Algoritmos

Pedro Hokama

Fontes

- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Desmistificando Algoritmos, Thomas H. Cormen.
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V
  https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6
- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende
- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

3-CNF-SAT

- Uma formula booleana é composta
  - de variáveis booleanas $x_1, x_2, \ldots, x_n$,
  - de conectivos $\neg$ (NOT), $\vee$ (OR), $\wedge$ (AND),
  - Parenteses.
- Uma variável ou sua negação são chamadas **literais**.
- Uma fórmula booleana está na forma normal 3-conjuntiva se
  - está dividido em clausuras
  - cada clausura tem 3 literais conectas por ORs
  - as clausuras estão conectadas por ANDs.

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

3-CNF-SAT

3-CNF-SAT

Dado uma fórmula booleana na forma normal 3-conjuntiva, decidir se existe alguma atribuição das variáveis que torna a formula verdadeira.

Teorema

O 3-CNF-SAT é NP-Completo

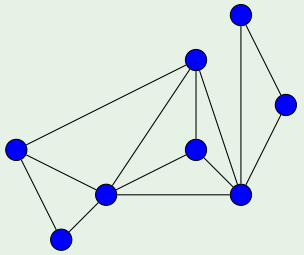
Prova: Veremos depois.

Problema do CLICK

Problema do CLICK

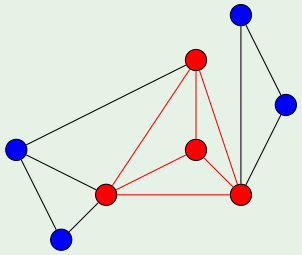
Dado um grafo não orientado $G = (V, E)$, e um inteiro k decidir se existe um subgrafo G' induzido de G que tenha k vértices e seja completo.

Exemplo



Existe uma click de tamanho 4?

Exemplo



Existe uma click de tamanho 4?

O Problema do CLICK

Será que CLICK é NP-Completo?

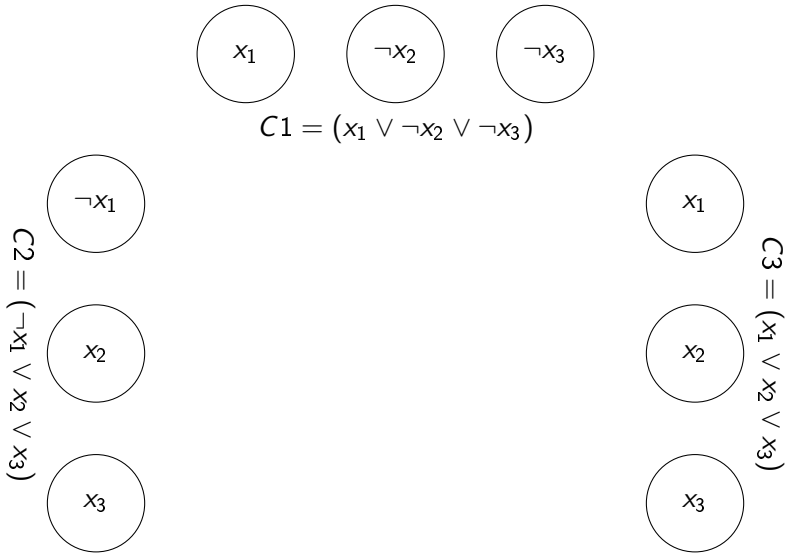
- CLICK $\in$ NP, basta apresentar o conjunto de vértices
- Vamos reduzir o 3-CNF-SAT para CLICK
  - Devemos converter qualquer instancia do 3-CNF-SAT para o problema do CLICK
  - Essa redução deve ter tempo polinomial
  - Aqui vamos mostrar o algoritmo da redução em um exemplo, mas é necessário garantir que funciona para qualquer instância.

