UNIT 5: NP and NP-Complete Problems

Basic concepts, nondeterministic algorithms,
P, NP, NP Complete,
NP-Hard class

Values (some approximate) of several functions important for analysis of algorithms

n	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 \cdot 10^1$	10^2	10^3	10^3	$3.6 \cdot 10^6$
10^2	6.6	10^2	$6.6 \cdot 10^2$	10^4	10^6	$1.3 \cdot 10^{30}$	$9.3 \cdot 10^{157}$
10^3	10	10^3	$1.0 \cdot 10^4$	10^6	10^9		
10^4	13	10^4	$1.3 \cdot 10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7 \cdot 10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0 \cdot 10^7$	10^{12}	10^{18}		

- Easy solved problems:

  constant, logarithmic (logn), linear (n), linear logarithmic (nlogn), quadratic (n^2)
- Hard solved problems

  Cubic(n^3), exponential(2^n), factorial (n!)
- Polynomial time $O(n^k)$
- Exponential time $O(k^n)$

Polynomial time algorithm

Algorithm that solves a problem in polynomial time i.e. its worst-case time efficiency belongs to **O(p(n)) where p(n) is a polynomial** of the problem's input size n.

(since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.)

Note:

e.g: solvable in polynomial time : constant, logarithmic (logn), linear (n), linear logarithmic (nlogn), quadratic (n^2) …

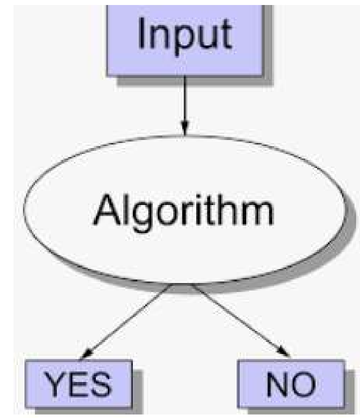
Tractable vs Intractable

Problems that can be solved in polynomial time are called **tractable**

Problems that cannot be solved in polynomial time are called **intractable**.

Decision problem (deterministic algorithms)

A decision problem is a question with a yes/no answer and the answer depends on the value of input.



Example:

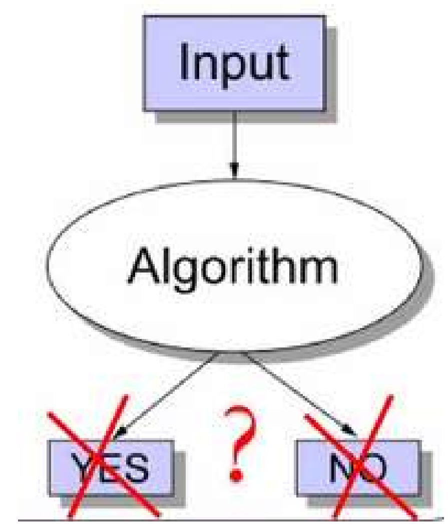
Given an array of n numbers, check if there are any duplicates or not?

Deterministic algorithms : Given a particular input, will always produce a the same output, with the underlying machine always passing through the same sequence of steps.

Example: State machine

Non-deterministic algorithms

- May use external state other than input, such as user input, global variable, random value, hardware timer, stored disk data.
- Is timing sensitive, eg: multiple processors writing to the same data at the same time.



Problem type: Decision vs Optimization

- **Optimization problem**: find a solution that maximizes or minimizes some objective function
- **Decision problem**: answer yes/no to a question

Note:

Many problems have decision and optimization versions.

Eg: TSP

- optimization: find Hamiltonian cycle of minimum length
- decision: find Hamiltonian cycle of length L

Decision problems are more convenient for formal investigation of their complexity.

Undecidable problems

- Some decision problems cannot be solved at all by any algorithm. Such problems are called **undecidable**.
- A famous example was given by Alan Turing in 1936.
- The **halting problem**: given a computer program and an input to it, determine whether the program will halt on that input or continue working indefinitely on it.

Complexity class

- A set of problems with related complexity
- Resources (time and space) required during computations are studied

- Types:
  - P class
  - NP class
  - NP-hard class
  - NP-complete class

Class P

- A set of decision problems that can be solved by a deterministic machine (algorithm) in polynomial time.

Examples:

computing the product and the greatest common divisor of two integers, sorting, searching, checking connectivity and acyclicity of a graph, finding a minimum spanning tree, finding the shortest paths in a weighted graph.

Class NP (Nondeterministic Polynomial)

- A set of decision problems that can be solved by a non-deterministic algorithm in polynomial time. **Set of problems whose solutions are hard to find BUT easy to verify.**

Example:

Hamiltonian circuit problem, the partition problem, decision versions of the traveling salesman, the knapsack, graph coloring

Nondeterministic Polynomial algorithm

is an abstract two-stage procedure that:

- Nondeterministic ("**guessing**") stage: generates a solution of the problem (on some input) by guessing
- Deterministic (**"verification"**) stage: checks whether this solution is correct in polynomial time. Returns yes/no.

- Most decision problems are in NP. NP class includes all the problems in P

$$P \subseteq NP.$$

Polynomially reducible

A decision problem D1 is said to be **polynomially reducible** to a decision problem D2 if there exists a function **T** that transforms instances of D1 to instances of D2 such that

1. T maps all yes instances of D1 to yes instances of D2 and all no instances of D1 to no instances of D2;
2. T is computable by a polynomial-time algorithm.

