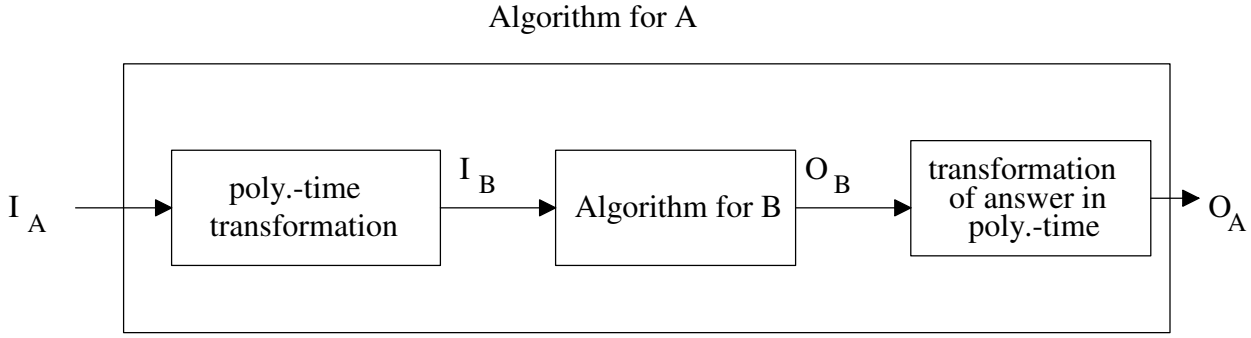
Extend the discussion to non-decision problems

If problem A can be solved in polynomial time using a hypothetical polynomial-time algorithm for problem B, then A is polynomial-time Turing reducible to problem B. A α_T B.



Definition: a problem B is called *NP-hard* if for some NP-complete problem A (a decision problem), A α_T B (A Turing reduces to B).

1. Why is the optimization problem corresponding to any NP-complete problem NP-hard?

Handling NP-hard Problems

1) Backtracking and Branch-and-Bound: Eventhough the worst case is still $\theta(2^n)$ or $\theta(n!)$, these often give efficient results for large problems.

2) Polynomial-time algorithms might exist for a subclass of NP-hard problem, e.g., for TSP with a restricted graph.

3) Approximation algorithm - give good, but not necessarily optimal solutions. Usually, the solution can bound how far from optimal its solution is.

In-Class Activity and HW: Reduction to Show NP-Completeness

Your task as a group is to show that the **Independent-set (IS) Problem** is NP-Complete. An *independent set* of a graph $G = (V, E)$ is a subset $V' \subseteq V$ of vertices such that each edge in E is incident on at most one vertex in V'. The *independent-set problem* is to find a maximum-size independent set in G.

The steps for showing that the IS problem is NP-complete is:

0) Phrase the IS problem as an appropriate decision problem.

1) Show that the IS problem is in NP, i.e., show that a claimed solution for the decision problem can be verified in polynomial time.

2) Select a known NP-complete problem to reduce to the IS problem. (HINT: Use the clique decision problem which is a known NP-complete problem.)

3) Describe an algorithm that transforms an instance of the clique problem to an instance of the IS problem.

4) Prove that the transformation of step 3 is correct, i.e., the resulting instances of IS problem answer "YES" if and only if the corresponding clique-decision problem instance answer is "YES".

5) Prove that the transformation of step 3 runs in polynomial time.

For this in-class part of the exercise, you are to concentrate on steps 0 and 3 (with some informal thought to step 4 and 5).

As homework, you should "formalize" steps 1, 4, and 5. Include a description of your transformation algorithm.