VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE 2003- DATA STRUCTURES AND ALGORITHMS

PROJECT TITLE:

OPTIMAL JOB ASSIGNMENT

Submitted By: Pranjael Bagaria

Registration No: 18BCE0495

Professor: Mr. Govinda K.

AIM

To allocate jobs to various employees in a company in such a way that minimum amount of money is required to complete all the jobs.

ABSTRACT

One of the biggest problem, the companies are facing these days is assigning jobs to various employees in their companies. Now-a-days, cost has become an important factor in companies. Every company wants their jobs to be done provided that the cost of doing it is least. In order to help out companies in this matter, I have tried to implement an algorithm (Hungarian Algorithm) which would help us to get the desired solution in a much efficient and faster way. Using Bipartite Graph and a double dimensional matrix, I have implemented this algorithm. Through various comparisons with brute force algorithm, it has been observed and analysed that the algorithm proposed by me is better than it. I have been able to achieve appropriate results during the analysis of the program. Thus, this algorithm brings to us one of the most efficient way to solve an assignment problem (optimal job assignment).

PROBLEM STATEMENT

In a company, there are N employees and M jobs. All N employees can do all M jobs but the amount of money charged by each of them for each job is different. Each job takes one day to be completed due to which an employee can be allocated only one job. Our aim is to determine which employee should be allocated with which job so that minimum amount of money is spent in order to complete all jobs and we have to find the minimum cost.

INTRODUCTION:

In Today's World, a lot of employees are required in all the companies of the world be it Google or Microsoft. Due to large number of employees, a huge amount of capital is required nowadays to ensure all the tasks of a day is completed well in time and with perfection. Depending upon the ease with which an employee can do a particular task, he/she charges money. These days all the employees are jack of all trades and a master in one. Thus, almost all the employees applying for a job in the company can do various other things. For example, a Software Engineer can also handle the Management of Internet along with developing software.

Nowadays, it has become very important for the company to allocate jobs to the employees in such a manner that they have to invest the least amount of money possible, which would help them to save money for future purposes and also ensure that the jobs are done.

In order to get a job in a company, one needs to first apply for that particular job and must have enough skill to compete with all other employees, who are applying for the job. Once the application is selected, one has to undergo certain tests to become eligible for the personal interview with the board members. Once this round is cleared, every selected employee is called for the interview, where they are shortlisted according to their eligibility. Now, if a company finds out that two or more employees are of the same talent and calibre to do a job, then it selects the one who offers to take least salary among all the shortlisted employees. This may also be the case for various other jobs in the same company. This process of choosing the correct employee becomes exhaustive once the number of jobs along with the number of shortlisted candidates increases.

Not only assigning jobs in a company can be solved by the algorithm proposed by me but also there are various other problems in our day to day life where this algorithm can be applied. For example, this algorithm would help us to select the roles of players in a baseball team. To be a part of a baseball team, one has to undergo various tests so that the coach can determine which position would be

perfect for a particular player. The tests include Hitting test, Speed Test, Pitching test, Throwing test, Fielding test, Reach test and Reaction test. Each and every player of a baseball team has to undergo these tests so that they can be amongst the playing nine. Now once all the tests are completed, the coach has to determine the team in such a way that all arenas of the team mainly defence and attack are covered. Since the number of players appearing for the test is large, it becomes difficult for the coach to determine the best players. So he can segregate the players on the basis of the role they want to play in the team. He can use Hungarian Algorithm for each category in order to find the combination which can provide him with players having maximum skills. Then, depending upon the number of players needed for a particular role, the coach can select that many players for the playing nine and thus, he can make the best team out of all the players. This is just another example of a daily life situation where Hungarian Algorithm can be applied.

Through the algorithm, one can also solve various other problems of assignment. For example, let us consider that there is a company having three employees at three different locations of the country Delhi, Kolkata and Chennai. The company has to send these employees to three other cities Hyderabad, Jaipur and Bangalore. The cost of them to travel from their respective cities to the desired cities is calculated. Now, it becomes important for the company to choose which employ should go to which city so that it has to spend minimum amount of money to accomplish the desired task. This problem can easily and effectively be solved by the algorithm proposed and selected by me.

LITERATURE REVIEW

The problem which I have chosen is known as the assignment problem. It is a special form of the transportation problem, which itself is a special form of the minimum cost flow program, which in turn is a specialized form of the linear programming approach.

