Algorithms and Computation

(grad course)

Lecture 5: NP problems, NP-Complete Problems

Instructor: Hossein Jowhari

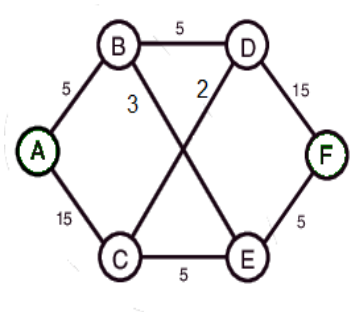
jowhari@kntu.ac.ir

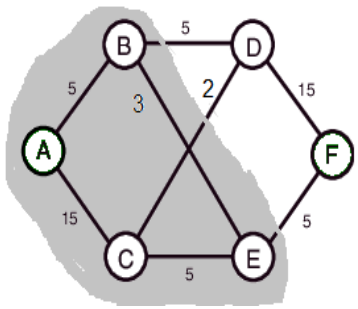
Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Fall 2021

An exercise

Can we use the Ford-Fulkerson algorithm for finding the global min cut in undirected graphs?





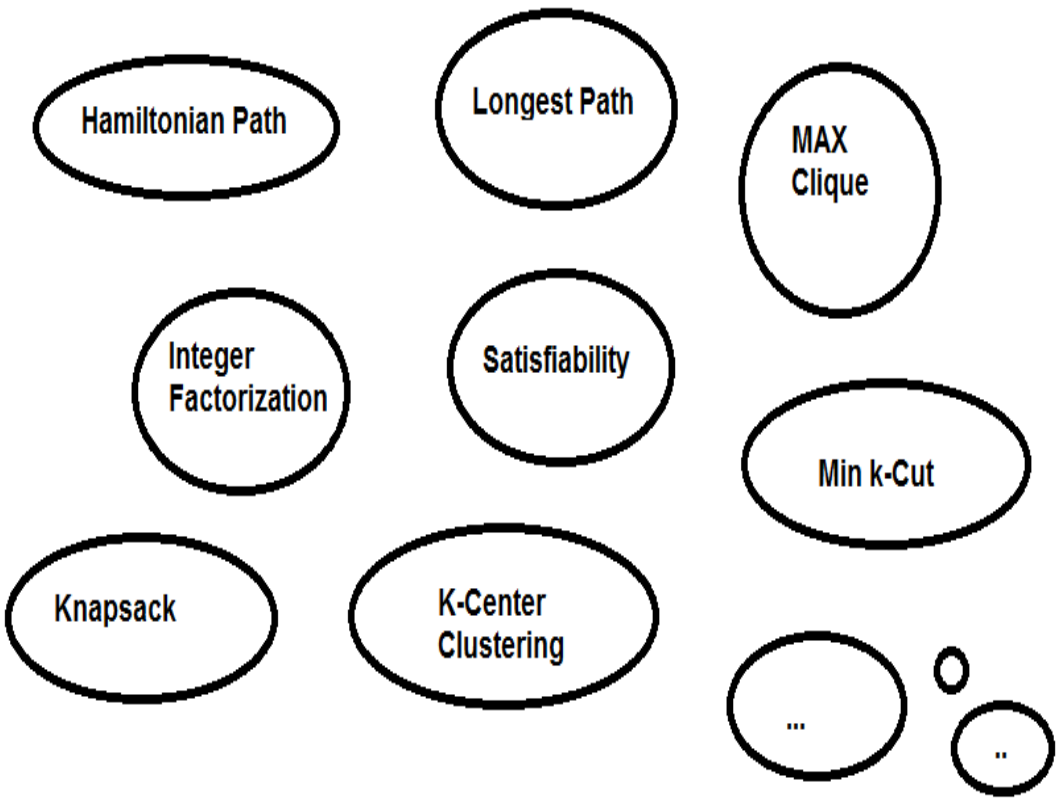
If yes, what is the time complexity of your solution?

Problems with no polynomial time algorithms

There is a long list of natural problems that are hard to solve. In other words they do not seem to have polynomial time algorithms. Some examples:

Hamiltonian path (does a given undirected graph has a Hamiltonian path?), **Longest path** (is there a simple path of length at least k between s and t?), **Maximum clique** (does given undirected graph has a clique of size k?), **Knapsack problem** (can we pick a set of items with total value at least v?), **Satisfiability** (does a given logical formula have a truth satisfying assignment?), **Integer factorization** (does a number n has a factor smaller than k?), k-**Center Clustering** (given a set of n points in the plane are there k points among them that forms clusters of radius at most r?), **Min** k-**cut** (given an undirected graph, can we remove at most t edges and create k disconnected components?), ...

