P/NP

Classifying Problems

Goal: Classify problems according to those that can be solved in polynomial-time $O(n^k)$ and those that cannot.

Huge number of fundamental problems have defied classification for decades.

We will look at a 3rd class of problems "computationally equivalent" and appear to be different manifestations of one really hard problem.

Class P

The class P consists of those problems for who exists an algorithm that can solve the problem in polynomial time.$O(n^k)$ time, for some constant k, where n is the size of the input to the problem. We say they are poly-time solvable.

Bellman-ford -->algorithm not a problem

Problem of shortest path with negative weights ---> belongs to class of P

Who is solvable in poly-time?

Longest Common Subsequence (LCS)?

Activity selection?

Class NP- Nondeterministically polynomial.

The class NP consists of those problems that are "verifiable" in polynomial time. What do we mean by a problem being verifiable? If we were somehow given a "certificate" of a solution, then there exists an algorithm that can verify that the certificate is correct in time polynomial in the size of the input to the problem.

- using exponential number of threads can be solved in poly-time

Meaning of NP

Given a hint to the solution --- i can solve the problem using an algorithm in poly-time!

Question

Is LCS problem NP?

Answer

Is LCS problem NP? Yes

Because there exists an algorithm that verifies given certificate and the problem in poly-time

Proof LCS in NP

By construction:

1.) Provide an algorithm that verifies a certificate given the instance of the problem
2.) Show that run-time of the algorithm is poly-time O(n^k)

Part 1 Algorithm

verifyLCS::

Input:

- An instance of the problem - two specific strings x ,y
- Certificate: solution common **subsequent and length**

Output:

True if the certificate verifies the solution to LCS of x,y

Part 1 & Part 2

Algorithm:

1.) Run LCS dynamic programming algorithm and compare the length of the certificate to the length returned by the dynamic program → $O(n^2)$
2.) Check that the subsequence of the certificate exists in both x and y → $2*O(n)$

Part 2:

The total run-time of step 1 & 2 is $O(n^2)$ → algorithm is poly-time

Small example

x=catty y=satty certificate=atty

1.) Lcs-dynamic(x,y) compare the result with atty
2.) Double-check that "atty" is a subsequence in both sequence x and y.

All P problems belong to NP

We can create a verification algorithm for any problem because by definition P problems are solved in poly-time and has verification can compare the certificate to the solution in p time.

P subset of NP ...

P=NP??

Proving a problem belongs to P

By construction → create an algorithm that solves the problem and show that it does so in poly time.

Does not need to be the most efficient solution just needs to be poly time.

Examples of questions for the final

Question: Prove the problem of Printing Neatly from hw4 is in P.

Solution: Provide an algorithm that solves Printing Neatly and its run-time. And state bc there exists an algorithm that solves the problem in poly-time the problem is in the class P.

Question: Prove the problem of Maximum Spanning Tree is in P.

Solution: Create an algorithm that solves maximum spanning tree problem and show that its run-time is is polynomial.

Question: Prove LCS is in NP.

Solution: See above algorithm and run-time. **Please note we didn't have to restate the LCS algorithm. Anythign we covered in class by name can be just state as run dynamic LCS from class.**

Takeaways

P class of all problems that have poly run-time.

NP does not mean none polynomial it actually stands for nondeterministically polynomial which is a set of problems that can be solved by running parallel threads of poly algorithms.