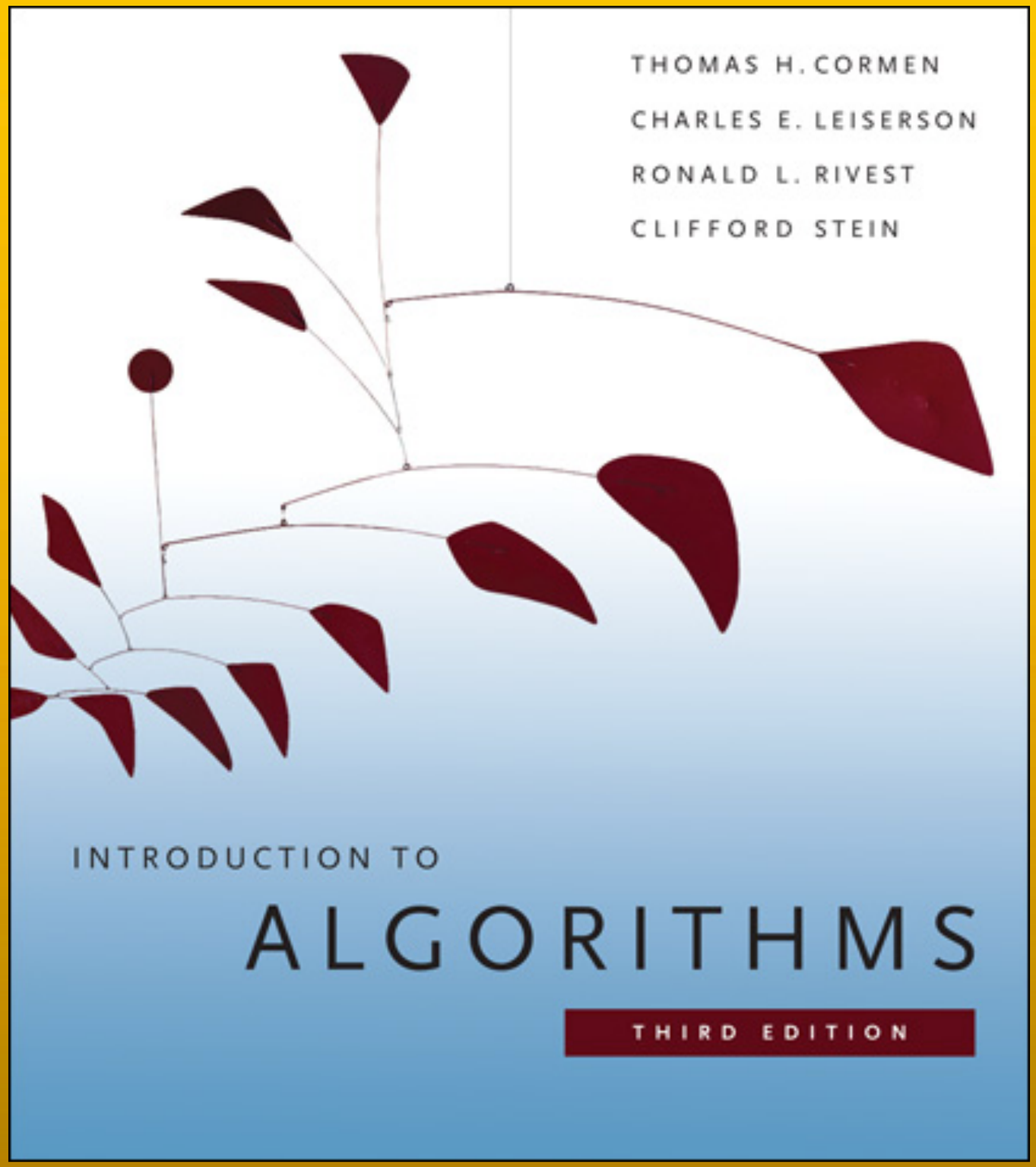
Special Topic 1 : Understanding NP-hard problem

Jonathan Irvin Gunawan
National University of Singapore


THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN
INTRODUCTION TO
ALGORITHMS
THIRD EDITION

Chapter 34

prerequisite

tau graph

bisa DP

motivation problem

IOI 2014 Friend

dikasih graph, tiap node punya weight

lu mo ambil beberapa node tapi gak boleh ada dua node yang lu ambil yang adjacent

lu mau total semua weight dari node2 yang lu pilih maksimum

definisi2

P =

bisa disolve in

polynomial time

NP =

bisa dicek in

polynomial time

NP-hard =
kalo lu bisa solve ini,
lu bisa solve semua
problem di NP

NP-complete =

NP hard dan NP

definisi gak resminya, biar gampang

NP-hard = belum ditemuin solusi polinomialnya

definisi gak resminya, biar gampang

research since 1971, unlikely buat lu bisa nemuin cuma dalam 5 jam

optimisation vs decision problem

decision problem

dikasih graph N vertex. **bisa ga** lu pilih at most K vertex sedemikian sehingga tiap edge (a,b), at least satu vertex lu pilih

optimisation problem

dikasih graph N vertex. **tentuin minimal vertex yang harus** lu pilih sedemikian sehingga tiap edge (a,b), at least satu vertex lu pilih

decision <=> optimisation

coba proof :)

cara buat tau sebuah problem itu NP-hard

reduction

notasi + definisi 1 :

