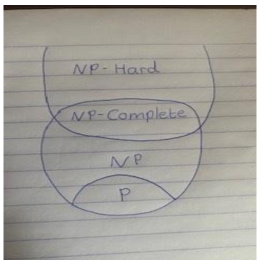
Assignment 4: P, NP, NP-Hard and NP-Compete Problems

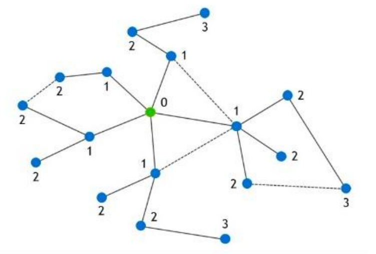


1.

a) Polynomial problems or ‘P problems’ are algorithmic problems that are solvable in polynomial time which means they can be classified as ‘tractable’. Tractable means that the algorithm can be solved efficiently with regard to time. The algorithm’s time complexity is O(p(n)) where p(n) is a polynomial function. Polynomial time means that the algorithm grows at most polynomial with respect to input size. Polynomial problems are the easiest problems to solve, examples of these problems in dynamic programming are Djikstra’s shortest path algorithm and binary search algorithm. P problems are an important study of computer science as it helps us understand the efficiency of algorithms. It enables us to create effective algorithm designs for real-world problems such as image processing in health care which can lead to better diagnosis and treatment for patients. This is done by the processing data with efficient algorithms.

b) Example: Dijkstra’s Shortest Path Algorithm

This shortest path algorithm has drastically improved the overall efficiency of logistics in the world. Logistics is the commercial transport of goods to customers, the overall goal is to transport goods from one location to another in the most efficient way possible whether that is through land, air or sea. By using Djikstra’s shortest path algorithm logistic company’s can find the most logical and practical path from the warehouse to the customers location in polynomial time. This would involve taking into consideration the transport costs, travelling routes and cost. Overall this will in turn create a more efficient transport system and therefore resulting in reduced costs, reduced delivery times and greater customer satisfaction. Djikstra’s shortest path algorithm improves the efficiency of logistics from one destination to another. It improved the overall time efficiency of travel in the world including apps such as Google Maps.



https://developer.nvidia.com/discover/shortest-path-problem

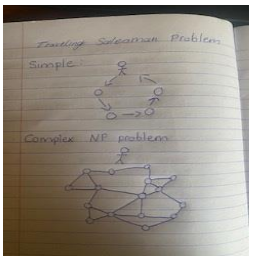
2.

a) Non-deterministic polynomial problems or ‘NP problems’ are algorithmic problems that cannot be solved in polynomial time, however they can be guessed and verified in polynomial time. The term non-deterministic means no particular rule is followed to make the guess. NP problems are intractable as they have an exponential complexity i.e. O(n^n).

These problem are as hard or harder than P problems that we haven't found an easy solution for in polynomial time. P is a subset of NP i.e. any problems that can be solved in deterministic machine in polynomial time can be solved by a non-deterministic machine in polynomial time. Examples of NP problems would be scheduling and routing. Np problems are difficult to solve as most are solved in exponential time. Np problems are important in so many aspects of the world such as cryptography, overall it functions our brains to understand the limitations of computational power and try to crack these limitations to turn them into P problems.

b) Example: Traveling salesman problem (TSP)

Given a set of cities and the distance between them, find the shortest possible route that visits each city exactly once and returns to the starting city. The TSP aims to increase the efficiency of the transport system as a whole, you should look at the shortest path algorithm as a subset of this problem, the number of permutations and combinations of different routes increases past the capabilities of computers, 15 destinations – possible routes = 87 billion. To solve this problems you could use the brute force approach or the nearest neighbour approach. To solve with brute force you must calculate the total number of routes and then list all the possible routes. Calculate the distance of each route and then list all the possible routes. Calculate the distance of each route and then chose the shortest one. By solving the traveling salesman problem, delivery companies or logistics can improve their overall efficiency as less distance is travelled and more fuel is saved. We can showcase this with graphing.

