

# **Constructive Comparison Between Two Algorithms**

**-AbssZy**

## **Table of Content**

1. Abstract
2. Introduction
3. Time Complexity
4. Program Input And Output Screenshots

## **Abstract**

In this project our problem statement was to calculate the minimum amount of profit that can be ensured to an investor in a cashflow. Suppose there are certain investments being made in a share market type of background. The weights in the graphs indicate the profit or the loss after the investment at every node. Positive weights indicate profit and negative weights indicate loss. By using this algorithm, we can find the minimum amount of profit which an investor will have at each node. We have used two algorithms using two different techniques

1. Bellman ford algorithm using dynamic programming.
2. A\* algorithm using heuristic search.

We have solved our problem statement using both these algorithms and have made a comparison between the time complexities of the optimal solutions given by both the algorithms.

## **Introduction**

In today's world investing in share market has become a common phenomenon for investing money and making quick money. Though investing in share markets gives quick and good profit returns but there are also high risks for losses. Thus, investors do not feel very convincing to invest in share market. In this project our aim was to solve this issue. Using graphs, we can map the transactions between various lenders into a pictorial description using graphs where the indicate the profit or the loss after the investment and the nodes represent the money lenders. The investor can find out the minimum amount of profit which he/she will definitely get at each node using these techniques.

## Time Complexity

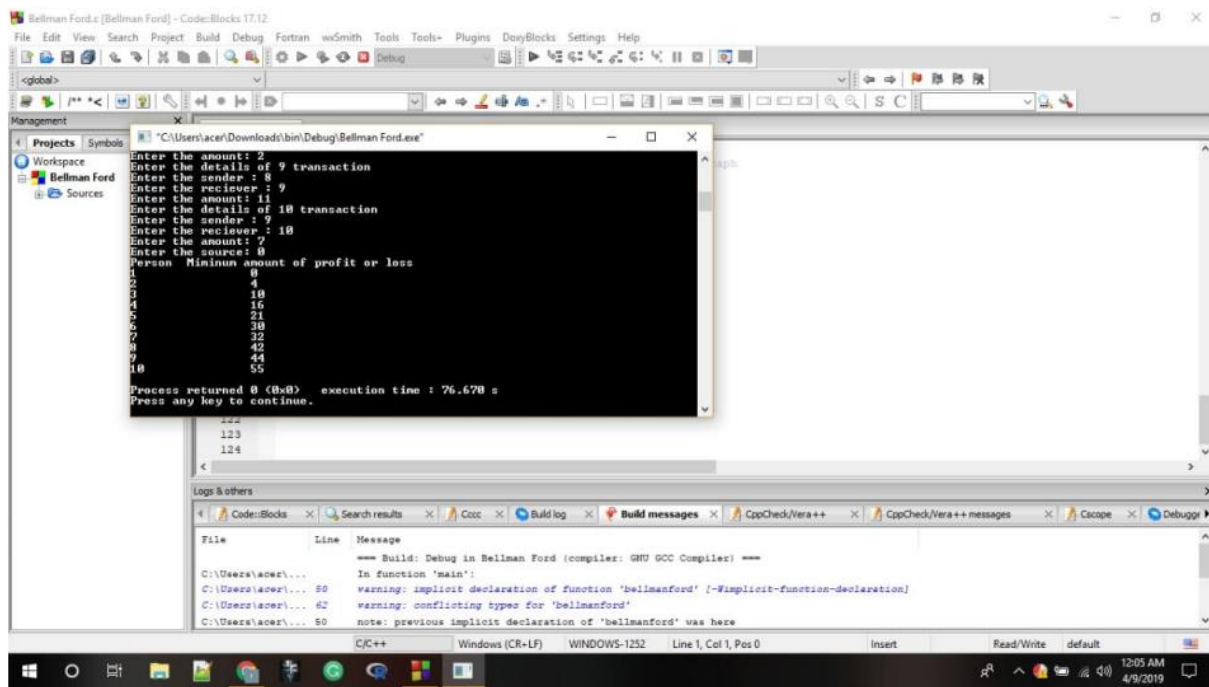
This solution uses two algorithms: -

**Bellman–Ford algorithm:** This algorithm runs in  $O(|V| \cdot |E|)$  where  $|V|$  and  $|E|$  are the number of vertices and edges respectively.

**A Star algorithm:** The time complexity of  $A^*$  depends on the heuristic. In the worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of the solution (the shortest path)  $d$ :  $O(b^d)$ , where  $b$  is the branching factor (the average number of successors per state). This assumes that a goal state exists at all, and is reachable from the start state; if it is not, and the state space is infinite, the algorithm will not terminate.