Y polynomial-time reduce ke X (ditulis $Y \leq_p X$) jika lu bisa solve X in polynomial time, maka lu bisa solve Y in polynomial time

suppose lu mau tau problem X itu NP-hard ato kagak

kalo lu bisa cari sebuah problem Y yang NP-hard, dan Y ≤p X

maka X itu NP-hard

prove by contradiction

mari kita mulai

problem pertama

3-SAT

Iu dikasih POS, tiap suku terdiri dari 3 variabel di OR semua sukunya di AND

tentuin ada solusi yang bikin

TRUE ato kagak

contoh

(a OR -a OR -b) AND
(c OR b OR d) AND
(-a OR -c OR -d)

contoh

(a OR b OR c) AND
(a OR b OR -c) AND
(a OR -b OR c) AND
(a OR -b OR -c) AND
(-a OR b OR c) AND
(-a OR b OR -c) AND
(-a OR -b OR c) AND
(-a OR -b OR -c)

untuk sementara, marilah

terima tanpa bukti bahwa,

$$3\text{-SAT} \in \text{NP-hard}$$

I have discovered a truly marvellous proof of this, which this margin is too narrow to contain.

nah, soal

MAX-CLIQUE

dikasih graph N vertex, lu mo pilih beberapa vertex (maksimal)

sedemikian sehingga
untuk tiap pair vertex
(u,v), ada edge (u,v)

kita prove MAX-CLIQUE itu NP-hard

3-SAT ≤p MAX-CLIQUE

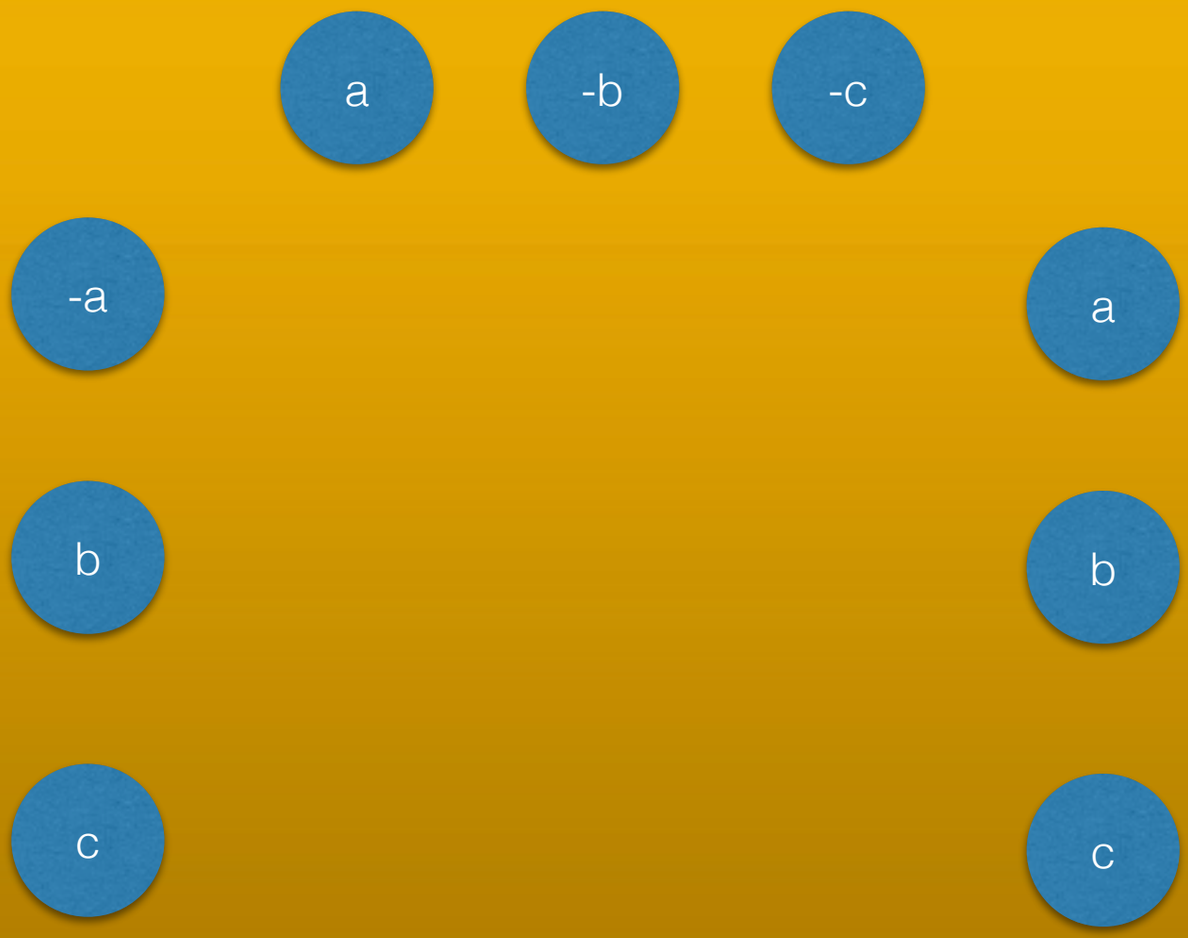
misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

