MODULE 5

Classifying problems

- Classify problems as **tractable** or **intractable** .
- Problem is *tractable* if there **exists at least one** polynomial bound algorithm to solve it.
- An algorithm is *polynomial bound* if its worst case growth rate can be bound by a polynomial $p(n)$ in the size n of the problem

$$p(n) = a_n n^k + \ldots + a_1 n + a_0 \text{ where } k \text{ is a constant}$$

Intractable problems

- Problem is *intractable* if it is not tractable.
- **All** algorithms that solve such a problem are not polynomial bound.
- It has a worst case growth rate *f*(*n*) which cannot be bound by a polynomial p(n) in the size n of the problem.
- For intractable problems the bounds are:

$$f(n) = c^n \text{ , or } n^{\log n}\text{, etc.}$$

Tractable and Intractable problems

- Almost all the algorithms we have studied thus far have been ***polynomial-time algorithms***
- On inputs of size n, their worst-case running time is $O(n^k)$ for some constant k.
- Whether *all* problems can be solved in polynomial time. The answer is no.
- For example, there are problems, such as Turing's famous "Halting Problem," that cannot be solved by any computer, no matter how much time we allow.
- There are also problems that can be solved, but not in time $O(n^k)$ for any constant k.
- Generally, we think of problems that are solvable by polynomial-time algorithms as being tractable, or easy
- Problems that require superpolynomial time as being intractable, or hard.

Tractable vs Intractable

- Some problems are *intractable*:
  as they grow large, we are unable to solve them in reasonable time
- What constitutes reasonable time? Standard working definition: *polynomial time*
  - On an input of size n the worst-case running time is $O(n^k)$ for some constant k
  - Polynomial time: $O(n^2)$, $O(n^3)$, $O(1)$, $O(n \lg n)$
  - Not in polynomial time: $O(2^n)$, $O(n^n)$, $O(n!)$

Hard practical problems

- There are many practical problems for which no one has yet found a polynomial bound algorithm.
- Examples: traveling salesperson, 0/1 knapsack, graph coloring, bin packing etc.
- Most design automation problems such as testing and routing.
- Many networks, database and graph problems.

TRACTABLE PROBLEMS	INTRACTABLE PROBLEMS
1) Solved in polynomial time.	1) Solved in exponential time.
2) Can be verified in polynomial time	2) Can be verified in polynomial time
3) It is easy to solve	3) It is not easy to solve
4) Class P are tractable	4) Class NP are intractable.

Complexity Classes

- There exist some problems whose solutions are not yet found, the problems are divided into classes known as **Complexity Classes**
- Complexity Class is a set of problems with related complexity
- The time complexity of an algorithm is used to describe the number of steps required to solve a problem
- The space complexity of an algorithm describes how much memory is required for the algorithm to operate.

Types of Complexity Classes

- P Class
- NP Class
- Co-NP Class
- NP hard
- NP complete

P Class

- The P in the P class stands for **Polynomial Time.**
- It is the collection of decision problems(problems with a “yes” or “no” answer) that can be solved by a deterministic turing machine in polynomial time.
- The solution to P problems is easy to find.
- P is often a class of computational problems that are solvable and tractable.
- Eg: Linear search, Bubble sort etc.

