P THINK FAST S

# Competitive Programming

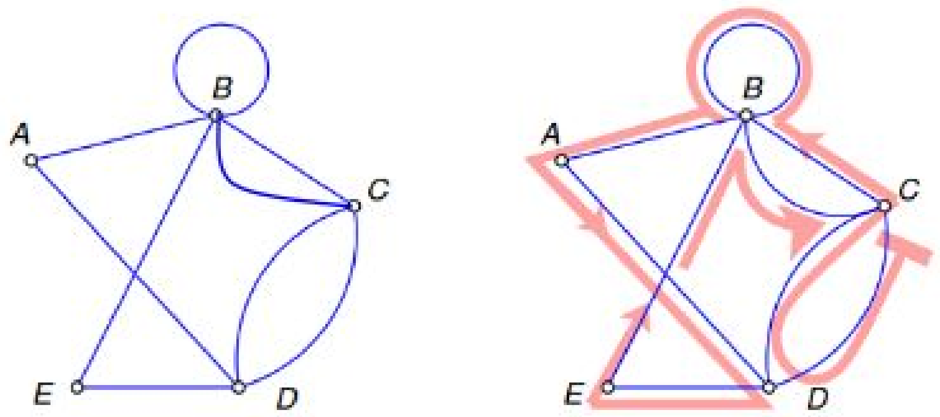
From Problem 2 Solution in O(1)

## NP-Completeness


**Mostafa Saad Ibrahim**
PhD Student @ Simon Fraser University


# Recall: Euler Cycle

- A cycle that uses every **edge** of exactly **once**.

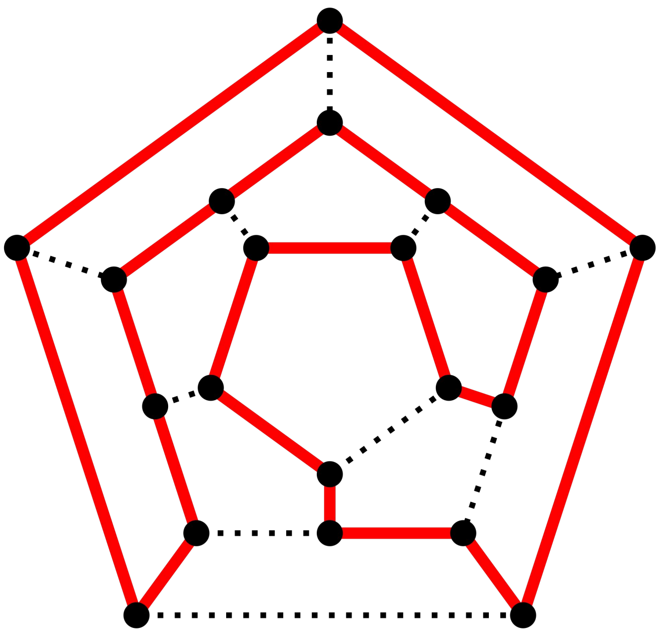



An Euler circuit: CDCBBADEBC

Src: https://www.math.ku.edu/~jmartin/courses/math105-F11/Lectures/chapter5-part2.pdf

# Recall: Hamiltonian Cycle

- A cycle that visits each **<u>node</u>** exactly **once**



Src: https://upload.wikimedia.org/wikipedia/commons/thumb/6/60/Hamiltonian_path.svg/2000px-Hamiltonian_path.svg.png

# Euler vs Hamiltonian Cycle

- We can solve efficiently Euler in O(E)
  - A polynomial time solution!
- We can't solve efficiently Hamiltonian **so far**
  - The only way..a backtracking algorithm: O(n!)
  - **Most probably** we won't solve it efficiently
- Summary of NP-Completeness
  - We can solve some problems efficiently
  - But we don't know for many other problems
  - And seems we won't be able
  - What are **characteristics** of these hard problems!

# NP-Completeness

- To be able to characterize them..we need to define some things
- Polynomial Reductions
- Decision Vs Optimization problems
- P, NP, NP-Hard, NP-Complete
- We will keep things informal as possible
  - We will avoid: Languages, Encoding, Turing Machine ( Deterministic, Nondeterministic), Co-NP, Cook Theorem
  - You need to study more beyond this session :)
  - We are giving only **big-picture**

# Polynomial-Time Reducibility

- Transform **problem** to another **problem**
  - E.g. We can transform min vertex cover to max independent set
  - E.g. We can transform Subset problem to partition problem
  - Lots of reductions here
- Reduction in **polynomial time** is of interest
- If problem A can be poly-reduced to B
  - Logically, Hardness(A) <= Hardness(B)
  - Hardness = E.g. its Algorithm Order
  - What if also B poly-reducible to A?

