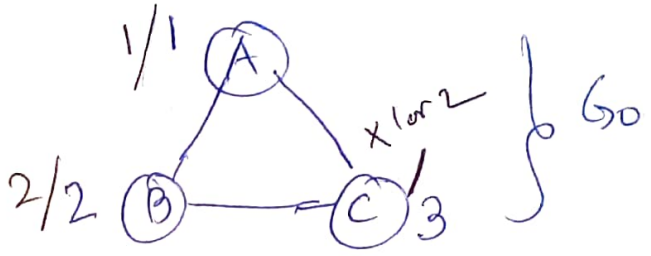
Dec 8

k-colorability



k-coloring of G is

$c: V \to \{1, \ldots, k\}$

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

Def (k-colorability / k-coloring)

i/p: $G = (V, E)$; k

o/p: 1 if G is k-colorable ← ∃ a k-coloring

0 o/w ← no k-coloring ∃

Ex: G_0; 3 ✓ G_0; 2 ✗

Claim 1: k-colorability $\in NP$

see book

THM: 3-SAT $\leq_p$ 3-colorability $\leq_p$ k-colorability

$\Rightarrow$ 3-colorability is NP-complete.

Goal: Given 3-SAT formula $\Phi = C_1, \ldots, C_m$ on $X = \{x_1, \ldots, x_n\}$

in poly time compute a graph G_Φ s.t

see book

$|G_\Phi| = \text{poly}(n, m)$ with

Φ is satisfiable $\iff$ G_Φ is 3-colorable

Algo SAT (Φ)

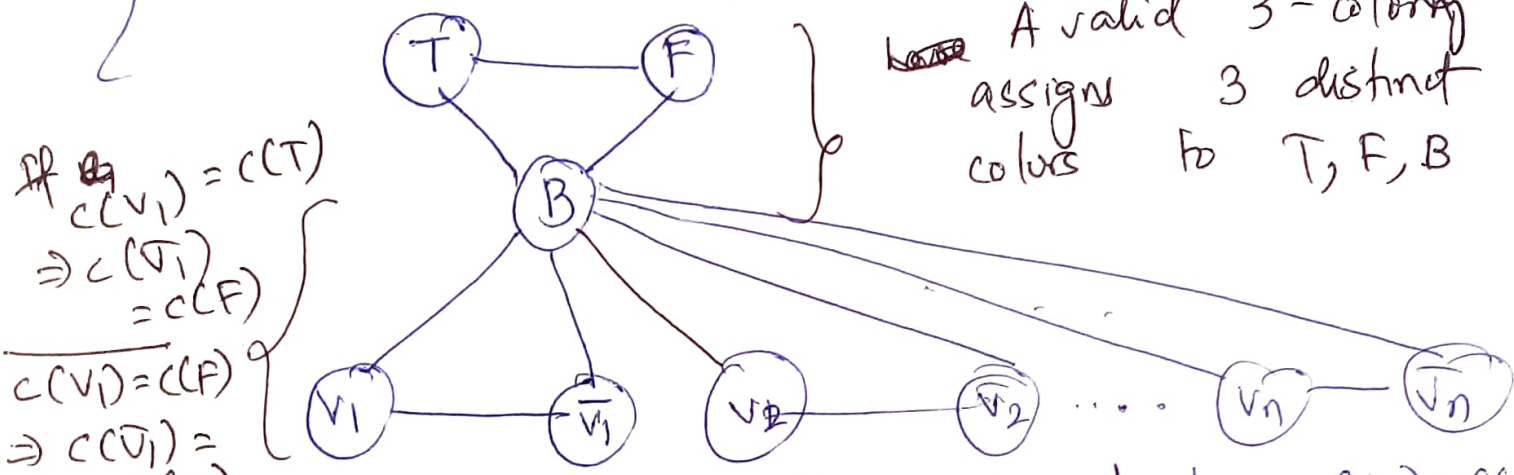
1. Convert Φ to G_Φ s.t
2. $b \leftarrow$ Algo 3-color(G_Φ)
3. return b.

Step 1: G_0: $2n$ vertices $v_1, \dots, v_n$, $\bar{v}_1, \dots, \bar{v}_n$

$v_i \equiv x_i$

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

3 special nodes T, F, B



A valid 3-coloring assigns 3 distinct colors to T, F, B

If $c(v_1) = c(T)$
$\Rightarrow c(\bar{v}_1) = c(F)$

$c(v_1) = c(F)$
$\Rightarrow c(\bar{v}_1) = c(T)$

Claim: $\exists$ a 3-coloring $\iff$ If $c(v_i) = c(T) \Rightarrow c(\bar{v}_i) = c(F)$ | else $c(v_i) = c(F) \Rightarrow c(\bar{v}_i) = c(T)$