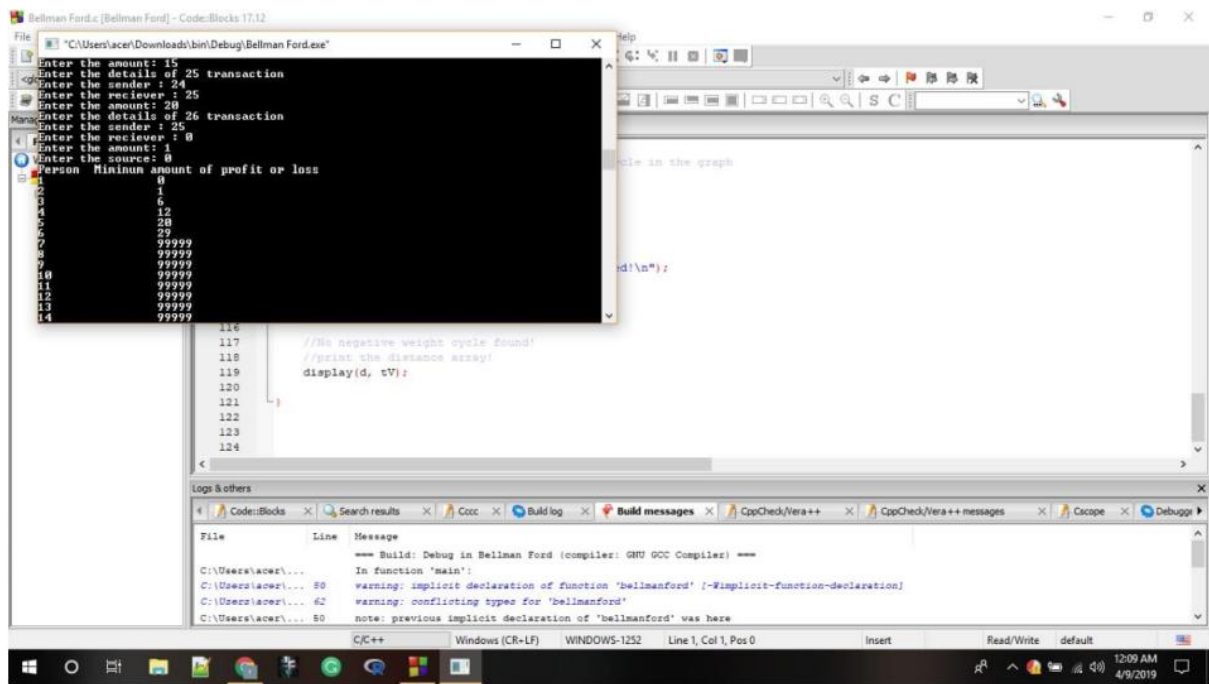
# Code Output

## 1. Bellman Ford Algorithm



```
Enter the amount: 2
Enter the details of 9 transaction
Enter the sender: 8
Enter the receiver: 9
Enter the amount: 11
Enter the details of 10 transaction
Enter the sender: 9
Enter the receiver: 10
Enter the amount: 7
Enter the source: 0
Person Minimum amount of profit or loss
1 0
2 4
3 10
4 16
5 21
6 30
7 32
8 42
9 44
10 55
Process returned 0 (0x0)   execution time : 76.670 s
Press any key to continue.
```

```
File Line Message
C:\Users\acer\... Build: Debug in Bellman Ford (compiler: GNU GCC Compiler)
C:\Users\acer\... 80 In function 'main':
C:\Users\acer\... 62 warning: implicit declaration of function 'bellmanford' [-Wimplicit-function-declaration]
C:\Users\acer\... 62 warning: conflicting types for 'bellmanford'
C:\Users\acer\... 80 note: previous implicit declaration of 'bellmanford' was here
```



```
Enter the amount: 15
Enter the details of 25 transaction
Enter the sender: 24
Enter the receiver: 25
Enter the amount: 20
Enter the details of 26 transaction
Enter the sender: 25
Enter the receiver: 0
Enter the amount: 1
Enter the source: 0
Person Minimum amount of profit or loss
1 0
2 1
3 6
4 12
5 20
6 29
7 99999
8 99999
9 99999
10 99999
11 99999
12 99999
13 99999
14 99999
116 //No negative weight cycle found!
117 //print the distance array!
118 display(d, tv);
119
120
121
122
123
124
```

```
File Line Message
C:\Users\acer\... Build: Debug in Bellman Ford (compiler: GNU GCC Compiler)
C:\Users\acer\... In function 'main':
C:\Users\acer\... 80 warning: implicit declaration of function 'bellmanford' [-Wimplicit-function-declaration]
C:\Users\acer\... 62 warning: conflicting types for 'bellmanford'
C:\Users\acer\... 80 note: previous implicit declaration of 'bellmanford' was here
```

```
Enter the receiver : 2
Enter the amount : 4
Enter the details of 3 transaction
Enter the sender : 2
Enter the receiver : 3
Enter the amount : -2
Enter the details of 4 transaction
Enter the sender : 3
Enter the receiver : 4
Enter the amount : 0
Enter the details of 5 transaction
Enter the sender : 4
Enter the receiver : 5
Enter the amount : 9
Enter the source : 0
Person Minimum amount of profit or loss
1 0
2 3
3 2
4 5
5 5
Process returned 0 (0x0) execution time : 29.699 s
Press any key to continue.
```

