## 1a.What is a polynomial problem?

A polynomial problem is a problem complexity class that includes deterministic algorithms that can be solved in a “reasonable” time. They are tractable, “easy” problems where you can both check and solve the solution in polynomial time, meaning the upper-bound of the problem is polynomial, and are also a subset of NP problems. Polynomial time means that the complexity of the algorithm is at it’s most complex O(n^k), where n is the input size and k is some constant, and can have a complexity as simple as O(1), constant time. In other words, it’s complexity is O(p(n)), where p(n) is a polynomial function. As these problems can be solved by an algorithm in polynomial time, we can say that Polynomial problems are tractable.

## 1b. Example of a polynomial problem.

An example of a P problem would be finding the sum of all integers in an array. The algorithm for this problem would involve iterating through the array and adding the sum of each number to a variable. This algorithm would have a time complexity of O(n), as the algorithm increases in complexity as the array grows in size. Testing this algorithm would also have a time complexity of O(n), as it would consist of finding the sum of the array and checking the sum against some value, therefore, as both solving and checking the solution for this algorithm has a time complexity of O(n), it is considered a P problem.

## 2a. What is a non-deterministic polynomial problem?

A non-deterministic polynomial problem is a problem class where there exists an algorithm that can verify a solution in polynomial time, but only solve the problem in at the quickest exponential time, an algorithm that can both solve and check a solution in polynomial time for an NP problem that is not a P problem does not exist. NP problems are non-deterministic, meaning that the algorithms used do not rely on the current state and inputs to make decisions. Both P problems and NP-complete problems are subsets of NP problems, as they both share the same characteristic of NP problems of being able to verify a solution in polynomial time, the main difference between NP and NP-complete problems is that NP-complete problems are at least as hard as the hardest problems in NP.

## 2b. Example of a NP Problem

An example of an NP problem would be the Subset Sum problem. The problem is : if given a set of elements S, and a target sum T, is there a subset of S where the elements in that subset add up to T? This problem is proven to be able to be verified in polynomial time, for example if S = {2, 4, 6, 8, 10, 12} and T = 20, if we’re given a subset {4, 6, 10}, we can easily check if this subset adds up to 20 in polynomial time. However, it hasn’t been proven that we can solve this problem in polynomial time yet, as if given the previous set and target sum, for the algorithm to try find a subset that adds up to T , at the worst the time complexity would be 2^n, or with the set {2, 4, 6, 8, 10, 12}, 2^6, which is exponential,  therefore this problem is in the NP class.

## 3a. What is a NP-Hard Problem?

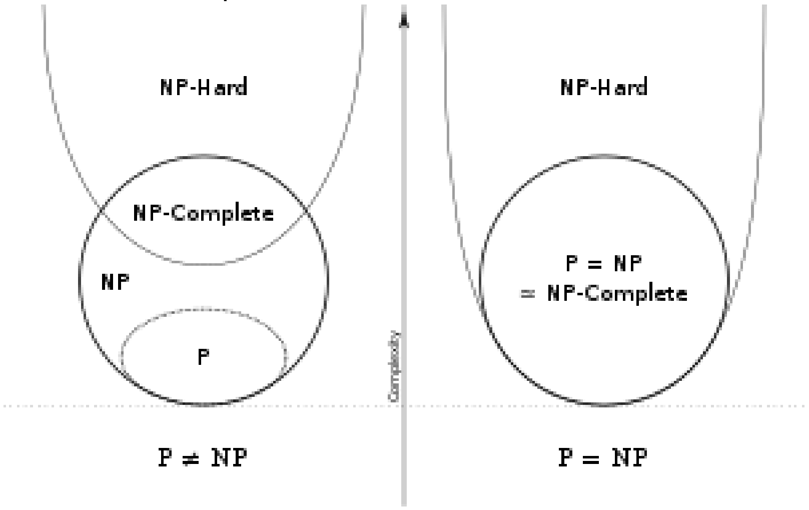
A NP-Hard problem is a class of problem where all problems in NP-Hard are at least as hard as the hardest problems in NP-Complete, or harder, in other words, where all problems in NP-Complete are polynomial reducible to it. Reducing a problem consists of rephrasing a problem to another problem so that the solution for the second problem can be used to find a solution for the first one, to expand on this: say you have Problem A and Problem B. if you have a solution to verify an answer for problem B, and you can reduce or “rephrase” problem A so that it is the same as Problem B, than you can use the solution for problem B on problem A. Problems in NP-Hard typically apply to, but are not contained  to decision problems, where the answer has a yes or no output, this includes verifying answers. Not all problems in the NP-Hard class are in the class of NP, because problems in NP-Hard cannot be solved in polynomial time and not all NP-Hard problems have solutions that can even be verified in polynomial time. Instead they are defined based on whether they are reducible to problems in NP. The only problems in NP-Hard that can verify solutions in polynomial time that are also found in the class of NP, have their own class called NP-complete,  I will explain more about NP-Complete in the next question.

