Universidade Federal do Ceará – Departamento de Computação

CK0019–T02A–2014.2 Avaliação Parcial 3 – Parte 1 2014-11-24

1. (3 pontos) Dada a entrada $((\neg C) \neq (A \vee B)) \rightarrow \neg(D \geq C)$ para o problema SAT, calcule a instância de 3-SAT correspondente, de acordo com a redução SAT $\leq_{\mathrm{P}}$ 3-SAT apresentada em sala, e supondo que

A	B	$A \neq B$	$A \geq B$
1	1	0	1
1	0	1	1
0	1	1	0
0	0	0	1

.

Resposta. Primeiro criamos cláusulas para cada operador da fórmula:

$$\begin{aligned} & y_6 \wedge (y_1 \iff \neg C) \\ & \quad \wedge (y_2 \iff (A \vee B)) \\ & \quad \wedge (y_3 \iff (D \geq C)) \\ & \quad \wedge (y_4 \iff (y_1 \neq y_2)) \\ & \quad \wedge (y_5 \iff \neg y_3) \\ & \quad \wedge (y_6 \iff (y_4 \rightarrow y_5)). \end{aligned}$$

Agora, para cada cláusula, obtemos uma fórmula CNF correspondente, negando a fórmula DNF correspondente à negação da cláusula, e então aplicando de Morgan:

C		DNF $\neg C$		CNF C
$y_1 \iff \neg C$	$\rightsquigarrow$	$(y_1 \wedge C)$ $\vee \ (\neg y_1 \wedge \neg C)$	$\rightsquigarrow$	$(\neg y_1 \vee \neg C)$ $\wedge \ (y_1 \vee C)$
$y_2 \iff (A \vee B)$	$\rightsquigarrow$	$(y_2 \wedge \neg A \wedge \neg B)$ $\vee \ (\neg y_2 \wedge A \wedge B)$ $\vee \ (\neg y_2 \wedge A \wedge \neg B)$ $\vee \ (\neg y_2 \wedge \neg A \wedge B)$	$\rightsquigarrow$	$(\neg y_2 \vee A \vee B)$ $\wedge \ (y_2 \vee \neg A \vee \neg B)$ $\wedge \ (y_2 \vee \neg A \vee B)$ $\wedge \ (y_2 \vee A \vee \neg B)$
$y_3 \iff (D \geq C)$	$\rightsquigarrow$	$(y_3 \wedge \neg D \wedge C)$ $\vee \ (\neg y_3 \wedge D \wedge C)$ $\vee \ (\neg y_3 \wedge D \wedge \neg C)$ $\vee \ (\neg y_3 \wedge \neg D \wedge \neg C)$	$\rightsquigarrow$	$(\neg y_3 \vee D \vee \neg C)$ $\wedge \ (y_3 \vee \neg D \vee \neg C)$ $\wedge \ (y_3 \vee \neg D \vee C)$ $\wedge \ (y_3 \vee D \vee C)$
$y_4 \iff (y_1 \neq y_2)$	$\rightsquigarrow$	$(y_4 \wedge y_1 \wedge y_2)$ $\vee \ (y_4 \wedge \neg y_1 \wedge \neg y_2)$ $\vee \ (\neg y_4 \wedge y_1 \wedge \neg y_2)$ $\vee \ (\neg y_4 \wedge \neg y_1 \wedge y_2)$	$\rightsquigarrow$	$(\neg y_4 \vee \neg y_1 \vee \neg y_2)$ $\wedge \ (\neg y_4 \vee y_1 \vee y_2)$ $\wedge \ (y_4 \vee \neg y_1 \vee y_2)$ $\wedge \ (y_4 \vee y_1 \vee \neg y_2)$
$y_5 \iff \neg y_3$	$\rightsquigarrow$	$(y_5 \wedge y_3)$ $\vee \ (\neg y_5 \wedge \neg y_3)$	$\rightsquigarrow$	$(\neg y_5 \vee \neg y_3)$ $\wedge \ (y_5 \vee y_3)$
$y_6 \iff (y_4 \rightarrow y_5)$	$\rightsquigarrow$	$(y_6 \wedge y_4 \wedge \neg y_5)$ $\vee \ (\neg y_6 \wedge y_4 \wedge y_5)$ $\vee \ (\neg y_6 \wedge \neg y_4 \wedge y_5)$ $\vee \ (\neg y_6 \wedge \neg y_4 \wedge \neg y_5)$	$\rightsquigarrow$	$(\neg y_6 \vee \neg y_4 \vee y_5)$ $\wedge \ (y_6 \vee \neg y_4 \vee \neg y_5)$ $\wedge \ (y_6 \vee y_4 \vee \neg y_5)$ $\wedge \ (y_6 \vee y_4 \vee y_5)$