3-CNF-SAT $\leq_p$ CLICK

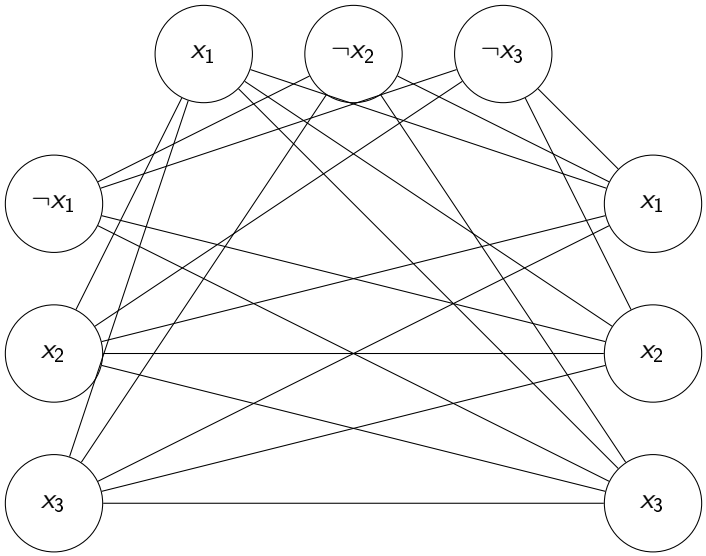
Considere uma formula booleana com k clausulas na forma normal 3-conjuntiva.

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

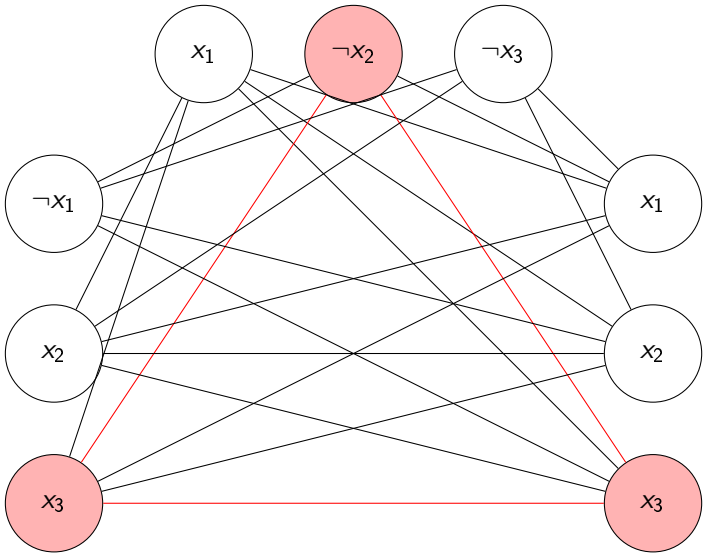
$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



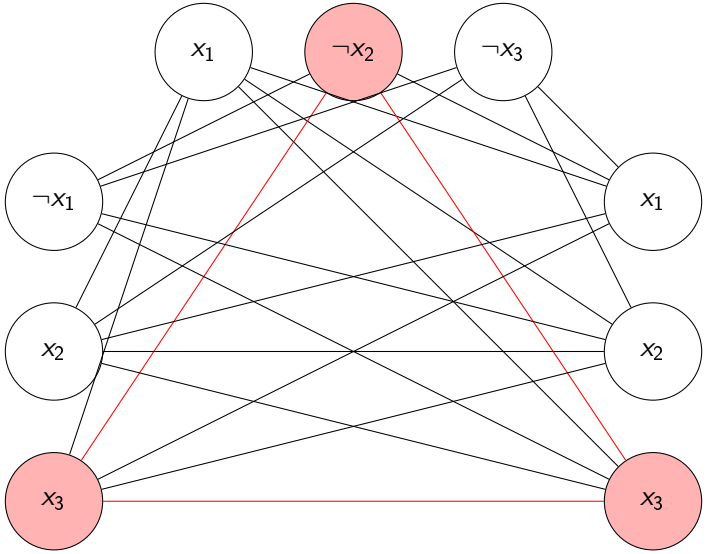
$(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



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



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



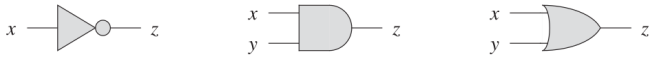
Um primeiro problema *NP*-completo

Satisfabilidade de Circuito

- O teorema de Cook-Levin é um pouco complicado (para mim, pelo menos).
- Ao invés disso vamos argumentar (um pouco informalmente) que de fato existe um problema *NP*-completo.
- O nosso primeiro problema será o problema da satisfabilidade de circuitos.

