P AND NP PROBLEMS

Submitted by: Nikita Uikey
Roll No.: 181210033
Branch: CSE 2ND YEAR

Many problems are solved where the objective is to maximize or minimize some values, whereas in other problems we try to find whether there is a solution or not. Hence, the problems can be categorized as follows –

OPTIMIZATION PROBLEM

an optimization problem is the problem of finding the best solution from all feasible solutions.
For example:

- SHORTEST PATH:

Find the path from ss to tt with the smallest weight

DECISION PROBLEM

There are many problems for which the answer is a Yes or a No. These types of problems are known as decision problems.
For example,

- SHORTEST PATH:

Is there a path pp from ss to tt with weight ⩽k⩽k (for some kk)?

INTRODUCTION TO P VS. NP PROBLEM

The P versus NP problem is to determine whether every language accepted by some non-deterministic algorithm in polynomial time is also accepted by some deterministic algorithm in polynomial time.

• WHAT IS 'P' ?

- P (stands for Deterministic Polynomial Time)
- Is the class of problems for which we have an efficient Algorithm that is capable of producing a solution in polynomial time.

• WHAT IS 'NP' ?

- P (stands for Non-Deterministic Polynomial Time)
- is the set of decision problems that are solvable in polynomial time by a Non-deterministic Turing Machine.

THE COMPLEXITY CLASS P

The complexity class P is the set of all decision problems that can be solved with worst-case polynomial time-complexity.

In other words, a problem is in the class P if it is a decision problem and there exists an algorithm that solves any instance of size n in
O(n^k)time, for some integer k.
(Strictly, n must be the number of bits needed for a 'reasonable' encoding of the input. But we won't get bogged down in such fine details.)

So P is just the set of tractable decision problems: the decision problems for which
we have polynomial-time algorithms.

THE COMPLEXITY CLASS NP

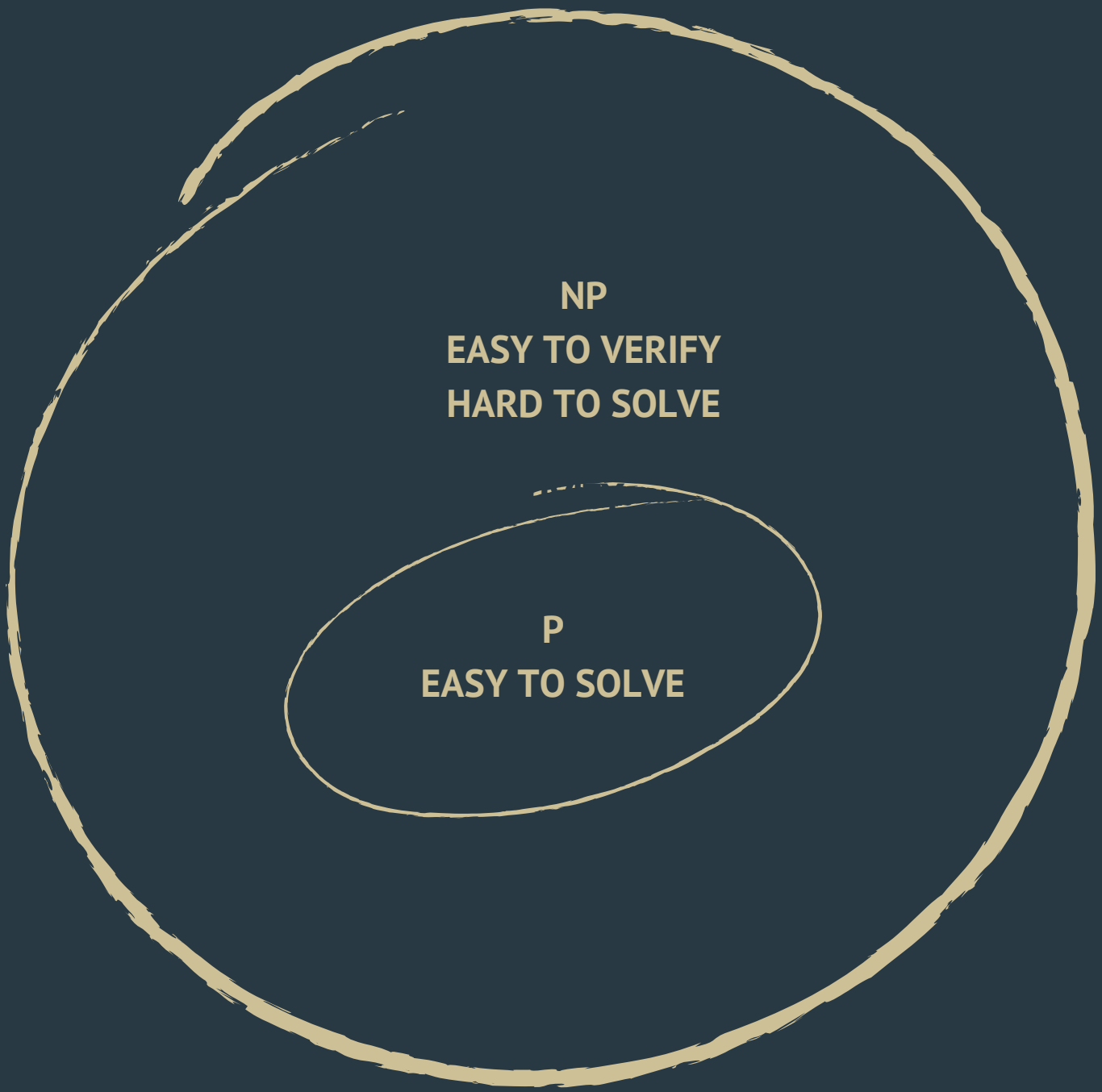
The complexity class NP is the set of all decision problems that can be non-deterministically accepted in worst-case polynomial time.