untuk tiap literal kita bikin nodenya



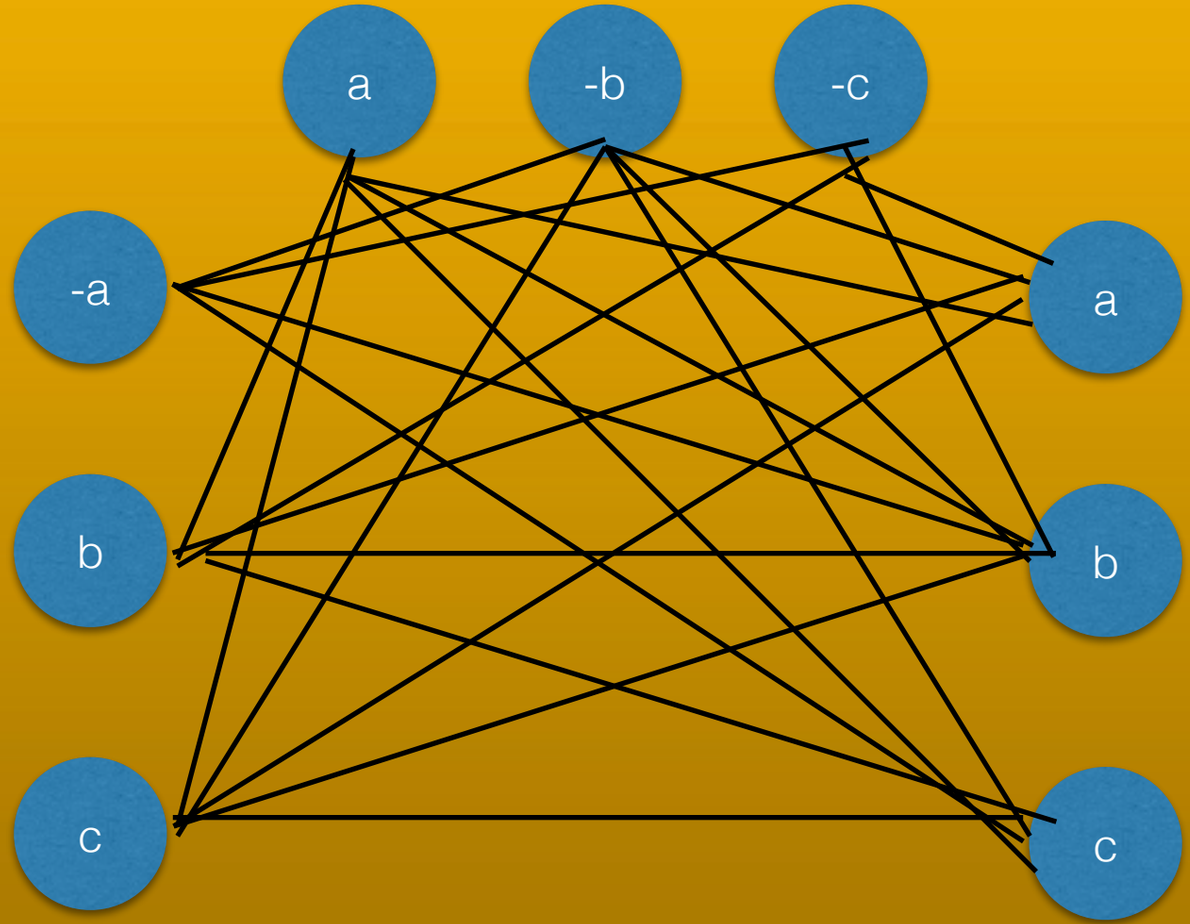
misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

