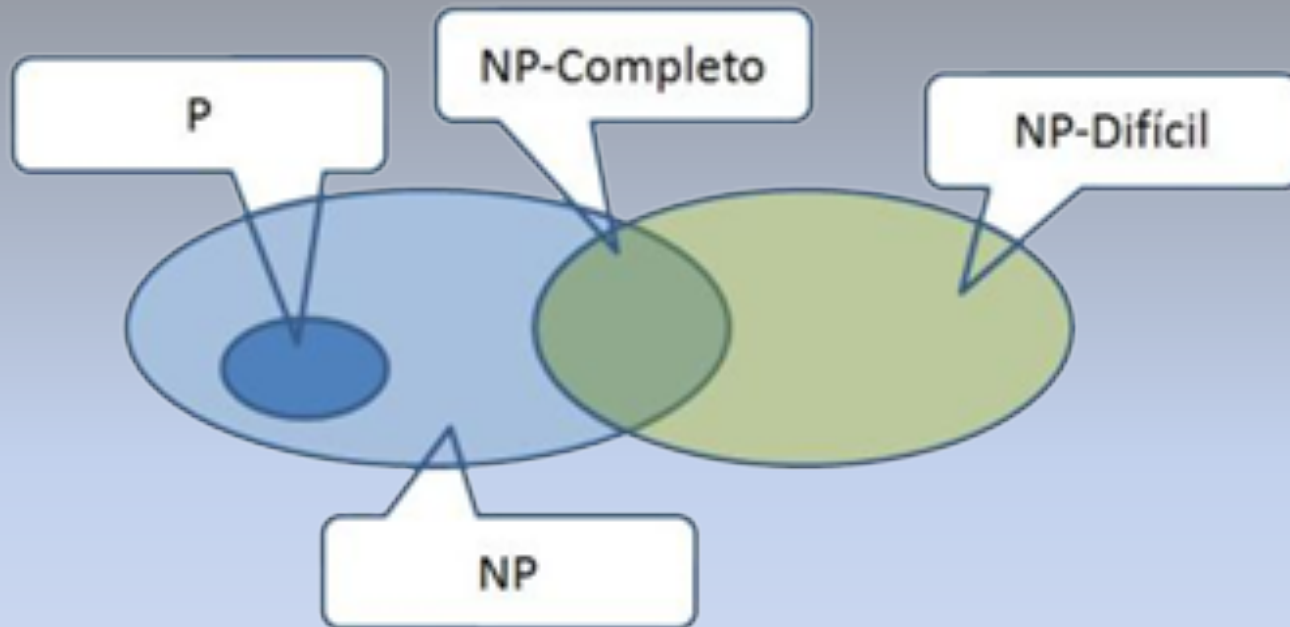


PROVA QUE O PROBLEMA DA SOMA DE SUBCONJUNTOS (SUBSET SUM) É NP-COMPLETO

COMPLEXIDADE DE ALGORITMOS

Nomes: Clemilson Luís de Brito Dias
Rafael Rebellato Trommenschläger

CLASSES DE PROBLEMAS



P – existe solução polinomial

NP – existe verificação polinomial

NP-I - existe verificação polinomial (não foi encontrado solução polinomial nem redução polinomial)

NP-Difícil – existe redução polinomial (não é garantido ter verificação polinomial)

Np-Completo – \in NP e \in NP-Difícil

PROBLEMAS DE DECISÃO x PROBLEMAS DE OTIMIZAÇÃO

Problema de otimização consiste em selecionar a melhor forma de resolver esse problema, ou seja, não basta resolvê-lo, temos que fazer isso da maneira mais eficiente.

Problema de decisão consiste em responder se existe ou não uma solução a um determinado problema.