↳ 3 coloring for G_0 $\iff$ $\exists$ an assignment to $x_1, \dots, x_n$

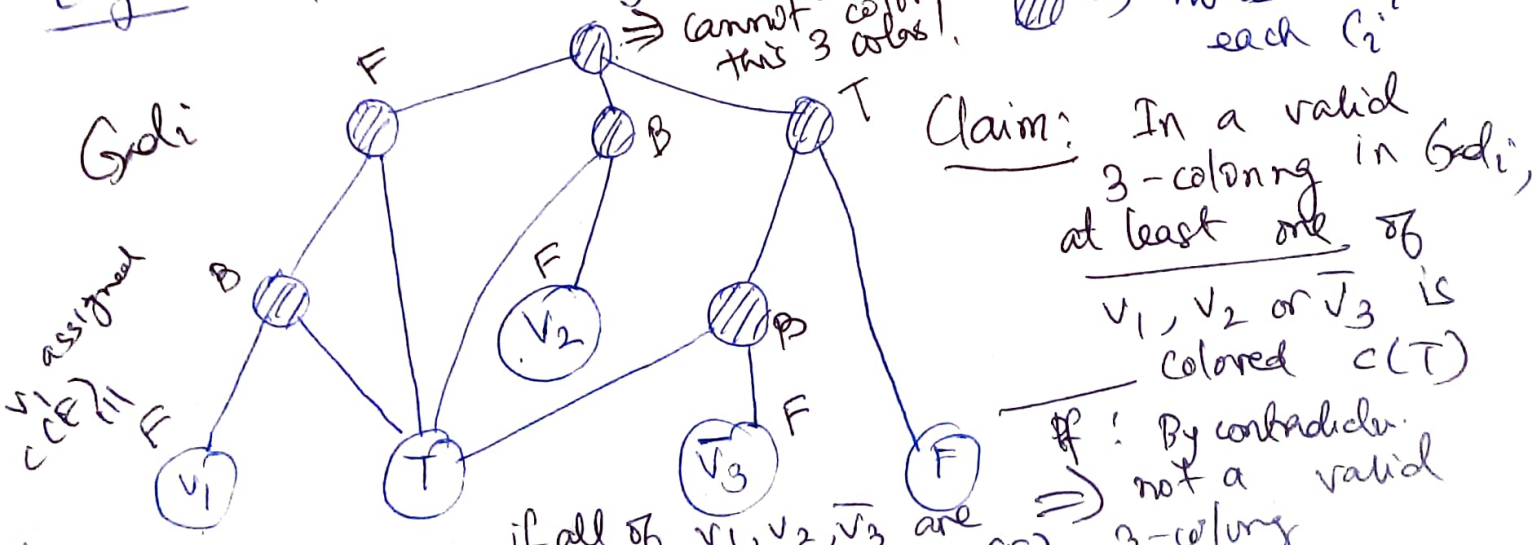
↳ If $c(v_i) = c(T) \Rightarrow x_i \leftarrow 1$

else $x_i \leftarrow 0$

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

E.g. $x_1 \vee x_2 \vee x_3 \equiv C_i$

$\Rightarrow$ cannot color this 3 colors!



Gad_i

nodes unique to each C_i

Claim: In a valid 3-coloring in Gad_i, at least one of v_1, v_2 or $\bar{v}_3$ is colored $c(T)$

If all of $v_1, v_2, \bar{v}_3$ are colored $c(F)$ $\Rightarrow$ not a valid 3-coloring

v_1 assigned $c(F)$

If: By contradiction

Claim: $\exists$ a coloring of Gad_i $\Leftrightarrow$ at least one literal in Gad_i is assigned C(T)

Pf: (book)

Formal ~~red~~ redux: Given $\Phi = C_1, \ldots, C_m$ on $x_1, \ldots, x_n$

1. Compute G_0 on $\{ v_1, \ldots, v_n, \bar{v}_1, \ldots, \bar{v}_n, T, F, B \}$ as above
2. Add Gad_i to G $\forall$ clauses C_i
   ↳ Let G_Φ be the resulting graph
3. Return output of Algo-3-color(G_Φ)

Thm/claim: Φ is SAT $\Leftrightarrow$ G_Φ is 3-colorable

Next up: Problems that are "harder" than NP-complete

HALTING problem:

Input: Program/code P, valid input I (for P) — (P, I) as pair of strings

o/p: 1 if P terminates on I
0 o/w

Q: Does $\exists$ an algo to solve the halting problem ← terminates in finite time.

Thm: NO!

Pf: By contradiction.
Assume $\exists$ an algo h that solves HALTING
$\forall$ P, I $h(P, I) = \begin{cases} 1 & \text{if } P(I) \text{ terminates} \\ 0 & \text{o/w} \end{cases}$

```
def c(x):                    ← string
    if h(x,x) = 1:
        loop forever
    else:
        return.
```

valid as c is a string

Consider the call $c(\langle c \rangle)$

Case 1: $h(\langle c, c \rangle) = 1 \Rightarrow$ $c(\langle c \rangle)$ loops ~~forever~~ forever

$\Rightarrow$ c terminates on c

Case 2: $h(\langle c, c \rangle) = 0 \Rightarrow$ $c(\langle c \rangle)$ terminates.

$\Rightarrow$ c loops forever on c

$\Rightarrow$ contradicts the claim that h solves the halting problem ∎