untuk tiap literal (x,y) kita kasih edge jika

(1) beda clause, dan

(2) x bukan negasi dari y



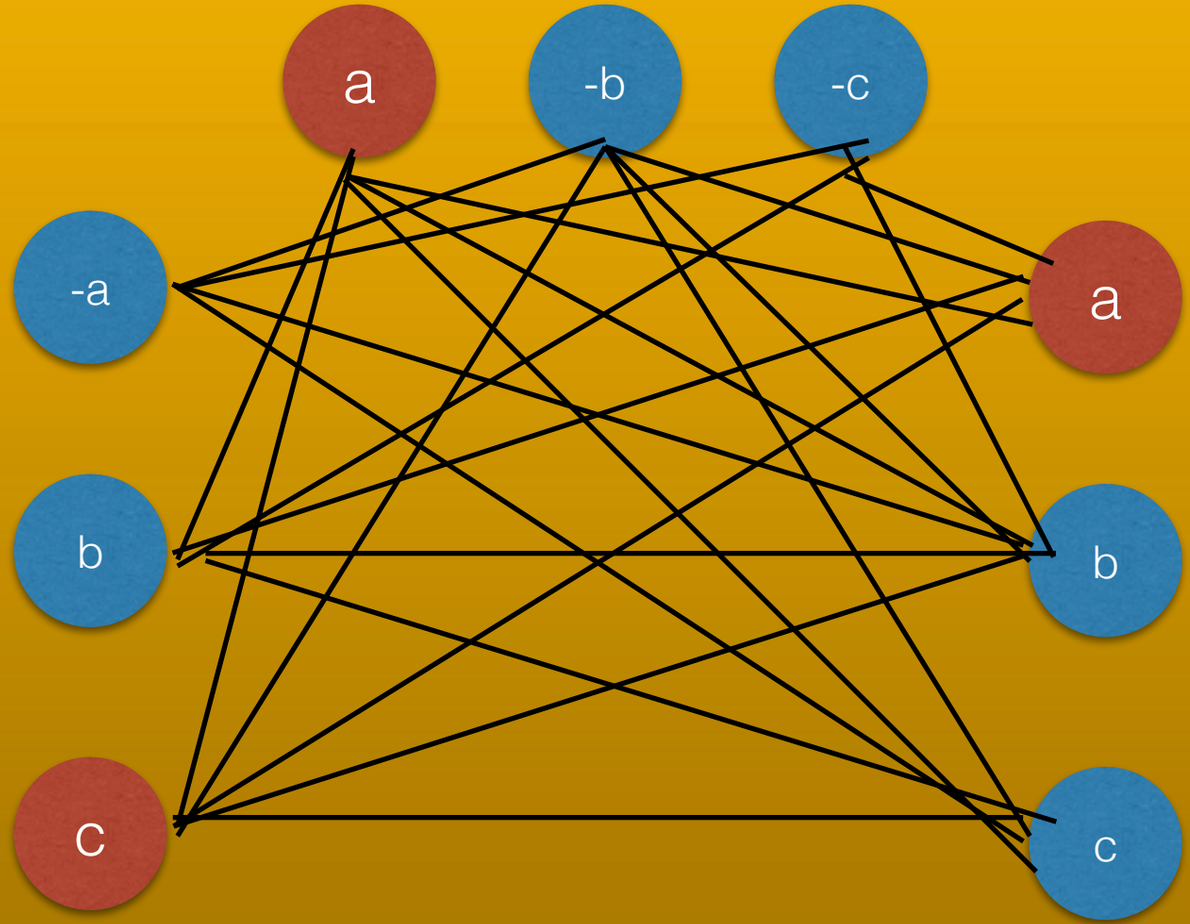
misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