Finalmente, transformamos as fórmulas CNF obtidas em 3-CNF, obtendo então:

$$
\begin{aligned}
& (y_6 \vee p \vee q) \wedge (y_6 \vee p \vee \neg q) \wedge (y_6 \vee \neg p \vee q) \wedge (y_6 \vee \neg p \vee \neg q) \\
\wedge\ & (\neg y_1 \vee \neg C \vee p) \wedge (\neg y_1 \vee \neg C \vee \neg p) \wedge (y_1 \vee C \vee p) \wedge (y_1 \vee C \vee \neg p) \\
\wedge\ & (\neg y_2 \vee A \vee B) \wedge (y_2 \vee \neg A \vee \neg B) \wedge (y_2 \vee \neg A \vee B) \wedge (y_2 \vee A \vee \neg B) \\
\wedge\ & (\neg y_3 \vee D \vee \neg C) \wedge (y_3 \vee \neg D \vee \neg C) \wedge (y_3 \vee \neg D \vee C) \wedge (y_3 \vee D \vee C) \\
\wedge\ & (\neg y_4 \vee \neg y_1 \vee \neg y_2) \wedge (\neg y_4 \vee y_1 \vee y_2) \wedge (y_4 \vee \neg y_1 \vee y_2) \wedge (y_4 \vee y_1 \vee \neg y_2) \\
\wedge\ & (\neg y_5 \vee \neg y_3 \vee p) \wedge (\neg y_5 \vee \neg y_3 \vee \neg p) \wedge (y_5 \vee y_3 \vee p) \wedge (y_5 \vee y_3 \vee \neg p) \\
\wedge\ & (\neg y_6 \vee \neg y_4 \vee y_5) \wedge (y_6 \vee \neg y_4 \vee \neg y_5) \wedge (y_6 \vee y_4 \vee \neg y_5) \wedge (y_6 \vee y_4 \vee y_5). \quad \square
\end{aligned}
$$

2. (2 pontos) Seja Coloração de Vértices o problema de, dados um grafo não-direcionado $G = (V, E)$ – sendo $V = \{1, \ldots, n\}$ – e um número natural k, responder se existe uma k-coloração própria de G. Uma k-coloração de G é uma atribuição $c : V \to \{1, \ldots, k\}$ de "cores" (formalmente, de números naturais) aos vértices de G. Uma coloração c de G é própria sse nunca atribui a mesma cor a vértices vizinhos, isto é, sse, $\forall \{u, v\} \in E$, $c(u) \neq c(v)$.

Assim sendo, mostre que Coloração de Vértices está na classe NP.

Resposta. Considere o seguinte algoritmo:

> Algoritmo: VerifColor
> Entrada: $x, y \in \{0, 1\}^*$
> Saída: 0 ou 1.
>
> 1. Compute o grafo $G = (V, E)$, onde $V = \{1, \ldots, n\}$, e o natural k codificados por x, ou retorne 0 se x não codifica uma entrada válida para o problema.
> 2. Compute a lista de naturais $c[1..n]$ representada por y, ou retorne 0 se y não representa uma lista de n números naturais, cada um deles de 1 a k.
> 3. Verifique se, $\forall \{u, v\} \in E$, $c[u] \neq c[v]$; caso isso não aconteça, retorne 0; caso aconteça, retorne 1.

Observe primeiramente que a linguagem L do problema Coloração de Vértices é o conjunto das sequências binárias que são verificadas pelo algoritmo acima por meio de certificados de tamanho polinomial:

1. De fato, para toda entrada $x = \langle G, k \rangle \in L$, isto é, para toda codificação de G e k, $G = (V, E)$ sendo um grafo k-colorível, existe um certificado $y = \langle c[1..n] \rangle$ para x, onde $n = |V|$: para qualquer k-coloração própria de G, basta considerar o vetor $c[1..n]$ tal que $c[i]$ é a cor do vértice i; nesse caso, claramente temos $\text{VerifColor}(x, y) = 1$.

   É importante observar que $|y|$ é polinomial sobre $|x|$, pois y consiste na codificação de uma sequência de n números ($n \leq |x|$), cada um dos quais vale no máximo k ($|\langle k \rangle| \leq |x|$).

2. Por outro lado, se $x \in \{0, 1\}^*$ não representa uma entrada válida para o problema, ou se $x = \langle G, k \rangle$ e G não é k-colorível, segue do texto do algoritmo que $\text{VerifColor}(x, y) = 0$ para todo y; em particular, observe que, se o algoritmo atinge a linha 3, então y codifica uma k-coloração de G; logo, como G não é k-colorível, então a coloração em questão não é própria, e esse fato é detectado durante a execução da linha 3.

