* Satisfatibilidade de Fórmulas (SAT) * (Cormen, 2ª ed., §34.4)

1. DEFINIÇÃO: o problema Satisfatibilidade de Fórmulas (SAT) é o de, dada uma fórmula booleana "F" composta por variáveis booleanas e operadores booleanos quaisquer de uma saída e uma ou duas entradas, além de parênteses, responder se "F" é ou não satisfatível.

2. EXEMPLOS:

   (Observação: abaixo, os conectivos lógicos comuns estão, por simplicidade, escritos por extenso.)

   a) "x1 e (não x1)": solução: "não" (insatisfatível).

   b) "(x1 ou (x2 e x3)) e ((x2 e x3) ou x4)": solução: "sim" (satisfatível), por exemplo atribuindo-se 1 a todas as variáveis envolvidas.

   c) "x0 § ((@ x0) -> x1)", onde "->" é a implicação comum e "@" e "§" estão denotando as funções

A	@ A
0	0
1	0

A	B	A § B
0	0	0
0	1	1
1	0	0
1	1	1

      A solução da fórmula em questão é "sim": qualquer atribuição a faz assumir o valor "1".

3. LEMA: SAT ∈ NP.

==========
ESBOÇO DE PROVA:

De forma semelhante ao caso de SAT-CIRCUITOS, um algoritmo de verificação para SAT pode receber como certificado uma atribuição de valores às variáveis booleanas da fórmula de entrada. A verificação pode então acontecer substituindo-se na fórmula os valores das variáveis, e então repetidamente calculando-se os valores das subfórmulas cujos operandos já estejam avaliados (isto é, cujos operandos sejam "0" ou "1"). Claramente, se a fórmula de entrada possuir "m" conectivos, então, após no máximo "m" "passagens" pelo texto da fórmula, ela estará avaliada. Como cada passagem pela fórmula tem custo linear sobre o tamanho dela (pois a avaliação de um operador booleano unário ou binário qualquer leva tempo constante), então a avaliação da fórmula pode ser feita em tempo $O(|x|^2)$, sendo "x" a sequência binária que codifica a fórmula. Logo, SAT pode ser verificada em tempo polinomial, e portanto SAT ∈ NP.
==========

4. EXERCÍCIO: reescreva o esboço de prova acima em maior nível de detalhe, como feito nas notas de aula anteriores, sobre a classe NP e o problema SAT-CIRCUITOS.

5. NOTAÇÃO: Se "O" é um objeto do nosso modelo de computação, então "<O>" denota a sequência binária que o codifica, com relação ao esquema de codificação definido pelo contexto.

6. TEOREMA: SAT é NP-difícil.

==========
PROVA:

Nós mostraremos que SAT-CIRCUITOS <=_P SAT; como SAT-CIRCUITOS é NP-difícil, seguirá então que SAT também o é, como desejado.

Considere então o seguinte algoritmo:

```
================================================================================
Algoritmo: SATC_SAT
Entrada: xc ∈ {0,1}*.
Saída: xf ∈ {0,1}*.
--------------------------------------------------------------------------------
1. Compute o circuito "C" de "ne" entradas e "np" portas lógicas representado
   por "xc", ou retorne "" (a sequência binária vazia) caso "xc" não represente
   um circuito.

2. Crie "ne" variáveis booleanas x_1 ... x_ne, correspondentes aos "ne" fios de
   entrada de "C", e mais "np" variáveis y_1 ... y_np, que corresponderão às
   saídas das portas lógicas de "C".

3. Se np = 0, retorne <"x_1">.

4. Para "i" de 1 a "np", crie uma fórmula f_i que represente o funcionamento da
   i-ésima porta lógica de "C". Exemplos:

   a) Se a porta "i" é um "E" e recebe como entrada os fios de entrada de
      números "a" e "b", então

        f_i = "y_i <-> (x_a e x_b)".

   b) Se a porta "i" é um "OU" e recebe como entrada o fio de entrada número "a"
      e a saída da porta "j", então

        f_i = "y_i <-> (x_a ou y_j)".

   c) Se a porta "i" é um "NÃO" da saída da porta "j", então

        f_i = "y_i <-> não y_j".

5. Retorne <"y_np e (f_1) e (f_2) e ... (f_np)">, sendo "f_np" a fórmula
   correspondente à porta lógica que dá a saída de "C".
================================================================================
```

Agora, observe primeiramente que SATC_SAT é um algoritmo polinomial:

a) A linha 1, na qual "xc" é decodificada, toma tempo polinomial sobre |xc|.

b) Como a criação de cada fórmula "f_i" leva tempo constante, então as linhas 2 a 4 executam em tempo O(|xc|). Em particular, observe que a linha 4 pode ser implementada por meio de apenas uma leitura do circuito "C".

c) Na linha 5, a construção da fórmula F = "y_np e (f_1) e ... (f_np)" também leva tempo O(|xc|), e o cálculo da sequência binária <F> leva tempo polinomial sobre o tamanho de "F", tamanho esse que é linear sobre o tamanho de "C", e portanto também linear sobre |xc|.

Observe agora que SATC_SAT reduz SAT-CIRCUITOS a SAT. De fato, sendo xf = SATC_SAT(xc), temos:

a) Se "xc" não representa um circuito (linha 1), então o algoritmo retorna a sequência vazia (xf = ""), que também não é a representação válida de nenhuma fórmula. Logo, xc ∉ SAT-CIRCUITOS e xf ∉ SAT.

b) Se "xc" representa o circuito que não possui portas lógicas, e que portanto consiste apenas em um fio de entrada e saída (linha 3), então o algoritmo retorna a (sequência que codifica a) fórmula "x1". Nesse caso, tanto o circuito de entrada quanto a fórmula de saída são satisfatíveis, isto é, xc ∈ SAT-CIRCUITOS e xf ∈ SAT.

c) Se "xc" representa um circuito com portas lógicas, então o algoritmo retorna a codificação da fórmula F = "y_np e G", onde G = "(f_1) e (f_2) e ... (f_np)". Nesse caso, observe que "G" é sempre satisfatível: basta atribuir a cada variável "y_i" o resultado da avaliação da parte direita da fórmula "f_i", avaliação essa que é determinada unicamente pelos valores atribuídos às variáveis x_1 ... x_ne. Logo, "F" é satisfatível sse existir uma atribuição que torne as fórmulas f_1 a f_np verdadeiras e também a variável y_np, o que ocorre exatamente sse o circuito de entrada em questão for satisfatível, dada a construção de "F". Assim, temos xc ∈ SAT-CIRCUITOS <-> xf ∈ SAT.

Logo, SATC_SAT é um algoritmo polinomial que reduz SAT-CIRCUITOS a SAT, e portanto SAT-CIRCUITOS <=_P SAT, CQD.
==========

7. EXERCÍCIO: pratique a redução acima, escolhendo um circuito qualquer e calculando a fórmula correspondente.

8. COROLÁRIO: SAT ∈ NPC.

   PROVA: Consequência direta do lema 3 e do teorema 6 acima.