CPSC 421/501 Homework Solutions 11

(1) (a) If $\sum_{i \in I} x_i = t$ for some

$I \subset [N]$, then

$$\sum_{j \notin I} x_j = \sum_{i=1}^{N} x_i - \sum_{i \in I} x_i = y - t$$

So

$$\sum_{i \in I} x_i = t$$

$$y - t = \sum_{j \notin I} x_j$$

so adding the two sides we have

$\langle x_1, \ldots, x_N, y-t, t \rangle \in$ PARTITION

(b) There are 2 problems:

(i) $y-t$ can be negative: e.g. $N=1$

$(x_1, t) = (1, 2)$ so $y = \sum_{i=1}^{N} = 1$, so

$y - t = 1 - 2 < 0$

(ii) there is another way to sum

$x_1, \dots, x_N, y-t, t$ to get equality

whether or not $\langle x_1, \dots, x_N, t \rangle \in$ SUBSET-SUM:

namely: $N = 1$, $x_1 = 3$, $t = 2$, so

$\langle x_1, t \rangle \notin$ SUBSET-SUM, but $y = 3 - 2 = 1$

and
$$x_1 = 3 = 2 + 1 = t + (y - t).$$

(c) The trick is to choose B so that

problems (i) and (ii) can't happen;

so we need

(i) $y - t + B \geq 1$, so $B \geq 1 + t - y$,

and

(ii) $y - t + B$ and $t + B$ must be on

opposite sides of the partition, so

that

$\langle x_1, \dots, x_N, \; y - t + B, t + B \rangle \in$ PARTITION

$\Longrightarrow$ $y - t + B, t + B$ are on opposite sides,

i.e. we can't have

$$(*) \begin{cases} (y - t + B) + (t + B) + \sum_{i \in I} x_i \\ \qquad = \sum_{j \notin I} x_j \end{cases}$$

But for any $I \subset [N]$,

$$\sum_{i \in I} x_i \geq 0, \quad \sum_{j \notin I} x_j \leq \sum_{j=1}^{n} x_j = y$$

So implies

$$(y - t + B) + (t + B) = \sum_{j \notin I} x_j - \sum_{i \in I} x_i \leq y - 0 = y$$

So

$$y + 2B \leq y$$

which can't happen if $B \geq 1$.

So $\langle x_1, \dots, x_N, t \rangle \in$ SUBSET

$\Leftrightarrow \langle x_1, \ldots, x_N, y-t+B, t+B \rangle \in$ PARTITION

if

$$B \geq \max(1+t-y, 1).$$

So it suffices to take

$$B = \max(1+t-y, 1)$$

Then B is at most $\max(1, t)$, so

$y-t+B$, $t+B$ have their size at most $y = \sum_{i=1}^{N} x_i$ plus t plus B. Hence

$$|\langle x_1, \ldots, x_N, y-t+B, t+B \rangle| \leq \text{poly}(|\langle x_1, \ldots, x_N, B \rangle|).$$

(d) PARTITION $\in$ NP by non-deterministically guessing, on input $\langle x_1, \ldots, x_N \rangle$, whether $i \in [N]$ is put into $I \subset [N]$ or not, and then comparing $\sum_{i \in I} x_i$ to $\sum_{j \notin N} x_j$.