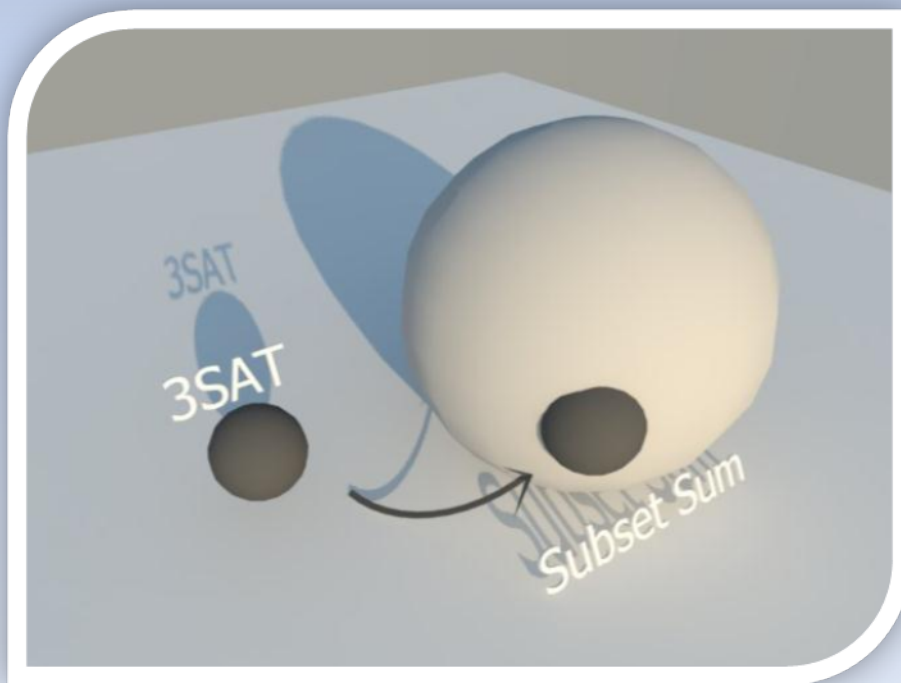
Vamos considerar os problemas de decisão para a apresentação

REDUTIBILIDADE

Redução de A em B

- B é, no mínimo, tão difícil quanto A.
- Se B é solúvel, A também será.
- A pode ser uma parte fácil de B.
- Se A não tem solução em tempo polinomial, B também não terá.
- A jamais será mais difícil que B.

3-SAT \leq_p SUBSET-SUM



SOMA DE SUBCONJUNTOS (SUBSET-SUM)

Dados números naturais p_1, \dots, p_n e t ,
questionamos se existe um subconjunto K_1 de $\{1, \dots, n\}$
tal que a soma dos k elementos de K_1 seja igual a t .

Exemplo:

Seja “K” o conjunto $\{10, 20, 46, 62, 70, 90\}$ e
o número alvo “t” 226.

Consideremos o subconjunto $K_1 = \{20, 46, 70, 90\}$
A soma dos valores de K_1 é igual a 226.

SOMA DE SUBCONJUNTOS (SUBSET-SUM)

Dados números naturais p_1, \dots, p_n e t ,
questionamos se existe um subconjunto K_1 de $\{1, \dots, n\}$
tal que a soma dos k elementos de K_1 seja igual a t .

Exemplo:

Seja “K” o conjunto $\{10, 20, 46, 62, 70, 90\}$ e
o número alvo “t” 226.

Consideremos o subconjunto $K_1 = \{20, 46, 70, 90\}$
A soma dos valores de K_1 é igual a 226.

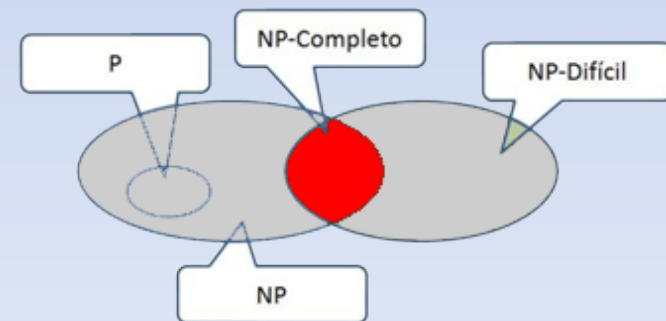
Então, para provar que o problema da Soma de Subconjuntos (Subset Sum) é NP-Completo, temos que provar que ele é:

- NP

- existe verificação polinomial

- NP-Difícil

- existe redução polinomial



PROVA NP

Verificação

ALGORITMO DE VERIFICAÇÃO

```
• Function Verifica_SubsetSum (K, K1, t) :  
• int soma = 0;  
• for each k in K1 :  
•     soma = soma + k;  
• if (soma != t)  
•     return FALSE  
• else  
• int array[sizeof(K1)];  
• array <= Fill with zeros;  
• for i = 0 to sizeof(K1)-1:  
•     for each n in K:  
•         if( K1[i] == n)  
•             array[i] = array[i] + 1;  
• for i = 0 to sizeof(K1) - 1:  
•     if( array[i] != 1)  
•         return FALSE  
• return TRUE  
• End Function Verifica_SubsetSum
```

