Algorithms & Data Structures I Week 19 Lecture Note copy

Notebook:	Algorithms & Data Structures I		
Created:	2020-10-21 4:14 PM	**Updated:**	2021-01-25 6:24 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** Computational Complexity, Part 2	Course: BSc Computer Science
		Class: CM1035 Algorithms & Data Structures I [Lecture]
		Date: January 25, 2021

Essential Question:

What is computational complexity?

Questions/Cues:

- What are some possible methods to solve NP-complete problems?
  - What is backtracking?
- What is method of randomized computation?

Notes

- NP-Complete problems(Solutions) = one way to solve NP-complete problems is to identify all possible proofs or candidate solutions to a problem and check them one at a time to see if we can solve the decision problem. The proofs are of polynomial size, so they'll be something like Big O 2 to the polynomial defined possible proofs.
  - With NP-Complete problems, the number of possible proofs will be exponential in the size of the input. So this is kind of like we have to search an exponentially large vector to solve our problem. Referring to linear in this instance, the worst-case time complexity of searching a vector of length n is O(n), we have an exponential time algorithm to solve the NP-Complete problem. Keeping in mind, we don't actually need to construct a vector storing the outputs for each candidate, we only to need to keep track of the relevant index
    - Because of this, hypothetically imagine we apply the BSA to our candidate solutions vector. The vector is still exponentially large, but the worst-case time complexity of BSA is logarithmic in the size of the vector. So taking a logarithm of an exponential of the size of the input will return the input size as the time complexity
      - So we could apply BSA to search our proofs for our NP-Complete problem, we could hypothetically show that P = NP, which is unlikely
  - Backtracking in terms of a NP-Complete problem means that if we a candidate solution or proof consisting of several values, we can try one value at a time to see if it leads to a correct solution and if at some later point all subsequent values fail to lead to a solution, we can go back and change that previous value. This referred to as backtracking
    - Backtracking is typically implemented recursively and so a call stack is used in this instance
- Randomized computation = a computer utilizes random numbers as a extra resource to help it decide which operation to do next. That is, the order of operations is not completely fixed but up to chance

Summary

In this week, we learned about possible methods for solving NP-Complete problems