Log & others

File	Line	Message
=== Build: Debug in Bellman Ford (compiler: GNU GCC Compiler) ===		
C:\Users\acer\...		In function 'main':
C:\Users\acer\... 50		warning: implicit declaration of function 'bellmanford' [-Wimplicit-function-declaration]
C:\Users\acer\... 62		warning: conflicting types for 'bellmanford'
C:\Users\acer\... 80		note: previous implicit declaration of 'bellmanford' was here

```
99999
10 99999
11 99999
12 99999
13 99999
14 99999
15 99999
16 99999
17 99999
18 99999
19 99999
20 99999
21 99999
22 99999
23 99999
24 99999
25 99999
Process returned 0 (0x0) execution time : 169.001 s
Press any key to continue.
```

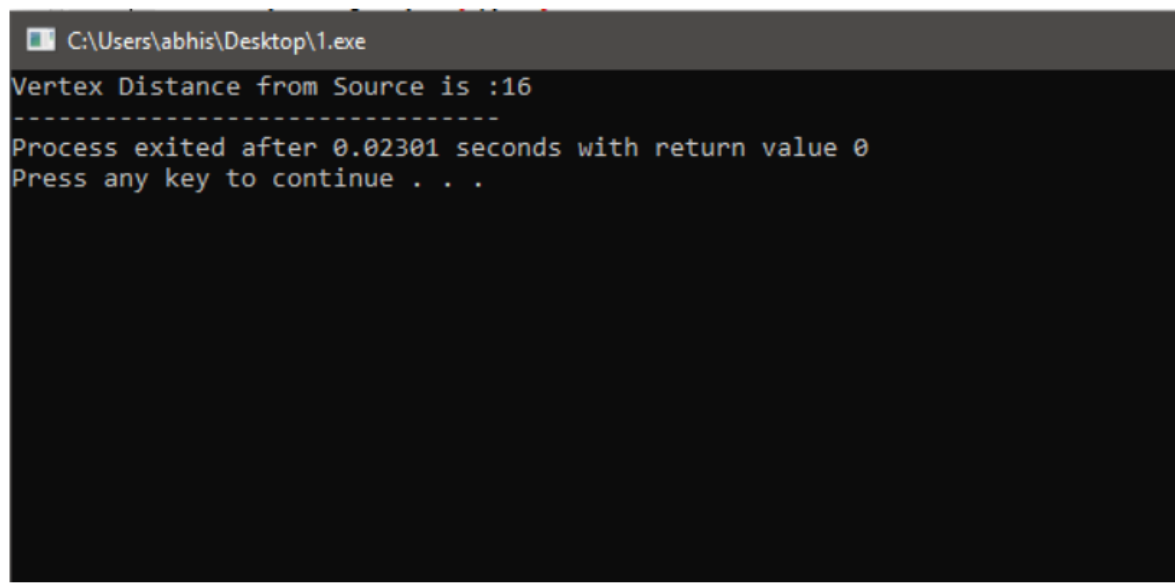
Log & others

File	Line	Message
=== Build: Debug in Bellman Ford (compiler: GNU GCC Compiler) ===		
C:\Users\acer\...		In function 'main':
C:\Users\acer\... 50		warning: implicit declaration of function 'bellmanford' [-Wimplicit-function-declaration]
C:\Users\acer\... 62		warning: conflicting types for 'bellmanford'
C:\Users\acer\... 80		note: previous implicit declaration of 'bellmanford' was here

## 2. A-Star Algorithm

```
C:\Users\abhis\Desktop\1.exe
Vertex Distance from Source is :5
-----
Process exited after 0.02604 seconds with return value 0
Press any key to continue . . .
```

