4. How to prove a problem is NPC

- The reducibility relation "$\leq_T$" is *transitive*, i.e,

$$A \leq_T B \quad \text{and} \quad B \leq_T C \quad \text{imply} \quad A \leq_T C$$

- Therefore, to prove that a problem A is NPC, we need to
  (1) show that $A \in$ NP
  (2) choose some known NPC problem B, i.e., $B \in$ NPC,
  define a polynomial transformation T from B to A
  show that $B \leq_T A$

4. How to prove a problem is NPC

- Why sufficient? the logic is as follows:

  Since B is NPC, all problems in NP is reducible to B.
  Show B is reducible to A.
  Then all problems in NP is reducible to A.
  Therefore, A is NPC

4. How to prove a problem is NPC

Example 1.

The directed HC is known to be NPC. Use this fact to prove that Undirected HC is NPC.

Proof:

(1) *By direct verification, we know that undirected HC is in NP.*

(2)

$\boxed{\text{Step A:}}$ *Define a transformation T*

$\boxed{\text{Step B:}}$ *Show that*

$$\textit{directed HC} \quad \leq_T \quad \textit{undirected HC}$$

- *By (1) and (2), we conclude that the undirected HC is NPC.*

4. How to prove a problem is NP-complete

Example 1, cont'd:

We now show that

$$\textit{directed HC} \quad \leq_T \quad \textit{undirected HC}$$

$\boxed{\textit{Step A}}$

- *Define transformation T:*

  Let $G = (V, E)$ be a directed graph. Define G to the undirected graph $G' = (V', E')$ by the following transformation T:

  - $\underline{v \in V} \quad \longrightarrow \quad \underline{v^1, v^2, v^3 \in V' \text{ and } (v^1, v^2), (v^2, v^3) \in E'}$
  - $\underline{(u, v) \in E} \quad \longrightarrow \quad \underline{(u^3, v^1) \in E'}$

- *T is polynomial-time computable.*

4. How to prove a problem is NP-complete

Example 1, cont'd:

An illustration of such transformation T:

4. How to prove a problem is NPC

Example 1, cont'd

$\boxed{\text{Step B:}}$ We show that

$$G \text{ has a HC} \iff G' \text{ has a HC.}$$

"$\Rightarrow$" Suppose that G has a directed HC: $v_1, v_2, \ldots, v_n, v_1$ Then

$$v_1^1, v_1^2, v_1^3, v_2^1, v_2^2, v_2^3, \ldots, v_n^1, v_n^2, v_n^3, v_1^1$$

is an undirected HC for G'.

4. How to prove a problem is NPC

Example 1, cont'd

$\boxed{\text{Step B:}}$ We show that

$$G \text{ has a HC} \iff G' \text{ has a HC}.$$

"$\Rightarrow$" Suppose that G has a directed HC: $v_1, v_2, \ldots, v_n, v_1$ Then

$$v_1^1, v_1^2, v_1^3, v_2^1, v_2^2, v_2^3, \ldots, v_n^1, v_n^2, v_n^3, v_1^1$$

is an undirected HC for G'.

"$\Leftarrow$"

1. Suppose that G' has an undirected HC, the three vertices v^1, v^2, v^3 that correspond to one vertex from G must be traversed **consecutively** in the order v^1, v^2, v^3 or v^3, v^2, v^1, since v^2 **cannot** be reached from any other vertex in G'.
2. Since the other edges in G' connect vertices with superscripts 1 or 3, if for any one triple the order of the superscripts is 1, 2, 3, then the order is 1, 2, 3 for all triples. Otherwise, it is 3, 2, 1 for all triples.
3. Therefore, we may assume that the undirected HC of G' is

$$\underline{v_{i_1}^1, v_{i_1}^2, v_{i_1}^3}, \underline{v_{i_2}^1, v_{i_2}^2, v_{i_2}^3}, \ldots, \underline{v_{i_n}^1, v_{i_n}^2, v_{i_n}^3}, \underline{v_{i_1}^1}.$$

Then $v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_1}$ is a directed HC for G. $\square$

4. How to prove a problem is NPC

Example 2: Show that

Subset-Sum $\leq_T$ Set-Partition

Since Subset-Sum is known to be NPC, the above reduction implies that Set-Partition is also NPC.