Satisfabilidade de Circuito

- Um circuito é formado por portas lógicas ligadas por fios.
- Em um extremo temos os fios de entrada e no outro temos os fios de saída.
- Existem três portas básicas. a porta NOT, AND e OR



x	$\neg x$
0	1
1	0

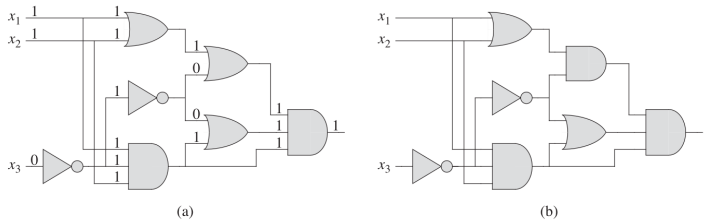
x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Satisfabilidade de Circuito

Problema da Satisfabilidade de Circuito

Dado um circuito C com fios de entrada e uma saída. Existe alguma atribuição dos valores de entrada que tornam a saída verdadeira?



- O primeiro circuito tem uma atribuição que a torna verdadeira.
- Enquanto o segundo é inviável.

Satisfabilidade de Circuito

Teorema

O Problema da Satisfabilidade de Circuitos é NP-completo.

- Primeiro provar que o Problema da Satisfabilidade de Circuitos é *NP*
- Depois provar que o Problema da Satisfabilidade de Circuitos é *NP*-Difícil.

Lema

O Problema da Satisfabilidade de Circuitos $\in$ *NP.*

Prova: Dado um circuito C e uma atribuição dos valores de cada fio de C. Um algoritmo A pode verificar cada porta lógica e verificar se os fios de entrada estão correspondendo corretamente ao fio de saída daquela porta. E verifica se o fio de saída do circuito é 1. Se C é viável ele tem um certificado, se C não é viável nenhum certificado pode enganar A. O algoritmo A roda em tempo polinomial. Então verificamos que o Problema da Satisfabilidade de Circuitos $\in$ *NP*.

Satisfabilidade de Circuito

Lema

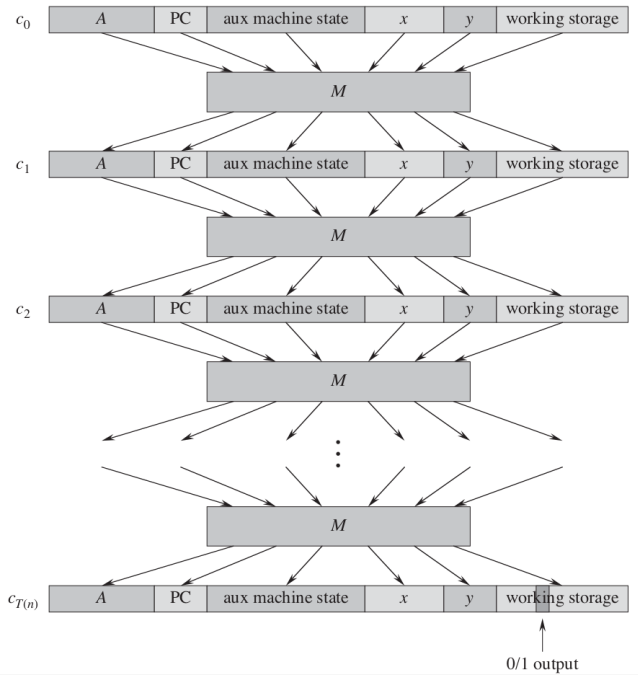
O Problema da Satisfabilidade de Circuitos é NP-Difícil.

- Primeiramente vamos relembrar de como funciona um computador.
- Um programa de computador é uma sequencia de instruções guardadas na memória do computador.
- Uma instrução tipicamente é composta da operação a ser executada, do endereço da memória dos operadores e do endereço onde vai ser armazenado o resultado.

- Um pedaço especial da memória é o **contador de programa** que sabe qual é a próxima instrução a ser executada.
- Sempre que uma instrução é executada, o contador é incrementado ou sobrescrito (no caso de um desvio)
- A qualquer momento, a memoria do computador (RAM, contadores, registradores, etc) guarda uma **configuração** completa de um passo na execução do programa.

