IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

DOI: 10.17148/IARJSET.2021.81109

Assessment of Computational Complexity and Methodologies of Pattern Matching Algorithms

Barkha Gupta[1]

Programme Assistant, Department of Computer, Agriculture University, Jodhpur, Rajasthan, India[1]

Abstract: Complexity Theory is a formal mathematical theory, in which we study computational problems and algorithms to solve those problems. In complexity theory we need to analyse the program in terms of the efficiency of the program, lines of the code, finding out that the program is able to find out the correct answer or output, figuring out whether one program is better than the other program. An algorithm here means the programming code through which the given problem will be solved. It is step by step procedure to perform a desired task.

Keywords: Computational theory, Complexity theory, Computational complexity theory, String matching algorithm, P and NP problem, NP complete problem.

I. INTRODUCTION

Computational Complexity theory focuses on many different aspects such as the number of resources need to solve the problem in terms of time efficiency and space required. Measure of complexity is also depending on amount of communication, number or circuit gates, number or processors required.

(i) **Time complexity** – It is simply meaning the amount of time required to execute the code or how fast the code runs, how much time it will take to run the program. Time complexity is hardware-independent. The time is basically depending on where you run the program or depends on processor so it is not a very deterministic metric but we know that the time is basically very proportional to the number of steps that are written in the programming language.

(ii) **Space complexity** – It is the amount of memory which is required by the program for its execution. All the variables occupy its memory in RAM. So, for a good and efficient program it is totally depends on the number of variables, loops statement used in the program. It should be minimized for saving the memory of the computer.

Example of Complexity in terms of Time and Space:

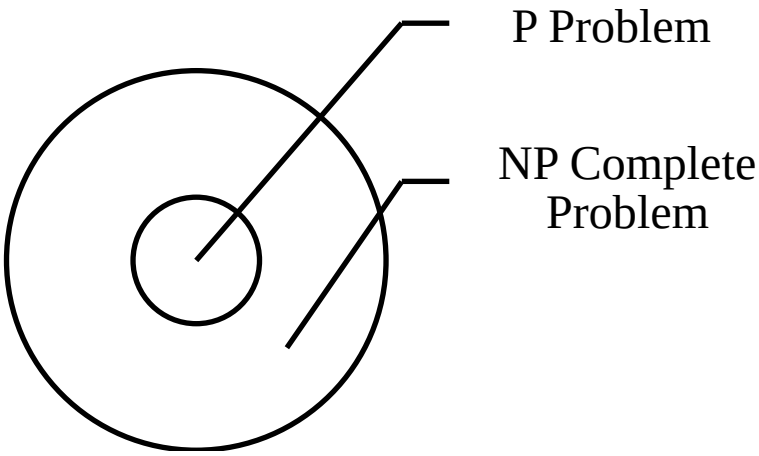
```
i = 10;                    1 second
while ( i >= 1 )           12 s
{
   a = 50;                 10s
   result = i * a;         10s
   printf ("Output is %d", result);  10s
   i--;                    10s
}
```

So, Time complexity is = 52 Seconds

Space complexity = 6 bytes (assuming each integer is of 2 bytes)

IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 8, Issue 11, November 2021
DOI: 10.17148/IARJSET.2021.81109

II. P PROBLEM



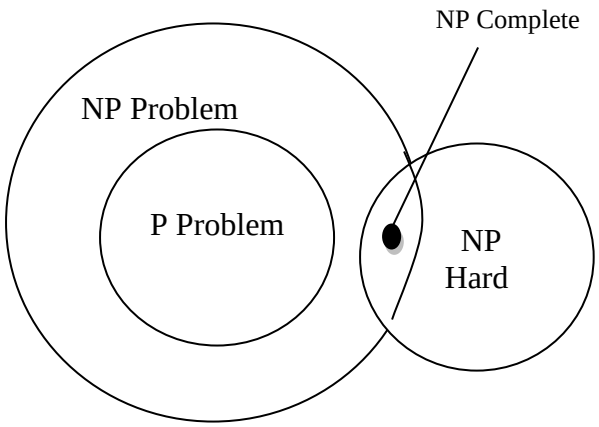
(i) P problem is solvable in polynomial time
(ii) It is solved in time $O(n^k)$ where n is the size of input and k is constant.
(iii) It is solved by deterministic machine
(iv) Its solution is easy to find
Example: - $O(n^2)$, $O(n^3)$ etc.

