

CS3230 Semester 1 2025/2026
Design and Analysis of Algorithms

Tutorial 09
Reductions and Computational Complexity
For Week 10

Document is last modified on: October 17, 2025

1 Lecture Review: Reductions and Computational Complexity

1.1 Revisiting Time Complexity

Time complexity is actually computed based on the input size.

- Example 1: Sorting
Input: N (32-bit) Integers.
Input Size: $O(32 \cdot N) = O(N)$.
Merge sort algorithm runs in $O(N \log N)$ – polynomial w.r.t. input size.
- Example 2: Fibonacci
Input: One single Integer, which has value N .
Input Size: $O(\log N)$ for just that one Integer.
DP algorithm (that sums the last two Fibonacci values) runs in $O(N)$ – this is NOT polynomial w.r.t. input size, as there is an exponential gap from $\log N$ to N , i.e., $2^{\log_2 N} = N$.
But it is pseudopolynomial if we consider the input is N and the DP runtime is $O(N)$.

1.2 Reductions

The Key Idea:

- We want to solve a problem A
- We know the solution to problem B
- To solve A, maybe we can translate/reduce problem A to B

Pseudo-code of a typical reduction algorithm:

```
Solve_A(instance_of_A):  
    instance_of_B = translate_A_to_B(instance_of_A)  
    solution_of_B = Solve_B(instance_of_B)  
    solution_of_A = translate_B_to_A(solution_of_B)  
    return solution_of_A
```

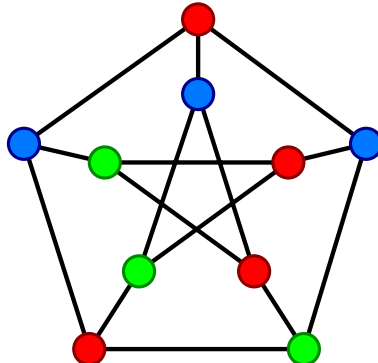
We call this **polynomial time reduction** if both sub-functions above (translate_A_to_B and translate_B_to_A) runs in polynomial time, and we denote this process as $A \leq_p B$.

1.3 Decision vs Optimization Problems

- Decision Problem: Any problem where the output is Boolean (YES/NO)
- Optimization Problem: Any problem where we want to optimize the output
Synonyms of optimize are: maximize, minimize, most optimal, longest, shortest, etc.

2 Tutorial 09 Questions

Q1). GRAPH-COLORING is the problem of assigning colors to vertices of a graph such that no two adjacent vertices share the same color.



Which statement(s) is/are True?

- If we can solve the optimization problem for GRAPH-COLORING in polynomial time, we would be able to solve the decision problem for GRAPH-COLORING in polynomial time.
- If we can solve the decision problem for GRAPH-COLORING in polynomial time, we would be able to solve the optimization problem for GRAPH-COLORING in polynomial time.
- If the decision problem for GRAPH-COLORING cannot be solved in polynomial time, the optimization problem for GRAPH-COLORING cannot be solved in polynomial time.
- If the optimization problem for GRAPH-COLORING cannot be solved in polynomial time, the decision problem for GRAPH-COLORING cannot be solved in polynomial time.

Q2). PARTITION versus BALL-PARTITION

- PARTITION: Given a set of positive integer S , can the set be partitioned into two sets of equal total sum? For example, if $S = \{18, 2, 8, 5, 7, 24\}$, we can partition S into $S_1 = \{18, 2, 5, 7\}$ and $S_2 = \{8, 24\}$, both sum to 32.
- BALL-PARTITION: Given k balls, can we divide the balls into two boxes with an equal number of balls? For example, if $k = 4$, we can divide the balls into $\{2, 2\}$.

We try to show that $\text{PARTITION} \leq_p \text{BALL-PARTITION}$ using the following transformation A :

1. From the problem PARTITION, we are given a set of positive integers S .
2. We define k as the total sum of all integers in S .
3. We use this number k for the BALL-PARTITION problem.

What is wrong with this transformation?

- (a) The transformation does not run in polynomial time.
- (b) This transformation is correct.
- (c) A YES solution to $A(S)$ does not imply a YES solution to S .
- (d) A YES solution to S does not imply a YES solution to $A(S)$.

Q3). PARTITION versus KNAPSACK (as in Lecture)

Transformation: Given a PARTITION instance $\{w_1, w_2, \dots, w_n\}$ with total sum $S = \sum_{i=1}^n w_i$, construct a KNAPSACK instance $\{(w_1, w_1), (w_2, w_2), \dots, (w_n, w_n)\}$ with capacity $W = \frac{S}{2}$ and threshold $V = \frac{S}{2}$.

Which statement(s) is/are True?

- (a) The transformation runs in polynomial time.
- (b) A YES answer to the PARTITION instance implies a YES answer to the KNAPSACK instance.
- (c) A YES answer to the KNAPSACK instance implies a YES answer to the PARTITION instance.

Q4). HAMILTONIAN-CYCLE (HC) versus TRAVELLING-SALESPERSON-PROBLEM (TSP)
(as in Lecture)

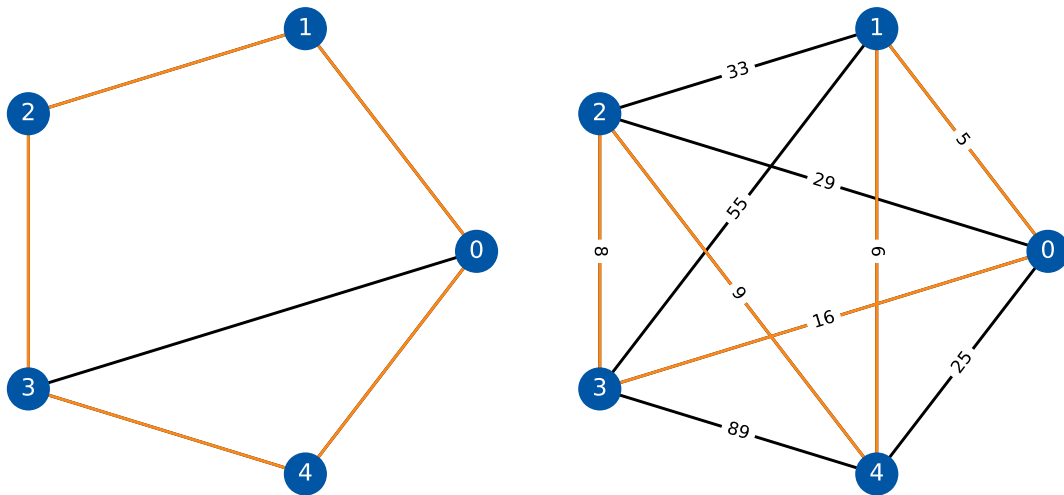


Figure 1: Illustration of Hamiltonian Cycle (left) and TSP Solution (right)

Show that $HC \leq_p TSP$!

The steps:

1. Show the transformation algorithm.
2. Show the transformation algorithm runs in polynomial time.
3. Show A YES answer to the HC instance implies a YES answer to the TSP instance.
4. Show A YES answer to the TSP instance implies a YES answer to the HC instance.

Optional Q5) - time permitting. Discuss the following LeetCode task during tutorial:

TA can choose to discuss in high-level only or show the partial/full code (in C++/Python/Java)

- Wednesday class: house-robber-ii
- Thursday class: partition-equal-subset-sum
- Friday class: shortest-path-visiting-all-nodes