Assignment of jobs or duties in a company or some other place has been a problem since ages. But this problem got intensified, when a large number of employees, capable of doing most of the things required by a company, started applying for jobs. As monetary capital became an important factor in the companies, it made the companies realize that they have to recruit employees in a manner such that least amount of money is invested [8]. Various other algorithms apart from the algorithm I have chosen can be used to solve this assignment problem. These algorithms are as follows:

- ***BRUTE FORCE ALGORITHM/ENUMERATION METHOD***

  Brute force algorithm or Enumeration method is one of the most inefficient ways in order to solve an assignment problem. It basically involves finding a list of assignments involving all the possible combinations and permutations among the given resources and activities (In my case, jobs and costs). Then an assignment involving the minimum cost, time or distance or maximum profit is selected. If two or more assignments are possible, then the problem has more than one optimal solution and we can assign the tasks in any of the ways. This algorithm makes the task of assignment easier when the size of the input is low but once the size of the resources (jobs) along with the activities (costs) starts increasing and crosses a mark of 4, then it becomes exhaustive and frustrating. The running time of the algorithm along with the amount of space needed starts increasing with size, which makes the algorithm inefficient for larger input size. Thus, this algorithm is not preferable as the input size of the real world is not small [10, 13].

- ***SIMPLEX METHOD***

  The simplex method is generally used to solve Linear programming problem (LPP) of any type involving two or more decision making variables. The simplex algorithm is an iterative process for finding the optimal/best solution to a linear programming problem. The objective function controls

the development and evaluation of each feasible/possible solution to the problem. If a feasible solution is present, it is located at a corner point of the feasible region determined by the constraints of the system. The simplex method simply selects the optimal solution amongst the set of feasible solutions of the problem. The efficiency of this algorithm is due to the fact that it considers only those feasible solutions which are provided by the corner points, and that too not all of them. One can consider obtaining a solution based on a minimum number of feasible solutions. Simplex algorithm, if applied, for assignment problem would make the task hectic and would create unnecessary problems to the programmer [9, 10].

- ***TRANSPORTATION METHOD***

  Transportation mechanism is an important class of linear programs. For a given supply at each source and a given demand at each destination, the model studies the minimisation of the cost of transporting a commodity from a number of sources to several destinations. One can observe that assignment problem (chosen by me) is a special case of transportation problem. Thus, assignment problem can also be solved using the transportation mechanism. But the degeneracy problem of solution makes the transportation method computationally inefficient for solving the assignment problem [10].

- ***BRANCH AND BOUND***

  Branch and Bound is an algorithm design hypothesis for discrete and combinational optimization problems. It is a technique of obtaining an optimal solution from various available solutions. This technique is generally used to solve problem which has an exponential complexity and problems which may require permutations (in the worst case). Branch and Bound algorithm can solve these problems in a relatively less time. The bounds present in the problem are merged with the latest best solution. It allows the algorithm to find all the possible solutions of the problem [6, 10].

- ***MM METHOD (A new Approach for solving assignment problem with Optimal Solution):***

  MM method can be used to solve assignment problem of balanced cost matrices. It is a very simple method but one of its drawbacks is that it can still not solve unbalanced matrices. It involves finding a minimum number in each row and subtracting each number of that row with the minimum number to get at least one zero in all the rows. Then, one of the distinct zero is eliminated from the table and with that the row as well as the column is also deleted. Then again from the original matrix (by removing the row and column), the process is continued until all assignments are made. If a column has two zeros, then the next minimum in the row is checked and the row having the maximum out the minimums is assigned the job in the column. The process is then repeated [7].

  MM method was published in an article in July, 2017 by IJSRD- International Journal for Scientific Research & Development.

  http://www.ijsrd.com/articles/IJSRDV5I70090.pdf

PROPOSED METHOD

The method proposed by me to solve the assignment problem, that is assigning jobs to various employees, is ***Hungarian Algorithm***. The Hungarian method is an optimization algorithm that solves the assignment problem in polynomial time. It was published and bought into light by Harold Kuhn, who gave the name "Hungarian Algorithm" because this algorithm was mainly based on the earlier works of two great Hungarian mathematicians: Denes Konig and Jeno Egervary.

In 1957, James Munkres reviewed the algorithm and observed that it is (strongly) polynomial. Since then the algorithm has been also known as the Kuhn-Munkres algorithm or Munkres assignment algorithm. This helped in improving the complexity of the already available algorithm and thus making the algorithm more efficient in terms of the time taken by it to solve a particular assignment problem. The time complexity of the initial algorithm was $O(n^4)$, however Edmonds and Karp, and independently Tomizawa noticed that it can be moulded and improved to achieve a time complexity of $O(n^3)$.

