Universidade de Brasília
Instituto de Ciências Exatas

Departamento de Ciência da Computação

CIC 117536 - Projeto e Análise de Algoritmos

Terceira Prova

Turma: B

NP-completude

Prof. Flávio L. C. de Moura

Alunos

Matheus Stauffer Viana de Oliveira 16/0071852

Nícolas M. Schumacher 13/0047660

7 de dezembro de 2018

1. **(2.5 pontos)** O problema 2-SAT tem como instâncias as fórmulas lógicas formadas por conjunções de disjunções de até dois literais, onde um literal é uma variável booleana ou a negação de uma variável booleana. Por exemplo, a expressão a seguir é uma instância de 2-SAT:

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

Prove que 2-SAT $\in$ P.

Solução.Como no exemplo, assumimos o mesmo 2-SAT C:

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

A fim de executar um algoritmo em tempo polinomial sobre a fórmula booleana para identificar se ela é capaz de satisfação, será adotada uma nova representação desta expressão para o formato de um grafo direcionado G = (V, E). Os seguintes passos demonstram a construção do grafo G a partir do exemplo C apresentado. Sua generalização é trivial. Segue a construção de G:

1) Cada variável booleana x_1, x_2 e x_3 de 2-SAT bem como todas as suas negações $\neg x_1, \neg x_2$ e $\neg x_3$ são vértices de G. Assim, $V = x_1, x_2, x_3, \neg x_1, \neg x_2, \neg x_3$.
2) Para cada cláusula do tipo (A $\vee$ B), para todo A, B $\in$ V, originam-se as arestas ($\neg$A, B), ($\neg$B, A) $\in$ E.

O passo 2 ocorre porque a expressão (A $\vee$ B) é equivalente a $(\neg A \rightarrow B) \wedge (\neg B \rightarrow A)$, conforme prova a **Tabela 1**:

A	B	$\neg A$	$\neg B$	$(\neg A \rightarrow B)$	$(\neg B \rightarrow A)$	$(\neg A \rightarrow B) \wedge (\neg B \rightarrow A)$	$(A \vee B)$
0	0	1	1	0	0	0	0
0	1	1	0	1	1	1	1
1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1

Tabela 1

Isto é, as arestas ($\neg$A, B) e ($\neg$B, A) significam, respectivamente, que se A não é verdadeiro, então B deve ser verdadeiro, e se B é falso, A deve ser verdadeiro. Dessa forma será buscado o resultado 1 para cada cláusula (A $\vee$ B). Se for possível que cada uma delas tenha o valor 1, então a expressão inteira é igual a 1 e, portanto, é capaz de satisfação.

O grafo G construído será capaz de satisfação se, e somente se, não houverem termos em contradição em qualquer um de seus componentes

fortemente conectados, ou seja, se o resultado não for 0 em qualquer uma das cláusulas.

Assim, quando existe um caminho em que é possível sair do vértice A e chegar em ¬A (ou ir de ¬A para A), haverá uma contradição e a fórmula será considerada incapaz de satisfação. Provamos isso por contradição:

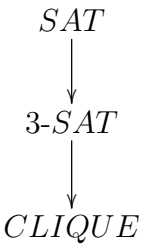
Seja o caminho x para ¬x e ¬x para x, para algum x em V, mas que também existe a atribuição p($x_1, x_2, x_3...x_n$) capaz de satisfazer a expressão. Assumimos que x = 1 de início. Digamos que no caminho de x para ¬x existe uma aresta (a, b). A partir da construção do grafo, essa aresta (a, b) existe se, e somente se, houver uma cláusula ($\neg a \vee b$) na expressão. Esta aresta mostra que, se a = 1, então b = 1 para que a cláusula ($\neg a \vee b$) seja verdadeira. Então, visto que x = 1, todos os literais no caminho de x até a, inclusive a, devem ser iguais a 1. Da mesma forma, todos os literais de b (inclusive) até ¬x devem ser iguais a 0, já que ¬x = 0. Porém isso resulta em uma aresta (a, b) tal que a = 1 e b = 0, tornando a cláusula falsa, o que é contraditório ao que foi assumido no início, portanto não há atribuição p($x_1, x_2, x_3...x_n$) capaz de satisfazer a expressão. Uma prova semelhante vale para o caso de x = 0.

A construção do grafo G a partir da expressão booleana pode ser realizada em O(n) para o passo 1, pois para cada literal diferente encontrado na expressão (ignorando a negação dele) este é adicionado ao conjunto de vértices V, bem como a negação dele e à medida que há mais literais diferentes, o tempo de execução deste trecho cresce linearmente. Para o passo 2, cada cláusula adicionará duas arestas ao conjunto E, e estas operações podem ser realizadas em tempo constante, fazendo também com que o tempo cresça linearmente à medida que o número de cláusulas aumenta. No total, para fins de simplificação e utilizando as propriedades da análise assintótica, a construção é realizada em O(n), ou seja, em tempo polinomial.

