Assignment 4 - CT2109 Object Oriented Programming: Data Structures and Algorithms

Daniel Hannon (19484286)

April 2021

1 Overview

The aim of this report is to give an Overview/Definition and example of:

- Polynomial Problems
- Non-Deterministic Polynomial Problems
- NP-Hard Problems
- NP-Complete Problems

And an introduction to the P versus NP problem along with some background and the reprecussions of the result when it is inevitably reached.

2 Polynomial Problems

2.1 Outline of a Polynomial Problem

A Polynomial Problem is that can be solved by a Deterministic algorithm in Polynomial time. A Deterministic algorithm is an algorithm that always reproduces the exact same results given the exact same inputs by performing the exact same steps. As a result problems in P can be defined as "Tractable" where a tractable problem is one that can be solved in reasonable time. They can essentially be boiled down to

$$O(p(n))$$

where *p(n)* denotes a Polynomial algortihm to calculate the time to solve it.

2.2 Example of a Polynomial Problem

A fairly straightforward example of a Polynomial Problem is the Karatsuba algorithm for multiplying two large numbers, it has an approximated complexity of:

$$O(n^{1.585})$$

and it is a proven more efficient method of multiplying two numbers than the classic schoolbook method it can be performed recursively.(Weiss 2006) It can be done in the following way:

1. Check if either number is less than 10, if yes return the product of both numbers.
2. Get the highest shared whole-number log10 value between the two numbers.
3. Record the value from step 2 divide it by two and round down.
4. Split the numbers at this log of 10 into the form $leading\_digits * 10^{step2\_value} + following\_digits$
5. Start from step 1 again with the following_digits values from both numbers and record this result
6. Start from step 1 again with $leading\_digits + following\_digits$ for either value and record this result
7. Start from step 1 with the leading digits from step 4 for either number and record the result
8. Return
$$(s\_7\_res * 10^{(s\_3\_res*2)}) + ((s\_6\_res - s\_7\_res - s\_5\_res) * 10^{s\_3\_res})) + s\_5\_res$$
where the form s_x_res denotes result from step x

As it follows a Deterministic algorithm to achieve a result and it is tractable it belongs to class P.

3 Non-Deterministic Polynomial Problems

3.1 Outline of a Non-Deterministic Polynomial Problem

A Non-Deterministic Polynomial problem is one that uses a Non-Deterministic Algorithm as a means to generate a potential solution, either randomly, or systematically, then it verifies whether it is true or not and repeats ad-infinitum unless it finds a solution.
If the verification step is Polynomial then the algorithm and by extension, the problem belong to NP. This of course implies that P is a subset of NP as anything that can be solved in Polynomial time can by definition be verified in Polynomial time.

3.2 Example of a Non-Deterministic Polynomial Problem

Although very often frowned upon, cracking a password by means of a brute force would constitute as an NP problem as it uses some form of algorithm to generate a potential password and then it is checked against the password and whether or not it is the password can be verified in Polynomial time.(Atkins-Bittner 2014)

4 NP-Complete Problems

4.1 Outline of an NP-Complete Problem

An NP-Complete Problem is a problem that is as at minimum, as hard as all other problems in NP. These algorithms are comparatible to others in NP but not necessecarily reducible to problems in P (this will be explored later on).
All NP-Complete problems can be solved by means of a bruteforce algorithm. All NP-Complete problems are also Polynominally Reduceable which means there exists some Polynomial-time transformation to convert inputs of one to those of another algorithm that's NP-Complete.

4.2 Example of an NP-Complete Problem

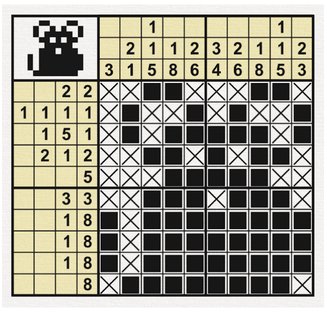

Figure 1: An Example of a Solved Nonogram(nonogramskatana.blogspot.com 2016)

A good example of an NP-Complete problem is a nonogram. A nonogram is a japanese picture puzzle where you are presented with an m x n grid and a list of groupings of colored squares for each row/column. A Nonogram is in in NP as a solution is extremely easy to verify and can be done in Polynomial time but is currently not in P as there is currently no known algorithm to solve every nonogram in Polynomial time and thus requires a bruteforce search. It was proven to be NP-Complete by demonstrating that the already known to be NP-Complete 3-Dimension Matching Problem (Karp 1972) could be reduced to a Nonogram and by extension the already known to be NP-Complete Another Solution Problem for 3 Dimensional Matching could be reduced to the Another Solution Problem for a Nonogram and consequently Nonograms are NP-Complete.(Ueda & Nagao 1996)

5 NP-Hard Problems

5.1 Outline of an NP-Hard Problem

An NP-Hard Problem is a problem that is at least as hard as an NP-Complete Problem, but unlike NP Problems, they do not need to be decision Problems which also means that some of them may be unsolvable by a computer.