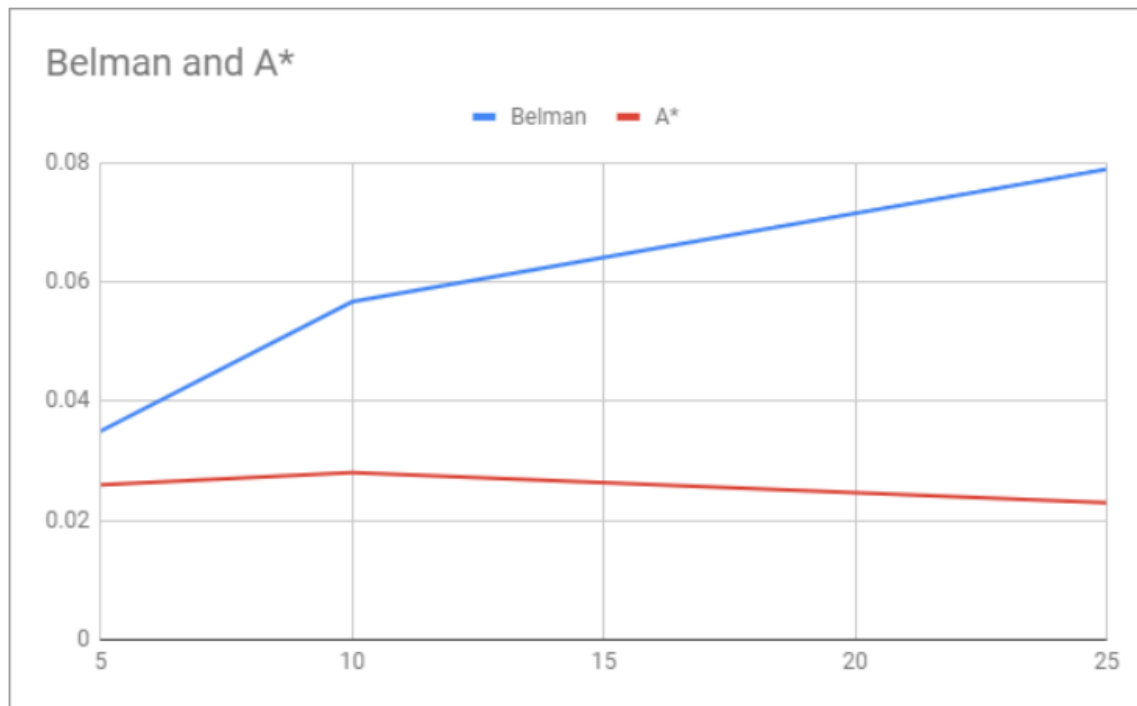
```
C:\Users\abhis\Desktop\1.exe
Vertex Distance from Source is :8
-----
Process exited after 0.02618 seconds with return value 0
Press any key to continue . . .
```



```
C:\Users\abhis\Desktop\1.exe
Vertex Distance from Source is :16
-----
Process exited after 0.02301 seconds with return value 0
Press any key to continue . . .
```



## Result



We have thus compared both the algorithms by plotting the graphs of the time complexities taken by both the algorithms.

## **Inference**

From the graph we infer that as  $n$  grows the bellman ford algorithm which uses dynamic programming, the time complexity also grows exponentially while for the A star algorithm which uses heuristics the time complexity remains nearly constant. This is because the astar algorithm does not visit every node and finds the minimum cost from source to destination by using the approximate heuristic function available for every node.

On the other hand the bellman ford algorithm which uses dynamic programming visits every node and also provide the minimum cost from source to every other node.

That is why dynamic programing gives the most optimal solution but the heuristic technique might not give the best optimal solution but it consumes less time than dynamic programing.

## **References**

A Star Algorithm base paper

<https://pdfs.semanticscholar.org/e2fc/bf5b2167c0c14d1be708e9a78b18616da474.pdf>

Bellman Ford Algorithm base paper

[https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.ijedr.org/papers/IJEDR1801130.pdf&ved=2ahUKEwi7ybvC08LhAhWLerwKHVsdDvsQFjADegQIARAB&usg=AOvVaw0t1lhfdHQ\\_r8kfMjRRU730](https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.ijedr.org/papers/IJEDR1801130.pdf&ved=2ahUKEwi7ybvC08LhAhWLerwKHVsdDvsQFjADegQIARAB&usg=AOvVaw0t1lhfdHQ_r8kfMjRRU730)