Of course, there's no computer than can toss magical coins or roll magical dice.
– No conventional computer can carry out the choose operation.
– No one has yet shown how even an unconventional computer (e.g. a quantum computer) could simulate a non-deterministic polynomial-time algorithm
in polynomial time.

So non-deterministic algorithms appear to be useless (we can't code them up and
run them anywhere).

The class NP may therefore seem pretty weird and pointless. Bare with it. It's a theoretical technicality in part of a broader argument that we will develop in the

DIAGRAMATIC REPRESENTATION



P VS. NP PROBLEM

“Can every problem whose solution can be quickly verified by a computer also be quickly solved by a computer?”

Every decision problem that is solvable by a deterministic polynomial time algorithm is also solvable by a polynomial time non-deterministic algorithm.

All problems in P can be solved with polynomial time algorithms, whereas all problems in NP - P are intractable.

It is not known whether P = NP. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If P ≠ NP, there are problems in NP that are neither in P nor in NP-Complete.The problem belongs to class P if it’s easy to find a solution for the problem.

The problem belongs to NP, if it’s easy to check a solution that may have been very tedious to find.

POLYNOMIAL TIME REDUCTION ALGORITHM

To prove P =NP we have to prove that every problem which lies in NP can be solved in polynomial time

There are millions of NP problems we can't solve each problem to prove this, here comes reduction.

A problem 'A' is said to be polynomial time reducible to a problem 'B' if :-

1. Every instance 'a' of 'A' can be transformed to some instance 'b' of 'B' in polynomial time.
2. Answer of 'a' is 'YES' if and only if answer of 'b' is 'YES'.

So If A is reduced to B in polynomial time then :-

- If B is easy then A is also easy.
- If B is in P then A is also in P.
- If this is proven that A can't be solved in polynomial time then B is also can't be solved in polynomial time.
- If A is not in P then B is also not in P.

WHAT IS 'NP HARD'?

NP-hard problems are essentially those that are at least as hard as the hardest NP problem, but don't need to be verifiable in polynomial time.

Enumerating all the possible subsets of the set of every individual atom in the universe is an NP-hard problem.

A problem X is NP-hard if every problem Y ∈ NP reduces to X i.e. X is at least as hard to solve as every problem in NP
(If P! = NP, then X does not belong to P)

WHAT IS 'NP-COMPLETE'?

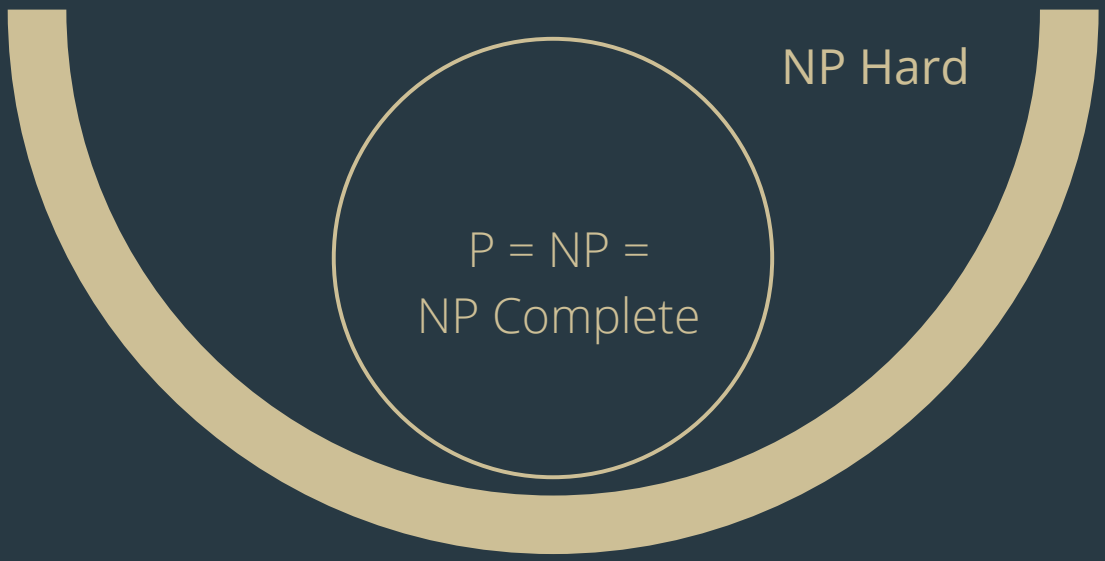
NP-complete is a special category of NP problems that have time complexities greater than polynomial time, are verifiable in polynomial time, and belong to a set of problems known as NP-hard.

