Exercises of chapter 5

February 26, 2020

Mayra Cristina Berrones Reyes.

1 Introduction

In previous excersies we analyzed the computational complexity of some of the more known problems. The efficiency and complexity to solve such problem gives us a clue as to how much time and space it will use to find an answer. There is a general knowledge of problems class $\mathcal{P}$, which are a set of problems that can be solved by a deterministic Turing machine in Polynomial time. Some examples of this problems can be seen in Figure 1. Then we have the $\mathcal{NP}$ problems, which can be described as a set of problems that can be solved by a non deterministic Turing machine in polynomial time [4].

Given this statement we can assume that $\mathcal{P}$ is a subset of $\mathcal{NP}$, because any problem that can be solved by a deterministic machine, can be solved by a non deterministic one in polynomial time. Next we have the $\mathcal{NP}$–complete problems, which are the hardest problems in a $\mathcal{NP}$ set. A $\mathcal{NP}$–complete problem can be verified in polynomial time, but there is no efficient known solution.

If we take this definition of $\mathcal{NP}$–complete problem, it seems impossible to prove that a problem $\mathcal{A}$ is $\mathcal{NP}$–complete. An alternative way of proving this, is to use reductions. In computational complexity theory a polynomial reduction can be computed by a deterministic Turing machine in polynomial time. Using reductions is very important because we can make many transformations between problems to prove their complexity, but to this date there is no reduction of a $\mathcal{NP}$ or $\mathcal{NP}$ complete problem in $\mathcal{P}$.

Now, given the concept of reductions, we can use them to prove that a problem is $\mathcal{NP}$–complete using the properties of transitivity. In [4] we can see some lemmas that helps us prove our problem, as seen below.

A decision problem $\mathcal{B} \in \mathcal{NP}$ is $\mathcal{NP}$–complete if:

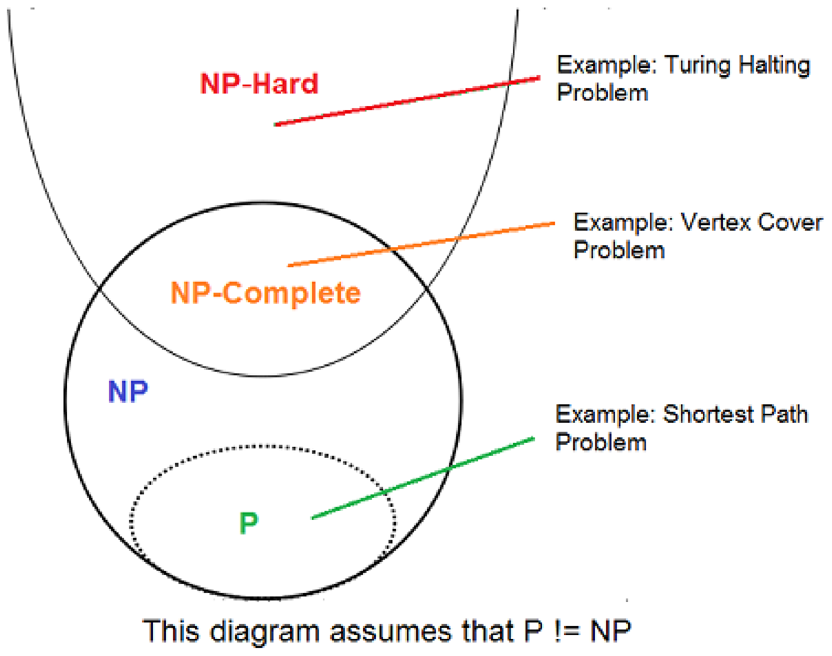

NP-Hard
Example: Turing Halting Problem
Example: Vertex Cover Problem
NP-Complete
NP
Example: Shortest Path Problem
P
This diagram assumes that P != NP

Figure 1: Example of some $\mathcal{P}$ and $\mathcal{NP}$ problems [4]

$$\mathcal{A} \leq_P \mathcal{B} \; \forall \; \mathcal{A} \in \mathcal{NP} \tag{1}$$

If you can solve $\mathcal{B}$ in polynomial time, then every other problem $\mathcal{A}$ in $\mathcal{NP}$ would be solvable in polynomial time. Given the rules of transitivity we have:

- $\mathcal{B} \in \mathcal{NP}$ and
- $\mathcal{A} \leq_P \mathcal{B}$ for some $\mathcal{NP}$ problem $\mathcal{A}$

2 Exercise 3. Hamilton

The HAMILTONCYCLE problem is defined as follows: Given a conected graph $G = (V, E)$, does the graph contain a tour that visits each node $v \in V$ exactly once? The HAMILTONPATH problem is defined as follows: Given connected graph $G = (V, E)$, does this graph contain a path that visits each node $v \in V$ exactly once? In HAMILTONPATH you can select a starting node and then visit all other nodes; you do not have to return to the starting node.

- Prove that HAMILTONCYCLE is $\mathcal{NP}$–complete by reduction from HAMILTONPATH.

