Complexidade de Algoritmos

3-SAT

Joana Campos, Matheus Costa e Marcos Reckers

Agosto | 2024

AGENDA

1 O QUE É O 3-SAT

2 3-SAT É NP?

3 3-SAT É NP-DIFÍCIL?

1 O QUE É O 3-SAT

CARACTERÍSTICA DO PROBLEMA

O problema **3-SAT** é um problema de **Decisão** em que, dada uma fórmula booleana na *forma normal conjuntiva* (CNF), com cláusulas contendo exatamente três literais cada, o objetivo é determinar se existe uma atribuição de valores verdadeiro ou falso às variáveis que torne pelo menos um literal verdadeiro em cada cláusula.

ENTRADA

Uma fórmula booleana 3-CNF

SAÍDA

Booleano

Determina se a fórmula é satisfazível.

EXEMPLOS DE INSTÂNCIAS DO 3-SAT

$$(a \vee \overline{b} \vee c) \wedge (c \vee d \vee \overline{e})$$

$$(a \vee \overline{b} \vee c) \wedge (a \vee \overline{b} \vee \overline{c}) \wedge (\overline{a} \vee b \vee c)$$

2 3-SAT É NP?

COMO VERIFICAR SE 3-SAT É NP?

Para isso, é necessário desenvolver um algoritmo de verificação que, dada uma instância do problema e um certificado, verifique, em tempo polinomial, se o certificado é de fato uma solução para essa instância do problema.

ALGORITMO DE VERIFICAÇÃO

Formato da entrada: O algoritmo receberá uma instância válida do problema e seu certificado.

- A instância será representada por uma matriz bidimensional (instancia[clausula][literal]), cujos elementos pertencem ao conjunto dos números inteiros ($\mathbb{Z}$). Valores positivos indicam variáveis em sua forma original, enquanto valores negativos representam variáveis negadas.

ALGORITMO DE VERIFICAÇÃO

- O certificado é um array de valores booleanos, onde cada posição corresponde ao valor da respectiva variável.

```
instancia = [  [1, -2, 3]  ,  [-1, 2, -3]  ,  [2, -3, 4]  ]
certificado = [ True , False , True , False ]
resultado = Verifica_3SAT(instancia, certificado)
```

ALGORITMO DE VERIFICAÇÃO

```
Função Verifica_3SAT(instancia, certificado):
    Para cada cláusula na instancia:
        cláusula_satisfeita = Falso

        Para cada literal na cláusula:
            var = abs(literal)
            valor = elemento do certificado na posição var

            Se (literal = positivo e valor = Verdadeiro) ou (literal = negativo e valor = Falso):
                Definir cláusula_satisfeita como Verdadeiro
                Sair do laço (break)

        Se cláusula_satisfeita = Falso:
            Retornar Falso  # Se uma cláusula não for satisfeita, o certificado não é válido

    Retornar Verdadeiro  # Todas as cláusulas foram satisfeitas pelo certificado
```

ALGORITMO DE VERIFICAÇÃO

```
Função Verifica_3SAT(instancia, certificado):
    Para cada cláusula na instancia:
        cláusula_satisfeita = Falso

        Para cada literal na cláusula:
            var = abs(literal)
            valor = elemento do certificado na posição var

            Se (literal = positivo e valor = Verdadeiro) ou (literal = negativo e valor = falso):
                Definir cláusula_satisfeita como Verdadeiro
                Sair do laço (break)

        Se cláusula_satisfeita = Falso:
            Retornar Falso  # Se uma cláusula não for satisfeita, o certificado não é válido

    Retornar Verdadeiro  # Todas as cláusulas foram satisfeitas pelo certificado
```

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{literal=1}^{3} (1) \right)$$

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{literal=1}^{3} (1) \right)$$

$$\sum_{clausula=1}^{n_{clausulas}} (3)$$

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{literal=1}^{3} (1) \right)$$

$$\sum_{clausula=1}^{n_{clausulas}} (3)$$

$$3 \; * n_{clausulas}$$

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{literal=1}^{3} (1) \right)$$

$$\sum_{clausula=1}^{n_{clausulas}} (3)$$

$$3 * n_{clausulas}$$

$$O(n_{clausulas})$$

3 3-SAT É NP-DIFÍCIL?

COMO SABER SE 3-SAT PERTENCE À CLASSE NP-DIFÍCIL?

Para isso, é necessário fazer uma redução de algum outro problema já comprovado NP-Difícil para o 3-SAT. Nossa prova será feita a partir do problema SAT, um problema sabidamente NP-Difícil.

O QUE É O PROBLEMA SAT?