# Subset Sum vs Partition Problem

- Subset Sum
  - Given Set of Integers A = {a1, a2..an} and T
  - Is there subset of A of sum T?
  - Assume A = {1, 2, 4} and T = 5
  - Answer is Yes. Subset {1, 4}
- Partition Problem
  - Given Set of Integers X = {x1, x2..xn}
  - Can we partition X to 2 subsets of same length?
  - E.g. X = {1, 2, 4, 9, 12}. Sum = 28. So we need 14/14
  - X1 = {1, 4, 9} and X2 = {2, 12}
- Can we reduce **instance** one to another?

# Partition Problem to Subset Sum

- Let X = {1, 2, 4, 9, 12}
  - Sum of X = 28
  - Each set will be 14
- To reduce it to subset sum
  - Let A = X
  - Let T = 14
  - Then subset solution will be either
  - {9, 1, 4} or {12, 14}…directly corresponds to partitioning

# Subset Sum to Partition Problem

- Assume A = {1, 2, 4} and T = 5
  - Let S = Sum of A = 1+2+4 = 7
- Create X as {A elements, <u>2S-T</u>, <u>S+T</u>}
  - E.g. X = {1, 2, 4, 14-5, 7+5} = {1, 2, 4, 9, 12}
  - X sum = S + 2S-T + S+T = 4S
  - Then X must be partitioned to 2S, 2S
  - Notice, (2S-T) + (S+T) = 3S. Hence can't be together
  - X1 = {9, Subset1}, X2 = {12, Subset2}
  - Subset1 will corresponds to items with sum T to have (2S-T) + T = 2 S => T = {1, 4}
- Then, we reduced subset sum to partition

# Decision & Optimization Problems

- Decision Problem: Yes/No Problem
  - bool divide(x, y)?  return true of x%y = 0
- Optimization Problem: Max/Min problem
  - Shortest Path, Min Vertex Cover, Max Indep Set
- We can solve optimization by calls to decision
  - Introduce answer at most / at least K
  - **Optimization**: Find Longest Path?
  - **Decision**: Is there a solution of at least k edges?
  - (Binary) Search for k
- NP focus is on Decision Problems

# Complexity Class P

- **Decision** Problem can be **solved** in **polynomial** time
  - Is x a multiple of y ?
  - Are x and y relatively prime ?
  - Is x prime ?
  - Is there a path between s and t in a graph G ?
- Typical orders: O(1), O(logn), O(nlogn), O(n^2), O(n^3)
- But NOT: $2^n$, n!, ...