By (c),

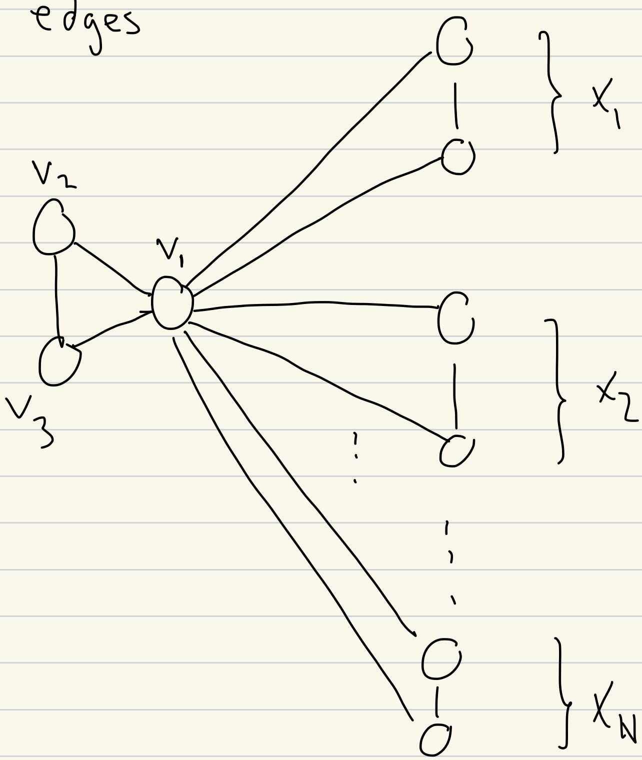
$$\text{SUBSET-SUM} \leq_p \text{PARTITION},$$

so PARTITION is NP-complete.

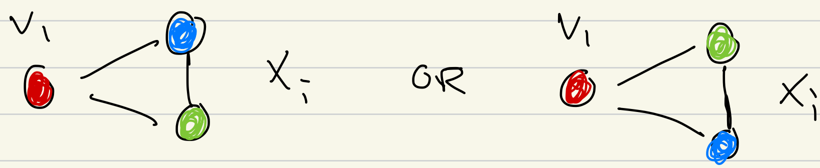
(2) 3COLOR is NP-complete:

In class we explained that 3COLOR $\in$ NP (by non-deterministically guessing a 3 colouring).

Given a 3CNF in variables $x_1, \dots, x_N$, for each variable introduce vertices and edges

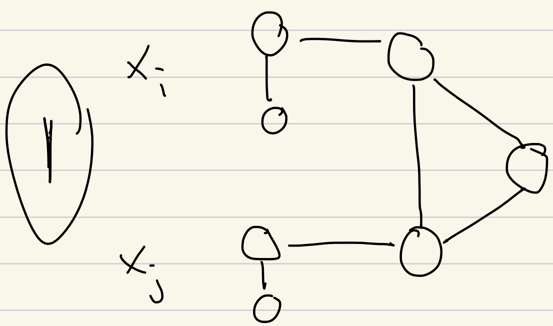


Hence if v_1 has the colour red (say the colours are R = red, G = green, B = blue) then each variable must be coloured G, B or B, G:



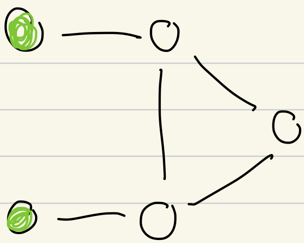
Now we view the top x_1 vertex as representing T (true), the bottom as F (false).

For each clause $C_1, \ldots, C_m$ of the 3CNF, we add the following: for a clause x_i OR x_j OR x_k we add:

