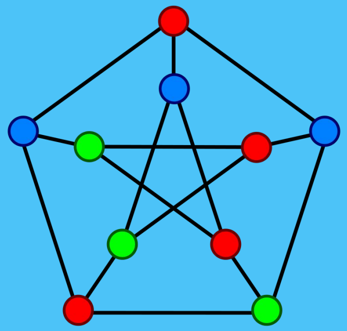
NP Problems

Week 7 AI Inspire Winter 2020

What is NP space?



Where does this come from?

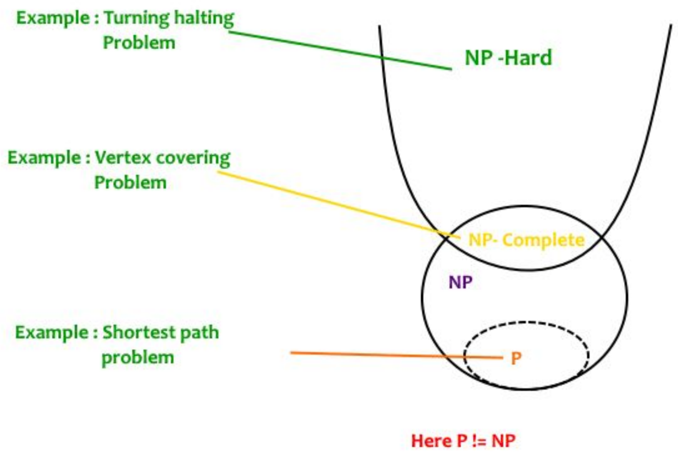
* Can every problem which can be checked/verified quickly by a computer also be solved quickly by a computer?
    * Ex: Finding prime #s:
        * checking if # is prime vs. actually finding prime #s

Another example...

* Rock piling problem ⇒ NP prob
    * Collection of rocks with diff masses
    * Want to make 2 towers with = masses
* Can easily verify if certain division of rocks work
    * Simply verify both towers have same mass by summing up, very efficient
* Very hard to solve
    * With 100 rocks calc how many combos there are
        * Even quantum computers won't be able to solve

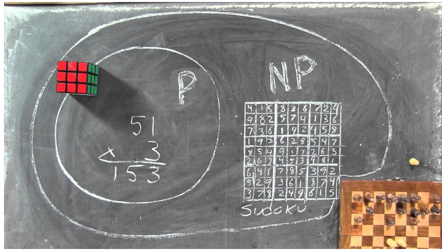
Classes of Problems

* Can classify problems into different types (P = Polynomial runtime, NP = Nondeterministic polynomial runtime, etc.)



P vs NP

* So far ⇒ have learned problems in P space
    * Can solve quickly
    * Can verify quickly
    * Except for a few like TSP
* Some problems fall in NP domain
    * Can verify quickly
    * Cannot always solve quickly
        * Only some problems part of NP are part of P which means they can be solved quickly
* No formal proof has been done to show that NP is harder than P in reality
* P equals NP means that can solve fundamentally difficult problems with easy solution like figuring out how to put together broken glass



So how do we work with NP problems?

* Compare relative difficulty of diff problems
  * Use inequalities
* Reduction
  * Problem X is AT LEAST as hard as Problem Y $\Rightarrow$ solving X would solve Y
    * Y <= pX

2 important lemmas

(8.1) *Suppose $Y \leq_P X$. If X can be solved in polynomial time, then Y can be solved in polynomial time.*

(8.2) *Suppose $Y \leq_P X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time.*

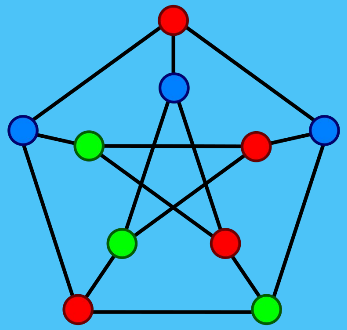
Credits: Algorithm Design by Tardos & Kleinberg book

Summary of NP

https://www.youtube.com/watch?v=OY41QYPI8cw

https://www.youtube.com/watch?v=EHp4FPyajKQ

Vertex Cover & Independent Set Problem



Problem Statement

* Vertex cover
  * Set of nodes s.t. each edge of the graph is incident to at least one node of this set
* Independent set
  * Set of nodes S is independent if no 2 nodes in S are joined by an edge or are adjacent