MAX-CLIQUE ≥ banyaknya clause

<=>

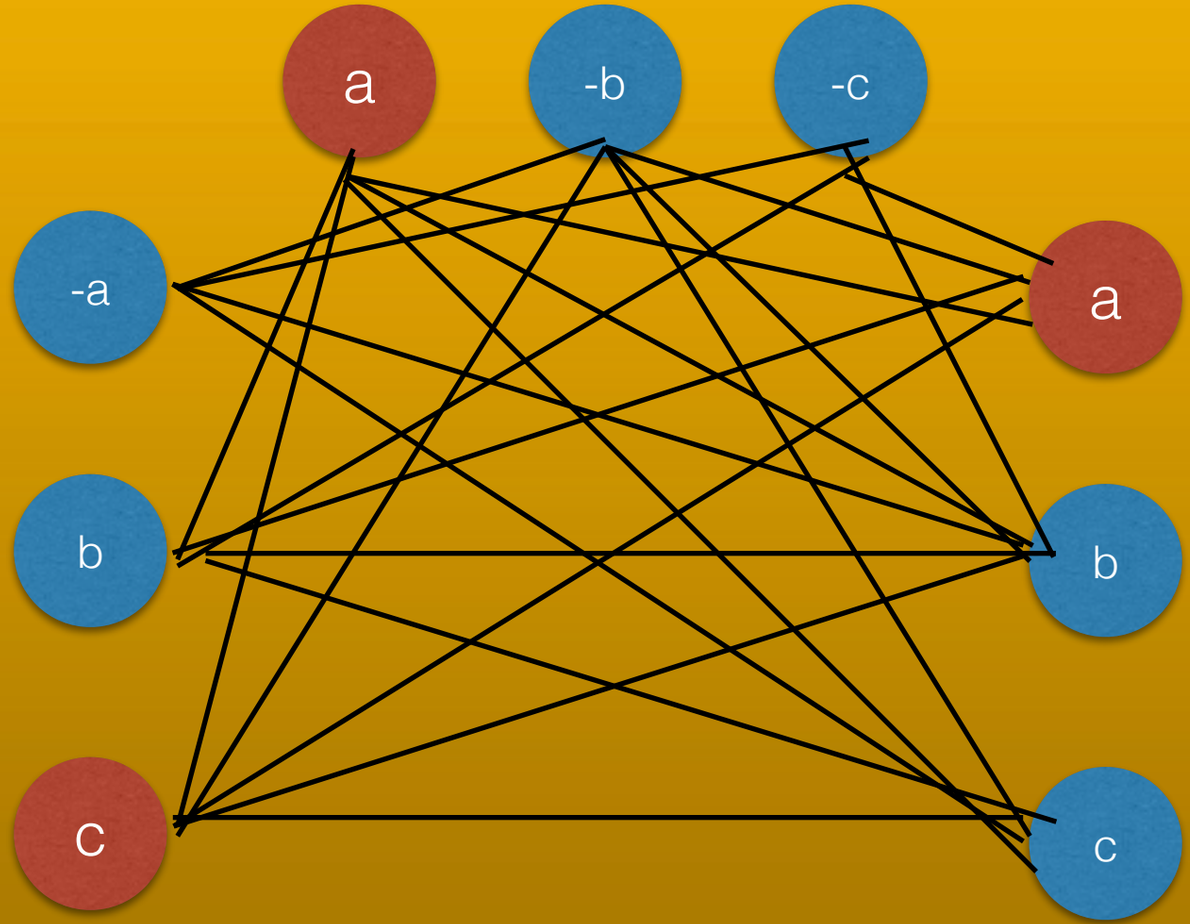
3-SAT nya satisfiable



misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

node yang dipilih CLIQUE <=> literal yang valuenya true

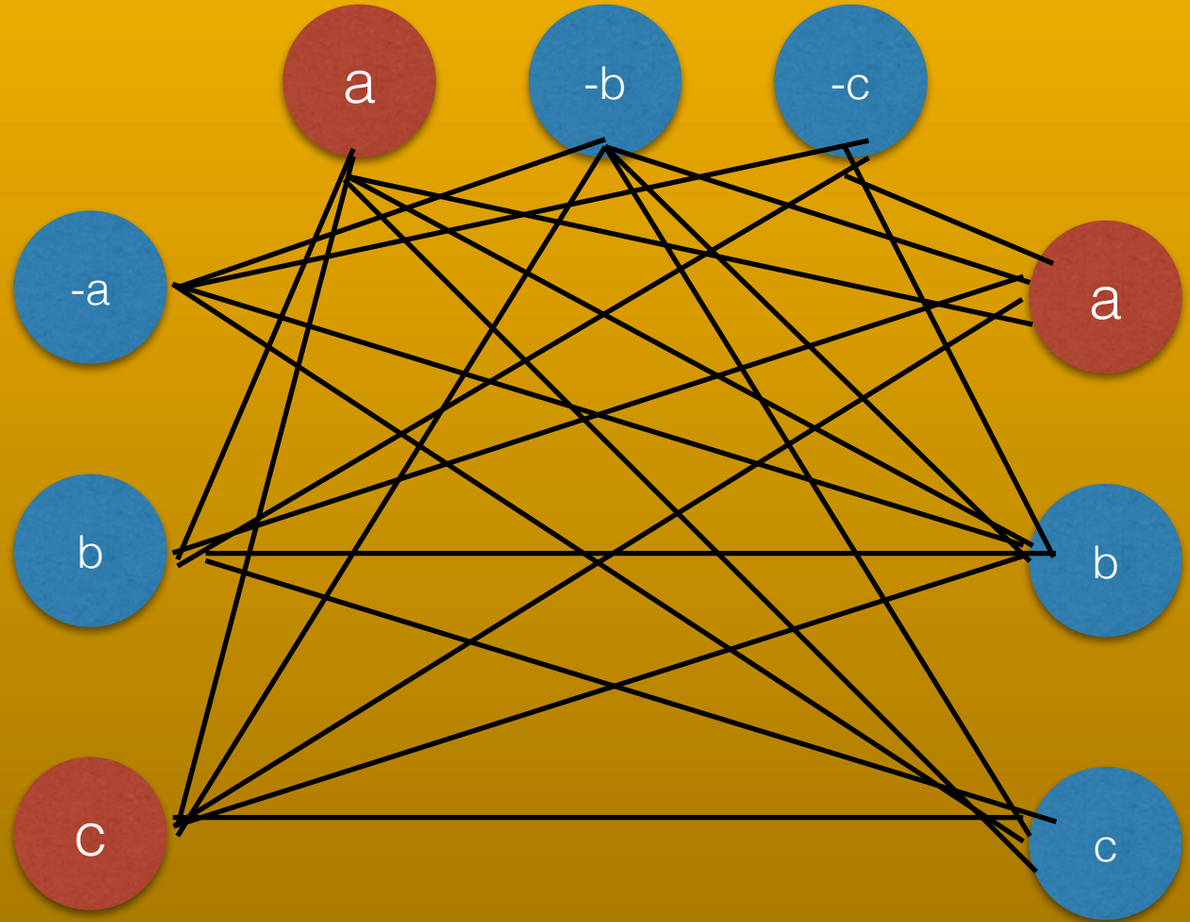


misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

CLIQUE pasti pilih vertex dari clause yang berbeda

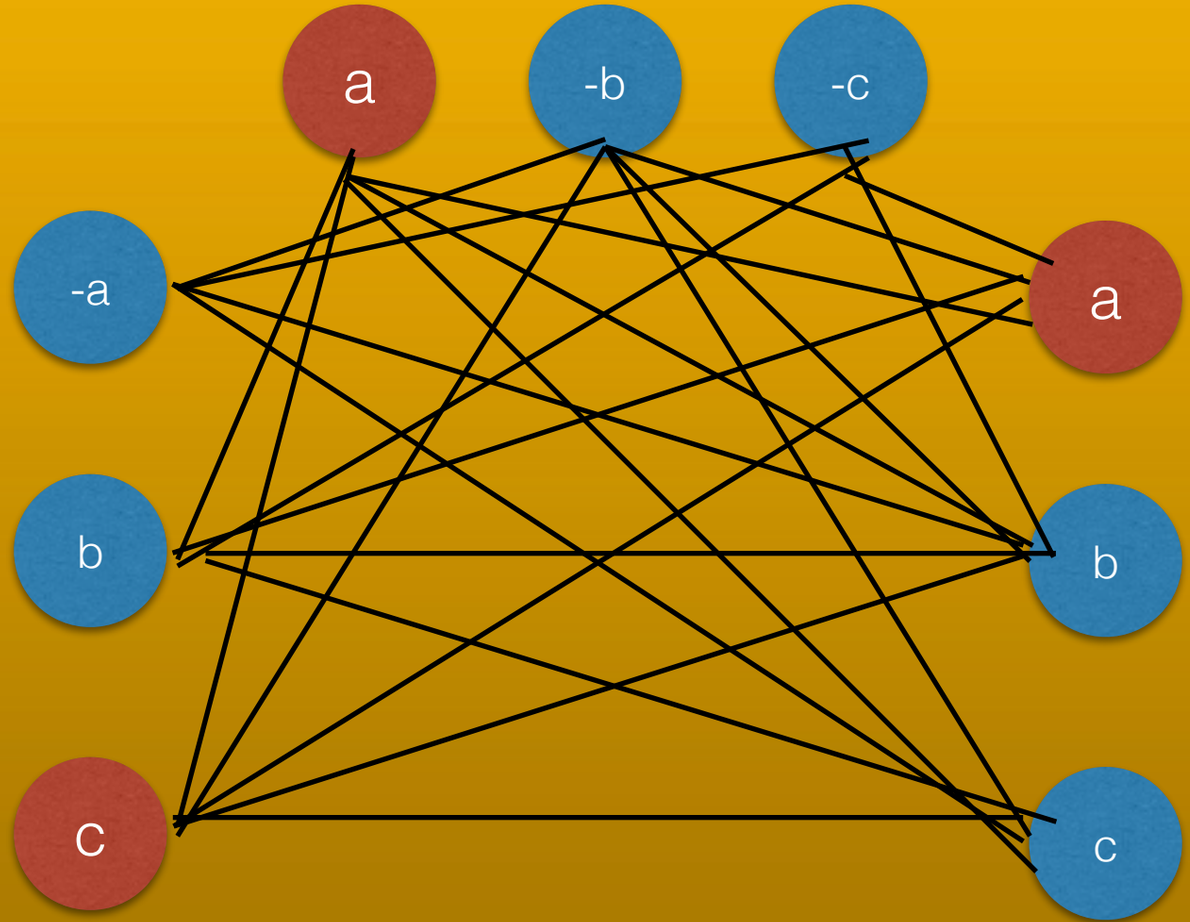
proof : karena dua vertex dari clause yang sama ga ada edge



misal :

(a v -b v -c) ^ (-a v b v c) ^ (a v b v c)

node yang dipilih ga bakal kontradiktif
x dan -x ga ada edge



proven :

kalo kita bisa solve MAX-CLIQUE in polynomial time, kita bisa solve 3-SAT in polynomial time

karena 3-SAT NP-hard,

MAX-CLIQUE juga NP-hard

terus gimana dong kalo ketemu soal NP-hard?

tips 1 : cek constraint

kalo $N \leq 16$, yaudah hajar solusi eksponensial,
otherwise, cari konstrain2 lain selain N yang mungkin bisa membantu

SUBSET SUM

dikasih array A, tentuin apakah lu bisa ambil subset dari A yang jumlahnya K

$1 \leq |A|,K \leq 1000$

SUBSET SUM itu NP-hard

3-SAT ≤p NP-hard

Construction. Given 3-SAT instance Φ with n variables and k clauses, form 2n + 2k decimal integers, each of n+k digits, as illustrated below.