## 3b. Example of a NP-Hard Problem.

An example of a NP-Hard problem is the travelling salesman problem. A salesman needs to visit a set of cities exactly once and return to the original city he started in, in the shortest route possible. This problem can be viewed as both NP-Complete and NP-Hard depending on which way it is perceived, there is a decision problem where the problem is to check whether a Hamiltonian cycle in a fully connected graph has a certain pathlength, this decision problem is NP-Complete, as this solution can be verified in polynomial time. Then there is the NP-Hard problem, the optimization problem, to find the shortest length Hamiltonian cycle of a fully connected graph. This problem cannot be solved in polynomial time as for n cities(nodes), there are (n-1)! Possible paths, giving it a complexity of O((n-1)!) in the worst case. The traveling salesman problem can be polynomially reduced to other NP-Complete problems, which satisfies the conditions for an NP-Hard problem This problem is often used to represent network optimization where data packets need to be forwarded between multiple routers, and the traveling salesman problem can be used to find the shortest route for the packets to use to reach their destination.

## 4a.What are NP-Complete problems?

NP-Complete problems refer to the subclass of problems which are essentially NP-Hard problems which are also included in the NP class, or NP-Hard problems which can verify a solution in polynomial time. These problems are at least as hard as the hardest problems in NP, and any NP-Complete problem can be reduced to an NP problem in polynomial time, but not P. As of now, no problems in NP complete can be solved in polynomial time. Much like NP-Hard problems, if it was found that an NP-Complete problem was able to be solved in polynomial time, it would finally solve the unanswered question of is NP=P. Here is the Euler diagram that shows the relationship between all the problem classes:



https://en.wikipedia.org/wiki/File:P_np_np-complete_np-hard.svg

## 4b. Example of a NP-Complete Problem.

The Boolean Satisfiability Problem is a NP-Complete Problem as given a solution, it can be verified in polynomial time, and is also as hard as the hardest problems in NP, meaning it is in both classes of NP-Hard and NP. The problem is about finding an assignment of truth values for variables in a given Boolean formula consisting of logical operators like AND, NOT and OR that makes the entire formula equal to true. It typically has 3 inputs, A,B, and C, and these variables are assigned truth values, for example, A=1, B=0, C=0, and put through the logical formula and checking if the formula will output a 1(true). What makes this problem NP-Complete is that given a potential solution, the solution can be verified in polynomial time, which means it is a part of the NP class. The problem cannot be solved in polynomial time, as it's complexity to solve in the worst case is O(2^n), which is exponential. Finally, what makes it NP-Complete is that it satisfies the condition that it is at least as hard or harder than the hardest problems in NP, meaning that it’s NP-Hard, and every problem in NP can be reduced to the Boolean satisfiability problem in polynomial time. Meaning it is both NP and NP-Hard, and therefore NP-Complete.

## P versus NP problem.

The overall question of this problem is asking whether every problem whose solution can be verified in polynomial time can also be solved in polynomial time. The question particularly relates to if an algorithm for a NP-Complete problem can be found that solves the problem in polynomial time, NP-Complete = p, as if there existed an algorithm that could do this, any problem in NP could be reduced to this as NP-Complete problems are as hard as the hardest problems in NP. The question was introduced by mathematician Steve Cook in 1971 in his paper ”The Complexity of Theorem Proving Procedures”(Fifty Years of P vs NP – Lance Fortnow). The question still hasn’t been solved to this day. There is a 1 million dollar reward being offered by the Clay Mathematics institute to anyone who can prove if P=NP. If it were found that every problem that could be verified in polynomial time could also be solved in polynomial time, it would have great benefits in almost all fields of research and would greatly improve sectors such as urban development, building complex road systems, computer science and AI development, cancer research and much more. However, one downside for P being equal to NP would allow hackers to easily access private data that relies on complex cryptography being difficult to solve, and sensitive data like bank details, passwords and private messages would be easily accessed by hackers, “If P = NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps", no fundamental gap between solving a problem and recognizing the solution once it's found.”- Scott Aaronson, UT Austin. Many computer scientists believe that P is not equal to NP as there is nearly no evidence supporting that they are equal, however this claim is not strong enough to be truly accepted.

## Sources:

https://link.springer.com/chapter/10.1007/978-3-031-17043-0_15

https://www.quora.com/What-is-the-difference-between-P-NP-and-NP-complete

https://www.britannica.com/science/NP-complete-problem

https://youtu.be/EHp4FPyajKQ

https://www.jntua.ac.in/gate-online-classes/registration/downloads/material/a159262902029.pdf

https://cacm.acm.org/?s=p+vs+np&orderby=relevance&fs%5Bcustom_date_range%5D%5B%5D=1958-01-01T00%3A00%3A00%2B00%3A00&fs%5Bcustom_date_range%5D%5B%5D=2024-03-05T13%3A34%3A27%2B00%3A00

https://en.wikipedia.org/wiki/P_versus_NP_problem#:~:text=If%20P%20%3D%20NP%2C%20then%20the,the%20solution%20once%20it's%20found.

https://www.quora.com/Why-is-the-traveling-salesman-problem-NP-complete#:~:text=The%20traveling%20salesman%20problem%20is%20a%20problem%20in%20graph%20theory,the%20problem%20is%20NP%2Dhard.

https://www.youtube.com/watch?v=uAdVzz1hKYY

(lecture notes)
https://universityofgalway.instructure.com/courses/9451/pages/3-dot-1-topic-three-lecture-notes?module_item_id=453267