Disc Week 9

1. Intro to P vs. NP

2. Reductions
   Note: Next Week We'll Do Sample Final Problems

K-Color

Input: Graph $G = (V, E)$, undirected, planar
Output: Yes/No, can $G$ be colored with $k$ colors?
(i.e. Vertices assigned a of $k$ colors; no edge $(u, v)$ has same color on $u$ and $v$)

• We've seen how to solve 2-color efficiently with BFS. 1 color is also solvable easily, efficient
• Can you suggest an algorithm for 3-Color?
   IF SO TELL ME

• Note: 4-color for planar graphs just recently solved by 4 Color Theorem, Ans: Always Yes for any planer $G$.

Evan's Country: Map Color: 8

K Efficient Alg?
1 2 3 4+ ✗ ✗ ✗ ✗
* 3-color is a member of a group of very hard problems called NP-complete. What is that?

* **Defn**: A search problem is any problem that has a checking algorithm \( C \), that takes input an instance \( I \) and solution \( S \) where

\[
C(I, S) = \begin{cases} 
\text{true} & \text{if } S \text{ is a solution to } I \\
\text{false} & \text{if } S \text{ is not a solution to } I 
\end{cases}
\]

and \( C \) runs in time polynomial in \( |I| \).

**Ex**: (3-Color)

\( I = \) a graph \( G \)

\( S = \) a potential 3-coloring of \( G \)

\( C = \) checks each edge to see if colors are the same in \( S \). Runs in \( O(|E|) \).

* **Defn**: NP is the class of all search problems.

  e.g. 3-Color, 2-Color, etc...

* **Defn**: P is the class of all search problems that can be solved in polynomial time, \( O(nc) \) for some \( c > 0 \) fixed.

  e.g. 2-Color.

* Note: By definition \( P \neq NP \).

**Big Question** - Does \( P = NP \)?

People think not, but nobody knows for sure...
Defn: A search problem is **NP-complete** if all other search problems reduce to it.

- What does reduce mean?

Defn: Problem A reduces to problem B if the following picture holds:

![Diagram showing reduction from A to B]

- Idea is A reduces to B if we can solve A efficiently using an efficient solution to B.

## Reductions

### 3SAT

Input: 3CNF formula \( \varphi \)

Output: Is there a satisfying assignment for \( \varphi \)? (Yes/No)

Example: \( \varphi = (x \lor y \lor z) \land (x \lor \neg y) \land (y \lor \neg z) \land (z \lor \neg x) \land (x \lor \neg y \lor \neg z) \)

Can we set \( x, y, z \) to make \( \varphi \) true? No, hard.
Independent Set

Input: Undirected $G = (V, E) \cup K$
Output: Does there exist $V' \subseteq V$ such that $\forall x, y \in V', (x, y) \notin E$ and $|V'| \geq K$.

e.g.

These two nodes are an independent set of size 2.

$\square$ is an ind. set of size 7.
$O$'s are """"""

Claim: 3SAT reduces to Independent Set.

Want:

\[
\begin{array}{c}
\text{3SAT instance } \varphi \\
\longrightarrow \text{Fulfil} \longrightarrow \text{Ind Set Solver} \prod_{S} \text{fail} \text{ SAT} \\
\text{No such } S \text{ exists} \\
\end{array}
\]

Idea: Construct a $G$ whose independent set is the truth assignment for $\varphi$

$\varphi = \text{as before in example}$

for each clause $B_k \Delta A$:

1. connect opposite literals
2. e.g. $(x \lor \neg y \lor \neg z)$

$\square\square\square\square$ etc.

$\square\square\square\square$
Note: Construction is poly-time since it takes $O(\#\text{clauses}^2)$.

Let $K_0$ be the # of clauses, then $O(\#\text{clauses}^2)$.

2 things to show we've solved 3SAT

1) If we find an ind. set of size at least $K_0$ in $G$, then $\varphi$ has an \textbf{invalid} assignment that make $\varphi$ TRUE.

   Since there are only $k$ clauses in $\varphi$ and
   each variable must be selected from each clause
   and $X, \bar{X}$ cannot be selected simultaneously, this
   is our truth assignment.
   e.g., if $X$ is on the ind. set, set $x = F$.

2) If there is no ind. set of size at least $k$ in $G$, then $\varphi$ is unsatisfiable.

   \textbf{Prove \textit{Contrapositive}}: If $\varphi$ is \textit{satisfiable}, then $G$ has
   an ind. set of size $\geq k$.

   Choose the one lit. $x$ in each clause that would satisfy the
   clause, be set to satisfy $\varphi$, and add to independent set.

   Thus must be $k$ of them since $\varphi$ is \textit{satisfiable}.

We've shown if we could solve ind. set efficiently, then we
could also solve 3SAT efficiently.