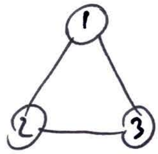
May 6

k-colorability (k-coloring)

$G = (V, E)$

Def: A k-coloring of G if $c : V \to \{1, \dots, k\}$
s.t. $\forall (u, w) \in E$, $c(u) \neq c(w)$



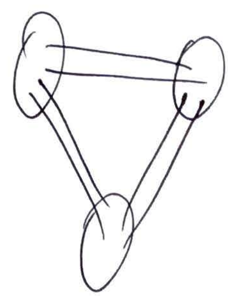
← 3-colorable but not 2-colorable

Def: G is k-colorable if $\exists$ a k-coloring for it.

Def: (k-colorability / k-coloring problem)
input: G, k
output: T if G is k-colorable
F o/w

---

k-partite
$k = 3$



Claim 1: k-colorability $\in NP$
Claim 2: 2-colorability $\in P$

THM: 3-SAT $\leq_p$ 3-colorability $\leq_p$ k-colorability ($k \geq 3$)

$\Longrightarrow$ 3-coloring is NP-complete

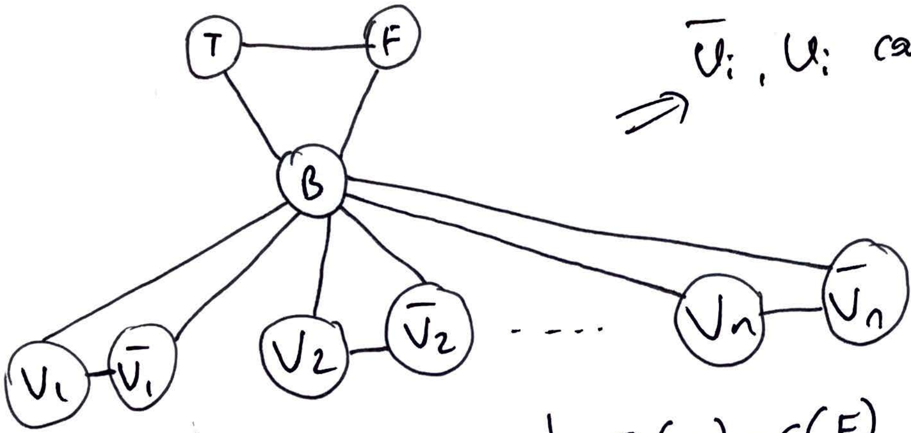
Claim 1

Goal: given any 3-SAT formula $C_1, \dots, C_m$ on $X = \{x_1, \dots x_n\}$
compute a graph G (s.t. $|G| = poly(n, m)$)
s.t. G is 3-colorable $\iff$ $C_1, \dots, C_m$ is satisfiable

Step 1: One node $v_i \equiv x_i$

$$\bar{v}_i \equiv \bar{x}_i$$

$\forall i$: v_i — $\bar{v}_i$ $\Rightarrow$ v_i & $\bar{v}_i$ need to be assigned diff colors



$\Rightarrow$ $\bar{v}_i$, v_i cannot have B color

Claim: If $c(v_i) = c(T)$ $\Rightarrow c(\bar{v}_i) = c(F)$ | $c(v_i) = c(F)$ $\Rightarrow c(\bar{v}_i) = c(T)$

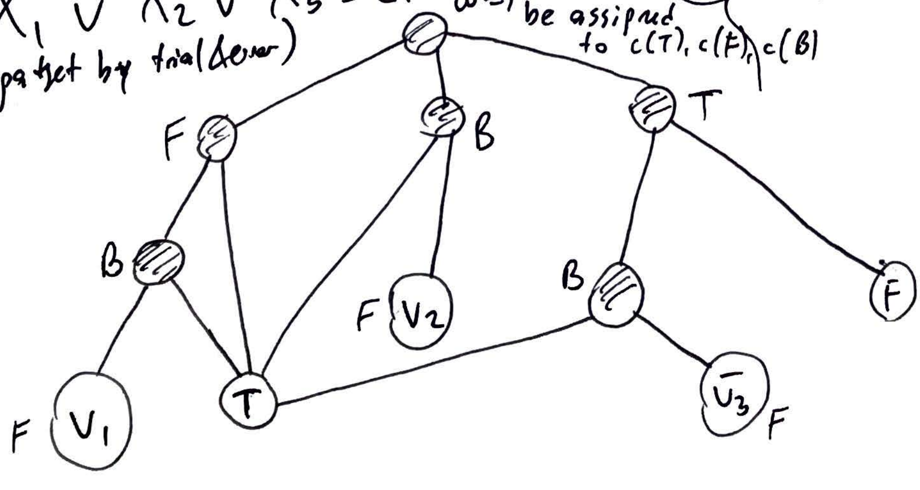
Claim: 3-coloring of $\Longleftrightarrow$ valid assignments to $x_1, \dots, x_n$

→ Encode the clauses $C_1, \dots, C_m$ in this graph

step 2) (adding more vertices + edges in G)

e.g. $x_1 \vee x_2 \vee \bar{x}_3 = C_i$ w⇒ cannot be assigned to c(T), c(F), c(B) — Gad$_i$

(Find a gadget by trial & error)



Claim: 3-coloring of $Gad_i \Rightarrow$ at least one of the literals in Gad_i is assigned $c(T)$

Pf(idea): Assume all $v_1, v_2, \bar{v_3}$ are assigned $c(F)$

$\Rightarrow w$ cannot be assigned any of $c(T), c(F), c(B)$

$\Rightarrow$ we do not have a valid 3-coloring!

Final redux: Given $C_1, \ldots, C_m$ on X

1. Compute G from X
2. Add Gad_i to G $\forall$ clauses $C_i \Rightarrow G'$
3. Output G' / Feed G' to an algo for 3-coloring

Claim: $C_1, \ldots, C_m$ is satisfiable $\Leftrightarrow G'$ is 3-colorable

---

Show: Problems that are harder to solve than NP-Complete

HALTING PROBLEM

(P, I) is pair of strings

Input: Program P, Input I for P

Output: Yes if P terminates on I

No o/w

Q: Does there $\exists$ an algo to solve the halting problem?

in any finite time

THM: No!

Pf: By contradiction

Assume $\exists$ algo h s.t. $h(P, I) = \begin{cases} \text{Yes} & \text{if } P(I) \text{ terminates} \\ \text{No} & \text{o/w} \end{cases}$

```
def c(x):
    if h(x,x) = Yes:
        loop forever
    else
        return
```

Consider the call $c(c)$

Case 1: $h(c,c) = \text{Yes} \Rightarrow c(c)$ loops forever

Case 2: $h(c,c) = \text{No} \Rightarrow c(c)$ terminates

$\Rightarrow$ contradicts h solving the halting problem!