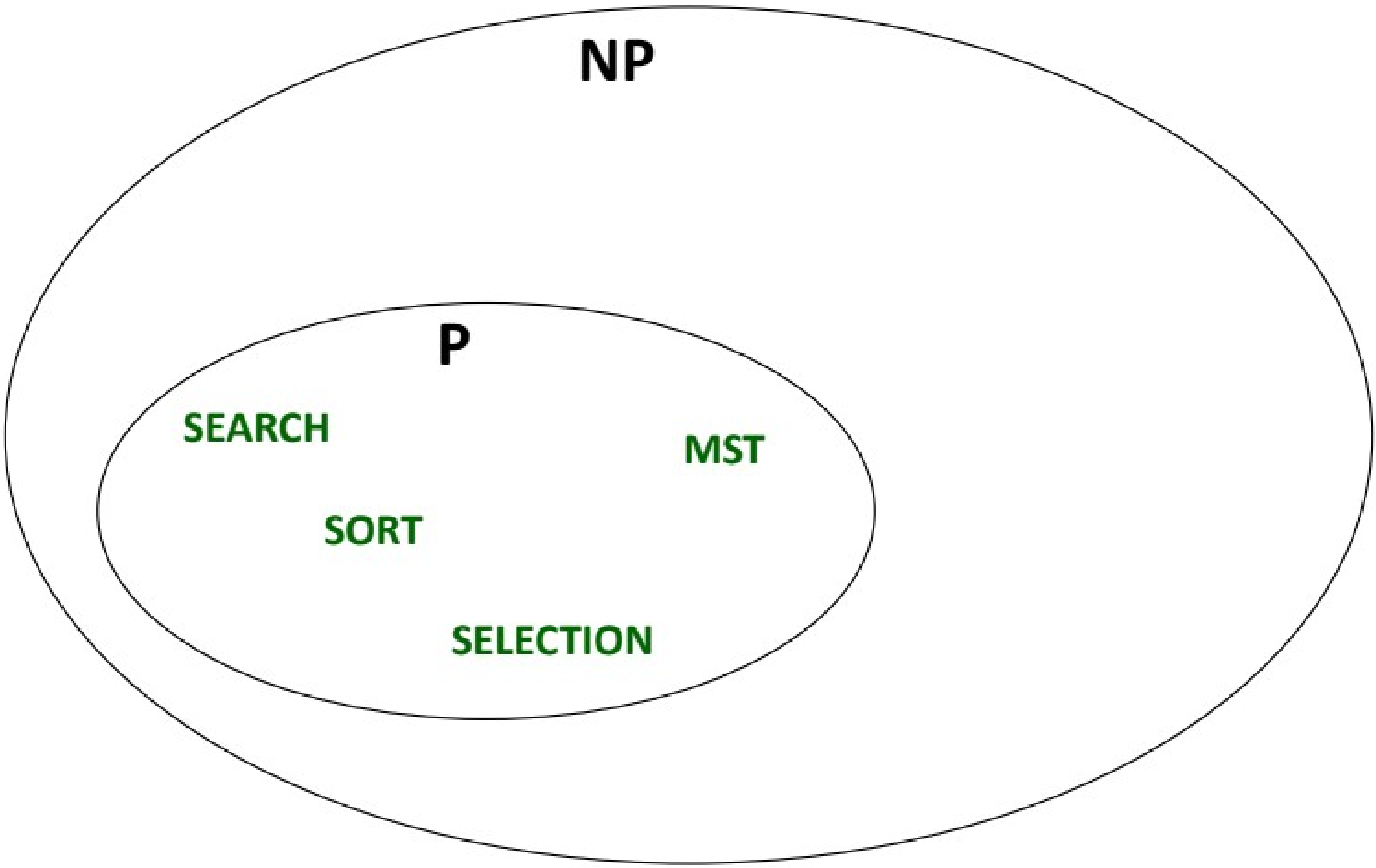
NP Class

- The NP in NP class stands for **Non-deterministic Polynomial Time**.
- It is the collection of decision problems that can be solved by a non-deterministic turing machine in polynomial time.
- The solutions of the NP class are hard to find since they are being solved by a non-deterministic machine but the solutions are easy to verify.
- Problems of NP can be verified by a Turing machine in polynomial time.
- eg: Graph coloring, TSP, SAT

Summary: P and NP

- Summary so far:
  - **P** = problems that can be solved in poly time
  - **NP** = problems for which a solution can be verified in polynomial time
  - **P** $\subseteq$ **NP**
  - Unknown whether **P = NP** (most suspect not)
- We've seen problems that belong to NP that may not belong to P
  - Hamiltonian path/cycle, TSP problems are in **NP**
  - Cannot solve in polynomial time
  - Easy to verify solution in polynomial time

Relation ship between P & NP



Co-NP Class

- Co-NP stands for the complement of NP Class.
- It means if the answer to a problem in Co-NP is No, then there is proof that can be checked in polynomial time.
- If a problem X is in NP, then its complement X’ is also is in Co-NP.
- For an NP and Co-NP problem, there is no need to verify all the answers at once in polynomial time, there is a need to verify only one particular answer “yes” or “no” in polynomial time for a problem to be in NP or Co-NP.
- Eg: **To check prime number , Integer Factorization.**

NP-hard class

- An NP-hard problem is at least as hard as the hardest problem in NP
- It is the class of the problems such that every problem in NP reduces to NP-hard.
- All NP-hard problems are not in NP.
- It takes a long time to check them. This means if a solution for an NP-hard problem is given then it takes a long time to check whether it is right or not.
- A problem A is in NP-hard if, for every problem L in NP, there exists a polynomial-time reduction from L to A.
- Eg: **Halting problem**

NP-complete class

- A problem is NP-complete if it is both NP and NP-hard. NP-complete problems are the hard problems in NP.
- NP-complete problems are special as any problem in NP class can be transformed or reduced into NP-complete problems in polynomial time.
- If one could solve an NP-complete problem in polynomial time, then one could also solve any NP problem in polynomial time.
- Eg:**0/1 Knapsack, Hamiltonian Cycle, Satisfiability, Vertex cover.**

Complexity Class	Characteristic feature
P	Easily solvable in polynomial time.
NP	Yes, answers can be checked in polynomial time.
Co-NP	No, answers can be checked in polynomial time.
NP-hard	All NP-hard problems are not in NP and it takes a long time to check them.
NP-complete	A problem that is NP and NP-hard is NP-complete.