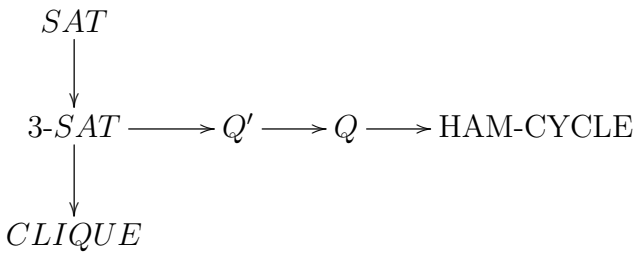
Portanto, a checagem se existe algum x $\in$ V tal que haja um caminho de x para x ou de x para x permite determinar se a expressão pode ser satisfeita ou não. Nesse caso, um algoritmo de busca em largura, por exemplo, pode determinar se tal caminho existe. Visto que as operações de construção do algoritmo levam O(n) e sabendo que a

busca em largura também pode operar em O(n), assim, podemos dizer que é possível determinar 2-SAT em tempo polinomial. Portanto, 2-SAT $\in$ P.

2. **(2.5 pontos)** Em aula, assumimos que SAT é um problema NP-completo (Teorema de Cook-Levin), e a partir deste fato mostramos que 3-SAT e CLIQUE também são problemas NP-completos. As reduções foram feitas de acordo com o seguinte diagrama:



Um ciclo Hamiltoniano é um ciclo simples que visita cada vértice de um grafo exatamente uma vez. Considere o problema de decisão HAM-CYCLE que pergunta se um dado grafo (não-dirigido) G possui um ciclo Hamiltoniano. Mostre que HAM-CYCLE é um problema NP-completo. Sua solução deve ser construída a partir de SAT, 3-SAT ou CLIQUE. Caso, você não veja como reduzir diretamente HAM-CYCLE a partir destes, mas sabe como fazê-lo a partir de um certo problema Q então inicialmente mostre que Q é NP-completo a partir de SAT, 3-SAT ou CLIQUE, e assim por diante. Digamos que você não saiba como mostrar que Q é NP-completo diretamente a partir de SAT, 3-SAT ou CLIQUE, mas você sabe como fazê-lo a partir de outro problema Q', e também sabe como mostrar que Q' é NP-completo a partir de 3-SAT, por exemplo. Então o diagrama correspondente à sua solução seria:



E todas as reduções (de 3-SAT para Q', de Q' para Q e de Q para HAM-CYCLE) devem ser detalhadas na sua solução.

Solução.Sabemos que HAM-CYCLE $\in$ NP, pois basta assegurar que, partindo de um vértice X, visitando cada vértice adjacente e seus vizinhos e retornando a X ao final, garantindo que todos os vértices em V foram visitados uma vez só, exceto X, que deve ser o vértice de partida e o de chegada. Essa análise pode ser realizada em tempo polinomial, por exemplo, por um Autômato Finito Não-Determinístico.

Para mostrar que HAM-CYCLE é NP-Completo, basta reduzir de 3-SAT. Para tanto, é necessário modelar uma instância de 3-SAT em um grafo G. Fazemos isso plotando G como um grafo que represente cada literal presente na fórmula de entrada como uma cadeia de possíveis estados - Verdadeiro ou Falso. Utilizaremos esses estados e a direção das arestas para modelar as cláusulas da entrada de 3-SAT. Representamos a presença de determinado literal em uma cláusula linkando o nó que representa esse literal com o nó-cláusula correspondente - a direção das arestas dessa ligação diz se o literal está negado ou não. A Figura 1 exemplifica o descrito:

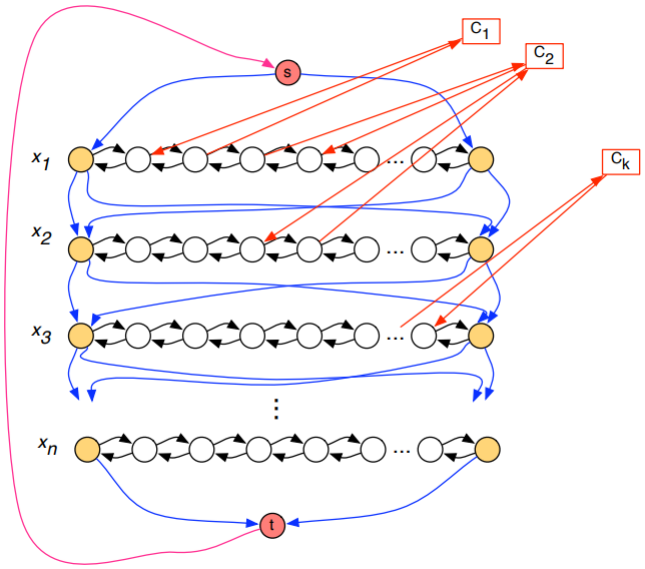


Figura 1: Descrição Grafo

Codificamos uma atribuição de valores de verdade para os literais presentes no grafo por meio de um *Caminho Hamiltoniano*. O ponto é que só conseguimos visitar um nó-cláusula se conseguimos satisfazer a cláusula respectiva. Também sabemos que um ciclo Hamiltoniano visita todos os nós do grafo - e, portanto, visita todos os nós-cláusula. Com isso, concluímos que o grafo apenas possuirá um ciclo Hamiltoniano se a instância 3-SAT de entrada possuir uma atribuição de valor de verdade que a satisfaça.