Satisfabilidade de Circuito

- Agora considere um problema *NP* qualquer.
- Por definição existe um algoritmo A que recebe uma instância x e um certificado y e devolve SIM (1) se aquele certificado comprova que a solução de x é SIM.
- Então vamos simular a execução desse algoritmo!



- Mas o computador M nada mais é do que um circuito lógico.
- O número de réplicas é polinomial no tamanho de x
- Então se removermos o y, e deixar entradas livres. Temos um circuito que só é satisfeito com a entrada adequada.

Satisfazibilidade de Fórmulas

- Uma fórmula booleana ϕ é composta de:
  - n variáveis booleanas: $x_1, x_2, \ldots, x_n$;
  - m conectivos booleanos: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
  - parênteses
- Ex. $\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$

Satisfazibilidade de Fórmulas

Problema da Satisfazibilidade de Fórmulas - SAT

Dada uma fórmula booleana, decidir se existe uma atribuição das variáveis booleanas que faz com que ela seja avaliada como 1 (verdadeira).

- No exemplo ϕ é satisfazível com a atribuição $< x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1 >$
- Um algoritmo ingênuo que testa todas as possibilidades é inviável pois existem 2^n atribuições diferentes.

Teorema

SAT é NP-completo

Lema

SAT ∈ NP

Lema

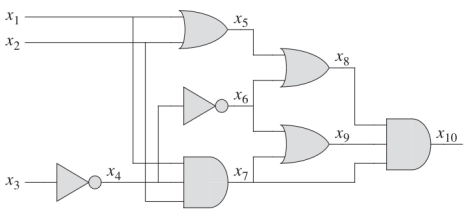
SAT ∈ NP-Difícil

Lema

SAT ∈ NP

- Para mostrar que SAT ∈ NP, mostramos que um certificado consiste em uma atribuição satisfatória das variáveis. Esse certificado pode ser verificado substituindo cada variável pelo valor dessa atribuição e avaliando a expressão, que pode ser feito em tempo polinomial. Se o resultado for 1 o algoritmo verificou que a formula é satisfazível.

- Para mostrar que SAT ∈ NP-Difícil mostraremos que CIRCUIT-SAT $\leq_p$ SAT.
- Considere um circuito C qualquer. Para cada fio x_i no circuito, a fórmula ϕ vai ter uma variável x_i, expressamos cada porta lógica como uma cláusula na formula booleana que representa o seu comportamento. No fim fazemos a conjunção das cláusulas.



$$\begin{aligned}\phi \;=\; & x_{10} \wedge (x_4 \leftrightarrow \neg x_3) \\ & \wedge (x_5 \leftrightarrow (x_1 \vee x_2)) \\ & \wedge (x_6 \leftrightarrow \neg x_4) \\ & \wedge (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4)) \\ & \wedge (x_8 \leftrightarrow (x_5 \vee x_6)) \\ & \wedge (x_9 \leftrightarrow (x_6 \vee x_7)) \\ & \wedge (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9))\,.\end{aligned}$$

Lema

C é satisfazível se e somente se ϕ é satisfazível.

- Esse algoritmo de redução executa em tempo polinomial.
- ($\rightarrow$) Se C tem uma atribuição que satisfaz, cada fio tem um valor bem definido e a saída do circuito é 1.
- Portanto se atribuirmos os valores de cada fio para as respectivas variáveis, a formula também terá valor 1.
- ($\leftarrow$) Se uma atribuição faz ϕ se verdadeira. Podemos atribuir o valor de cada fio com o valor das suas respectivas variáveis. E C terá saída igual a 1.
- Portanto mostramos que CIRCUIT-SAT $\leq_p$ SAT. $\square$

Satisfazibilidade 3-CNF

- Uma formula está na forma normal conjuntiva (*conjunctive normal form* - CNF) se é expressa como uma conjunção (ANDs) de clausulas, e cada clausula como disjunções (ORs) de uma ou mais literais.
- Uma fórmula booleana está na forma normal 3-conjuntiva (3-CNF) se está na forma normal conjuntiva e cada cláusula tem exatamente três literais distintas.
- Por exemplo:

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

3-CNF-SAT

Dada uma fórmula booleana na forma normal 3-conjuntiva, decidir se existe uma atribuição das variáveis booleanas que faz com que ela seja avaliada como 1 (verdadeira).

