NP and NP-Completeness

Introduction to Decision and Optimization Problems

- Decision Problem: computational problem with intended output of “yes” or “no”, 1 or 0
- Optimization Problem: computational problem where we try to maximize or minimize some value
- Introduce parameter k and ask if the optimal value for the problem is a most or at least k. Turn optimization into decision

Complexity Class P

- Deterministic in nature
- Solved by conventional computers in polynomial time
  - O(1) Constant
  - O(log n) Sub-linear
  - O(n) Linear
  - O(n log n) Nearly Linear
  - $O(n^2)$ Quadratic
- Polynomial upper and lower bounds

Complexity Class NP

- Non-deterministic part as well
- choose(b): choose a bit in a non-deterministic way and assign to b
- If someone tells us the solution to a problem, we can verify it in polynomial time
- Two Properties: non-deterministic method to generate possible solutions, deterministic method to verify in polynomial time that the solution is correct.

Relation of P and NP

- P is a subset of NP
- "P = NP"?
- Language L is in NP, complement of L is in co-NP
- co-NP ≠ NP
- P ≠ co-NP

Polynomial-Time Reducibility

- Language L is polynomial-time reducible to language M if there is a function computable in polynomial time that takes an input x of L and transforms it to an input f(x) of M, such that x is a member of L if and only if f(x) is a member of M.
- Shorthand, $L \overset{poly}{\rightarrow} M$ means L is polynomial-time reducible to M