Hungarian Algorithm uses a concept of Bipartite Graph matching which comes under Graph theory. It involves the usage of weighted undirected edges from a set of jobs to a set of workers or vice versa. It can be observed that it is a much better and efficient algorithm in terms of the time it takes to compute the result. This makes it much better than the Brute Force algorithm, which computes all possible permutations of the assignment of jobs to employees. Brute Force is not preferred as when the input size grows, the time to compute the result would also grow and even a huge space will be needed to store the various permutations. Thus Hungarian Algorithm is better than Brute Force Algorithm in terms of both Time complexity as well as Space complexity. Therefore, Hungarian Algorithm is preferred to Brute Force Algorithm.

MATHEMATICAL REPRESENTATION OF ASSIGNMENT PROBLEM:

COST MATRIX	JOB 1	JOB 2	JOB 3	.....	JOB N
Worker 1	Cij	Cij	Cij	Cij	Cij
Worker 2	Cij	Cij	Cij	Cij	Cij
Worker 3	Cij	Cij	Cij	Cij	Cij
. .	Cij	Cij	Cij	Cij	Cij
. .. Worker n	Cij	Cij	Cij	Cij	Cij

Let C_{ij} represent the cost of assigning jth job to ith worker.

Let Xij be equal to 1 if job j is assigned to worker i otherwise let it be 0.

Since only one job can be assigned to one worker,

$$\sum_{i=1 \text{ to } n} Xij=1 \text{ (for all i and for all j)}$$

Thus the total cost would be,

$$C=\sum_{i=1 \text{ to } n} \sum_{j=1 \text{ to } n} Cij*Xij$$

Thus the main objective of our algorithm is to minimize this cost function C, so that minimum amount of money needs to be spent to complete all the tasks.

This minimization process can be carried out using Hungarian Algorithm and the optimal assignment can be easily found out.

DATA STRUCTURES USED:

The concept of Bipartite Graph used in the algorithm is implemented using dynamically controlled double dimensional and single dimensional array. Bipartite Graph is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V. Vertex sets U and V are usually called the parts of the graph. In the context of solving the desired assignment problem, U can be considered as a set of jobs and V can be considered as a set of workers or vice versa (since it is an undirected graph).

STEPS OF HUNGARIAN ALGORITHM:

Hungarian algorithm uses various steps in order to get the desired result (in my case, getting the minimum cost). The following steps describe the way Hungarian Algorithm works:-

1. Find the minimum number present in all rows and subtract the elements of all the rows by the minimum number present in that row, such that we get at least one zero in each row.
2. Now, find the minimum number in each column and subtract all elements of a column by the minimum number present in that column, in the similar manner as done for the rows.
3. Cover all the zeros by minimum number of horizontal/vertical lines.
4. If the number of lines is equal to the order of the cost matrix, then we have found the assignment and we end the process otherwise we go to step 5.
5. We find the minimum uncovered element and subtract it with all the uncovered elements and add it with all the elements present at the intersection of the lines. After this we again, go to step 1 and repeat this process until an optimal assignment is possible.

PSEUDO CODE OF HUNGARIAN ALGORITHM:

Step 1: Begin

Step 2: Input the matrix

Step 3: Reduce the matrix by the reduction process mentioned in the algorithm.

Step 4: Cover all the zeros by minimum number of lines (N).

Step 5: Run a while loop until N =the order of the matrix and then go to step 6

Step 5.1: Find the minimum uncovered element (m).

Step 5.2: Add m to all the elements which are covered twice and subtract from all the uncovered elements.

Step 5.3: Again reduce the matrix and cover all the zeros by minimum number of lines (N).

Step 6: Display the assignment made.

Step 7: End.