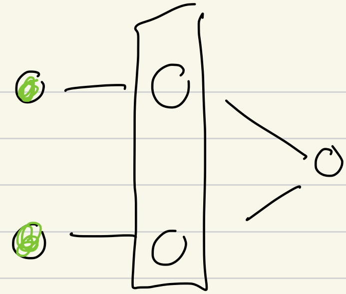


this can be blue iff one of the top x_i or top x_j vertices is blue

Since

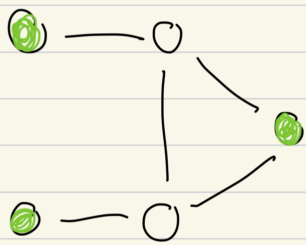


implies

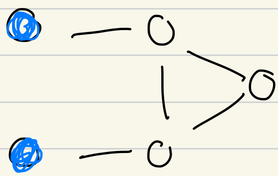


one is blue,
one red

implies

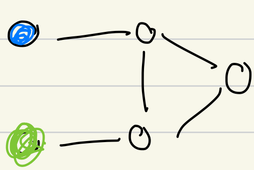


but, similarly

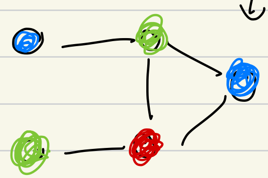


implies this
must be blue,

and

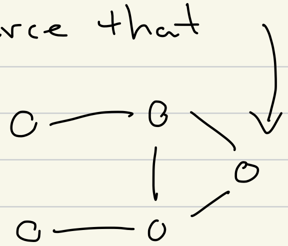


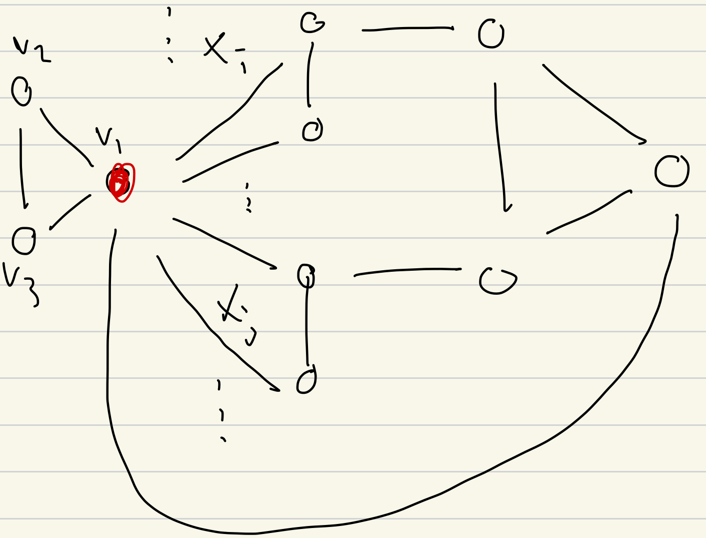
allows for



can be
blue

To enforce that is blue or green, we can add an edge to "red"