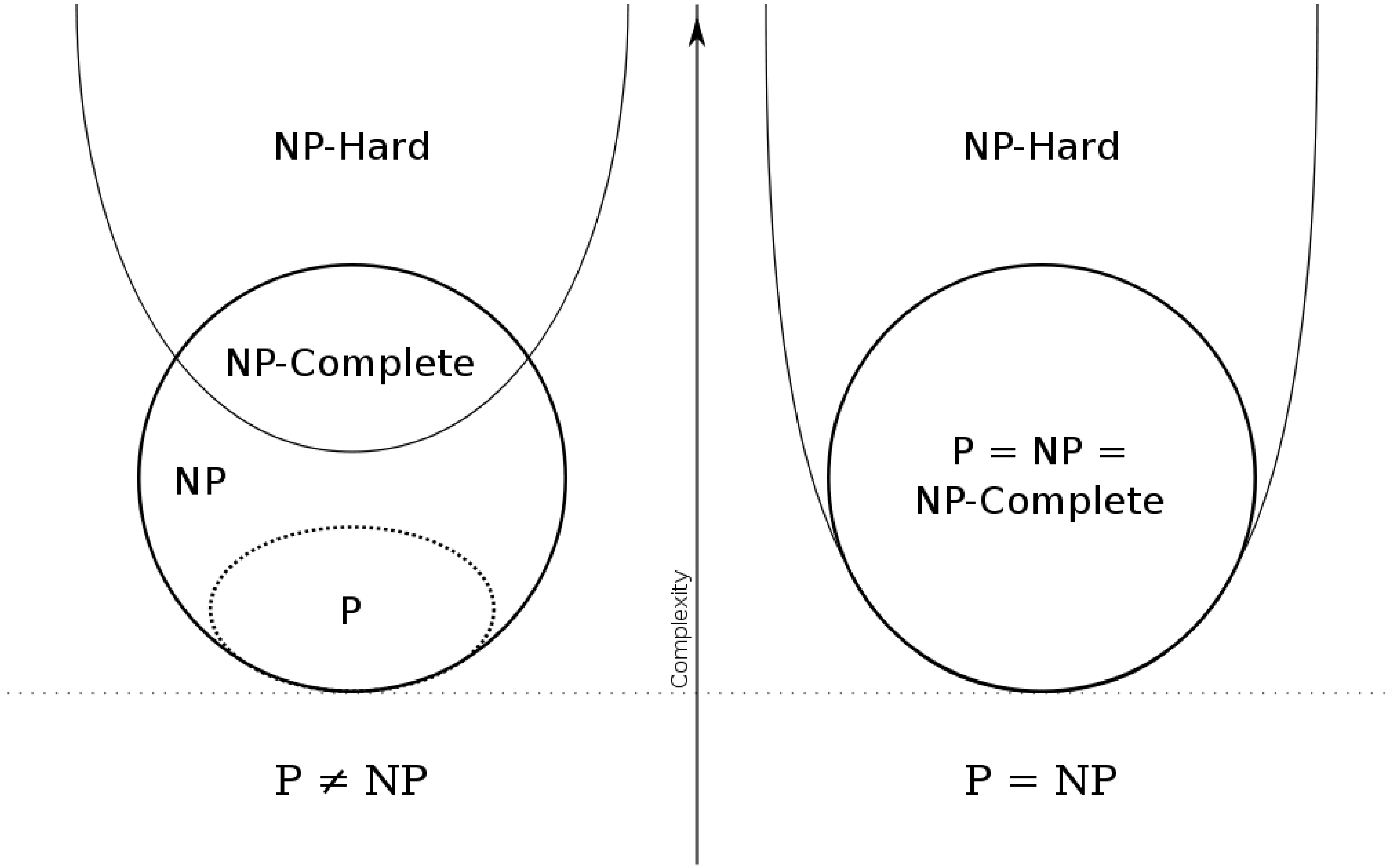
# Complexity Class NP

- Given a **solution** to a decision problem, we can **verify** "yes" answer in **polynomial** time
  - E.g. Given N and and **Factor K**, Does N % K = 0 ?
  - E.g. Given solution to subset sum: does it sum to T?
  - E.g. Given solution to vertex cover, we can see in O(E) if it covers all edges or not
  - E.g. Given solution to CNF, does evaluates to TRUE?

# NP-Hard and NP-Complete

- NP-Hard: **Every** NP Problem Can be poly-reduced to it
  - So all problems...easy and hard ones!
  - This means..everything reduce to it = it is so hard
- NP-Complete (NPC): Both NP and NP-Hard
  - No efficient algorithm is know
- if any NP-complete problem can be solved quickly, then every problem in NP can
  - Because “EVERY” reduction, by definition