FLOW CHART OF HUNGARIAN ALGORITHM:

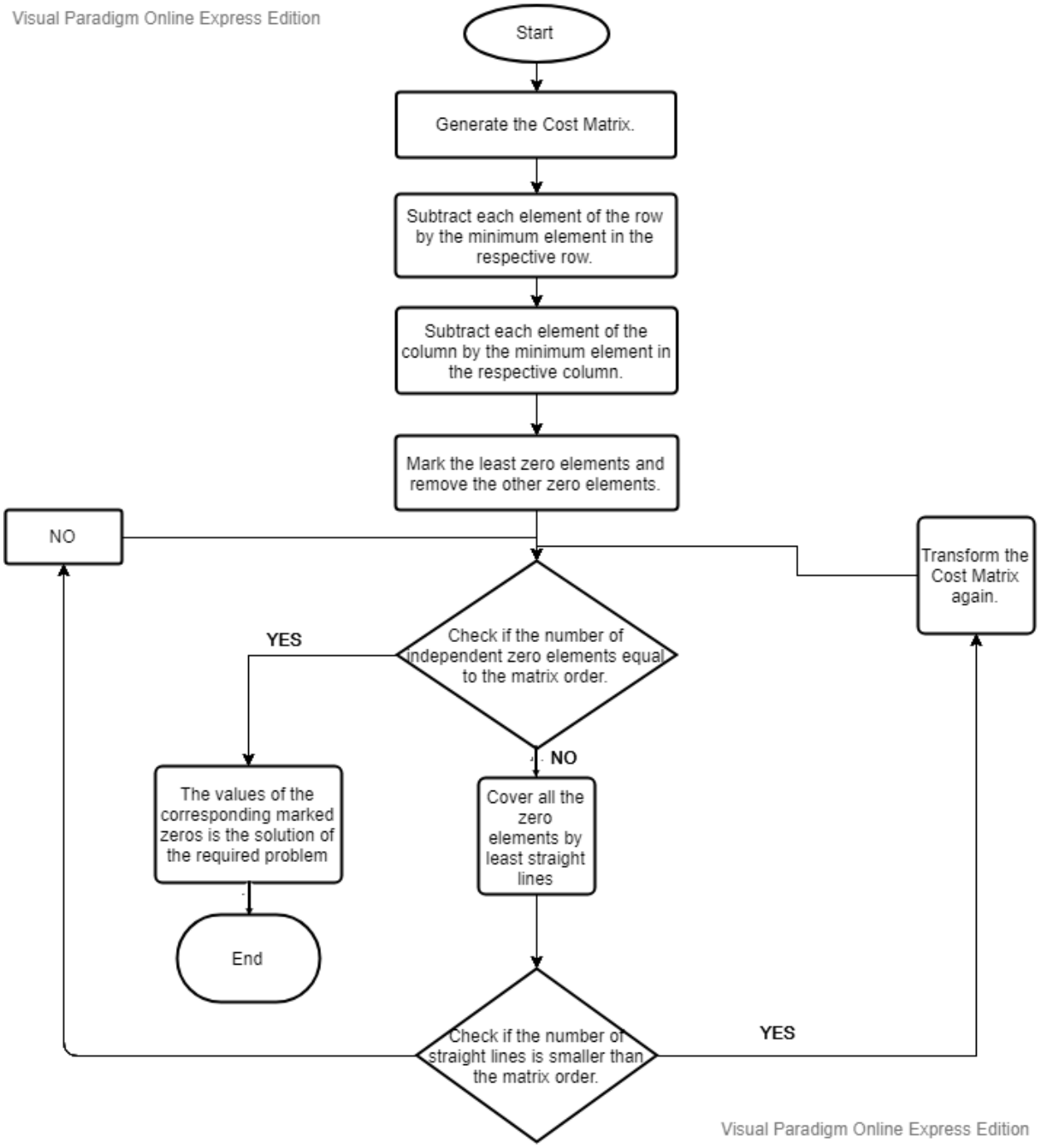

FIGURE 1: FLOWCHART OF HUNGARIAN ALGORITHM

PSEUDO CODE OF BRUTE FORCE ALGORITHM:

Step 1: Begin

Step 2: Input the cost matrix from the user.

Step 3: Find all the permutations of the matrix, i.e. all the possible combinations of jobs and workers.

Step 4: Store the cost of each assignment in a single dimensional array (a1) and in another array store the way of assignment (a), i.e. which job would be allocated to which worker.

Step 5: Declare m=10000; Run a loop from i=0 to i<length of a1.

Step 5.1: if(a1[i]<m), then assign a1[i] to m and store the value of i in pos.

Step 6: Using pos, find the corresponding a[pos]. Using a[pos], display the assignment which needs to be made.

Step 7: End.