What do these problems have in common? (except being hard to solve). Are these problems somehow connected? Perhaps they are completely unrelated.

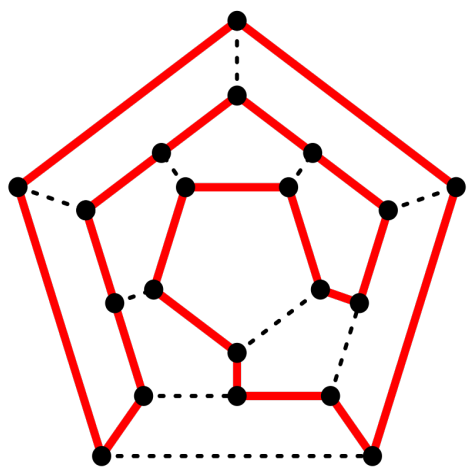


These problems have one thing in common though.

If I were given a solution for any of these problems, I can efficiently (in polynomial time) check the correctness of the solution. Note that this is different from finding the solution.

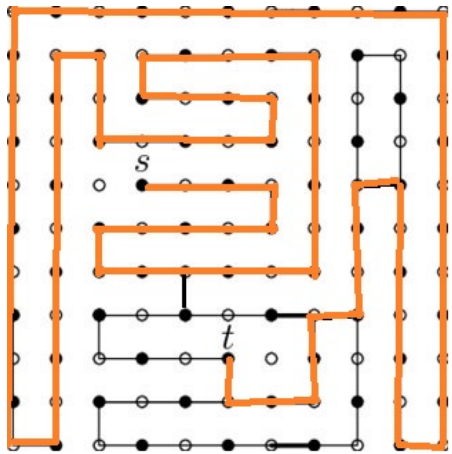
Lets examine this fact in some of these problems.

- **Hamiltonian path.**



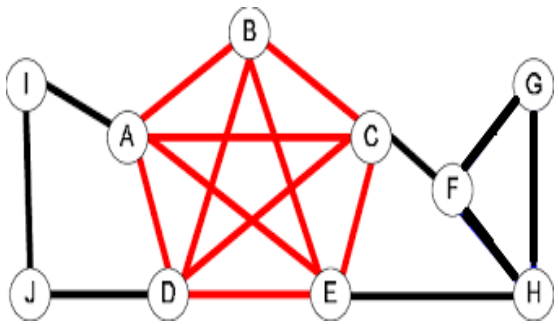
We can check in $O(n)$ time whether a given path is Hamiltonian or not.

- (**Longest path**) Is there a path of length at least k between s and t?



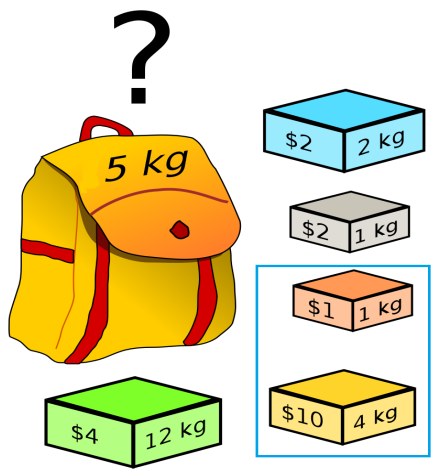
We can check in $O(n)$ time whether a given path connect s to t and has length at least k.

- (**MAX Clique**) Is there a clique of size at least k in the graph G?



We can check in $O(k^2)$ time whether a given subset of k nodes is in fact a clique of size k or not.

- (**Knapsack**) Is there a set of items with total value at least t and total weight less than W?



We can check in $O(n)$ time whether the given items satisfy the desired property.

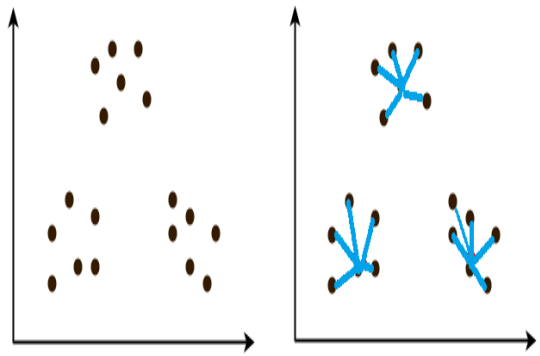
- (**Satisfiablity**) Is there a satisfying assignment for a given logical formula?

$$(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee x_4)$$

x1 = False
x2 = True
x3 = True
x4 = True

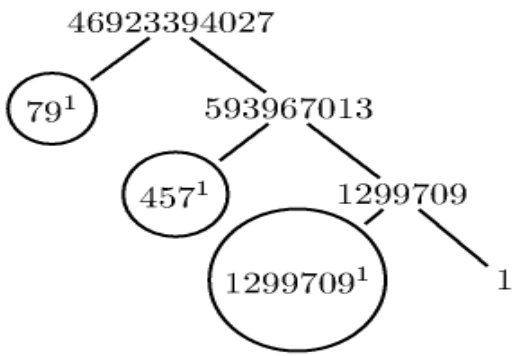
We can check in $O(m)$ time whether a given assignment satisfy the given formula or not. m is the length of the formula.