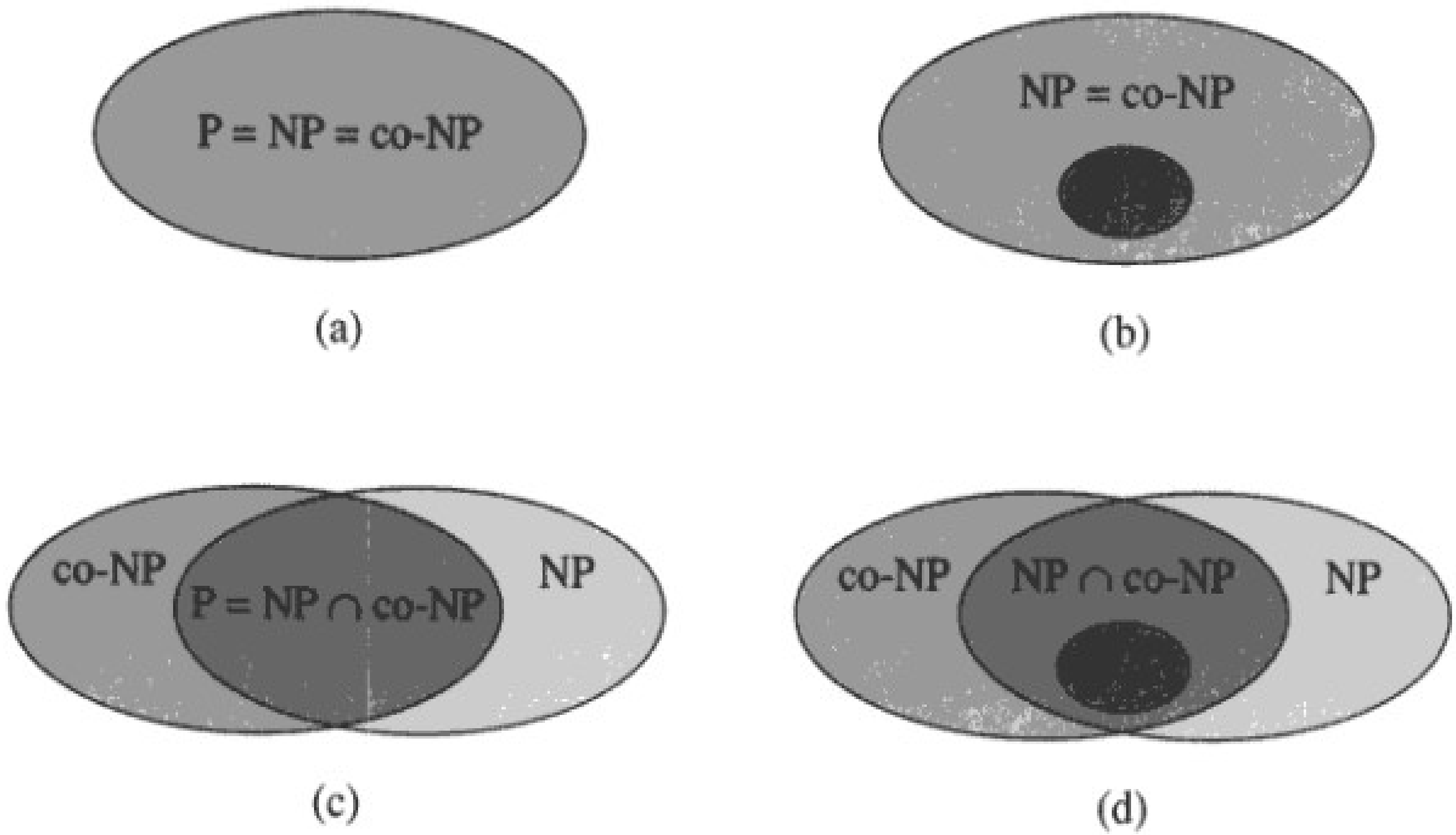
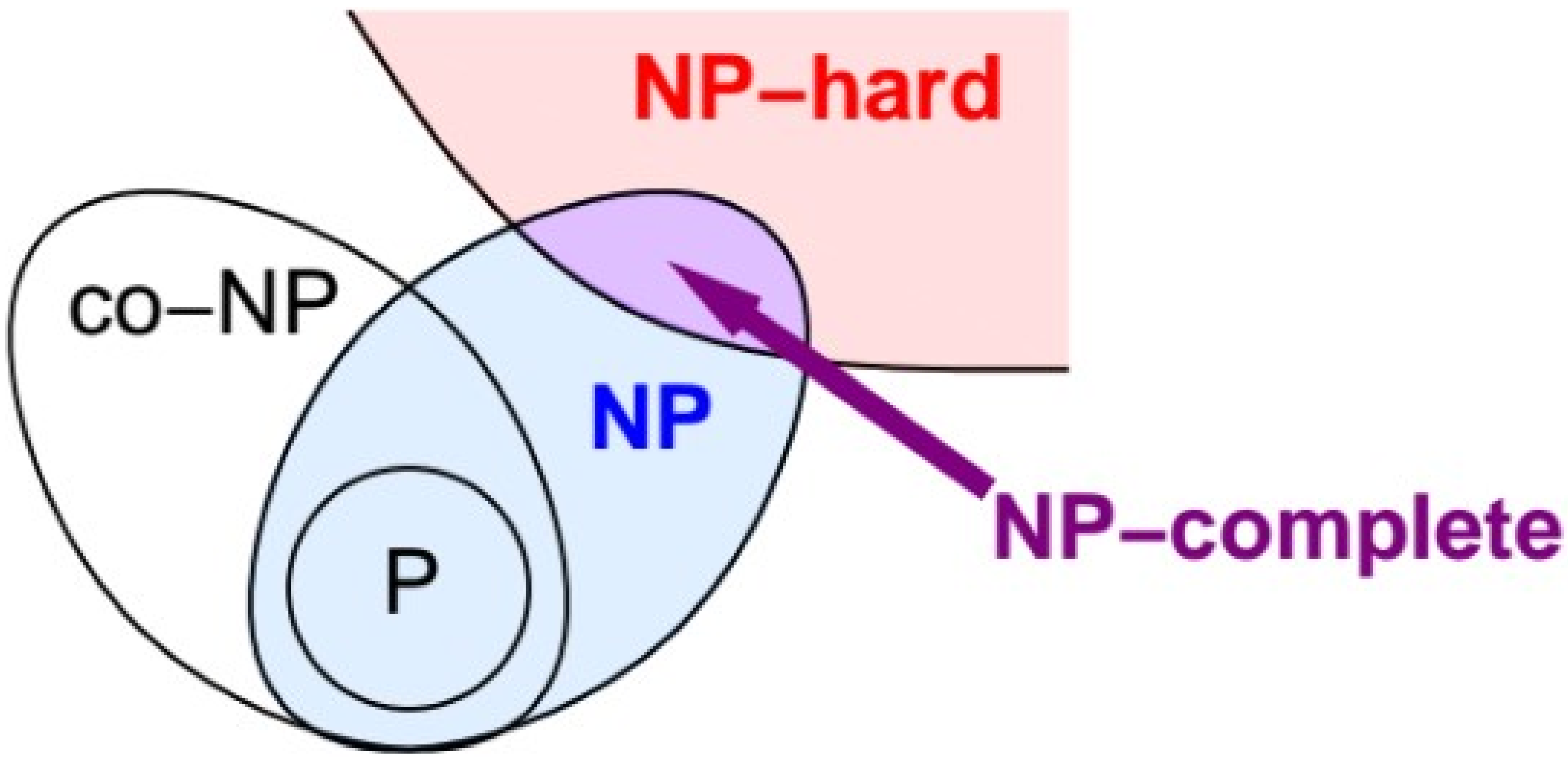


Figure 34.3 Four possibilities for relationships among complexity classes. In each diagram, one region enclosing another indicates a proper-subset relation. **(a)** $\text{P} = \text{NP} = \text{co-NP}$. Most researchers regard this possibility as the most unlikely. **(b)** If NP is closed under complement, then $\text{NP} = \text{co-NP}$, but it need not be the case that $\text{P} = \text{NP}$. **(c)** $\text{P} = \text{NP} \cap \text{co-NP}$, but NP is not closed under complement. **(d)** $\text{NP} \neq \text{co-NP}$ and $\text{P} \neq \text{NP} \cap \text{co-NP}$. Most researchers regard this possibility as the most likely.

Relationship between P, NP, Co-NP, NP-hard and NP-Complete



How to prove a Problem Is NP-Complete?

- NP-Complete problems are the ones that are both in and -Hard.
- To prove that problem is -Complete we need to show that the problem:

  i) belongs to NP

  ii) It is -Hard