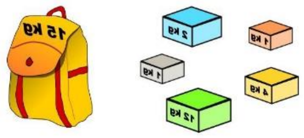


3.

a) NP-Hard Problems are at least as hard as if not harder than NP-complete problems. They are intractable as they have an exponential complexity and cannot be verified in polynomial time, the time to solve the problem is unknown. A problem is NP-hard if there is an NP-complete problem Y, where Y is reducible to X in polynomial time. The reason these NP-hard problems are classified as hard is because we have not found an efficient algorithmic solution to solve in polynomial time. NP-hard problems include the knapsack problem, the traveling salesman is also apart of the NP-hard problems too and the Boolean satisfiability problem. NP-hard problems involve the most complex problems in the world today, we continue to study them to improve the overall understanding of computational complexity and therefore finding efficient algorithms to solve them will have significant impact on our world today i.e. AI.

b) Example: The Knapsack problem

We are given M items where each item has a weight W and a knapsack with a capacity C. The reason this problem is so important is because it can be modelled as many real world problems such as machine scheduling and space allocation. The problem aims to provide the understanding and importance of how we effectively gain the greatest value for the task given at hand, it is an optimization problem. In algorithmic scheduling we can use the knapsack problem to spend our memory wisely i.e using pre-emptive algorithms to improve overall time efficiency. It is classes as an NP-Hard problem due to its exponential complexity but we continue to study this problem in the search for optimising the algorithmic approach and therefore find faster solutions.

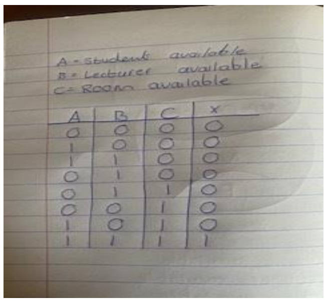


4.

a) Np-Complete Problems are part of both NP and NP-Hard and are as hard as NP problems. Any problem in NP can be reduced to NP-Complete in polynomial time. All problems are unknown to whether they are tractable or intractable as the we can estimate the solution in polynomial time but the answer won't be obtained in polynomial time. They are apart of the harder problems in NP. NP-Complete are decision problems. These are important as if we can find an efficient algorithm for NP-complete problems that would mean an efficient algorithm can be found for all NP and NP complete problems.

b) Example: Boolean satisfiability problem (SAT)

In this problem we determine whether a Boolean formula is satisfiable: if the Boolean value can be assigned values that result in the formula being true, or unsatisfiable: if it is not possible to assign values that can make the formula true. SAT can be applicable to many areas such as travel, logistics and software/hardware design. SAT problems are written in conjunctive normal form using the logical AND/OR statements. An everyday example would be scheduling college lectures as we must satisfy true that it will only be applicable if and only if, any student in the module must not have another class, the lecturer must only have that class at that specific time and the classroom must be scheduled for only that class. If one value hold false then we return false. i.e. (x1 ^x2^x3) must all be true.

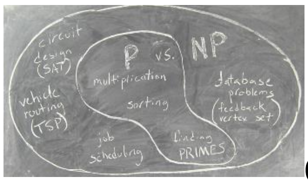


A = Student available
B = Lecturer available
C = Room available

A	B	C	x
0	0	0	0
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
1	0	1	0
1	1	1	1

5.

a) P stands for polynomial which refers to the time it can be solved in, it is a class that contains the problems that can be solved very easily such as multiplication. NP problems which means non-deterministic are outside and including P. They are harder problems that we haven't found optimal solutions for in polynomial time i.e. exponential complexity. Over time we have found many NP problems to have P solutions which has given rise to this question. P problems are more like puzzles as they can be checked if they are correct in no time when given the answer whereas NP problems are hard to check an answer e.g. best chess moves. For most NP problems the scalability of the problems make them hard to solve for all solutions. The P vs NP problem is so hard to crack is because the proving the question itself is an NP problem.



(https://youtu.be/YX40hbAHx3s)

b) The NP vs P problem is part of the Millenium Prize Problems that the Clay institute offered one million dollars if anybody could find the solution. It was conceived in 1971. It is considered one of the most important problems. P is a subset of NP but NP problems cant be inside of P. The question arose when we started to find solutions optimal for NP problems that caused us to think, do all NP problems have a P solution? Or are there just some problems that are impossible to find in polynomial time? If we were to find the solution to P vs NP the world would change almost instantly, problems such as curing cancer and online banking cryptography would become easy to solve.

- Baeldung, November 25, 2022, *P, NP, NP-Complete and NP-Hard Problems in Computer Science*, 11th march 2023, (https://www.baeldung.com/cs/p-np-np-complete-np-hard)
- Geeks for Geeks, 24 Feb 2023, *Knapsack problem*, 12/03/2023, (https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)
- Routific, Jan 1st 2020, *Solving The Traveling Salesman Problem For Deliveries*, 14th March 2023, (https://blog.routific.com/blog/travelling-salesman-problem)
- Hackerdashery, 26TH Aug 2014, *P vs NP and the Computational Complexity Zoo*,14th March 2023, (https://youtu.be/YX40hbAHx3s)
- Michael Sipser, *The history and status of the P versus NP Question*, 14th March 2023, (https://www.cs.princeton.edu/courses/archive/spr09/cos522/SipserNP.pdf)