- (**K-Center Clustering**) Given a set of n points in the plane, is there a set of k centers that induce a clustering with radius at most r?



We can check in $O(n)$ time whether the given centers satisfy the desired property.

- (**Integer Factorization**) Does a given number n has a factor less than k?



We can check in $O(1)$ time whether a given number divides the number n.

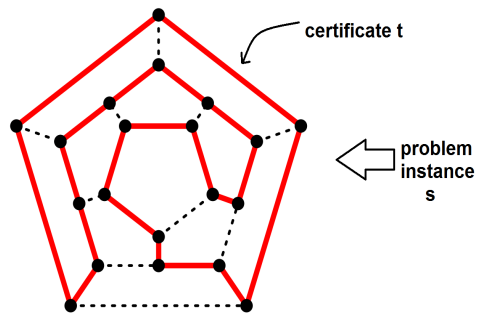
Definition: We say the algorithm B is an *efficient certifier* for the decision problem X if the following properties holds.

- B runs in polynomial time and takes two input strings s and t.
- There is a polynomial function p such that for any s,

$$s \in X \iff \text{there is a string } t \text{ of length } p(|s|) \text{ such that } B(s,t) = yes$$

We call the string t a certificate for the problem instance s

Example:
X = Hamiltonian Graphs.



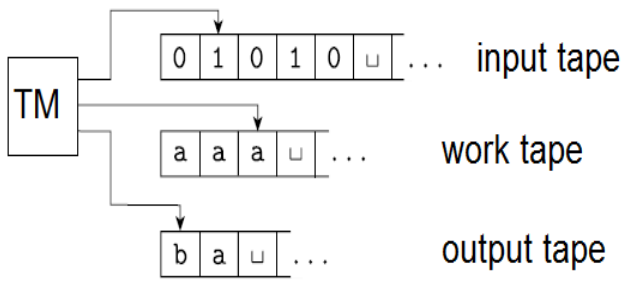
The class NP

Definition: NP is the set of all problems that have efficient certifiers. In other words, the probelm $X \in NP$ if and only if X has an efficient certifier.

NP stands for Nondeterministic Polynomial time.

An equivalent definition of class NP.

Definition: NP is the set of all problems that have a nondeterministic Turing machine that runs in polynomial time.



Examples of problems (sets) that do not seem to be in NP:

- Set of Non-Hamiltonian graphs
- Set of graphs with max clique less than k
- set of logical formulas that do not have a satisfying assignment
- ...

Theorem: $P \subseteq NP$

Proof: Problems in P have polynomial time algorithms. A polynomial time algorithm for a problem X in P is also an efficient certifier for X. Here $t = \varnothing$. (The algorithm does not need a certificate!)

Question: Is $NP \subseteq P$? in other words is $P = NP$?

This is a fundamental question of the computational complexity theory and it is still undecided. It is widely believed that $NP \neq P$ but we do not have a proof for that.

If $NP = P$, then all the problems in NP have polynomial time solutions (algorithms). For example we will be able to answer if a given graph on n vertices is Hamiltonian or not in time $O(n^c)$ (when c is a constant).

NP-Complete Problems

What is the hardest problem in NP? Is there a central problem in NP?

Stephen Cook

Leonid Levin

Cook / Levin Theorem 1971

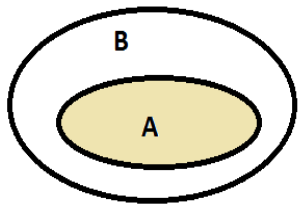
Definition: A problem X is NP-Complete if and only if

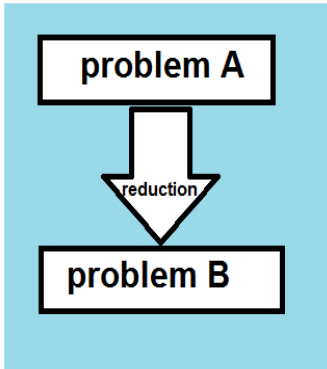
- $X \in$ NP
- All the problems in NP are *polynomial-time reducible* to X.