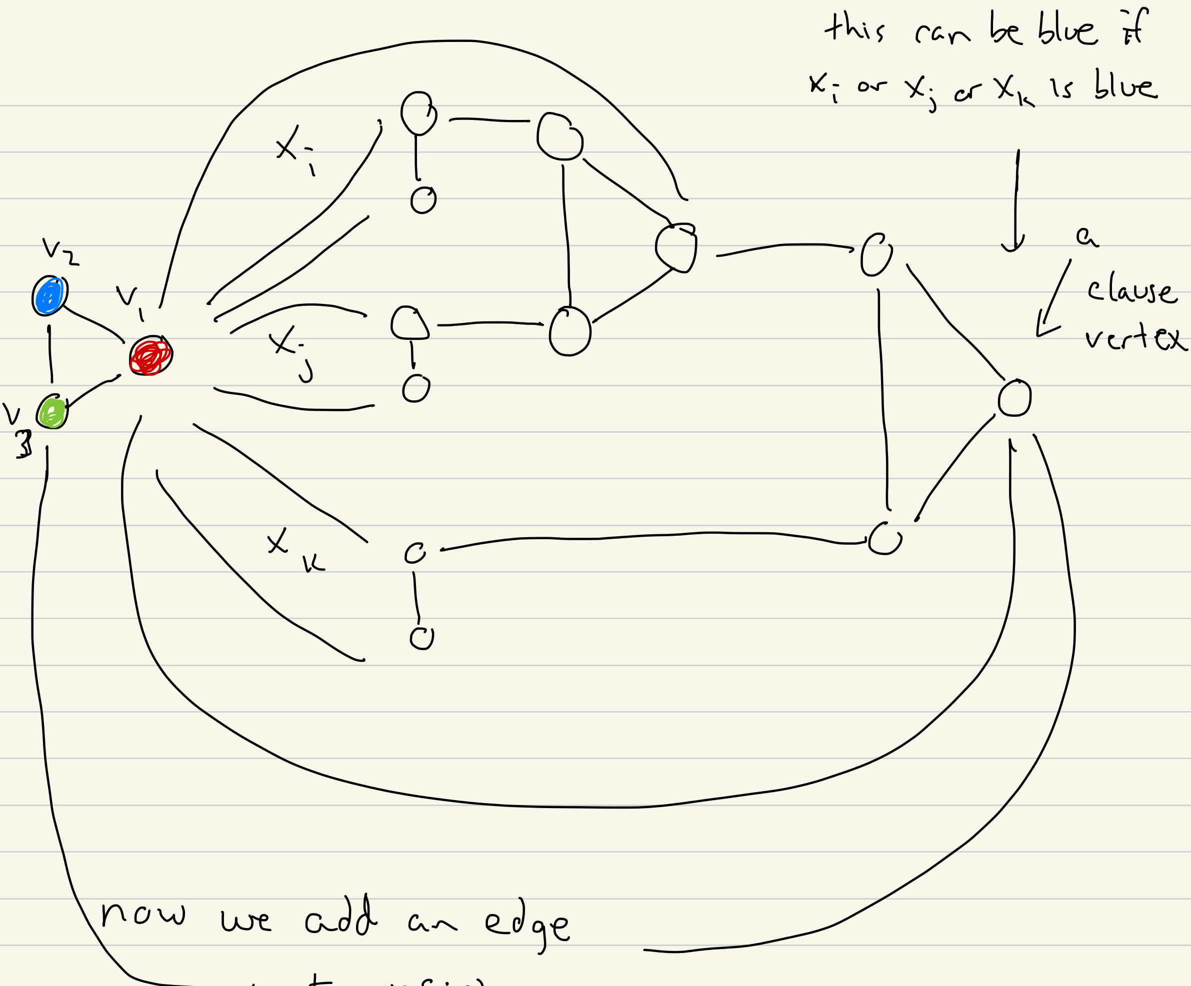




" x_i OR x_j "

enforce green or blue

② We now add a similar "OR" gadget between " x_i OR x_j " and " x_k "



now we add an edge to insist

hence the rightmost vertex must be blue, i.e. the colour at v_2 on the left. We do the same for every other clause, but connecting to the bottom vertex of x_i if $\neg x_i$ appears, and similarly for $x_j, \neg x_j$ and $x_k, \neg x_k$.

Hence we can satisfy the 3CNF iff

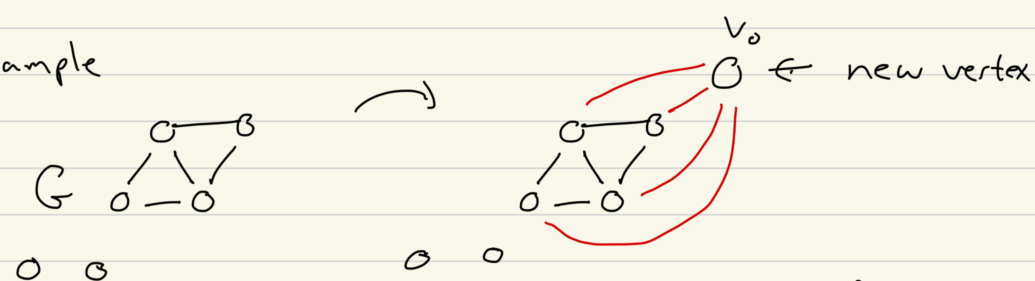
each clause vertex is colourable (with the color at V_2, here shown in blue).

Hence the 3CNF in question is satisfiable this graph can be 3-coloured.

(3) 4COLOR $\in$ NP by non-determinstically guessing a 4-colouring of the graph (i.e. for each vertex connected to at least one edge).

