Cathal McSweeney:19731485

RESEARCH ASSIGNMENT OF POLYNOMIAL PROBLEMS

1. Polynomial Problems:
    a. Definition:

    This type of problem also referred to as "P" is when an algorithm can be solved in polynomial time $O(n^k)$ (k = constant, n = input size) this also includes any problems with a time complexity of less than polynomial time such as quadratic, linear, logarithmic and constant time. Polynomial problems are easy to solve with the problem also being easy to verify that the answer is correct. Another word for these algorithms is deterministic problems which means the algorithm is not random and is essentially a state-machine where depending on what state it is in the next state is decided. Any given input will always produce the same output given that the input is the same.

    b. Example:

    A Binary to Decimal algorithm that identifies the binary representation of a decimal number is an example of a Polynomial Problem where the program just needs to modulus any given input by 2 with the result being stored in an array which upon completion is reversed.

2. Non-Deterministic Polynomial Problems:
    a. Definition:

    A Non-Deterministic Polynomial problem also referred to as NP are problems that cannot be solved within polynomial time but can be verified as correct within Polynomial time. These problems may have more than one solution to be found resulting in a time complexity analysis of $O(k^n)$. Verifying that a solution has been found however is found within polynomial time.

    b. Example:

    An example of a NP problem would be a sudoku game where finding the solution to the game would not be able to be complete within polynomial time however checking if a solution was correct would be faster by checking each row, column and 3x3 square would prove faster than finding a solution.

3. NP Hard Problems:
    a. Definition:

    NP Hard problems are considered the hardest problems in computer science. This is a result of them not only being hard to solve but also being hard to verify. To be classified as NP-Hard the algorithm for solving it must be able to be translated to solve any NP problem, then we can say that the problem is as hard as any other NP problem but is much more complex.

    b. Example:

The Travelling Salesman problem is a problem where a salesman must travel between N cities in any order. He must finish where he started and visit each city once. The aim of this problem is to keep the “cost” as low as possible. Each trip between nodes differs depending on length and type of journey.

4. NP Complete Problems:
   a. Definition:

      NP-Complete problems are very similar to NP but are amongst the hardest of these problems. These problems are considered to be complete when they can be transformed into any other NP complete problem through reduction in polynomial time. They are both NP and NP-Hard.

   b. Example:

      The knapsack problem in which an algorithm is used to identify the what items and how many, each with a different value, can make up a collection to that the total weight is less than or equal to what an object can carry while determining the value is as large as possible.

5. P vs NP:
   a. P vs NP was first introduced by Stephen Cook in 1971 and is considered today as one of the most important open problems in computer science. It is one of the Seven Millennium Prize Problems carrying a $1,000,000 prize for a proof. The problem focuses around whether a question which can be verified within polynomial time can be solved within polynomial time. If this is the case there would be massive implications both negative and positive not only in the computer science community but also in other areas such as biotechnology. In the case where P != NP, which is widely believed by a number of those in the computer science community, would have less profound implications however it would give guidance for future research. It would show in a formal capacity that many common problem and algorithms cannot be solved efficiently.

References

baeldung. (2020, October 20). *P, NP, NP-Complete and NP-Hard* . Retrieved from baeldung.com: https://www.baeldung.com/cs/p-np-np-complete-np-hard

Daniel, G. G. (2013). *Deterministic and Nondeterministic Turing Machine*. Retrieved from link.springer.com: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4020-8265-8_200983

Juho. (2013, Aug 07). *cs.stackexchange.com*. Retrieved from stackexchange.com: https://cs.stackexchange.com/questions/13625/what-exactly-is-polynomial-time

maycontainmaths. (2020, Dec). *p-vs-np-for-dummies-part-1*. Retrieved from maycontainmaths.wordpress.com: https://maycontainmaths.wordpress.com/2014/12/21/p-vs-np-for-dummies-part-1/#respond

Yun, P. (2019, aug 27). *P, NP, NP-Hard and NP-Complete Problems*. Retrieved from medium.com: https://medium.com/@p.yun1994/p-np-np-hard-and-np-complete-problems-fe679bd1cf9c