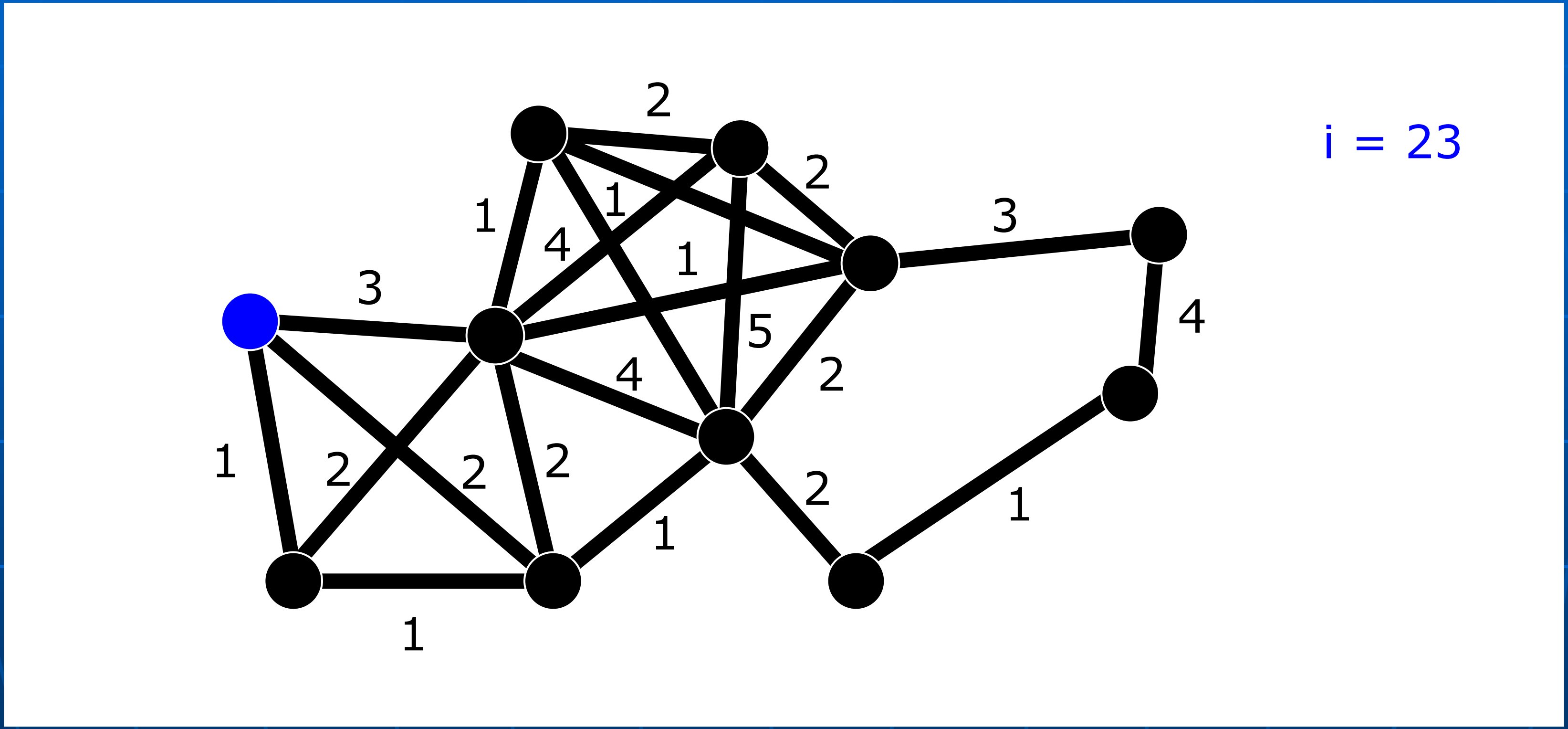
NP-Hard and NP-Complete

- Language M is NP-hard if every other language L in NP is polynomial-time reducible to M
- For every L that is a member of NP, $L \overset{poly}{\rightarrow} M$
- If language M is NP-hard and also in the class of NP itself, then M is NP-complete

NP-Hard and NP-Complete

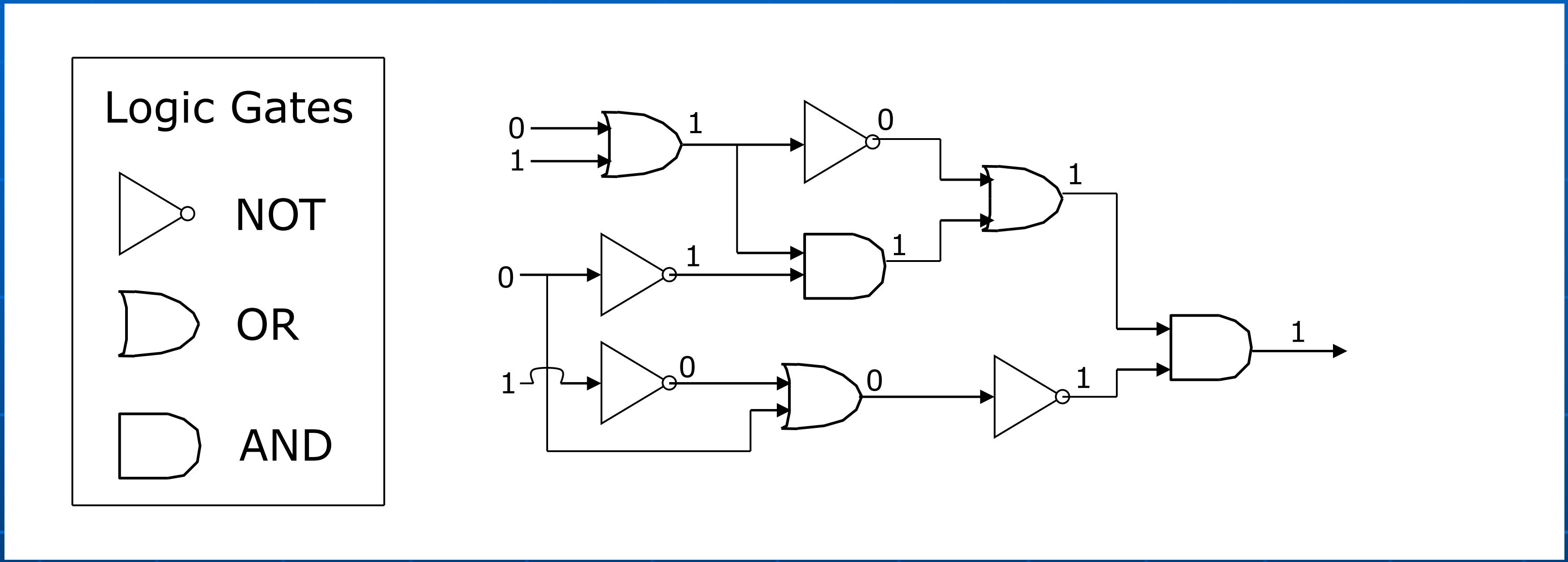
- Restriction: A known NP-complete problem M is actually just a special case of L
- Local replacement: reduce a known NP-complete problem M to L by dividing instances of M and L into “basic units” then showing each unit of M can be converted to a unit of L
- Component design: reduce a known NP-complete problem M to L by building components for an instance of L that enforce important structural functions for instances of M.

