Dec 9

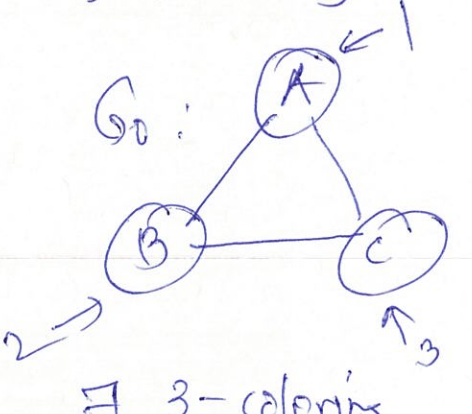
k-colorability (k-coloring)

$G = (V, E)$

Def: k-coloring $C : V \to \{1, .., k\}$

s.t $\forall (u, w) \in E \quad C(u) \neq C(w)$

Def (k-colorability / k-coloring problem)



Input: $G = (V, E)$; k

o/p: $\begin{cases} 1 & \text{if } G \text{ is k-colorable} \\ & (\exists \text{ a k-coloring of } G) \\ 0 & \text{o/w} \end{cases}$

∃ 3-coloring
no 2-coloring

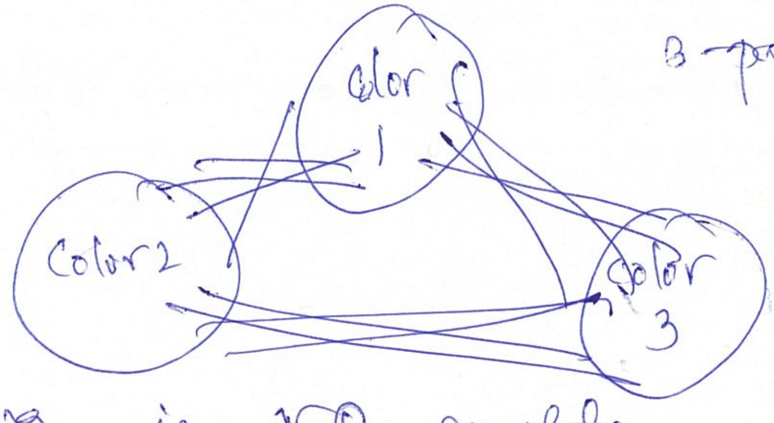
E.g: $G_0 ; 3$ ✓ $\quad G_0 ; 2$ ✗

Claim 1: k-colorability $\in NP$

Pf idea: witness : $C : V \to \{1, .., k\}$

3-colorable graph

k-colorable graph
= k-partite graph

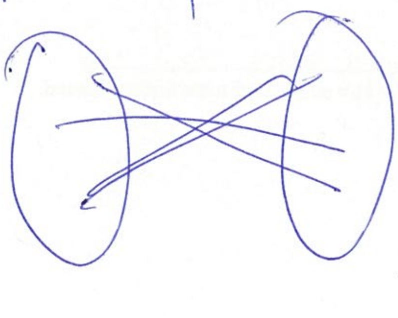


3-partite graph.

Skip this on Dec 9

THM: 3-coloring is NP-complete.

2-colorability $\in P$



Bipartiteness

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

↑ (book)

$\Rightarrow$ 3-colorability is NP-complete

Pf (idea) Goal: Given a 3-SAT formula

$\Phi = C_1, \ldots, C_m$ on $X = \{x_1, \ldots, x_n\}$

in poly time (poly(n,m)) compute a graph G_Φ s.t.

$|\Phi| = O(m+n)$ with (i) $|G_\Phi| \leq$ poly(n,m)

Φ is satisfiable $\Longleftrightarrow$ G_Φ is 3-colorable.

```
Algo Sat(Φ)
    1. Convert Φ to G_Φ
    2. b ← Algo3-coloring(G_Φ)
    3. Return b
```

Idea: Use gadgets again ("Δ" like but more complicated)

Step 1: G_0 $2n+3$ vertices $v_1, \ldots, v_n$ $\quad v_i \equiv x_i$

$\overline{v}_1, \ldots, v_n$ $\quad \overline{v}_i \equiv \overline{x}_i$

3 special vertices T, F, B

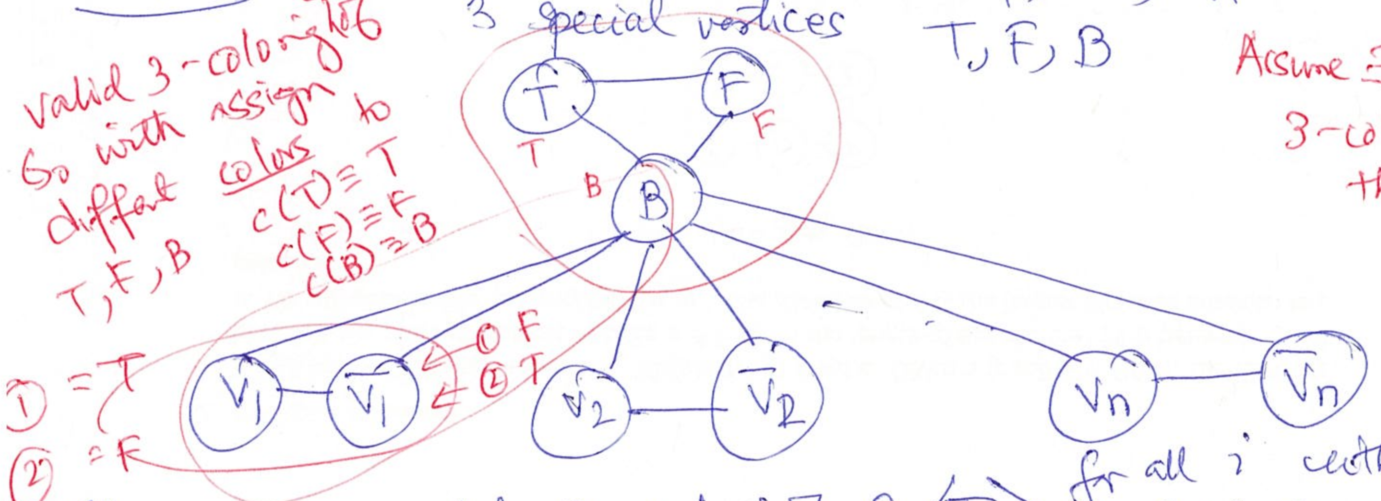
valid 3-coloring of G_0 with assign to different colors T, F, B