This implies that if a problem D1 is polynomially reducible to some problem D2 that can be solved in polynomial time, then problem D1 can also be solved in polynomial time.

Example:

Hamiltonian circuit: Determine whether a given graph has a Hamiltonian circuit (a path that starts and ends at the same vertex and passes through all the other vertices exactly once).

Traveling salesman: Find the shortest tour through n cities with known positive integer distances between them (find the shortest Hamiltonian circuit in a complete graph with positive integer weights).

Introduce parameter k and ask if the optimal value for the problem is at most or at least k. Turn optimization into decision

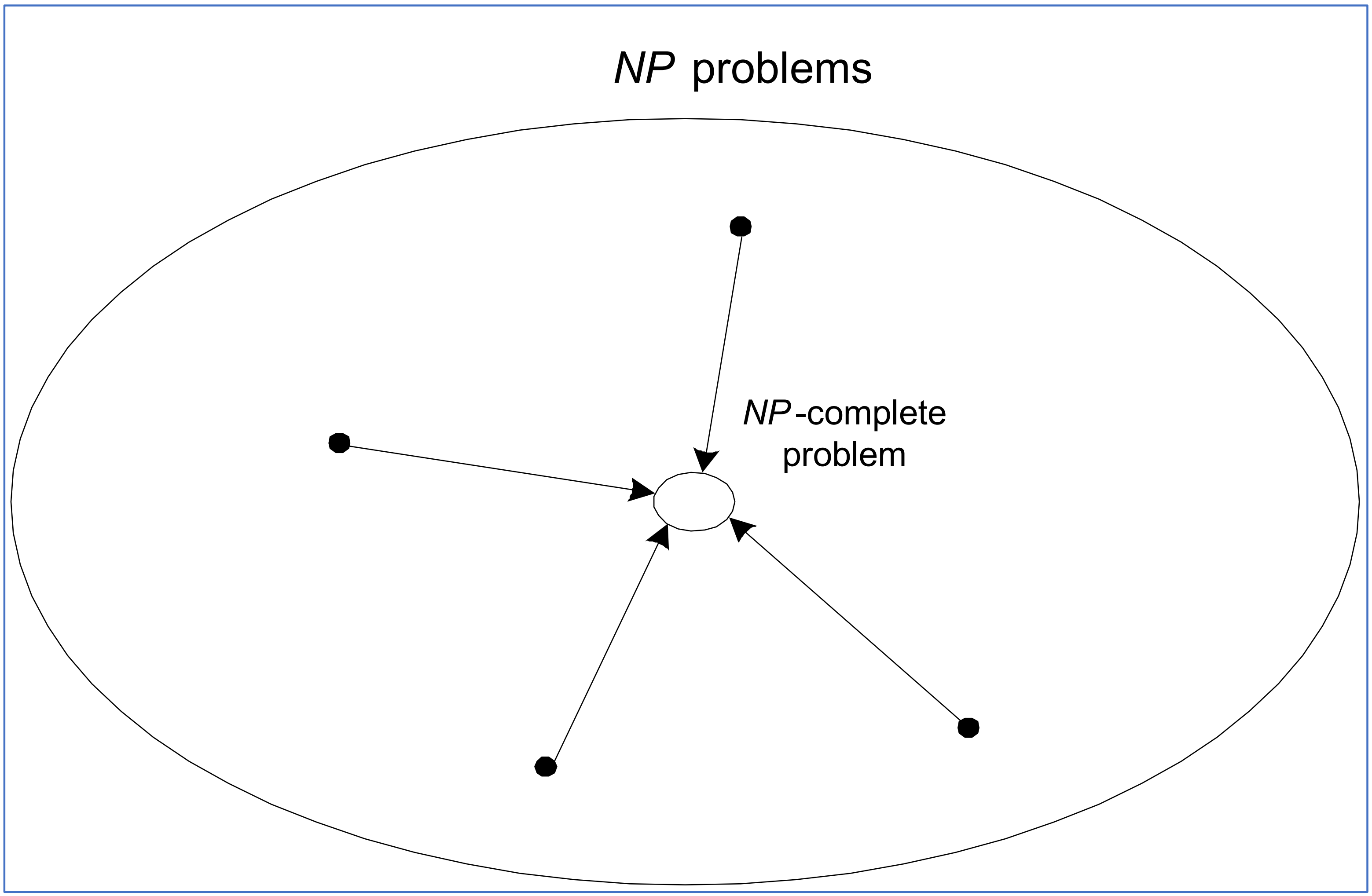
NP-Complete problems

NP-complete problem is a problem in NP that is as difficult as any other problem in this class because, by definition, any other problem in NP can be reduced to it in polynomial time.

A decision problem D is said to be NP-complete if

1. it belongs to class NP;

2. every problem in NP is polynomially reducible to D



NP problems
NP-complete
problem

P = NP ? Dilemma

- P = NP would imply that every problem in NP, including all NP-complete problems, could be solved in polynomial time
- If a polynomial-time algorithm for just one NP-complete problem is discovered, then every problem in NP can be solved in polynomial time, i.e. P = NP

- Most but not all researchers believe that P ≠ NP , i.e. P is a proper subset of NP. If P ≠ NP, then the NP-complete problems are not in P, although many of them are very useful in practice.

NP Hard class

Definition:

The complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time.

When a decision version of a combinatorial optimization problem is proved to belong to the class of NP-complete problems, then the optimization version is NP-hard.