Dada uma fórmula booleana na forma normal conjuntiva (CNF), o objetivo é determinar se existe uma atribuição de valores verdadeiro ou falso às variáveis que torne pelo menos um literal verdadeiro em cada cláusula, podendo haver um ou mais literais em cada cláusula. Note que é um problema extremamente parecido com o 3-SAT, apenas sem a restrição de cada cláusula conter exatamente 3 literais.

EXEMPLOS DO PROBLEMA SAT

$$(a \lor b \lor c) \land (\overline{a}) \land (\overline{f} \lor e \lor g \lor \overline{d})$$

$$(\overline{a} \lor b) \land (\overline{a} \lor c) \land (e \lor f \lor g \lor \overline{h}) \land (a)$$

$$(a) \land (a \lor \overline{b} \lor \overline{c}) \land (\overline{a} \lor b)$$

REDUÇÃO DO SAT PARA O 3-SAT

É necessário um algoritmo que transforma qualquer instância do problema SAT a uma instância do problema 3-SAT, de modo que a satisfazibilidade mantenha-se preservada, ou seja, se a instância do SAT é insatisfazível, a 3-CNF gerada a partir dele também deverá ser insatisfazível, assim como, se a instância do SAT for satisfazível, a 3-CNF gerada também deverá ser satisfazível.

EXEMPLOS

COMO FAZER ESSA REDUÇÃO?

1. Dada a fórmula booleana em CNF com qualquer número de literais por cláusula, faça:

2. Crie uma fórmula nova vazia: *3SAT_form*

COMO FAZER ESSA REDUÇÃO?

3 Se a cláusula <= 3:

a se cláusula = 1:

$C' = [(Z_1 \vee Y_1 \vee Y_2) \wedge (Z_1 \vee \overline{Y_1} \vee Y_2) \wedge (Z_1 \vee Y_1 \vee \overline{Y_2}) \wedge (Z_1 \vee \overline{Y_1} \vee \overline{Y_2})]$

Adiciona C' à *3SAT_form*

b se cláusula = 2:

$C' = [(Z_1 \vee Z_2 \vee Y_3) \wedge (Z_1 \vee Z_2 \vee \overline{Y_3})]$

Adiciona C' à *3SAT_form*

c se cláusula = 3:

Adiciona C' à *3SAT_form*

TABELAS VERDADE

Cláusula de tamanho = 1

<table>
<tr><th>Z1</th><th>Y1</th><th>Y2</th><th>(Z1 v Y1 v Y2)</th><th>AND</th></tr>
<tr><td rowspan="4">F</td><td>F</td><td>F</td><td>F</td><td rowspan="4">F</td></tr>
<tr><td>F</td><td>V</td><td>V</td></tr>
<tr><td>V</td><td>F</td><td>V</td></tr>
<tr><td>V</td><td>V</td><td>V</td></tr>
<tr><td rowspan="4">V</td><td>F</td><td>F</td><td>V</td><td rowspan="4">V</td></tr>
<tr><td>F</td><td>V</td><td>V</td></tr>
<tr><td>V</td><td>F</td><td>V</td></tr>
<tr><td>V</td><td>V</td><td>V</td></tr>
</table>

Cláusula de tamanho = 2

<table>
<tr><th>Z1 v Z2</th><th>Y3</th><th>(Z1 v Z2 v Y3)</th><th>AND</th></tr>
<tr><td rowspan="2">V</td><td>F</td><td>V</td><td rowspan="2">V</td></tr>
<tr><td>V</td><td>V</td></tr>
<tr><td rowspan="2">F</td><td>V</td><td>V</td><td rowspan="2">F</td></tr>
<tr><td>F</td><td>F</td></tr>
</table>

COMO FAZER ESSA REDUÇÃO?

4 Se não, para cada cláusula C' = $(Z_1 \vee Z_2 \vee Z_3 \ldots \vee Z_k)$ da fórmula original

a Crie novas variáveis:

$Y_1 \vee Y_2 \vee Y_3, \ldots, \vee Y_{k-3}$

b Defina:

C' = $[(Z_1 \vee Z_2 \vee Y_1) \wedge (\overline{Y_1} \vee Z_3 \vee Y_2) \wedge (\overline{Y_2} \vee Z_4 \vee Y_3) \wedge \ldots \wedge (\overline{Y_{k-3}} \vee Z_{k-1} \vee Z_k)]$

c Adiciona C' à 3SAT_form

EXEMPLOS

$$(a) \wedge (a \vee b) \wedge (a \vee \overline{b} \vee \overline{c}) \wedge (a \vee \overline{b} \vee c \vee d \vee e)$$