To reduce 3COLOR to 4COLOR, given a graph G, add a new vertex, v_0, to G, connected to every vertex of G (connected to some edge)

Example



$G + v_0 +$ a bunch of edges

The new graph, G', can be described by having one more vertex, and at most 2 new edges for each edge of G. Hence $|\langle G' \rangle|$ can be generated from G in poly time.

Any 4-colouring of G' gives a 3-colouring of G with the 3 colours different than the colour of v_0; conversely, any 3-colouring of G gives a 4 colouring of G' by colouring v_0 with the 4^{th} colour. Hence

$$\langle G \rangle \in 3\,\text{COLOR} \iff \langle G' \rangle \in 4\,\text{COLOR}.$$

So the map $G \mapsto G'$ gives a reduction $3\,\text{COLOR} \leq 4\,\text{COLOR}$.

Hence 4COLOR is NP-complete.

(4) (a) If for some $i \neq j$, $x_i = x_j = T$, then

either ① i, j are both in $\{1, \dots, m\}$

OR ② i, j are both in $\{m+1, \dots, n\}$

OR ③ one each of i, j are in one of each of $\{1, \dots, m\}$ and $\{m+1, \dots, n\}$.

Conversely, if $i \neq j$ are both in $\{1, \dots, m\}$ with $i \neq j$, then $Th_{2,n}(x_1, \dots, x_n) = T$, and similarly if they are both in $\{m+1, \dots, n\}$ or if one is in $\{1, \dots, m\}$, the other in $\{m+1, \dots, n\}$. Hence the equation holds

(b) Taking $m = n/2$, we have $Th_{2,n}$ is written as the $\wedge$ (AND) of two $Th_{2,n/2}$ formulas, and of $(x_1 \vee \dots \vee x_m) \wedge (x_{m+1} \vee \dots \vee x_n)$, the latter of size n.

Hence $L_n \le 2 \cdot L_{n/2} + n$.

(c) By induction on ~~k~~ k:

for $k=1$, $n = 2^k = 2$, $n \log_2 n = 2 \cdot 1$, and

$Th_2(x_1, x_2) = x_1 \wedge x_2$, which is of length 2.

Hence this holds for $k=1$

Assuming it holds for some k, then

$$L_{2^{k+1}} \le 2 L_{2^k} + 2^{k+1}$$

$$= 2 \cdot 2^k \cdot k + 2^{k+1}$$

$$= 2^{k+1}(k+1)$$

so it holds for $k+1$. Hence, by induction, it holds for all k.

(d) (a,b,c) is a slight variant, which is of the same size but is the OR of more clauses. For example, (a,b,c) gives, after rearranging x_2, x_3:

$Th_{2,4}(x_1, x_3, x_2, x_4)$

$= Th_2(x_1, x_3) \vee Th_2(x_2, x_4) \vee \big((x_1 \vee x_3) \wedge (x_2 \vee x_4)\big)$

$= (x_1 \wedge x_3) \vee (x_2 \wedge x_4) \vee \underbrace{\big((x_1 \vee x_3) \wedge (x_2 \vee x_4)\big)}$

But in class we wrote

$$= \big((x_1 \vee x_4) \wedge (x_2 \vee x_3)\big) \wedge \quad \text{same}$$

c_2 $\qquad$ c_1

So the c_2 expands to 8 terms

$x_1x_2, x_1x_3, x_4x_2, x_4x_3$ (we drop the $\wedge$, and write commas for the $\vee$)

Wheras (a,b,c) writes only

x_1x_3, x_2x_4

since the other two, x_1x_2, x_4x_3, already appear when expanding c_1

So the formulas are of the same size, but the one in (a,b,c) is the OR of more clauses: they each have a largest clause of size $n=2^k$ literals, but the others in (a,b,c) are of size 2^{k-1}, or 2^{k-2}, etc.