Claim. Φ is satisfiable iff there exists a subset that sums to W.

Pf. No carries possible.

$$C_1 = \bar{x} \vee y \vee z$$
$$C_2 = x \vee \bar{y} \vee z$$
$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

	x	y	z	C_1	C_2	C_3	
x	1	0	0	0	1	0	100,110
¬ x	1	0	0	1	0	1	100,001
y	0	1	0	1	0	0	10,000
¬ y	0	1	0	0	1	1	10,111
z	0	0	1	1	1	0	1,010
¬ z	0	0	1	0	0	1	1,101
dummies to get clause columns to sum to 4	0	0	0	1	0	0	100
	0	0	0	2	0	0	200
	0	0	0	0	1	0	10
	0	0	0	0	2	0	20
	0	0	0	0	0	1	1
	0	0	0	0	0	2	2
W	1	1	1	4	4	4	111,444

SUBSET SUM

dikasih array A, tentuin apakah lu bisa ambil subset dari A yang jumlahnya K

$1 \leq |A|, \mathbf{K \leq 1000}$

udah tau lah ya solusinya apa

tips 2 : cek special case soalnya

given S = first N fibonaci number
{1,1,2,3,…}
determine whether lu bisa bagi set S sedemikian sehingga jumlahnya sama

PARTITION-SUM itu NP-hard

SUBSET-SUM ≤p PARTITION-SUM

SUBSET-SUM
dikasih array A dan cari subset yang totalnya K

<=>

PARTITION-SUM
bikin array A + (K - (sum of all elements in A - K))

SUBSET-SUM

dikasih array A dan cari subset yang totalnya K

PARTITION-SUM

bikin array A + (K - (sum of all elements in A - K))

SUBSET-SUM

A : {1,2,**3**,4,**5**} K = 8

K - (1 + 2 + 3 + 4 + 5 - K) = 8 - (15 - 8) = 1

PARTIITION-SUM

A : {1,2,**3**,4,**5**,1}

PARTITION-SUM itu NP-hard

so?

OBSERVASI !!!

fibonaci number
kalo N itu partitionable, maka N + 3 juga partitionable

$$A' = A + (f(N+1) + f(N+2))$$
$$B' = B + f(N+3)$$

$$A = B$$
$$f(N+1) + f(N+2) = f(N+3)$$
$$A' = B'$$

OBSERVASI !!!

N = 2 partitionable
A = {1}, B = {1}

N = 3 partitionable
A = {1,1}, B = {2}

N % 3 == 2 || N % 3 == 0 partitionable for all N

OBSERVASI !!!

N % 3 == 1

totalnya pasti ganjil

fibo = {1,1,2,3,5,8,...}

prefix_sum_fibo = {**1**,2,4,**7**,12,20,**33**}

OBSERVASI !!!

```
int main()
{
   int n;
   cin >> n;
   puts(n % 3 == 1 ? “no” : “yes”);
}
```

YEAY

IOI 2014 - Friend

dikasih graph, tiap node punya weight

lu mo ambil beberapa node tapi gak boleh ada dua node yang lu ambil yang adjacent

lu mau total semua weight dari node2 yang lu pilih maksimum

MAX INDEPENDENT SET

3-SAT ≤p MAX INDEPENDENT SET

3 Satisfiability Reduces to Independent Set

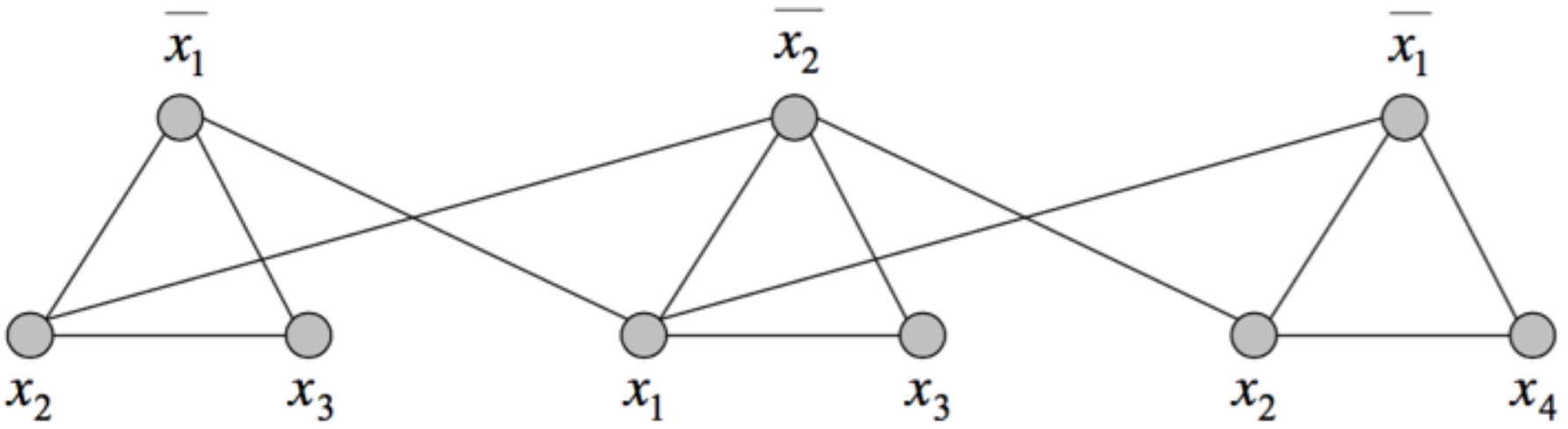
Claim. 3-SAT $\leq_p$ INDEPENDENT-SET.

