Algorithms & Data Structures I Week 19 Lecture Note

Notebook:	Algorithms & Data Structures I		
Created:	2020-10-21 4:14 PM	**Updated:**	2021-01-25 5:48 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** Computational Complexity, Part 1	Course: BSc Computer Science
		Class: CM1035 Algorithms & Data Structures I [Lecture]
		Date: January 25, 2021

Essential Question:

What is computational complexity?

Questions/Cues:

- What is meant by saying a problem is hard?
- What is a complexity class?
- What is a decision problem?
- What are regular languages?
- What is the complexity class P?
- What is the complexity class EXP?
- What is the complexity class NP?
- What is NP Hardness?
- What is NP-Complete?

Notes

- Problem (Hard) = a problem is hard if there no efficient algorithm to solve it
    - In topic 6, we stated that quantifying the input to a problem is the size of the input. That is if we assume our input is stored in an array, where each element of the array stores either a bit or nothing, then the size of the input is the number of elements in that array.
    - So an appropriate quantity in computational complexity is the size of the input according to the standard way of encoding the input into an array with each element being only at most a bit. Then the worst-case time complexity will be computed in terms of the size of the input. Given this, we can now classifying problems in terms of the resources required to solve the problems.
        - Alongside this, since there are multiple possible inputs for each possible size of the input, we'll need to always consider the worst-case
    - Problems can be viewed as set or collection of inputs. All of these inputs will satisfy a particular property as defined by the problem.
        - A complexity class is then the set of all of these problems according to some classification. When a complexity class is thought of we are thinking of a set of sets
    - So every problem is a set of inputs, but is also an element in the class of problems that can be solved with a particular set of resources
- Decision problem = In which a machine will decide whether to accept or reject an input. In simple words, a decision problem is where there are two possible answers. The answers could true/false, accept/reject or maybe correct/incorrect. The essential fact is that the output is binary

- Regular language(s)= a set of words that is accepted by a finite state automaton.
    - A problem defined in a rigorous manner is the set of inputs that are accepted because they share a common property
    - Every input can be considered as a word since it's just a string of symbols (assumed to be in binary). A language is collection of words or strings.
        - So from this, a decision problem defines a language since the inputs or words that are accepted belong to a language and the words that are rejected don't belong to the language

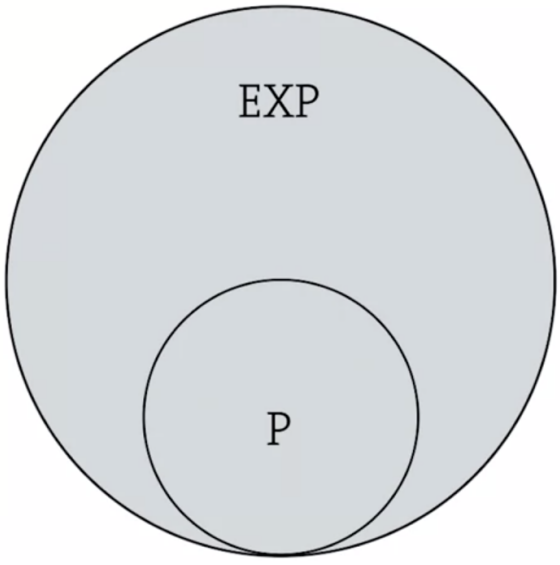
P: the set of all languages that can be decided by an algorithm in the RAM model with worst-case time complexity at most **polynomial** in the **size** of the input

The set of all problems can be decided efficiently

EXP: the set of all languages that can be decided by an algorithm in the RAM model with worst-case time complexity at most **exponential*** in the **size** of the input

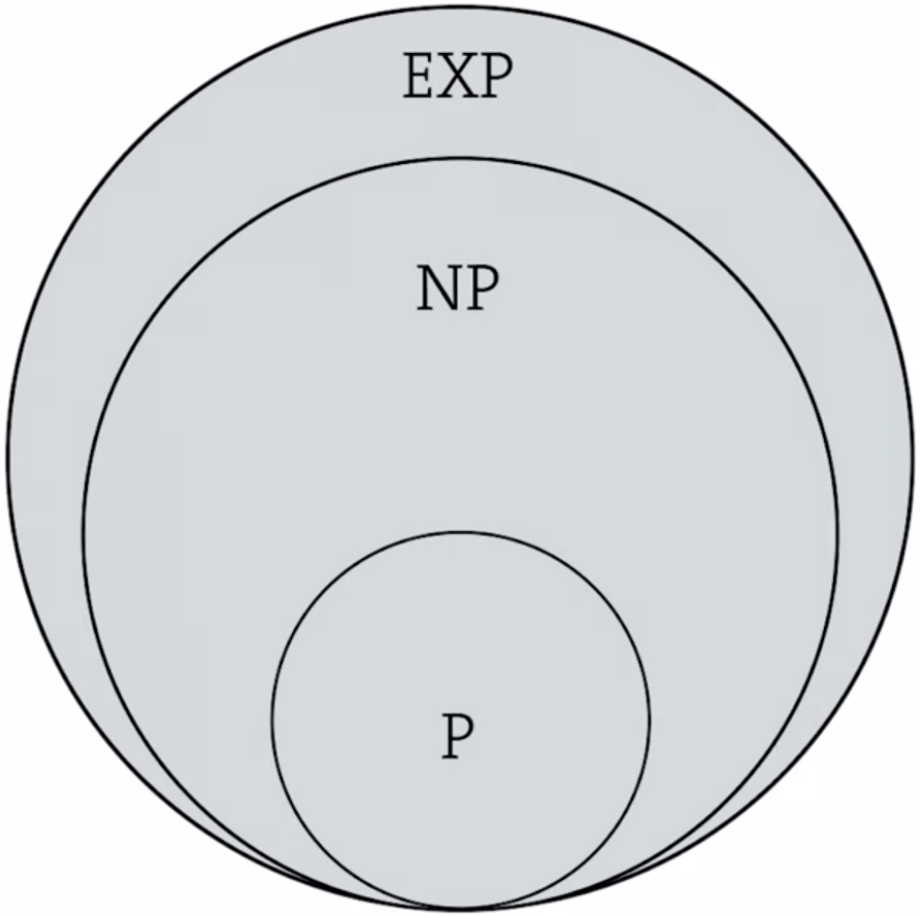
$*O(2^{poly(n)})$

This containment is **strict**



Remember: all $O(n^k)$ are contained in $O(2^n)$

NP: the set of all languages that can be decided by an algorithm in the RAM model **that has access to a proof (of polynomial size)** with worst-case time complexity at most **polynomial** in the **size** of the input



- NP in some sense is the class of all problems where solutions to other problems be checked efficiently. So these solutions are then the proofs in the definition of NP. All problems in P are also in NP, since NP in some sense is P but without the bit about proofs. Furthermore, NP is in EXP since all we need to do is identify all possible proofs and check them, which can be done in exponential time
- So if P is contained in NP, we distinguish the hard problems in NP where there may not be an efficient algorithm for a solution by the notion of NP hardness
- NP hardness denotes problems that are as hard as any of the hardest in the class of NP. More precisely, a problem Q is NP-Hard if an algorithm to solve Q could be then efficiently used to solve another problem R in NP
- A problem is NP-complete if it's both NP-Hard and in the class of NP. So NP-complete can be seen as the hardest problem inside the class NP.
- Because the NP-Complete problems are the hardest in the class of NP, if we could show that these problems have efficient algorithms, then we could show that P = NP, though it's generally held that $P \neq NP$

Summary

In this week, we learned about decision problems and complexity classes.