A further classification is that An algorithm H is NP-Hard if there is a when any problem L in NP-Complete can be solved by means of H in Polynomial-time (Knuth 1974)

5.2 Example of an NP-Hard Problem

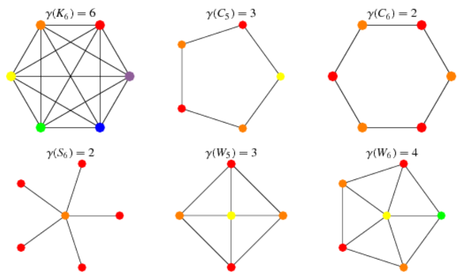

Figure 2: Examples of graphs and their chromatic numbers Weisstein (n.d.)

A very good example of an NP-Hard problem is that of the ”Minimum-Color Problem” where you must calculate the chromatic number (Minimum number of Colors needed to ensure no two nodes of the same color are adjacent) of an undirected graph G.

6 The P versus NP Problem

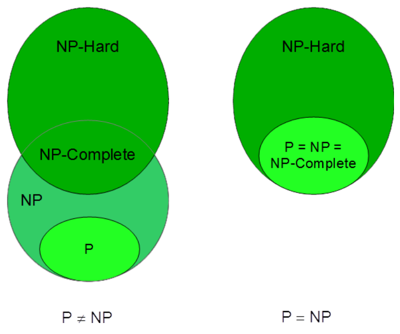

Figure 3: Graph for if $P = NP$ and if $P \neq NP$ (n.d.)

”P = NP where N = 1” ∼Unknown

The P versus NP problem is one very important problem in all fields of Computer Science. It was first Introduced by Stephen Cook in 1973 when he published the paper ”The complexity of theorem proving procedures” and in 2000 the Clay Mathematics Institute declared it as one of the seven ”Millennium Prize Problems”, each of these problems was at the time an unsolved problem in Mathematics, and if someone were to offer an accepted solution either proving, or disproving it, they would recieve $1,000,000. Thusfar only one of these problems has been solved. In Essence the P versus NP problem poses the question that for any problem that can be quickly verified does there exist a means in which it can be quickly solved? Some academics such as Donald Knuth believe that P = NP but he stated the runtime of an algortihm for such a proof would be so large that it would be ”nonconstructive” (informit 2014)

6.1 If $P = NP$

If P happens to equal NP that would imply all NP-Complete problems such as all modern encryption systems could theoretically be rendered useless. This as a result would compromise things such as Crypto currency and the recently ”discovered” Non-Fungible Token as one could easily ”Steal” such assets as they could modify the blockchain by spoofing digital signatures rather than needing to perform the theoretical 51% Attack (Where a person or group could modify contents within a blockchain at will if they managed to control 51% of it.), Although it would heavily hinder if not destroy almost all forms of cryptography, It would make efficient navigation/distribution a lot easier to achieve as the Travelling Salesman Problem would be far easier to solve, if p=np it would also extend into lifesciences as Protein Folding problems are NP-Complete. (Berger & et al. 1998)

6.2 If $P \neq NP$

If this happens to be the Case it would be nowhere near as exciting as it would only reassure the already large (88%) of people who believe P does not equal NP (Gasarch 2020) Consequently, it would mean that there is no solution in P to a lot of very exciting and difficult problems.

References

(n.d.).
URL: *https://ict.senecacollege.ca/ gpu621/pages/content/intro$_p$.html*

Atkins-Bittner, T. (2014), ‘A quick primer on p and np’.
URL: *https://www.xanthir.com/b4Zb0*

Berger, B. & et al. (1998), ‘Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete’.

Gasarch, W. I. (2020), ‘Guest column: The third p =? np poll’.
URL: *https://www.cs.umd.edu/users/gasarch/BLOGPAPERS/pollpaper3.pdf*

informit (2014), ‘Twenty questions for donald knuth’.
URL: *https://www.informit.com/articles/article.aspx?p=2213858*$WT.rss_f = ArticleWT.rss_a = TwentyQuestionsforDonaldKnuthWT.rss_ev = a$

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* ‘Complexity of computer computations’, Springer, pp. 85–103.

Knuth, D. E. (1974), ‘Postscript about np-hard problems’, *SIGACT News* **6**(2), 15–16.
URL: *https://doi.org/10.1145/1008304.1008305*

nonogramskatana.blogspot.com (2016), ‘10x10 mouse jpg’.

Ueda, N. & Nagao, T. (1996), ‘Np-completeness results for nonogram via parsimonious reductions’, *preprint* .

Weiss, M. A. (2006), *Data structures and algorithm analysis in C*, Pearson Education.

Weisstein, E. W. (n.d.), ‘Chromatic number. From MathWorld—A Wolfram Web Resource’. Last visited on 15/4/2021.
URL: *http://mathworld.wolfram.com/ChromaticNumber.html*