RESULT AND ANALYSIS

Running time of both the algorithms, Hungarian Algorithm and Brute Force Algorithm was computed using codes of java language. Data was collected and stored in a tabular format. Graphs of both the algorithms were formulated and drawn using the various data collected. The following table displays the running time data which was collected and then follows the graphs of both the algorithms.

DATA COLLECTED

INPUT SIZE (n)	***HUNGARIAN TIME (sec)***	***BRUTE FORCE TIME (sec)***
0	0	0
1	1.854	1.72
2	4.73	3.54
3	7.13	5.64
4	11.98	10.602
5	16.795	17.885
6	18.024	22.125
7	24.636	32.229
8	26.219	40.919
9	35.321	81.929

GRAPHS FOR THE TWO ALGORITHMS:

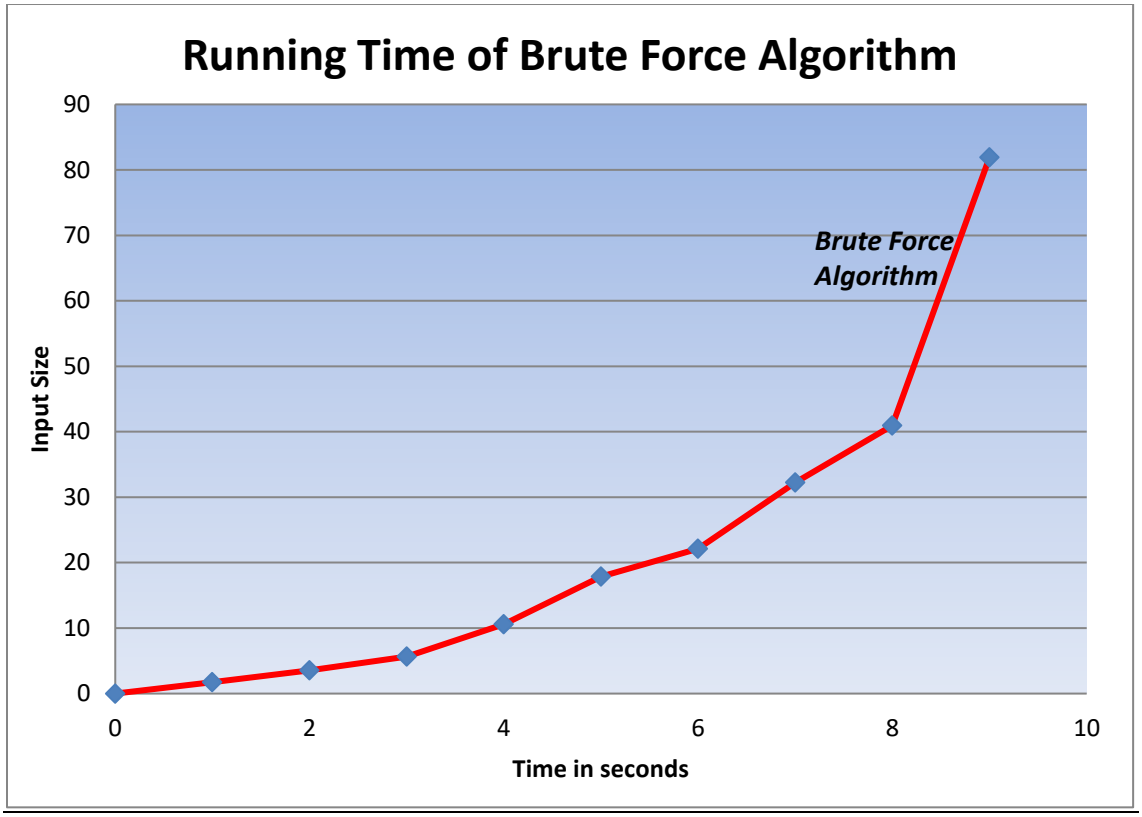

FIGURE 2: GRAPH OF RUNNING TIME OF BRUTE FORCE ALGORITHM

ANALYSIS

The above graph represents the running time of the code implemented through brute force algorithm. It can be observed from the graph that initially when the input size was small the running time was very small but as the input size grew, a steep increase in the running time was observed which shows us that this algorithm is efficient for problems having smaller input size. This algorithm would not be suitable for larger input size because the time needed to compute the result would be very high. In today's world, we need algorithms which can do its job in a quick span of time.