TSP



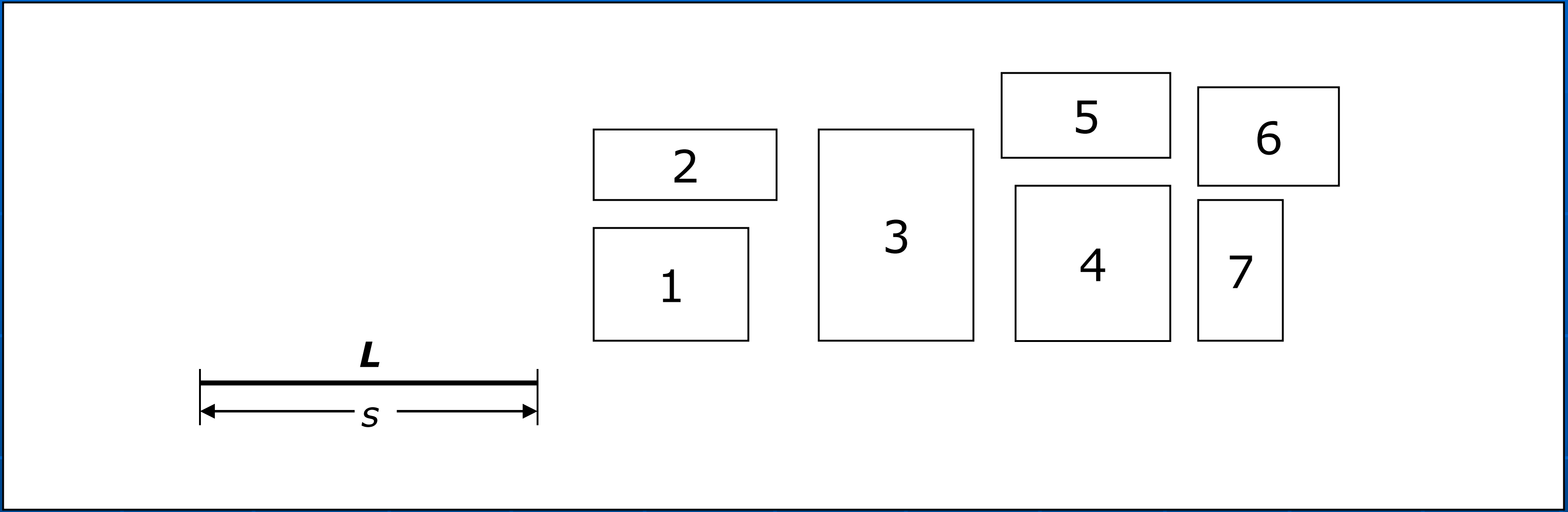
- For each two cities, an integer cost is given to travel from one of the two cities to the other. The salesperson wants to make a minimum cost circuit visiting each city exactly once.

Circuit-SAT



- Take a Boolean circuit with a single output node and ask whether there is an assignment of values to the circuit's inputs so that the output is "1"

Knapsack



- Given *s* and *w* can we translate a subset of rectangles to have their bottom edges on L so that the total area of the rectangles touching L is at least *w*?

PTAS

- Polynomial-Time Approximation Schemes
- Much faster, but not guaranteed to find the best solution
- Come as close to the optimum value as possible in a reasonable amount of time
- Take advantage of rescalability property of some hard problems

Application

- Bin packing problem
- knapsack problem
- Mininum spanning tee
- Longest path problem

Assignment

Q.1)Differentiate between NP-hard & NP-Complete.

Q.2) What is polynomial time reducibility?

Q.3)What is relation between P and NP.