Sendo:

K = conjunto com original

n = elemento em K

K1 = subconjunto de K com
entrada válida (certificado)

k = elemento em K1

t = valor alvo

ALGORITMO DE VERIFICAÇÃO

```

Function Verifica_SubsetSum(K, K1, t) :
•int soma = 0;
•for each k in K1 :
•    soma = soma + k;
•if (soma != t)
•    return FALSE
•else
•    int array[sizeof(K1)];
•    array <= Fill with zeros;
•    for i = 0 to sizeof(K1):-1
•        for each n in K:
•            if( K1[i] == n)
•                array[i]= array[i]+1;
•    for i = 0 to sizeof(K1) :-1
•        if( array[i] != 1)
•            return FALSE
•    return TRUE
End Function Verifica_SubsetSum

```

Sendo:

K = conjunto com original

n = elemento em K

K1 = subconjunto de K com entrada válida (certificado)

k = elemento em K1

t = valor alvo

Soma = operação elementar

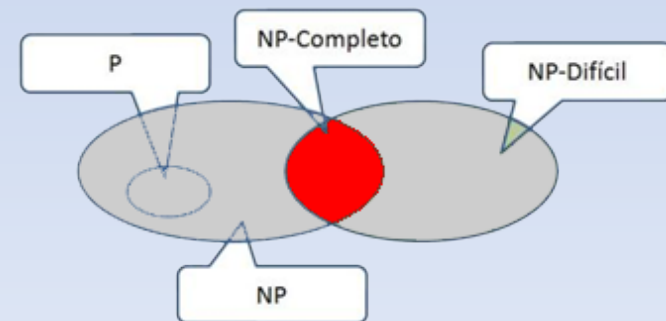
Iteração da linha 10 e 11 executará $k * n$ vezes

Outras complexidades = não influenciam, pois $k * n$ é a complexidade dominante.

Código acima possui complexidade **$O(kn)$** , ou seja, **polinomial**

Então, para provar que o problema da Soma de Subconjuntos (Subset Sum) é NP-Completo, temos que provar que ele é:

- **NP**
 - **existe verificação polinomial**
- NP-Difícil
 - existe redução polinomial



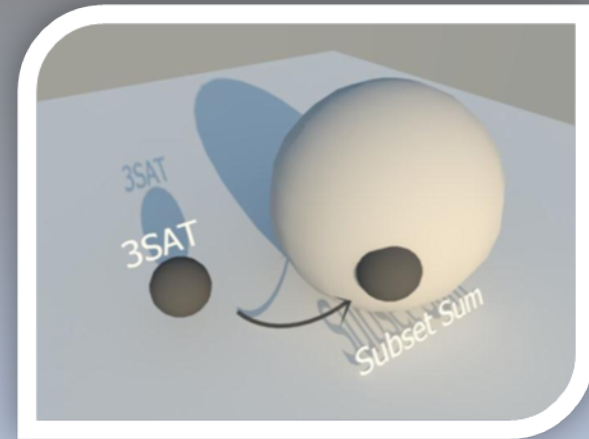
PROVA

NP-Completo

Redução

3-SAT

O problema 3-SAT consiste em verificar a satisfabilidade de fórmulas booleanas com três literais em cada cláusula



3-SAT \leq_p SUBSET-SUM

Exemplo:

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Redução

Exemplo para melhor entendimento:

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Literais:

Cláusulas:

Redução

Exemplo para melhor entendimento:

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$ ← negações

Cláusulas:

Redução

Exemplo para melhor entendimento:

$$\begin{array}{cccc} \text{C1} & & \text{C2} & & \text{C3} & & \text{C4} \\ \hline (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \end{array}$$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Primeiro verificar se:

- Cada cláusula possui 3 literais (característica básica de 3-SAT)**
- Nenhuma cláusula da fórmula booleana pode conter, ao mesmo tempo, um literal e sua negação (não faria sentido).**
- Cada literal deve aparecer em, pelo menos, uma cláusula.**

$$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

Segundo, gostaríamos que a soma de suas linhas selecionadas na tabela resultasse em um valor no padrão 111*4444*.

Onde a quantidade de 1's é igual a quantidade de literais.