NP-complete problems are the hardest problems in NP set.

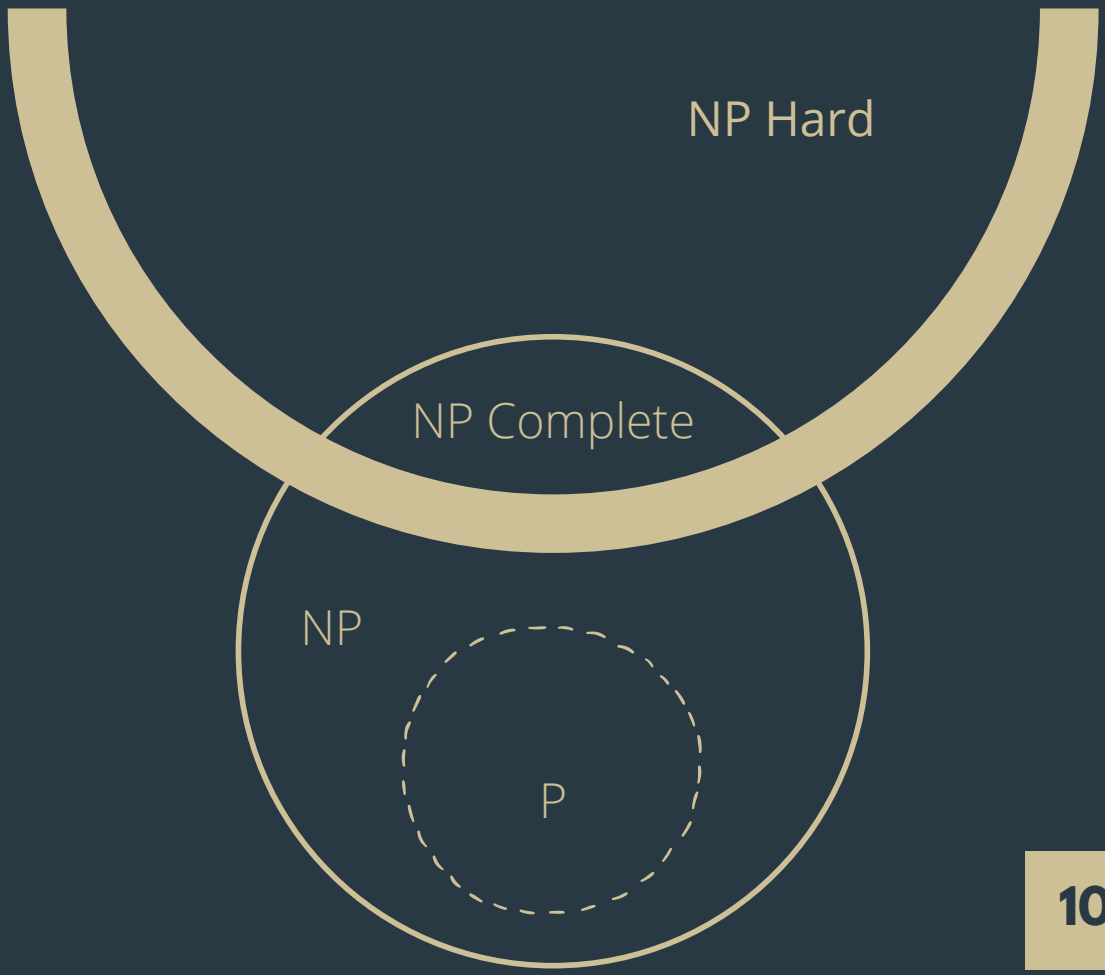
A decision problem L is NP-complete if:

1. L is in NP (Any given solution for NP-complete problems can be verified quickly, but there is no efficient known solution).
2. Every problem in NP is reducible to L in polynomial time

P = NP



P =! NP



DIFFERENCE BETWEEN P & NP

P PROBLEMS	NP PROBLEMS
These can be solved in polynomial time by deterministic algorithms.	These can be solved in non-deterministic polynomial time.
Such problems can be solved and verified in polynomial time.	NP problems solution cannot be obtained in polynomial time but if solution is given it can be verified in polynomial time.
P problems are subset of NP problems.	NP problems are a superset of P problems.
Example: Searching, Sorting, Addition, Multiplication, etc.	Example: Sudoku Puzzle,Travelling Salesman Problem, etc.

REAL LIFE EXAMPLES