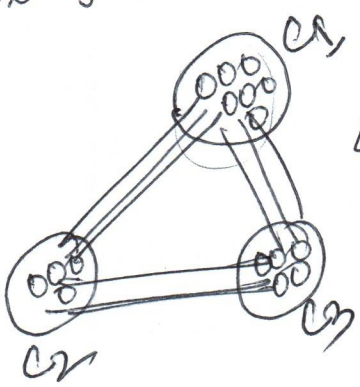
claim: K-colorability $\in NP$.

↳ has an efficient verifier

witness: $c: V \to \{1, \dots K\}$

e.g. c_0: $A \to 1$, $B \to 2$, $C \to 3$

e.g. 3-colorability.



← 3partite

K-colorability $\equiv$ K-partite.

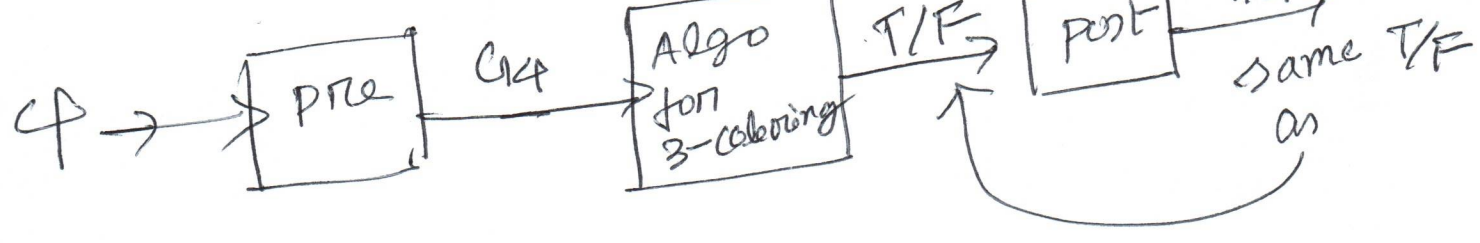
2-colorability $\in P$ → Uses the concept of bipartiteness → section 3 in KT.

THM: 3-SAT $\leq_P$ 3-colorability $\leq_P$ K-colorability (Book)

claim + THM ⇒ 3-colorability is NP-complete.

May 12

→ Given a 3-SAT formula $\varphi = c_1, \dots, c_m$ on $X = \{x_1, \dots, x_n\}$, compute G_φ with $|G_\varphi| = \text{poly}(m, n)$ s.t. φ is satisfiable ⇔ G_φ is 3-colorable.



Algo SAT (φ):

1. Convert φ to G_φ
2. $b \leftarrow$ Algo-3-color(G_φ)
3. return b

---

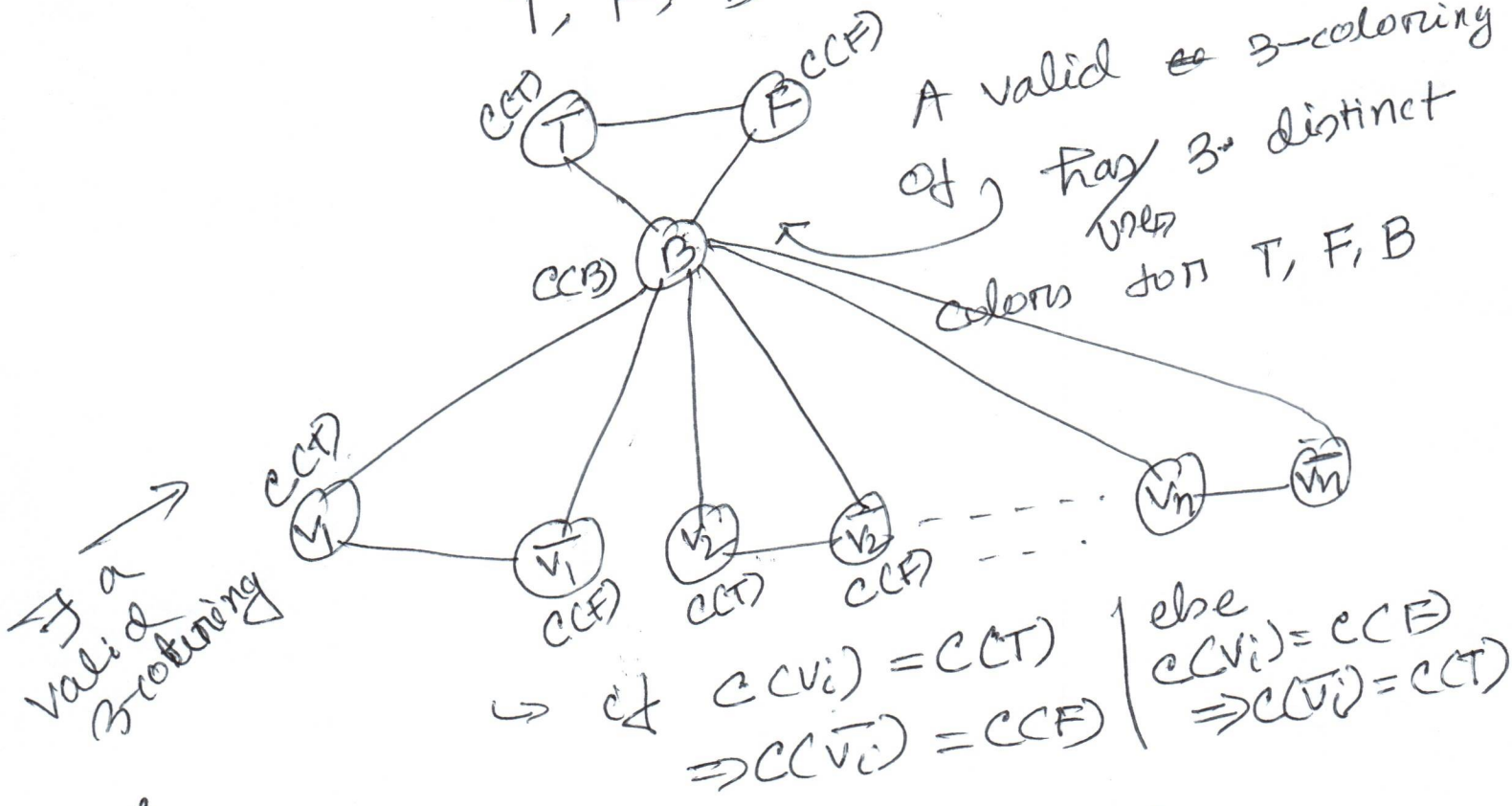
idea: Use a gadget.