Cada cláusula pode ser validada por 1, 2 ou 3 variáveis.

Como só temos um número alvo para a soma, precisamos de um valor fixo maior que 3 para completar os demais dígitos, sem estender muito o conjunto de valores K.

Logo, selecionamos o 4, primeiro maior que 3. Para a soma de uma coluna de cláusula chegar a 4, precisamos somar 3, 2 ou 1 ao resultado.

Criando TabelaLiterais: x_1, x_2, x_3 $\neg x_1, \neg x_2, \neg x_3$ Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$ C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$ C4: $(x_1 \vee x_2 \vee x_3)$

Ao criar uma tabela, devemos calcular seu tamanho antes.

Para isso, devemos fazer o seguinte cálculo:

Largura : $C + X$

Altura : $2 \cdot C + 2 \cdot X$

Sendo “C” o número de cláusulas e “X” o número de literais

$$4 + 3$$

$$7$$

$$2 \cdot 4 + 2 \cdot 3$$

$$8 + 6$$

$$14$$

Criando Tabela

Literais: x_1, x_2, x_3 $\neg x_1, \neg x_2, \neg x_3$ Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$ C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$ C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	0	0	0	0	0	0	0
v1' =	0	0	0	0	0	0	0
v2 =	0	0	0	0	0	0	0
v2' =	0	0	0	0	0	0	0
v3 =	0	0	0	0	0	0	0
v3' =	0	0	0	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$


C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

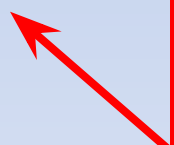
X são os literais e C as cláusulas

V são os valores e S os coringas

Os coringas são utilizados para que a redução ocorra da forma certa (a soma dar 4).



	x1	x2	x3	c1	c2	c3	c4
v1 =	0	0	0	0	0	0	0
v1' =	0	0	0	0	0	0	0
v2 =	0	0	0	0	0	0	0
v2' =	0	0	0	0	0	0	0
v3 =	0	0	0	0	0	0	0
v3' =	0	0	0	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0



Criando Tabela

Literais: x_1, x_2, x_3
 $\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$
C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$
C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$
C4: $(x_1 \vee x_2 \vee x_3)$

Preencher os literais na tabela com os valores 1

	x1	x2	x3	c1	c2	c3	c4
v1 =	0	0	0	0	0	0	0
v1' =	0	0	0	0	0	0	0
v2 =	0	0	0	0	0	0	0
v2' =	0	0	0	0	0	0	0
v3 =	0	0	0	0	0	0	0
v3' =	0	0	0	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3
 $\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$
 C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$
 C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$
 C4: $(x_1 \vee x_2 \vee x_3)$

Preencher os literais na tabela com os valores 1

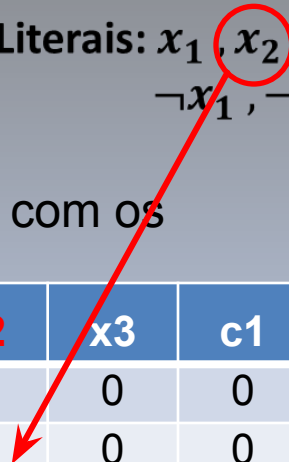
	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	0	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	0	0	0	0	0	0
v2' =	0	0	0	0	0	0	0
v3 =	0	0	0	0	0	0	0
v3' =	0	0	0	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3
 $\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$
 C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$
 C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$
 C4: $(x_1 \vee x_2 \vee x_3)$

Preencher os literais na tabela com os valores 1



	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	0	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	0	0	0	0
v3 =	0	0	0	0	0	0	0
v3' =	0	0	0	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3
 $\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$
 C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$
 C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$
 C4: $(x_1 \vee x_2 \vee x_3)$

Preencher os literais na tabela com os valores 1

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	0	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	0	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Preencher as cláusulas na tabela com os valores 1

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	0	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	0	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	0	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$**

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Preencher as cláusulas na tabela com os valores 1

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	1	0	0	0
s1 =	0	0	0	0	0	0	0
s1' =	0	0	0	0	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$**

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Preencher o coringa respectivo a sua cláusulas

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	1	0	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$**

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Preencher o coringa respectivo a sua cláusulas

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	1	0	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	0	0	0
s2' =	0	0	0	0	0	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Pq 1 e

2??