Theorem: (Cook-Levin 1971) SAT is NP-Complete.

Polynomial-time Reduction

Definition: Problem A is reducible to the problem B if given an algorithm for B we can solve the problem A. We write $A \leq B$





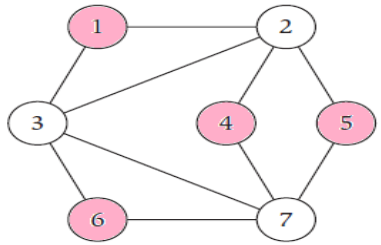
Definition: Problem A is *polynomial-time reducible* to the problem B if we can solve an instance of the problem A in polynomial number of steps plus polynomial number of calls to an algorithm that solves B. We write $A \leq_p B$

Conclusion 1: If $B \in P$ and $A \leq_p B$ then $A \in P$.

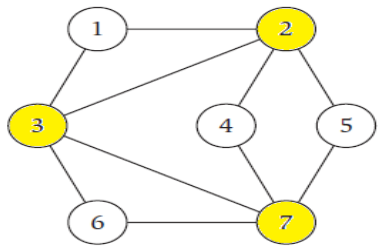
Conclusion 2: If $A \notin P$ and $A \leq_p B$ then $B \notin P$.

Examples of Polynomial-time Reductions

Problem A: (Independent Set) Given the graph G on n nodes, does G have an independent set of size at least k? An independent set is a subset of nodes with no edges between them.



Problem B: (Vertex Cover) Given the graph G on n nodes, does G have a vertex cover of size at most k? A vertex cover is a subset of nodes that covers every edge in the graph.



Theorem: Let $G = (V, E)$ be a graph. S an independent set of G if only if $V - S$ is a vertex cover of G.

- Independent Set $\leq_p$ Vertex Cover

  We want to know if G has an independent set of size at least k (Problem A). Given an algorithm for Vertex Cover (Problem B) we can solve problem A. By the theorem above, it is enough to ask does G have a vertex of cover of size at most $n - k$?

- Vertex Cover $\leq_p$ Independent Set

  We want to know if G has a vertex cover of size most k (Problem B). Given an algorithm for Independent Set (Problem A) we can solve problem B. By the theorem above, it is enough to ask does G have an independent set of size at least $n - k$?

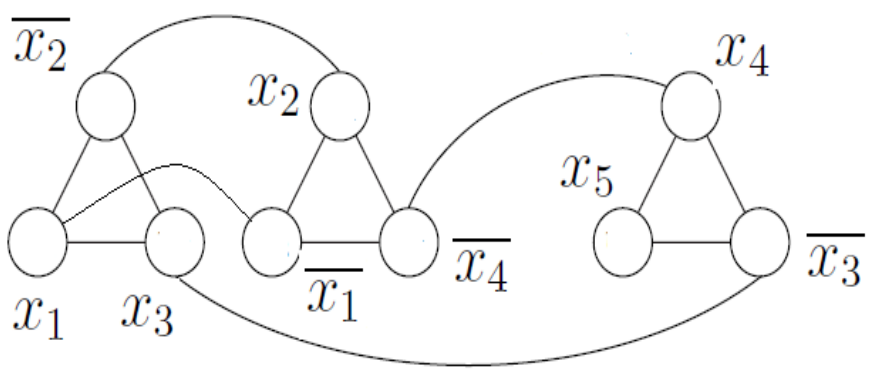
Examples of Polynomial-time Reductions

Problem A: (3-SAT) Given a set of clauses $C_1, \ldots, C_k$ each of length 3 defined over n variables $x_1, \ldots, x_n$, is there an assignment of the variables that satisfy the formula $C_1 \wedge \ldots \wedge C_k$

Example: $\underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{C_1} \wedge \underbrace{(x_2 \vee \overline{x_4} \vee \overline{x_1})}_{C_2} \wedge \underbrace{(\overline{x_3} \vee x_4 \vee x_5)}_{C_3}$

Problem B: (Independent Set) Given the graph G on n nodes, does G have an independent set of size at least k?

Each clause contains exactly 3 literals. For each literal we add a node. We put an edge between the nodes corresponding to the literals of a clause creating a triangle for each clause. Also we put an edge between the pair of nodes u, v if u corresponds to x_i and v corresponds to $\overline{x_i}$ (or vice versa).



Any independent set contains at most one node from each triangle.

$$\left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_2 \vee \overline{x_4} \vee \overline{x_1}\right) \wedge \left(\overline{x_3} \vee x_4 \vee x_5\right)$$