# P vs NP?

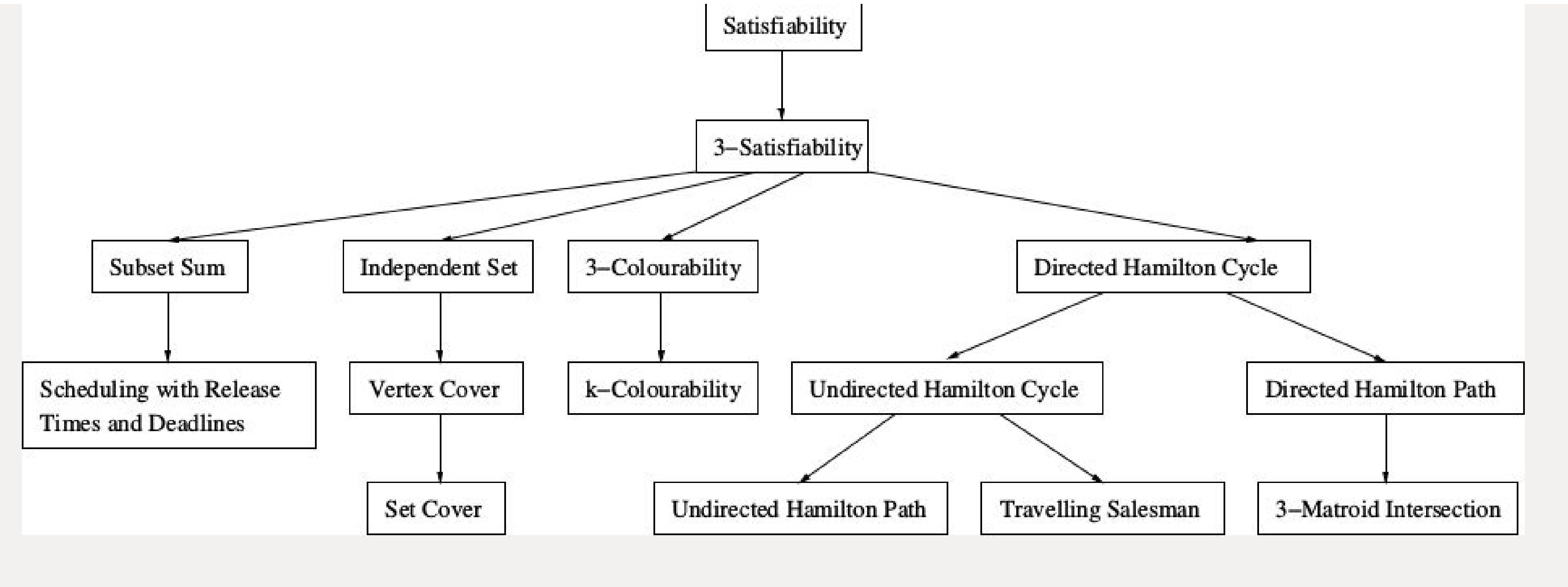


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP =
NP-Complete
P ≠ NP
P = NP

# Cook–Levin theorem

- Proving EVERY NP problem is reducible?!!
- Cook theorem proved Boolean satisfiability problem is NP-Complete
  - Find solution to: (x1 | x2) & (!x1 | x2 | x3) & (.....
  - Solution: $O(2^n)$ try all subsets!
- To prove NP problem X is NP-Complete
  - Just find another NP-Complete Problem Y that we can poly reduce to X and prove correctness of reduction
- More and more NPC problems discovered

# NP-Complete Problems




- SAT is the first problem to be NP-Complete...Every Arrow is reduction
- E.g. We Can prove **Traveling Salesman problem** is NP-Complete by reduction
- Who can solve one….can solve all

# NP-Complete Graph Problems

- Vertex Cover, Independent Set
- Clique, Subgraph Isomorphism
- Hamiltonian path/cycle - directed/undirected
- Traveling Salesman
- 3D Graph Matching, k-Colorability
  - But not 2D matching, 2 Colorability
- Max Cut (but not min cut)
- Longest Path (but not shortest path)
- ...

# Approximation

- What if need an NPC problem for a real app!
- We have to find an approximation to it
  - Some heuristics have **no guarantees** on performance
  - Others can have some **factor** far from optimal
  - E.g. Min solution is 100, some approx solution can have up to 200. E.g. 2-approx algorithm
- Some approx. techniques doesn't have a **const** factor..solution quality is based on requested ε and have (1+ε) approximation

# Real Life Apps

- Why studying NPC is important?
- Many real life apps need NPC problems
  - E.g. Facility Location Problem
- If you don’t know NPC, you may try forever to find efficient a solution!
- If couldn’t outline solution in short time?
  - try to prove it is NPC
  - If so, then no efficient solution
- Try to find an approximation solution for it
  - Try to bound it…or prove no guarantee

# Programming Competitions

- Sometimes, a competition has NPC problem
- Either with very small limits
  - E.g. n = 20 for 2^n   or   n = 10 for n!
- Or higher limits for Pseudo-Poly algorithms
  - E.g. for subset-sum and knapsack: N = 100, W = 10000
- Or much higher limits, but **constraints** make it in P class NOT in NPC
  - Find these restrictions and utilize them
  - E.g. Longest Path in **DAG**
  - E.g. Min Vertex Cover in **Trees**

# تم بحمد الله

علمكم الله ما ينفعكم

ونفعكم بما تعلمتم

وزادكم علماً