Criando Tabela

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$**

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Preencher o coringa respectivo a sua cláusulas

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	0	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	0	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	1	0	0	0
s1 =	0	0	0	1			
s1' =	0	0	0	2			

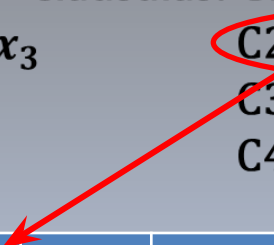
Pq 1 e

Pois desejamos que as cláusulas válidas resultem em 4. Se nenhum literal da cláusula for válido, a soma máxima resulta em 3.

Criando Tabela

Literais: x_1, x_2, x_3 $\neg x_1, \neg x_2, \neg x_3$ Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$ C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$ C4: $(x_1 \vee x_2 \vee x_3)$

Continuando a preencher as cláusulas e coringas




	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	1	0	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	1	0	0
v3 =	0	0	1	0	0	0	0
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	0	0
s3' =	0	0	0	0	0	0	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3 $\neg x_1, \neg x_2, \neg x_3$ Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$ C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$ C4: $(x_1 \vee x_2 \vee x_3)$

Continuando a preencher as cláusulas e coringas



	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	0
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	0
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	0
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	0
s4' =	0	0	0	0	0	0	0

Criando Tabela

Literais: x_1, x_2, x_3 $\neg x_1, \neg x_2, \neg x_3$ Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$ C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$ C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$ C4: $(x_1 \vee x_2 \vee x_3)$

Continuando a preencher as cláusulas e coringas

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Criando Tabela

Literais: x_1, x_2, x_3
 $\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$
 C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$
 C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$
 C4: $(x_1 \vee x_2 \vee x_3)$

Tabela Completa!!!

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Exemplos

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Para testar se a tabela realmente funciona, testaremos 2 exemplos:

Exemplo 1 (válido - certificado):

Entradas: $x_1 = 0, x_2 = 0$ e $x_3 = 1$

Exemplo 2 (inválido):

Entradas: $x_1 = 0, x_2 = 0$ e $x_3 = 0$

Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v2 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Inicialmente, selecionamos as linhas

Indicadas pelas variáveis

Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	0	1
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2
t =	1	1	1	1	2	3	1

Essas linhas resultariam na seguinte

soma



Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0

Mas desejamos a soma no padrão

111*4444*

s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2
t =	1	1	1	1	2	3	1

Exemplo

S

Exemplo 1 (válido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 1$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: C1: $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

Pronto!
Temos
uma
respost
a
válida!

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2
t =	1	1	1	4	4	4	4

Exemplo

S

Exemplo 2 (inválido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 0$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2

Exemplo

S

Exemplo 2 (inválido):

Entradas: $x_1 = 0$, $x_2 = 0$ e $x_3 = 0$

Literais: x_1, x_2, x_3

$\neg x_1, \neg x_2, \neg x_3$

Cláusulas: **C1:** $(x_1 \vee \neg x_2 \vee \neg x_3)$

C2: $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

C3: $(\neg x_1 \vee \neg x_2 \vee x_3)$

C4: $(x_1 \vee x_2 \vee x_3)$

	x1	x2	x3	c1	c2	c3	c4
v1 =	1	0	0	1	0	0	1
v1' =	1	0	0	0	1	1	0
v2 =	0	1	0	0	0	0	1
v2' =	0	1	0	1	1	1	0
v3 =	0	0	1	0	0	1	1
v3' =	0	0	1	1	1	0	0
s1 =	0	0	0	1	0	0	0
s1' =	0	0	0	2	0	0	0
s2 =	0	0	0	0	1	0	0
s2' =	0	0	0	0	2	0	0
s3 =	0	0	0	0	0	1	0
s3' =	0	0	0	0	0	2	0
s4 =	0	0	0	0	0	0	1
s4' =	0	0	0	0	0	0	2
t =	1	1	1	4	4	4	3

**Soma das
linhas
resulta
em um “t”
inválido!**

**Cláusula
4 possui
a soma
máxima
de 3!**

ALGORITMO DE REDUÇÃO

- Function Reducao_3SATtoSubsetSum (Equation)
 - if(\neq verify_clauses(Equation.clauses);)
 - return ERROR
 - else
 - M = create_matrix(sizeof(Equation.clauses), sizeof(Equation.Literals));
 - for each different lit in Equation.literals
 - create_lits_T_F_column(M,lit);
 - for each clause in Equation.clauses
 - create_var_clause_column(M,clause));
 - if(everything_is_ok(M,K,t))
 - return OK
 - else
 - return ERROR
- End Function Reducao_3SATtoSubsetSum