Teorema

3-CNF-SAT é NP-completo

Lema

3-CNF-SAT $\in$ NP

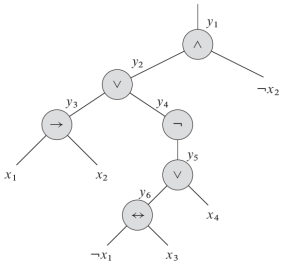
Prova igual à prova que SAT $\in$ NP.

Lema

3-CNF-SAT $\in$ NP-Difícil

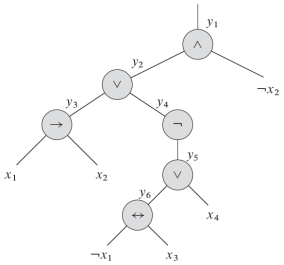
- Para mostrar que 3-CNF-SAT $\in$ NP-Difícil mostraremos que SAT $\leq_p$ 3-CNF-SAT.
- Considere ϕ uma formula booleana qualquer.
- Primeiramente construímos uma árvore de análise para ϕ em que cada literal é uma folha e os conectivos são nós internos. Essa construção sempre é possível (usando a associatividade podemos colocar parenteses para explicitar uma ordenação)

- Por exemplo $\phi = ((x_1 \to x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$.



- Depois dessa construção, introduzimos uma variável para cada aresta que sobe dos nós internos.
- A seguir escrevemos cláusulas para cada nó interno. E fazemos a conjunção dessas cláusulas e criamos uma formula ϕ'.

- Por exemplo $\phi = ((x_1 \to x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$.



- No exemplo ao lado a formula obtida seria:

$$\begin{aligned}\phi' \;=\; & y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ & \wedge\ (y_2 \leftrightarrow (y_3 \vee y_4)) \\ & \wedge\ (y_3 \leftrightarrow (x_1 \to x_2)) \\ & \wedge\ (y_4 \leftrightarrow \neg y_5) \\ & \wedge\ (y_5 \leftrightarrow (y_6 \vee x_4)) \\ & \wedge\ (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))\ .\end{aligned}$$

- Dessa forma obtemos clausulas que tem no máximo 3 literais. Mas ainda não está na forma normal 3-conjuntiva. Então podemos escrever a tabela verdade de cada clausula, e encontrar os valores que tornam ela FALSA. Por exemplo a cláusula $\phi'_1 = y_1 \leftrightarrow (y_2 \wedge \neg x_2))$

y_1	y_2	x_2	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

- Analisando as linhas da tabela que tornam a cláusula FALSA obtemos:

  $(y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2)$

- Agora aplicamos a lei de DeMorgan

  $\phi''_1 = (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2)$

- Caso a cláusula resultante só tenha 2 literais $(l_1 \vee l_2)$, então incluímos uma literal auxiliar e a seguinte formula $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$
- No caso de uma cláusula com uma única literal l, incluímos 2 literais auxiliares $(l \vee p \vee q) \wedge (l \vee p \vee \neg q) \wedge (l \vee \neg p \vee q) \wedge (l \vee \neg p \vee \neg q)$.
- Seja ϕ''' a fórmula em 3-CNF-SAT resultante.

Lema

ϕ''' é satisfazível se e somente se ϕ é satisfazível.

- Como todas as transformações preservam o valor algébrico da formula, tanto ϕ quanto ϕ''' são equivalentes.
- A redução é calculada em tempo polinomial. Construir ϕ' a partir de ϕ insere no máximo uma variável e uma cláusula por conectivo.
- Construir ϕ'' a partir de ϕ' introduz no máximo oito cláusulas para cada cláusula. E a construção de ϕ''' no máximo multiplica por 4 o número de cláusulas.
- Portanto mostramos uma redução de tempo polinomial de SAT para 3-CNF-SAT e concluímos a prova. □