Pf. Given an instance Φ of 3-SAT, we construct an instance (G, k) of INDEPENDENT-SET that has an independent set of size k iff Φ is satisfiable.

Construction.

- G contains 3 vertices for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

G



k = 3

$$\Phi = \left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(\overline{x_1} \vee x_2 \vee x_4\right)$$

so, MAX INDEPENDENT SET is NP-hard

kita harus go back to problem statement dan cari special casenya

The people are added to the network in n stages, which are also numbered from 0 to $n - 1$. Person i is added in stage i. In stage 0, person 0 is added as the only person of the network. In each of the next $n - 1$ stages, a person is added to the network by a *host*, who may be any person already in the network. At stage i ($0 < i < n$), the host for that stage can add the incoming person i into the network by one of the following three protocols:

- *IAmYourFriend* makes person i a friend of the host only.
- *MyFriendsAreYourFriends* makes person i a friend of *each* person, who is a friend of the host at this moment. Note that this protocol does *not* make person i a friend of the host.
- *WeAreYourFriends* makes person i a friend of the host, and also a friend of *each* person, who is a friend of the host at this moment.

The people are added to the network in n stages, which are also numbered from 0 to $n - 1$. Person i is added in stage i. In stage 0, person 0 is added as the only person of the network. In each of the next $n - 1$ stages, a person is added to the network by a *host*, who may be any person already in the network. At stage i ($0 < i < n$), the host for that stage can add the incoming person i into the network by one of the following three protocols:

- *IAmYourFriend* makes person i a friend of the host only.
- *MyFriendsAreYourFriends* makes person i a friend of *each* person, who is a friend of the host at this moment. Note that this protocol does *not* make person i a friend of the host.
- *WeAreYourFriends* makes person i a friend of the host, and also a friend of *each* person, who is a friend of the host at this moment.

subtask	% of points	n	`confidence`	protocols used
1	11	$2 \le n \le 10$	$1 \le$ `confidence` $\le 1{,}000{,}000$	All three protocols
2	8	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only MyFriendsAreYourFriends
3	8	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only WeAreYourFriends
4	19	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only IAmYourFriend
5	23	$2 \le n \le 1{,}000$	All `confidence` values are 1	Both MyFriendsAreYourFriends and IAmYourFriend
6	31	$2 \le n \le 100{,}000$	$1 \le$ `confidence` $\le 10{,}000$	All three protocols

kita coba yang ini dulu

subtask	% of points	n	`confidence`	protocols used
1	11	$2 \le n \le 10$	$1 \le$ `confidence` $\le 1{,}000{,}000$	All three protocols
2	8	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only MyFriendsAreYourFriends
3	8	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only WeAreYourFriends
4	19	$2 \le n \le 1{,}000$	$1 \le$ `confidence` $\le 1{,}000{,}000$	Only IAmYourFriend
5	23	$2 \le n \le 1{,}000$	All `confidence` values are 1	Both MyFriendsAreYourFriends and IAmYourFriend
6	31	$2 \le n \le 100{,}000$	$1 \le$ `confidence` $\le 10{,}000$	All three protocols

yang keatasnya gampang lah ya

maximum unweighted independent set on bipartite graph

|A| + |B| - bipartite matching

solusi penuh

kita process querynya dari belakang

tiap process, kita “hapus” node barunya

MyFriendsAreYourFriends(A->B)

solusi yang bisa pake A <=> bisa pake B
ambil dua2nya ato tidak sama sekali

w(A) = w(A) + w(B)

WeAreYourFriends(A->B)

cuma bisa ambil A ato B, ga ngefek ke pengambilan node2 lain

w(A) = max(w(A),w(B))

IAmYourFriend(A->B)

ini tricky. assume B diambil.
ans += w(B).

however, kalo akhirnya kita ambil A,
ada cost buat apus B.

w(A) = w(A) - w(B)

solusi akhir

ans = ans + w(0)

contoh2 konstrain2 aneh umum yang mengubah segalanya

graph :

1. satisfies triangle inequality
2. planar
3. bipartite

EOF

Q&A?