$c(T) \equiv T$

$c(F) \equiv F$

$c(B) \equiv B$

Assume ∃ a 3-coloring of this graph



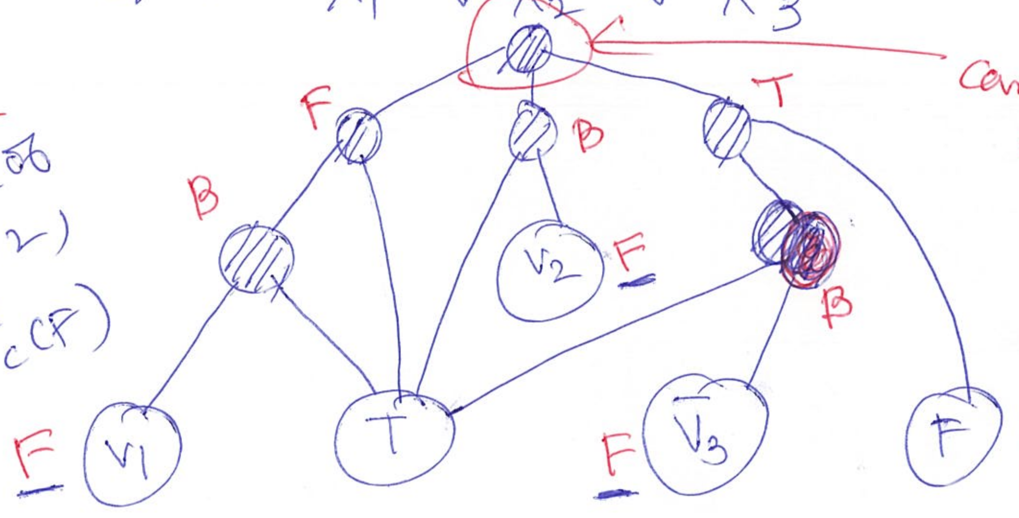
Claim: ∃ a valid 3-coloring of G $\Longleftrightarrow$ for all i either $c(v_i) = T \wedge c(\overline{v}_i) = F$ OR $c(v_i) = F$, $c(\overline{v}_i) =$

$\Rightarrow$ 3-coloring of G $\Longleftrightarrow$ an assignment to $x_1, \ldots, x_n$

Step 2: Encode each clause C_i with a gadget Gad_i that we "add" to G_0

E.g.: $C_i = X_1 \vee X_2 \vee \overline{X_3}$

▨ nodes unique to ~~to~~ Gad_i



Say not i.e all of $v_1, v_2, \overline{v_3}$ are assigne C(F)

Cannot be assigned C(T) C(F) or C(B) ⇒ no valid 3-coloring

Claim! In a valid 3-coloring of Gad_i, (+G_0) at least ~~v~~1, v_2 or $\overline{v_3}$ is assigned C(T)

$C_i = t_1 \vee t_2 \vee t_3$ ↑ t_2 ↑ t_2 ↑ t_3

(Book detail)

Formal redux! $\Phi = C_1, \ldots, C_m$ on $X_1, \ldots, X_n$

1. Compute G_0 on $\left\{ \begin{matrix} v_1, \ldots, v_n \\ \overline{v_1}, \ldots, \overline{v_n} \end{matrix} \right\}$ + T, B, F
2. Add Gad_i to G_0

→ Results is G_Φ

THM: Φ is satisfiable ⇔ G_Φ is 3-colorable (pf in book)

↑ Final exam until here

Next up: Problems "harder" than NP-complete

HALTING problem

P, I as strings

Input: Program code P, valid input I for P

Output: $\begin{cases} 1 & \text{if } P(I) \text{ terminates} \\ 0 & \text{o/w} \end{cases}$

Q: Does $\exists$ an algo to solve the ~~halt~~ HALTING problem.

↑ algo just needs to terminate.

THM: NO!

Pf: By contradiction

Assume $\exists$ a algo h that solves HALTING

$$\forall P, I \qquad h(P, I) = \begin{cases} 1 & \text{if } P(I) \text{ terminates} \\ 0 & \text{o/w} \end{cases}$$

Program ← string

```
def C(X):
    If h(X,X) = 1:
        loop forever
    else
        return.
```

Call $C(C)$

Case 1: $h(C,C) = 1$ — but as per def of C, C loops forever on C

↳ C terminates on C (by def of h)

Case 2: $h(C,C) = 0$ — but as per def of C, C terminates on C

↳ C loops forever on C (by def of h)

$\Longrightarrow$ NO such h exists! ∎