III. NP PROBLEM

NP Problem is a problem whose solution is verifiable in polynomial time. Let problem 'P' is given and its solution 'S' is already known or already given by the algorithm 'A'. If this algorithm verifies in polynomial time that whether solution 'S' for this problem 'P' is correct or not. Np problems are solved by No-deterministic machine in polynomial time.

Thus, NP problem are the collection of problems whose solution is difficult to find out but once the solution is known than its easy to verify that solution.

IV. NP HARD PROBLEM



NP Hard problem is a set of problem if there is an NP-Complete problem P, such that X is reducible to Y in polynomial time. NP-Hard problems are as hard as NP-Complete problems. Few Examples of NP Hard problems are set cover, vertex cover, subset sum etc.

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

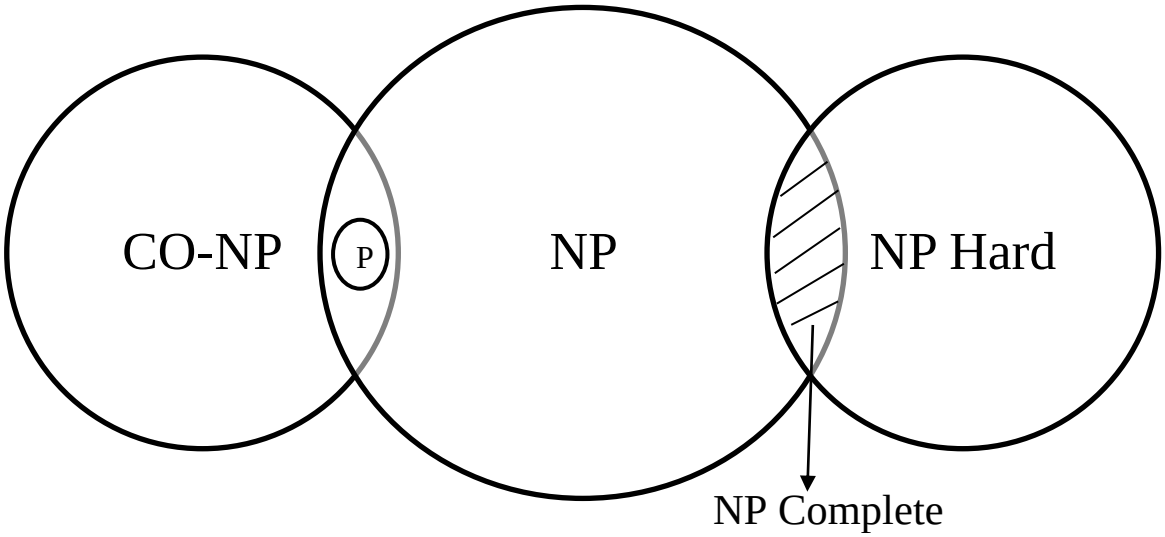
DOI: 10.17148/IARJSET.2021.81109

V. NP COMPLETE PROBLEM

NP Problem are set of problem which is combination of NP Hard as well as NP Problems. It implies that the problems which is not solvable in polynomial time but if there is solution exists then that can be verifiable in polynomial time. Subset problems are NP Complete and NP Problems. If any problem in NP can be reduced to it in polynomial time and it is also in NP.

VI. CO-NP CLASS

CO-NP class is a compliment of NP class. CO-NP is the class of decision problems for which there is a polynomial time algorithm that can verify "no" instances given the appropriate certificate. CO−NP problems are easy to find that there isn't a solution.



VII. PATTERN/STRING MATCHING ALGORITHM

NAIVE STRING MATCHING ALOGORITHM

Naive string-matching algorithm is a simplest method for pattern matching. It performs checking at all the position in the text whether an occurrence of pattern starts there or not. It finds all the matches. It keeps track of the pointer by the value called SHIFT. (S) After each attempt, the algorithm moves the pointer and increment the value of SHIFT by exactly one position to the right. Let's understand this by some examples in details

Example 1:

T = abcabaabc
P = abaa

Solution: Here T is the text string

a	b	c	a	b	a	a	b	c
0	1	2	3	4	5	6	7	8

String length here is n +1 = 9

and P denotes pattern to match in T string

a	b	a	a
0	1	2	3

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

DOI: 10.17148/IARJSET.2021.81109

Initially SHIFT = 0

Step:1 when initially SHIFT = 0, then it will match T[0] character with the P[0] character. As here match found then

Step 2: It will increment both the value by one that is T[1] and P[1]. Here also the match is found it will again increment the value of T[2] and P[2]. Here there is no match

Step 3: As no match found, now SHIFT will increment by one and its new value will become SHIFT=1 and it will increment the value of T by one and it will become T[1].