Subset-Sum decision problem:

Given a positive integer c, and a set $S = \{s_1, s_2, \ldots, s_n\}$ of positive integers s_i for $i = 1, 2, \ldots, n$. Is there a $J \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in J} s_i = c$? assume that $w = \sum_{i=1}^{n} s_i \geq c$.

Set-Partition decision problem:

Given a set S of numbers. Can S be partitioned into two sets A and $\bar{A} = S - A$ such that $\sum_{x \in A} x = \sum_{x \in \bar{A}} x$?

4. How to prove a problem is NPC

Example 2, cont'd

- Let S be an instance of Subset-Sum with $w = \sum_{s_i \in S} s_i$ and the target c.
- Define the set S' (i.e., the transformation T from S to S') as follows:

$$S' = S \cup \{u, v\}, \quad \text{where} \quad u = 2w - c, \quad v = w + c.$$

- Next to show that

$$\text{Yes of Subset-Sum of } S \iff \text{Yes of Set-Partition of } S'$$

4. How to prove a problem is NPC

Example 2, cont'd

$\Longrightarrow$ Let $J \subseteq S$ and the elements in J sum to c. Then $J \cup \{u\}$ sum to $2w$. Note that the elements in $\overline{J} = S - J$ sum to $w - c$. Hence, $\overline{J} \cup \{v\}$ also sums to $2w$. Therefore, S' can be partioned into $J \cup \{u\}$ and $\overline{J} \cup \{v\}$ where both partitions sum to $2w$. Thus, Yes of Subset-Sum *transforms* to a Yes of Set-Partition.

4. How to prove a problem is NPC

Example 2, cont'd

$\Longleftarrow$ Assume S' can be partitioned into two sets, T and $\overline{T} = S' - T$, such that

$$\sum_{x \in T} x = \sum_{x \in \overline{T}} x. \tag{1}$$

Since $w + u + v = 4w$, the sum of the elements in both sets must be equal to $2w$. Therefore, u must be in one set and v must be in the other because $u + v = 3w$. Without loss of generality, let $u \in T$. Then

$$2w = \sum_{x \in T} x = u + \sum_{x \in T - u} x = 2w - c + \sum_{x \in T - u} x.$$

It implies that

$$\sum_{x \in T - u} x = c$$

Thus, Yes of Set-Partition *transforms* to Yes of Subset-Sum. $\square$

5. How to solve a NPC problem

Example 1: Bin Packing problem

Suppose we have an unlimited number of bins, each of capacity 1, and n objects with sizes $s_1, s_2, \ldots, s_n$, where $0 < s_i \leq 1$.

- Optimization problem: Determine the smallest number of bins into which objects can be packed and find an optimal packing.
- Decision problem: Do the objects fit in k bins?

Theorem. Bin Packing problem is NPC

Proof. reduced from the subset sum.

5. How to solve a NP-complete problem

Approximate algorithm for the Bin Packing

- *First-fit strategy (greedy):*

  places an object in the first bin into which it fits.

- Example: Objects $= \{0.8, 0.5, 0.4, 0.4, 0.3, 0.2, 0.2, 0.2\}$
- *First-fit strategy* solution:

B_1	B_2	B_3	B_4
		0.2	
0.2	0.4	0.3	
0.8	0.5	0.4	0.2

- Optimal packing:

B_1	B_2	B_3
	0.2	0.2
0.2	0.3	0.4
0.8	0.5	0.4

5. How to solve a NP-complete problem

Theorem. Let $S = \sum_{i=1}^{n} s_i$.

1. The optimal number of bins required is at least $\lceil S \rceil$
2. The number of bins used by the first-fit strategy is never more than $\lceil 2S \rceil$.

5. How to solve a NP-complete problem

The vertex-cover problem:

- A vertex-cover of an undirected graph $G = (V, E)$ is a subset set of $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ (inclusive) or $v \in V'$.
- In other words, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all edges in E.
- The size of a vertex cover is the number of vertices in it.
- Decision problem: determine whether a graph has a vertex cover of a given size k
- Optimization problem: find a vertex cover of minimum size.
- **Theorem.** The vertex-cover problem is NPC.

5. How to solve a NP-complete problem

The vertex-cover problem:

- An approximate algorithm
  $C = \emptyset$
  $E' = E$
  while $E' \neq \emptyset$
  let (u, v) be an arbitrary edge of E'
  $C = C \cup \{u, v\}$
  remove from E' every edge incident on either u or v.
  endwhile
  return C
- **Theorem.** The size of the vertex-cover is no more than twice the size of an optimal vertex cover.