ALGORITMO DE REDUÇÃO

Algoritmos auxiliares:

Function **verify_clauses**(clauses)

```

for each clause in clauses
    if (sizeof(clause.literals)) != 3)           //verifica se tem 3 literais em cada cláusula
        return FALSE
    elseif (clause.literals(1)== clause.literals(2) ||
            clause.literals(2)== clause.literals(3) ||
            clause.literals(1)== clause.literals(3) ) // verifica se todos são diferentes
        return FALSE
    elseif (clause.literals(1)==not clause.literals(1) || // verifica se não tem um literal
            clause.literals(2)==not clause.literals(2) || // e seu negado na mesma
            clause.literals(3)==not clause.literals(3) ) // cláusula
        return FALSE
    else
        return TRUE

```

Function **create_matrix**(c,x);

```

int matriz [sizeof(c + x)][sizeof(2*c + 2*x)]; //cria matriz de tamanho c+x por 2*c+2*x
for (int i = 0; i < sizeof(c + x); i++) // zera todos elementos da matriz
    for (int j = 0; j < sizeof(2*c + 2*x); j++)
        matriz[i][j] = 0;
return matriz;

```

ALGORITMO DE REDUÇÃO

Algoritmos auxiliares:

Function create_lits_T_F_column(M,lit)

*//adiciona na matriz em seu respectivo lugar o valor 1 em seu literal
//e 1 na sua negação para seus "v" e "v'" respectivos
//ex.: (lit1=1 em i=0 e j=0,1
// lit2=1 em i=1 e j=2,3
// lit3=1 em i=2 e j=4,5
// etc...)*

Function create_var_clause_column(M,clause)

*//para cada cláusula, adiciona na matriz em seu respectivo
// lugar o valor 1 nos lugares indicados pela
//cláusula criar também os coringas, com os valores 1 e 2 para os seus "s" e "s'"
respectivos*

Function everything_is_ok(K)

*//concatena cada linha da matriz e gera um número em K.
// A concatenação da última linha gera t, logo, não é
//adicionada a K. Se t não estiver no padrão 111*444* retorna FALSE, senão TRUE*

ALGORITMO DE REDUÇÃO

Complexidad

e

Analisando a **complexidade dos algoritmos auxiliares**, resumidamente temos:

verify_clauses: $O(c)$, visto que percorre o array de cláusulas 1 vez.

create_matrix: $O((2c + 2x)(c+x))$, já que é necessário zerar a matriz, percorrendo-a.

create_lits_T_F_column: $O(2c+2x)x$, visto que apenas preenche a coluna da matriz.

create_var_clause_column: $O(2c+2x)c$, visto que apenas preenche a coluna da matriz.

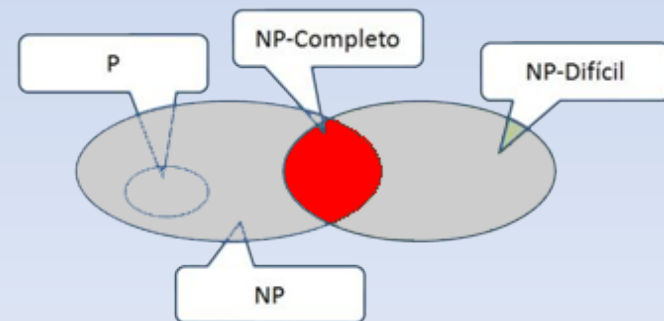
everything_is_ok: $O((2c + 2x)(c+x))$, já que é necessário percorrer a matriz.

Fazendo a soma das complexidades do algoritmo de redução com seus algoritmos auxiliares, temos a complexidade $O(6c^2 + 6x^2 + 12cx + c)$.

Nesse caso, conforme já dito anteriormente, x indica o número de literais e c o número de cláusulas da equação booleana. Com isso, a complexidade final do algoritmo de redução é $O(c^2 + x^2)$.

Então, para provar que o problema da Soma de Subconjuntos (Subset Sum) é NP-Completo, temos que provar que ele é:

- **NP**
 - existe verificação polinomial
- **NP-Difícil**
 - existe redução polinomial



FIM