Step 1: G_0 of $2n+3$ vertices.

$v_1, \ldots, v_n,$
$\overline{v_1}, \ldots, \overline{v_n}$

$v_i = x_i$
$\overline{v_i} = \overline{x_i}$

T, F, B



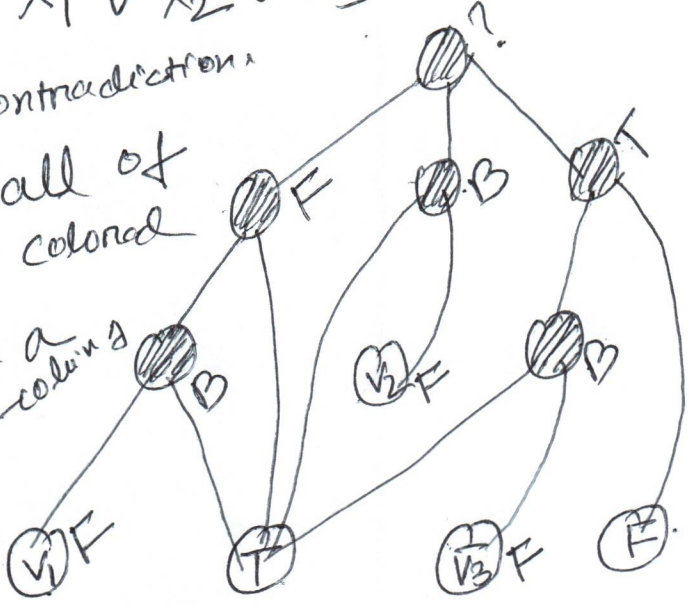
If a valid 3-coloring

↪ if $c(v_i) = c(T)$ ⇒ $c(\overline{v_i}) = c(F)$ | else $c(v_i) = c(F)$ ⇒ $c(\overline{v_i}) = c(T)$

↪ Claim: a valid 3-coloring of G_0
$\Leftrightarrow \exists$ an assignment to $x_1, \ldots x_n$
s.t. if $c(v_i) = c(T)$ $x_i \leftarrow 1$
else $x_i \leftarrow 0$

Step 2°: Encode each clause C_i with a gadget G_{adi} that we will add to G_0.

$C_i = x_1 \vee x_2 \vee \overline{x_3}$



● ← unique to each clause

Pf: By contradiction:
Assume all of $v_1, v_2, \overline{v_3}$ colored C(F)
⇒ Not a valid 3-coloring

claim: In a valid 3-coloring of G_{adi}, at least one of $v_1, v_2, \overline{v_3}$ is colored C(T)

↳ claim: ∃ A valid 3-coloring of G_{adi} ⇔ (Book) at least one literal in G_{adi} is colored with C(T).

Redux: Given $\varphi = C_1, \ldots, C_m$ on $x_1, \ldots, x_n$

1. Compute G_0 on $\{v_1, \ldots, v_n, \overline{v_1}, \ldots, \overline{v_n}, T, F, B\}$ as above

2. Add G_{adi} to G_0 $\forall C_i$
   Let the resulting graph G_φ

3. return the output of Algo-3-color(G_φ).

* problems that are harder than NP-complete *

HALTING Problem:

Input: a program/code P, and a valid input I (for P)

(P, I) ~ pair of strings

Output: 1 if P halts/terminates on I
0, o/w

Q: ∃ an algo to solve the Halting problem? (in finite time)

THM: NO!

Pf. (by contradiction)
Assume ∃ an algo h that solves the Halting problem.

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

```
def C(x):  ← a string
    if h(x, x) = 1:
        loop forever
    else
        terminate/return
```

Consider the call C(c)

Case 1: if h(c, c) = 1 $\Rightarrow$ C(c) loops forever.
$\Rightarrow$ c terminate one

Case 2: if h(c, c) = 0 $\Rightarrow$ C(c) terminates.
$\Rightarrow$ c loops forever one

Undecidability