This problem was a bit more difficult to understand than the one below (which I did first), because I found several different ways of seeing it. The first one was that given the example below, about the reduction of the Hamitlon cycle, if we rejoin the two vertex v^1, and v^2, we obtain the vertex V and the path gets converted into a Hamilton cycle, proving that if the Hamilton path is $\mathcal{NP}$–complete, then the Hamilton cycle must be $\mathcal{NP}$–complete as well. Another way of viewing it, is if you add another vertex to the graph, and use it to join the begin node and end node of the path. This way as well as the first one, we prove that the Hamilton cycle is $\mathcal{NP}$–complete with the verification of $\mathcal{NP}$–complete we had of the Hamilton path.

- Prove that HAMILTONPATH is $\mathcal{NP}$–complete by reduction from HAMILTONCYCLE.

So, this is the first one that I understood, thanks to an explanation by [3]. Basically what we do here is, we have a Hamiltion cycle $\mathcal{A}$, and we would like it to help us prove that a Hamilton path $\mathcal{B}$ is $\mathcal{NP}$–complete. So first we make sure that $\mathcal{A}$ is a cycle. After we have our verification, we take the node v in which the cycle started, and make it into two nodes, v^1, and v^2; v^1 only has incoming direction, and v^2 only has outgoing direction. Now, after doing this, we have a hamiltonian path, which can be proved to be $\mathcal{NP}$–complete with the verification that we had of the cycle $\mathcal{A}$, that this (cycle $\mathcal{A}$) was $\mathcal{NP}$–complete. With this we show that the answer to the reduced problem is the answer to the original problem, thanks to the rules of transitivity that we discussed earlier.

3 Exercise 5. Parcels and two trucks

A company has two trucks, and must deliver a number of parcels to a number of addresses. They want both drives to be home at the end of the day. This gives the following decision problem.

Instance: Set V locations, for each pair of locations $v,\, w \in V$, a distance $d(v,\, w) \in \mathbb{N}$, a starting location $s \in V$, and an integer K.

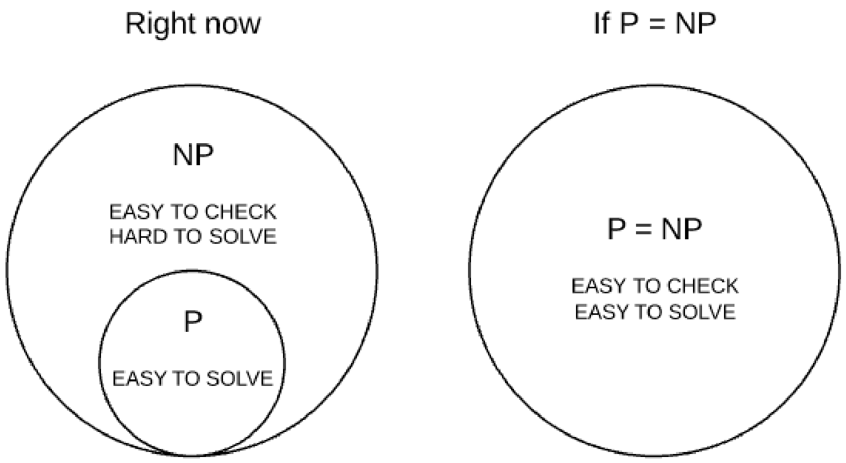
Question: Are there two cycles, that both start in s, such that every location in V is on at least one of the two cycles, and both cycles have length at most K?

Show that this problem is $\mathcal{NP}$–complete.

This is similar to the TSP problem. The only difference is that there are two trucks, and I have a little bit of a hard time trying to form the answer. I think this reduction can be managed as a sub problem type of thing. What I mean by this, is something along the lines of proving that with only one driver, the problem is $\mathcal{NP}$–complete. In that case, I can assume that two drivers is also $\mathcal{NP}$–complete, based on the concepts we stablished for reductions and their rules of transitivity. So in this my graph $\mathcal{A}$ is one cycle that goes trough all the vertex with only one driver. And my graph $\mathcal{B}$ is the one made by the two drivers. Given the implication that $\mathcal{A} \in \mathcal{NP}$–complete, then $\mathcal{B}$ is also $\mathcal{NP}$–complete, since a general problem must be as difficult to solve as one of their subproblems, which is what I understood from the reductions. So, if $\mathcal{A}$ is verified as a TSP problem, then by theory of computational complexity, we know that this problem belongs in the $\mathcal{NP}$–complete class, therefor we can say that $\mathcal{B}$ is also $\mathcal{NP}$–complete.

4 Conclusions

For this exercises, I had to do a lot of research, and the videos by [1, 2, 3] really helped me grasp the concept of reductions. At the end of this work, my concept on this subject is that reductions help analyze the complexity of different problems, and I can see why is important to study them, specially after reading about the $\mathcal{P}$ vs. $\mathcal{NP}$ debate that is going on. If someone can find a reduction that can simplify a $\mathcal{NP}$–complete problem, than maybe they can finally reach the connection to polynomial resolution of all of the different $\mathcal{NP}$–complete problems.



5 Reference

References

[1] Design and Analysis of algorithms. Lecture notes March 1996. `https://www.ics.uci.edu/~eppstein/161/960312.html`

[2] NP complete problems. UHMICSAlgorithms. `https://youtu.be/J5l-crl0LgA`

[3] NP complete problems. MIT OpenCourseWare.`https://youtu.be/G7mqtB6npfE`

[4] K. Seshagiri. Study of NP-complete problems: Polynomial time reductions and solutions. Indian Institute of Information Technology.