Função de Redução

$$(a) \rightarrow (a \vee x_1 \vee x_2) \wedge (a \vee \overline{x_1} \vee x_2) \wedge (a \vee x_1 \vee \overline{x_2}) \wedge (a \vee \overline{x_1} \vee \overline{x_2}) \wedge$$

$$(a \vee b) \rightarrow (a \vee b \vee x_3) \wedge (a \vee b \vee \overline{x_3}) \wedge$$

$$(a \vee \overline{b} \vee \overline{c}) \rightarrow (a \vee \overline{b} \vee \overline{c}) \wedge$$

$$(a \vee \overline{b} \vee c \vee d \vee e) \rightarrow (a \vee \overline{b} \vee x_4) \wedge (\overline{x_4} \vee c \vee x_5) \wedge (\overline{x_5} \vee d \vee e)$$

COMO FUNCIONA?

a

Se $(Z_1 \vee Z_2 \vee ... \vee Z_k)$ for Falsa, então sabemos que cada Z_k é Falsa. É fácil perceber, então, que em Z' o valor de Y_1 deve ser verdadeiro (a partir da primeira cláusula), e consequentemente Y_2 deve ser verdadeiro (2ª cláusula), e assim por diante... até percebermos que Y_{k-3} deve ser verdadeiro, mas então a última cláusula precisa ser falsa. Portanto, simplesmente não somos capazes de satisfazer Z'.

$$Z' = [(Z_1 \vee Z_2 \vee Y_1) \wedge (\overline{Y_1} \vee Z_3 \vee Y_2) \wedge (\overline{Y_2} \vee Z_4 \vee Y_3) \wedge ... \wedge (\overline{Y_{k-3}} \vee Z_{k-1} \vee Z_k)]$$

COMO FUNCIONA?

b Se $(Z_1 \vee Z_2 \vee \ldots \vee Z_k)$ for Verdadeira, então algum Z_i é verdadeiro. Nesse caso, atribuímos os valores verdadeiros a $Y_1, \ldots, Y_{i-2}$ e os demais Y's recebem valores falsos. Uma verificação rápida mostra que satisfazemos Z'.

$$Z' = [(Z_1 \vee Z_2 \vee Y_1) \wedge (\overline{Y_1} \vee Z_3 \vee Y_2) \wedge (\overline{Y_2} \vee Z_4 \vee Y_3) \wedge \ldots \wedge (\overline{Y_{k-3}} \vee Z_{k-1} \vee Z_k)]$$

ALGORITMO DE REDUÇÃO

Função Reduz_3SAT(*instancia_SAT*):

```
1.    formula_3sat = [ ]        #Cria uma lista vázia

2.    Para cada cláusula na instancia_SAT:
3.        SE comprimento(cláusula)  <= 3
4.            SE comprimento(cláusula) = 1
5.                clausula'=  [ (z1,' Y1 , Y2),
                               (z1, -Y1', Y2),
                               (z1, Y1 , -Y2'),
                               (z1, -Y1 , -Y2) ]
6.                formula_3sat.adiciona(clausula')

7.                SE comprimento(cláusula) = 2
8.                    clausula'= [ (z1, z2 , Y3),
9.                                 (z1, z2, -Y3)]
10.                   formula_3sat.adiciona(clausula')
```

ALGORITMO DE REDUÇÃO

```
            SE comprimento(cláusula) = 3
                formula_3sat.adiciona(clausula)

    SENÃO:      #comprimento(cláusula) > 3
        variaveis_ligação[ comprimento(cláusula) - 2 ]
        clausula'= [ ( z1, z2, variaveis_ligação[0] ) ]

        Para cada literal:2 em comprimento(cláusula) - 2
            clausula'.adiciona( (-variaveis_ligação[i-1] , z_{i+2}, variaveis_ligação[i])

        clausula'.adiciona( (-variaveis_ligação[comprimento(cláusula) - 2] , z_{k-1},z_k])
        formula_3sat.adiciona(clausula')

Retorna formula_3sat
```

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{i=2}^{comp.\_clausula-2} (1) \right) + 2$$

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{i=2}^{comp.\_clausula-2} (1) \right) + 2$$

$$\sum_{clausula=1}^{n_{clausulas}} (comp.\_clausula - 2)$$

CÁLCULO DE COMPLEXIDADE

$$\sum_{clausula=1}^{n_{clausulas}} \left(\sum_{i=2}^{comp.\_clausula-2} (1) \right) + 2$$

$$\sum_{clausula=1}^{n_{clausulas}} (comp.\_clausula - 2)$$

$$O(n_{clausulas} * maior\_clausula)$$

CONCLUSÃO

Nesse sentido, como provamos que o 3-SAT pertence a **NP** e fizemos uma redução do problema SAT, **NP-Difícil** para o 3-SAT, podemos afirmar que 3-SAT é um problema **NP-Completo**.