Step 4: Now here it will match T[1] and P[0] and here there is no match found. So, it will repeat the step 3 again.

Step 5: SHIFT and T will become 2 and now it will match T[2] and P[0] and here there is no match found. So, it will repeat the step 3 again.

Step 6: SHIFT and T will become 3 and now it will match T[3] and P[0]. As here the match is found then it will repeat step 2

So, when SHIFT = 3 pattern match is successful.

So, it defines 0<=SHIFT <= n-m

Where n= length of text string and m = length of pattern to be matched string

So, in above example 0<=SHIFT<=9-4

Simplify, 0<=SHIFT<=5

VIII. RABIN KARP ALGORITHM

Rabin Karp algorithm is developed by Michael O Rabin and Richard M Karp. It works on the hashing function. Here in this algorithm T is denotes by text string and P is denoted by string to be searched and Q is any random prime number. In this algorithm paired of string is been framed from the P string of the length equivalent of P string and by incrementing each time by the value of one.

Example 2:

T=31234862
P=234

Solution:

T=

3	1	2	3	4	8	6	2

P =

2	3	4

Let Q be 13

P mod q = 234 mod 13 = 0
So, 0 is the hash to be pattern that is to be found.

Now let's make the pair of T string. Length of the pair will be 3 (length of P string). So, first pair will be

3	1	2

Now increment the value of T by one and again make the next pair.

1	2	3

Now repeat the process and make all the pairs. So, all the pairs will be as follows: -

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

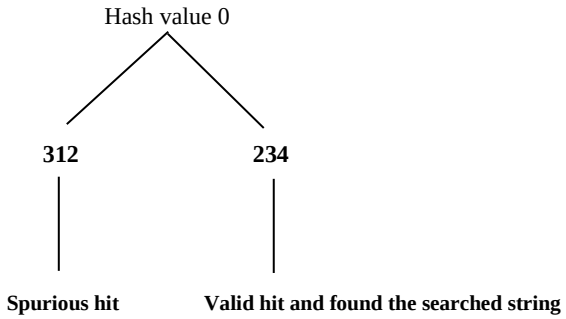
DOI: 10.17148/IARJSET.2021.81109

3	**1**	**2**
1	**2**	**3**
2	**3**	**4**
3	**4**	**8**
4	**8**	**6**
8	**6**	**2**

Now, compute its hash value by taking the mod (pair mod q)

Pairs	Mod	Hash Value
312	**312 mod 13**	**0**
123	**123 mod 13**	**6**
234	**234 mod 13**	**0**
348	**348 mod 13**	**10**
486	**486 mod 13**	**5**
862	**862 mod 13**	**4**

So, wherever hash value matches it with hash pattern, pick that pair. We have computed hash pattern as 0 by (234 mod 13). So, its matching pair is



IX. APPROXIMATION ALGORITHM

Approximation algorithm is a way of dealing with NP completeness for optimization problem. The goal of approximation algorithm is come close as possible to optimal solution in polynomial time instead of giving optimal solution.

C = Cost of solution
C* = Cost of optimal solution
P (n) = Approximation ratio
N = Input size

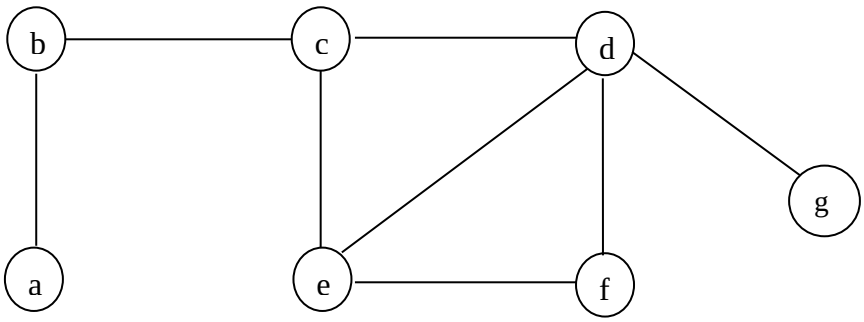
Maximization = $\frac{C*}{C}$ <= P(n) where P(n) >= 1 and P(n) ≠ 1 or P(n) ≠ < 1