Por fim, observe que VerifColor executa em tempo polinomial sobre $|x| + |y|$, pois as linhas 1 e 2 claramente podem ser executadas em tempo polinomial sobre $|x|$ e $|y|$, respectivamente, e a linha 3 pode ser executada em tempo $O(|E|) = O(|x|)$.

Assim, nós concluímos que VerifColor verifica L em tempo polinomial, e portanto que $L \in \text{NP}$, CQD. $\square$

3. (2,5 pontos) Informalmente, dois grafos $G = (V, E)$ e $G' = (V', E')$ são isomorfos sse eles têm a mesma forma, diferindo então no máximo com relação aos nomes dados aos vértices. Formalmente, G e G' são isomorfos sse existe uma função bijetiva $f : V \to V'$ tal que, $\forall u, v \in V$, $\{u, v\} \in E \iff \{f(u), f(v)\} \in E'$.

Um grafo $G = (V, E)$ é subgrafo de um grafo $G' = (V', E')$ sse $V \subseteq V'$ e $E \subseteq E'$.

O problema de decisão Isomorfismo de Subgrafo é o de, dados grafos não-direcionados G_1 e G_2, responder se G_1 é isomorfo a algum subgrafo de G_2 – isto é (informalmente falando), o problema de responder se G_1 tem a mesma forma de alguma parte de G_2.

Mostre que Isomorfismo de Subgrafo é NP-difícil.

Resposta. Por redução a partir do problema de decisão Clique, que é NP-difícil.

Considere o seguinte algoritmo:

> Algoritmo: Clique_IsoSub
> Entrada: uma sequência binária $x \in \{0, 1\}^*$
> Saída: uma sequência binária.

1. Verifique se x codifica uma entrada válida para o problema Clique; se isso não acontecer, retorne a sequência binária vazia; se acontecer, sejam $G = (V, E)$ o grafo não-direcionado e k o natural positivo representados por x.
2. Se $k > |V|$, retorne $\langle (H, H') \rangle$, sendo H um grafo com dois vértices e uma aresta e H' um grafo com um vértice e nenhuma aresta.
3. Seja G_1 um grafo completo com k vértices.
4. Retorne $\langle (G_1, G) \rangle$.

Assim sendo, observe primeiramente que Clique_IsoSub reduz Clique a Isomorfismo de Subgrafo. De fato, para qualquer sequência binária x, a chamada Clique_IsoSub(x) sempre retorna uma sequência binária z, e o seguinte ocorre, sendo LC e LIS as linguagens correspondentes ao problema Clique e ao problema Isomorfismo de Subgrafo, respectivamente:

- Se x não representa uma entrada válida para o problema Clique, então z é a sequência vazia (linha 1), que também não codifica uma entrada válida para Isomorfismo de Subgrafo. Logo, temos $x \notin$ LC e $z \notin$ LIS.
- Se x representa uma entrada (G, k) e $k > |V|$, então $z = \langle (H, H') \rangle$, H e H' possuindo 2 e 1 vértices, respectivamente (linha 2). Nesse caso, temos $x \notin$ LC – pois G não pode possuir uma clique de k vértices se $|V| < k$ – e $z \notin$ LIS – pois todo subgrafo de H' tem menos que 2 vértices, e portanto não pode ser isomorfo a H.
- Se x representa uma entrada (G, k), sendo $G = (V, E)$ e $k \leq |V|$, então $z = \langle (G_1, G) \rangle$, sendo G_1 um grafo completo de k vértices (linhas 3 e 4). Nós argumentamos agora que G possui uma clique de k vértices sse G_1 é isomorfo a um subgrafo de G. De fato:

– Se G possui uma clique de k vértices, então seja C uma tal clique e G' o subgrafo de G induzido por C – isto é, G' possui os vértices de C e todas as arestas de G entre vértices de C. Como C é uma clique de G, então G' é um grafo completo. Além disso, G' possui exatamente k vértices. É evidente, então, que G_1 é isomorfo a G', pois ambos são grafos completos de exatamente k vértices.

– Se G_1 é isomorfo a um subgrafo de G, então G possui como subgrafo um grafo completo de k vértices, ou seja, G possui uma clique de tamanho k.

Assim sendo, nós concluímos que $x \in \text{LC} \iff z \in \text{LIS}$.

Nós concluímos então que $x \in \text{LC} \iff z \in \text{LIS}$, e portanto que o algoritmo acima reduz LC a LIS.