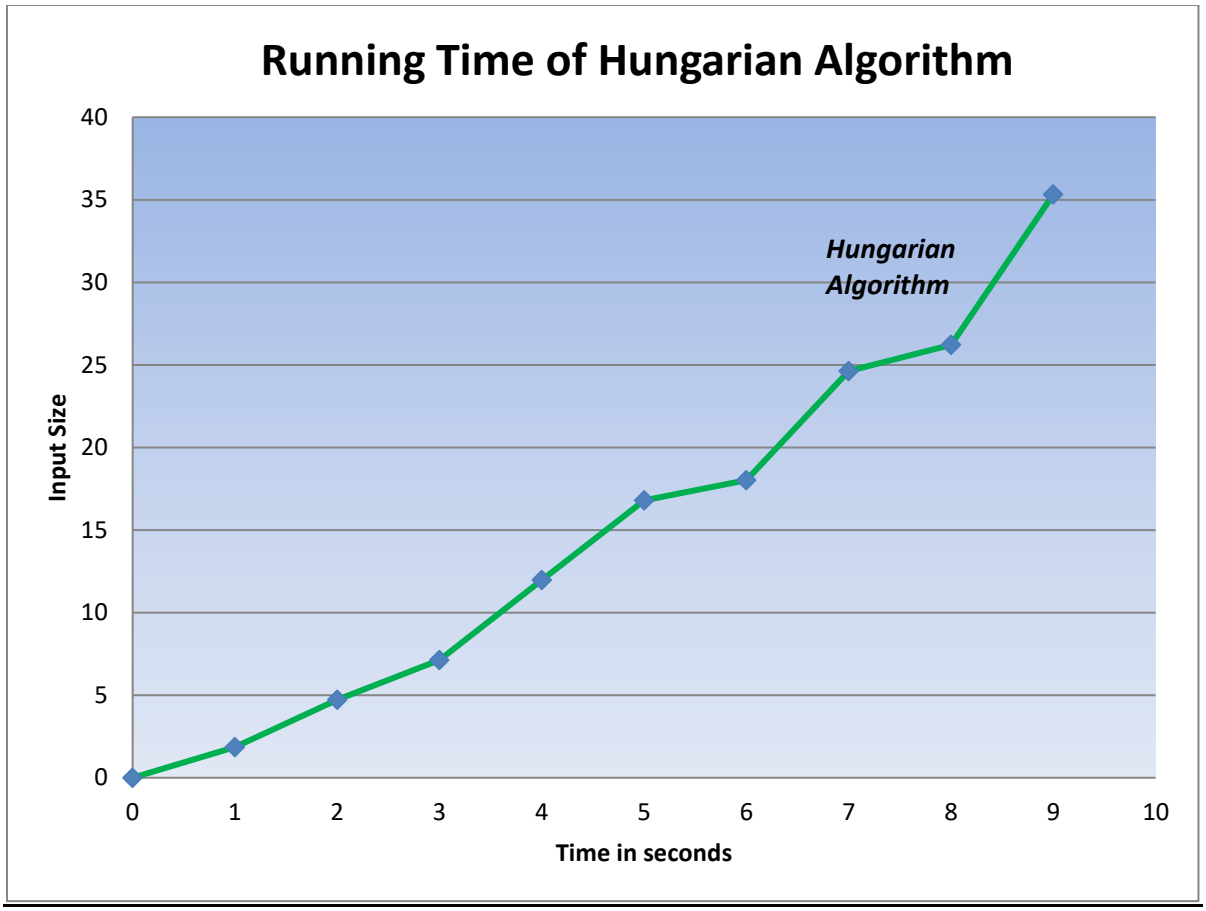

FIGURE 3: GRAPH OF RUNNING TIME OF HUNGARIAN ALGORITHM

ANALYSIS

The above graph represents the running time of the code implemented through Hungarian algorithm. It can be observed from the graph that initially when the input size was small the running time was less (but larger than the code implemented by brute force algorithm) but as the input size grew, there was a steady increase (not steep as in brute force algorithm) in the running time, from which we can infer that the code would take a reasonable amount of time to compute the desired solution when the size of the input is large. Thus, we can say that this algorithm should be preferred when the input size is large. As in the real world input size is generally large, we can say that Hungarian Algorithm is of more practical use in our day to day life than Brute Force Algorithm.