3. **(2.5 pontos)** Considere o seguinte jogo em um grafo (não-dirigido) G, que inicialmente contém 0 ou mais bolas de gude em seus vértices: um movimento deste jogo consiste em remover duas bolas de gude de um vértice $v \in G$, e adicionar uma bola a algum vértice adjacente de v. Agora, considere o seguinte problema: Dado um grafo G, e uma função $p(v)$ que retorna o número de bolas de gude no vértice v, existe uma sequência de movimentos que remove todas as bolas de G, exceto uma? Mostre que este problema é NP-completo. A mesma observação feita no exercício anterior vale aqui: a prova deve ser feita a partir de problemas que provamos serem NP-completos, e reduções intermediárias, caso existam, devem ser incluídas na solução.

   Solução.Escreva aqui sua solução.

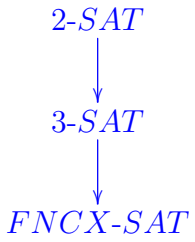
4. **(2.5 pontos)** Uma fórmula booleana em *forma normal conjuntiva com disjunção exclusiva (FNCX)* é uma conjunção de diversas cláusulas, e cada cláusula é uma disjunção exclusiva (XOR) de diversos literais. Lembre-se que a disjunção exclusiva é dada por:

a	b	$a \oplus b$
V	V	F
V	F	V
F	V	V
F	F	F

   O problema FNCX-SAT pergunta se uma dada fórmula em FNCX é satisfatível. Mostre que o problema FNCX-SAT está em P, ou então que FNCX-SAT é NP-completo. No último caso, a mesma observação feita nos dois exercícios anteriores vale aqui: a prova deve ser feita

a partir de problemas que provamos serem NP-completos, e reduções intermediárias, caso existam, devem ser incluídas na solução.

Solução.Em nossa solução iremos realizar reduções do tipo



O primeiro ponto a se considerar é que 2-SAT$\in$P. De fato, podemos resolver instâncias de 2-SAT em tempo polinomial, conforme foi provado na questão 1.

Suponhamos uma transformação t que converte instâncias de 2-SAT em 3-SAT. Imaginemos o seguinte exemplo φ de 2-SAT:

$$\varphi : (x_1 \vee x_2) \wedge (x_3 \vee x_4)$$

A transformação t consiste em adicionar a todas as n cláusulas possíveis de φ um dos literais já existentes em cada respectiva cláusula. Para o φ exemplo, um possível resultado seria:

$$\varphi_t : (x_1 \vee x_2 \vee x_1) \wedge (x_3 \vee x_4 \vee x_4)$$

Essa transformação é passível de ser realizada em tempo polinomial:

1. Para cada cláusula C de φ faça
2. Adicione à C um dos literais já existentes em C

No pseudocódigo acima, percorre-se cada cláusula C de φ uma única vez, e supondo que temos n literais no total, nossa complexidade resultante é O(n).

Agora queremos uma transformação r que mapeie instâncias de A $\vee$ B em A $\oplus$ B. Dessa maneira será possível reduzir instâncias de 3-SAT para FNCX-SAT.

Conhecendo a tabela verdade de ambas as operações, uma possível proposição para r seria verificar se os valores de verdade dos literais de uma

cláusula são ambos verdadeiros. Se sim, o valor de A $\vee$ B seria verdadeiro, e, nesse caso, inverteríamos o valor de verdade resultante. Dessa maneira, teríamos uma tabela verdade resultante igual à operação $\oplus$.

A	B	A $\vee$ B	A $\oplus$ B	(A $\vee$ B)$_r$
V	V	V	F	F
V	F	V	V	V
F	V	V	V	V
F	F	F	F	F

Essa transformação é possível de ser realizada em tempo polinomial:

1. Para cada cláusula C de uma entrada φ faça
2. se algum literal de C possui valor de verdade FALSO então
3. FIM
4. retorne FALSO

No pior caso, o pseudocódigo acima fará a verificação de todos os literais de todas as cláusulas. Tomando n como o número total de literais φ, o pior caso do algoritmo possui complexidade O(n).

Sabendo que 2-SAT $\in$ P, como foi mostrado na questão 1, e que conseguimos reduzir instâncias de 2-SAT em instâncias de 3-SAT por meio da transformação t, além de podermos reduzir instâncias de 3-SAT em FNCX-SAT por meio da transformação r, temos que FNCX-SAT $\in$ P.