Examples

* Want to find largest possible independent set
    * Ex: {3, 4, 5} vs {1, 4, 5, 6}
* Want to find smallest vertex cover
    * Ex: {1, 2, 6, 7} vs {2, 3, 7”{

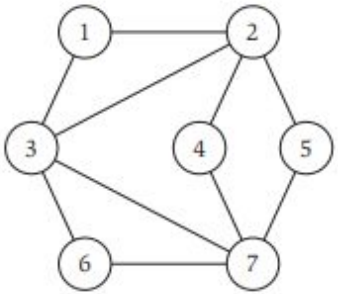


Figure 8.1 A graph whose largest independent set has size 4, and whose smallest vertex cover has size 3.

Credits: Algorithm Design by Tardos & Kleinberg book

Objective

* Try to reduce this problem and compare with another problem so we can understand its relative difficulty because this is NP problem and is unsolvable
* Pose question differently so we have **<u>YES OR NO answer</u>**
    * Give graph G and # k, can we construct and independent set of size k?

Objective (cont.)

* Already simplified question
* Want to find problems relative difficulty and reduce it's runtime complexity using **INEQUALITY SIGN**
* Can't solve either independent set or vertex cover but can show they are equally hard by showing IS <= p * VC & VC <= p * IS

Important Lemma

(8.3) *Let $G = (V, E)$ be a graph. Then S is an independent set if and only if its complement $V - S$ is a vertex cover.*

Try showing this is true!!!

Credits: Algorithm Design by Tardos & Kleinberg book

Using this lemma we can conclude...

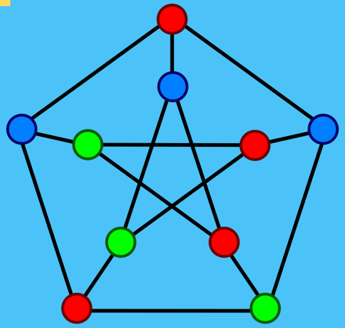
(8.4) Independent Set $\leq_P$ Vertex Cover.

(8.5) Vertex Cover $\leq_P$ Independent Set.

- 8.4: if we can solve Vertex Cover ⇒ we can decide whether G has an IS of size at least k by asking if G has a vertex cover at most K
- 8.5: if we can solve IS ⇒ we can decide whether G has a VC of of at most K by asking if G has an IS of size at least K

Credits: Algorithm Design by Tardos & Kleinberg book

Reducing 3-SAT to Independent Set



3-sat problem

- Boolean problem
- Want the problem to yield answer 1
- Different clauses with several variables

$$(x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (x_2 \vee \overline{x_3}).$$

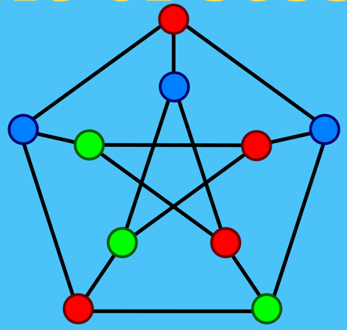
- Truth assignment v that sets all variables to 1 is NOT a satisfying argument vs. to 0 is a satisfying assignment ⇒ each clause MUST evaluate to 1 and have OR operators in between terms of each clause
  - 3 sat when 3 variables
  - Assign 0 or 1 to each variable (F and T)

Credits: Algorithm Design by Tardos & Kleinberg book

3-SAT <=$_p$ * IS

- Want to reduce 3-SAT
- Somehow need to encode boolean constraints of 3-SAT into graph for IS
- ***Homework Task: Go over this problem and try to see if can try solving***
    - ***Will post solutions and see if enough time in next session to go over***

Graph Coloring Problem: Introducing a problem we will look at in the last couple of sessions



Problem statement

* Assign colors to nodes of graph s.t. it satisfies the following constraint

Constraint:

No pair of adjacent nodes have same color

Some vocabulary

* Chromatic number = MINIMUM # of colors required to color graph
    - Varies depending on problem statement
* Vertex coloring
    - Coloring nodes so that no pair of adj nodes have same color
* Edge coloring
    - Coloring edges so no pair of adj edges have same color

Summary of NP

https://www.youtube.com/watch?v=OY41QYPI8cw

https://www.youtube.com/watch?v=EHp4FPyajKQ