MERGED GRAPH OF THE TWO ALGORITHMS:

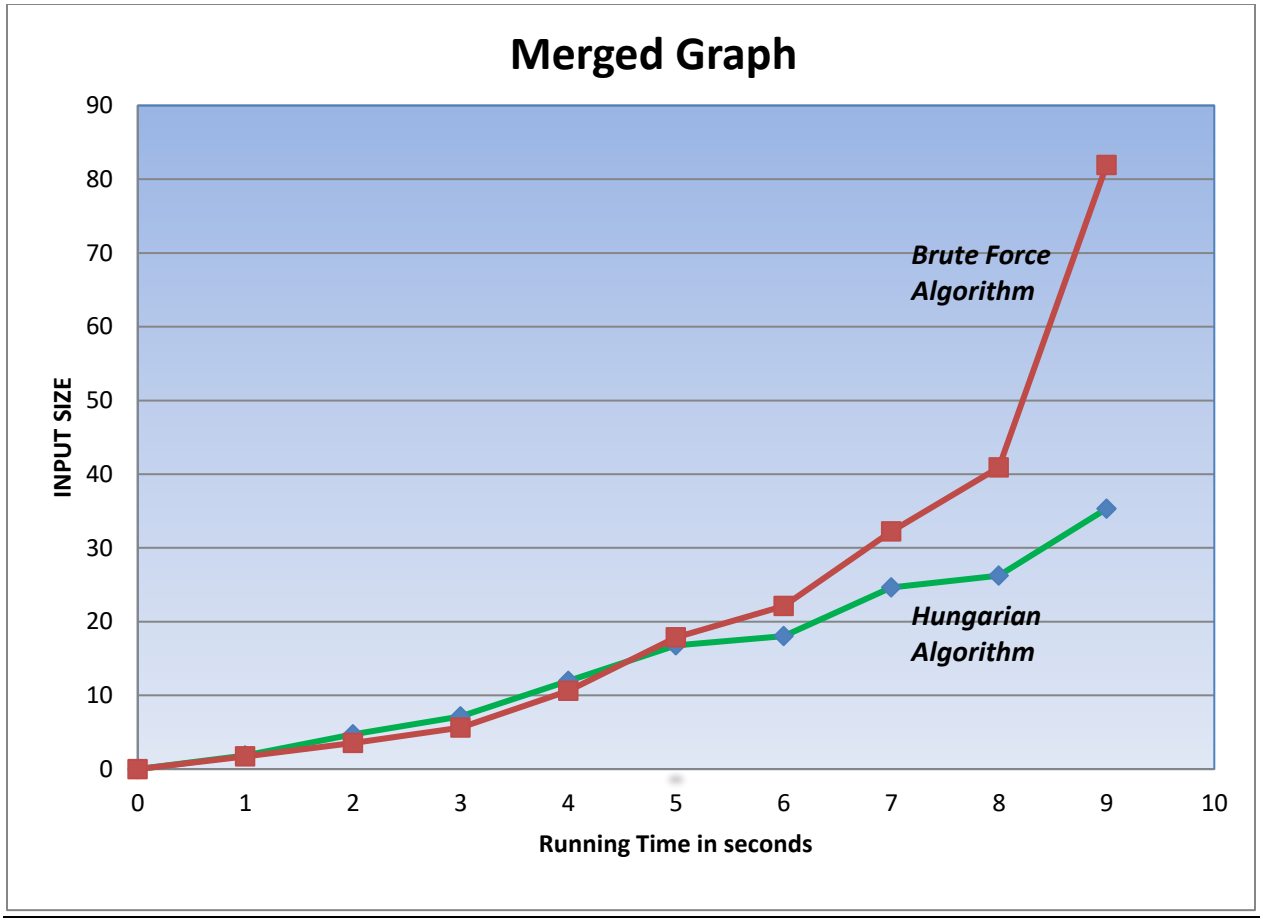

FIGURE 4: MERGED GRAPH OF RUNNING TIME

ANALYSIS

From the graphs displayed above, we analyse that initially when the input size is less than 5, brute force algorithm is a better algorithm than the Hungarian Algorithm. But as the size of the input grows the running time of brute force algorithm is much higher than that of Hungarian Algorithm. As in the real world, we deal with problems involving larger input size; Hungarian algorithm would be a better choice when compared to brute force algorithm. Brute force algorithm has a time complexity of O(n!) whereas Hungarian Algorithm has a time complexity of $O(n^3)$. This can also be observed from the graph. Thus, we can conclude that Hungarian Algorithm is much better than brute force algorithm.