Minimization = $\frac{C}{C*}$ <= P(n)

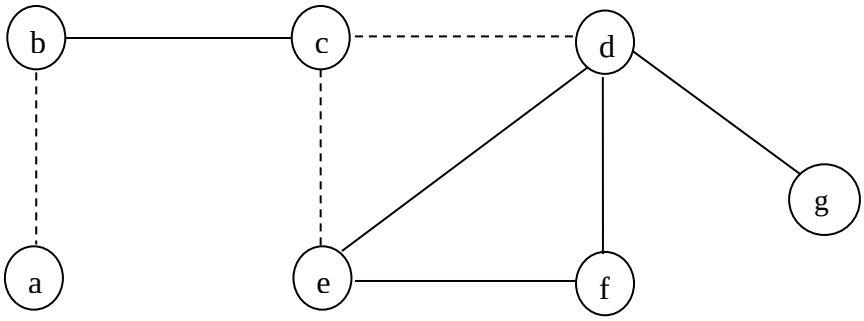
IARJSET
ISSN (O) 2393-8021, ISSN (P) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 8, Issue 11, November 2021
DOI: 10.17148/IARJSET.2021.81109

X. VERTEX COVER PROBLEM

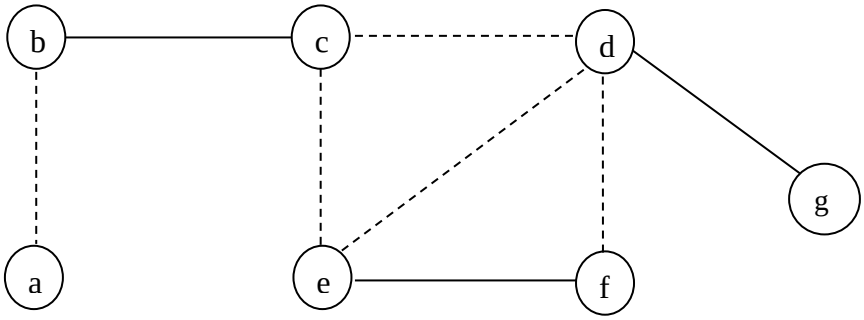
Vertex cover problem is a NP hard problem. It is a subset of vertex which cover every edge. Vertex cover problem is the minimum size vertex cover.



Step 1: C = Ø
Step 2: E' = contains all edges
E' = {(a, b), (b, c), (c, e), (c, d), (d, f), (e, f), (e, d), (d, g)}
Step 3: Repeat till E' is not equal to Ø
Step 4: Take any edge and do C = C U {u, v}. Suppose we have taken an edge (b, c) than they act as (u, v) and remove its all adjacent (covered) edges from E'.
Step 5: C = C U {u, v}
= Ø U {b, c}
C = {b,c}



And E' = {(d, f), (e, f), (e, d), (d, g)} as {(a, b), (c, d), (b, c), (c, e)}
Step 6: Now suppose we have taken edge (d, g)
C = C U {u, v}
= {b, c, e, f}



And E' = {d, g}
Step 7: Now we have taken edge (d, g)
C = {b, c, e, f, d, g}

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

DOI: 10.17148/IARJSET.2021.81109

E'= Ø

So, this provide 6 vertices but this is not optimal solution. Optimal is {b, e, d} which contain 3 vertices.

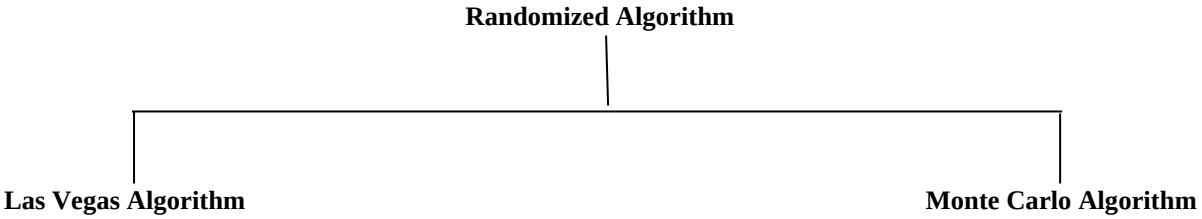
Maximization = $\frac{C*}{C}$ <= P(n)