Observe agora que toda saída $z = \text{Clique\_IsoSub}(x)$ é retornada pelo algoritmo em tempo polinomial sobre $|x|$, pois:

1. A linha 1 realiza apenas a decodificação de x.
2. A linha 2 apenas verifica se G possui menos do que k vértices e, se isso ocorrer, computa uma saída em tempo constante.
3. A linha 3 apenas produz a representação de um grafo completo G_1 de k vértices, sendo $k \leq |V|$ – o que leva tempo $O(|V|^2) = O(|x|^2)$.
4. A linha 4 apenas retorna a codificação do par (G_1, G).

Assim sendo, o algoritmo acima reduz Clique a Isomorfismo de Subgrafo em tempo polinomial, e portanto este último problema é NP-difícil, CQD. □

4. (2,5 pontos) Um grafo não-direcionado $G = (V, E)$ é k-colorível sse existe uma função $c : V \to \{1, \ldots, k\}$ tal que, $\forall \{u, v\} \in E$, $c(u) \neq c(v)$.

Seja, agora, 2-Coloração o problema de, dado um grafo G, responder se G é 2-colorível. Mostre que 2-Coloração $\in P$.

Resposta. Considere o seguinte algoritmo:

```
=====================================================================
Algoritmo: testa2c
Entrada: uma sequência binária "x"
Saída: 0 ou 1.
---------------------------------------------------------------------
01. Seja G o grafo codificado por "x",
    ou retorne 0 se "x" não codifica um grafo não-direcionado.
02. PARA cada vértice "u" de G
03. | cor[u] := -1   // "u" ainda não colorido
04. PARA cada vértice "y" de G
05. | SE cor[y] = -1
06. | | cor[y] := 0
07. | | L := lista contendo apenas "y"
08. | | ENQUANTO "L" não estiver vazia
09. | | | Remova um vértice "u" qualquer de L
10. | | | PARA cada vizinho "v" de "u"
11. | | | | SE cor[v] = cor[u]
```

```
12. | | | | | RETORNE 0.
13. | | | | SE cor[v] = -1
14. | | | | | cor[v] := (cor[u] + 1) mod 2
15. | | | | | Insira "v" em L
16. RETORNE 1.
=======================================================================
```

Nós argumentamos que "testa2c" decide a linguagem 2-Coloração. Uma demonstração rigorosa desse fato é longa e está além do escopo deste material; assim, a seguir apresentaremos uma argumentação informal do fato. Seja então uma sequência $x \in \{0,1\}^*$; temos:

1. Se x não codifica uma entrada para o problema, então o algoritmo corretamente retorna 0 (linha 1).

2. Se $x = \langle G \rangle$ e G é 2-colorível, então o algoritmo realiza uma 2-coloração própria de G e ao final retorna 1. De fato, observe que, se um grafo é 2-colorível, então só existe essencialmente uma maneira de 2-colorí-lo: o que pode mudar de uma coloração para outra são apenas os "nomes" das cores. Assim, como G é 2-colorível, a coloração produzida pelo algoritmo é essencialmente a mesma de qualquer 2-coloração própria do grafo – pois o algoritmo sempre colore primeiro os vizinhos de vértices que já foram coloridos, e em tais casos só existe uma opção de cor que leva a uma 2-coloração própria, conforme feito pelo algoritmo –, e portanto não acontece de o algoritmo atribuir cores iguais a vértices vizinhos no grafo (que é o que poderia fazer o algoritmo retornar 0).

3. Se $x = \langle G \rangle$ e G não é 2-colorível, então o algoritmo detectará o fato e retornará 0. De fato, observe que o algoritmo procede tentando 2-colorir o grafo de maneira própria, e que, para todo vértice colorido, é verificado se a coloração continua própria; logo, como G não é 2-colorível, o algoritmo acabará por descobrir uma aresta cujas extremidades receberam cores iguais, e então retornará 0, corretamente.

Observe agora que o algoritmo acima executa em tempo polinomial sobre o tamanho da sua entrada x. De fato, a linha 1 apenas decodifica a entrada e, pela suposição da nossa teoria sobre as codificações utilizadas, pode ser executada em tempo polinomial sobre $|x|$. Além disso, sendo $G = (V, E)$, $n = |V|$ e $m = |E|$, observe que as linhas 2–7 claramente executam em tempo $O(n)$, que o mesmo vale para as linhas 8 e 9 (pois cada vértice é colorido apenas uma vez, e portanto é inserido em L apenas uma vez, e portanto é removido de L apenas uma vez), e que as linhas 10–15 executam em tempo $O(m)$ (pois cada aresta $\{a, b\}$ é considerada no laço da linha 10 exatamente 2 vezes: quando a é removido de L e quando b é removido de L); logo, as linhas 2–16 executam em tempo $O(n + m) = O(|x|)$.

Assim sendo, nós concluímos que "testa2c" decide 2-Coloração em tempo polinomial, e portanto que 2-Coloração $\in P$, CQD. □

— Boa prova! —