CONCLUSION:

There is always more than one method/algorithm to solve a particular problem. It is the most efficient algorithm which a programmer should choose in order to solve a given problem because in this world, time is very precious and efficiency is the key to successful programming. Assignment problems are present all around us. It is just that we should know the right approach to solve this kind of problem. During the process of analysis, all the outputs were checked and it gave the desired result. My main aim in the project was to highlight Hungarian Algorithm as the best algorithm to solve an assignment problem. By analysing the graphs obtained from the data collected, I can infer that Hungarian Algorithm is a much better algorithm than Brute Force Algorithm.

REFERENCES:

1. https://www.geeksforgeeks.org/hungarian-algorithm-assignment-problem-set-1-introduction/
2. https://math.stackexchange.com/questions/1283207/hungarian-method-algorithm
3. https://stackoverflow.com/questions/23278375/hungarian-algorithm
4. https://www.youtube.com/watch?v=rrfFTdO2Z7I
5. https://www.topcoder.com/community/competitive-programming/tutorials/assignment-problem-and-hungarian-algorithm/
6. https://en.wikipedia.org/wiki/Branch_and_bound
7. http://www.ijsrd.com/articles/IJSRDV5I70090.pdf
8. https://www.sciencedirect.com/science/article/pii/0167637786900738
9. https://en.wikipedia.org/wiki/Simplex_algorithm
10. https://nikhatshahin.wordpress.com/2011/10/20/mb0048-q5-state-and-discuss-the-methods-for-solving-an-assignment-problem-how-is-hungarian-method-better-than-other-methods-for-solving-an-assignment-problem/
11. https://study.com/academy/lesson/using-the-hungarian-algorithm-to-solve-assignment-problems.html
12. https://en.wikipedia.org/wiki/Hungarian_algorithm
13. https://www.wiwi.uni-kl.de/bisor-orwiki/Enumeration_methods_5
14. https://www.veltech.edu.in/wp-content/uploads/2016/04/Paper-10-2016.pdf

APPENDIX:

- **Enumeration**: The action of doing a number of things one by one.
- **Time Complexity**: Time complexity is the quantification of the amount of time a code or algorithm takes to produce the desired output. It is a function of the input size.
- **Space Complexity**: Space complexity refers to the amount of space required by an algorithm to execute itself.
- **Bipartite Graph**: It is a graph whose vertices can be divided into two disjoint sets which are independent such that each edge connects one vertex in one set to a vertex in the other set.
- **Simplex**: According to Google, Simplex means composed of or characterized by a single part or structure
- **Polynomial time:** The time required by an algorithm to solve a problem, where the time is determined by a simple polynomial function having input size as the variable.

OUTPUT OF BOTH THE ALGORITHMS:

HUNGARIAN ALGORITHM:

BlueJ: Terminal Window - Pranjael

Options

```
Enter the dimentsions of the cost matrix:
r:
5
c:
5
Enter the cost matrix: <row wise>
9 8 6 5 7
8 7 5 4 9
2 6 5 1 3
6 9 8 5 1
7 4 5 8 9
        Job1    Job2    Job3    Job4    Job5
Worker1 9.0     8.0     6.0     5.0     7.0
Worker2 8.0     7.0     5.0     4.0     9.0
Worker3 2.0     6.0     5.0     1.0     3.0
Worker4 6.0     9.0     8.0     5.0     1.0
Worker5 7.0     4.0     5.0     8.0     9.0
Worker 1 gets job no: 3
Worker 2 gets job no: 4
Worker 3 gets job no: 1
Worker 4 gets job no: 5
Worker 5 gets job no: 2
Execution time is 22.021 seconds
```

BRUTE FORCE ALGORITHM:

BlueJ: Terminal Window - Pranjael

Options

```
Enter the dimentsions of the cost matrix:
r:
5
c:
5
Enter the cost matrix: <row wise>
9 8 6 5 7
8 7 5 4 9
2 6 5 1 3
6 9 8 5 1
7 4 5 8 9
        Job1    Job2    Job3    Job4    Job5
Worker1 9.0     8.0     6.0     5.0     7.0
Worker2 8.0     7.0     5.0     4.0     9.0
Worker3 2.0     6.0     5.0     1.0     3.0
Worker4 6.0     9.0     8.0     5.0     1.0
Worker5 7.0     4.0     5.0     8.0     9.0
Worker 1 gets job no: 3
Worker 2 gets job no: 4
Worker 3 gets job no: 1
Worker 4 gets job no: 5
Worker 5 gets job no: 2
Execution time is 22.138 seconds
```