Maximization = $\frac{6}{3}$ <= P(n)

P(n) >=2 satisfied as P(n) ≠ 1 or P(n) ≠ < 1

XI. NON-DETERMINISTIC PROBLEM

Non-Deterministic problem are those unpredictable problem that depends on some other variable. It generally depends on randomizer variable. This randomize is generated from randomizer algorithm (deterministic algorithm). Randomizer generate random value known as Pseudo Random Number because it is not purely random number, it is generated from an algorithm. It totally depends on initial value and a truly random number is completely unpredictable.



(i) **Las Vegas Algorithm** – Las Vegas Algorithm is introduced by Laszlo Babai in 1979. It is used where number of possible solutions is limited. In this verifying solution is easy while the calculation is complex. It gives correct output but there is no fixed time. Running time is bounded by input size. It informs about failure as well. Las Vegas is generally performed if there is no time constraint as it may take long time to return the exact result.

(ii) **Monte Carlo Algorithm** – Monte Carlo Algorithm is introduced by Nicholas Metropolis in 1947. It may not return output at all the time, but if they do so then may or may not guarantee of correctness. It has bounded running time. In this algorithm the probability of failure is minimized by repeated operations. Monte Carlo Algorithm is generally performed if there is constraint of time as it has to execute a fixed number of iterations, but it may not return the exact result.

XII. RUSSEL PARADOX

The Russel paradox says that there is no set of all set or collection of all sets is not a set. Russel Paradox is a standard way to show naïve fact

A is set, A ∉ A, A ⊆ A

Naive set theory uses the comprehension principle. This states that given any property there exists a set containing all the objects that have that property. So, paradox is neither R *E* R nor R ∉ R.

To resolve this and have a consistent system we have to reject this principle and use on alternative set of axioms. Most well known consistent axiomatic framework foe set theory is Zermelo Fraenkel Set Theory (ZFC) with axiom of choice.

XIII. POST CORRESPONDENCE PROBLEM

Post Correspondence Problem also known as PCP problem is undecidable decision problem that was introduced by Emil Post in 1946. It is simpler then halting problem and used in proof of undecidability. It has dominos or tiles which is divided into two parts.

IARJSET

ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Vol. 8, Issue 11, November 2021

DOI: 10.17148/IARJSET.2021.81109

$\frac{B}{CA}$	$\frac{A}{AB}$	$\frac{CA}{A}$	$\frac{ABC}{C}$

Here we need to find a sequence of dominos such that top and bottom strings are same. We can change sequence and we can use any dominos any number of times. We should always try to start with dominos whose first symbol of top and bottom are same. As in first dominos first letter of top and bottom are different so, we cannot start with first dominos. So, we have to start with second dominos.

$\frac{A}{AB}$	$\frac{B}{CA}$	$\frac{CA}{A}$	$\frac{A}{AB}$	$\frac{ABC}{C}$	⇒	$\frac{ABCAAABC}{ABCAAABC}$

XIV. CONCLUSION

There is a belief that the overall result of the P vs. NP problem will be inconsequential in the end. This is because of sizeable majority of computer scientists believe that P ≠ NP as is. If P ≠ NP is proven, then nothing will change because of the assumptions already made within the computer science industry. Advances in theoretical computation will thereby rely on things like quantum computing to radically disrupt the speed at which we solve certain problems. However, there are several computer scientists that are optimistic about the potential about the P = NP problem.

XV. ACKNOWLEDGEMENTS

Author is grateful to Agriculture University, Jodhpur for encouraging writing in my field.

XVI. REFERENCES

[1]. Sanjeev Arora. Boaz Barak. Computational Complexity. Cambridge University Press Publisher
[2]. Steven Homer. Alan L. Selman. Computability and Complexity Theory. Springer US Publisher.
[3]. Michael Sipser. Introduction to the Theory of Computation. Publisher Cengage Laerning.
[4]. Sanjeev Arora. Boaz Barak. Computational Complexity A Modern Approach. Publisher Cambridge University Press
[5]. Oded Goldreich. Computational Complexity. Cambridge University Press Publisher.