IMPLEMENTATION OF DIJKSTRA’S & FLOYD WARSHALL ALGORITHM IN FINDING THE SHORTEST PATH FOR ROAD NETWORKS

2CSOE52: DATA STRUCTURES

SPECIAL ASSIGNMENT

B.TECH, SEMESTER – VI

ELECTRONICS & COMMUNICATION ENGINEERING

NIRMA UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED ‘A’ GRADE

SUBMITTED BY,

VINCENT ABRAHAM, ROLL NO. 18BEC122

DIVYESH RANPARIYA, ROLL NO. 18BEC083

SUBMITTED TO,

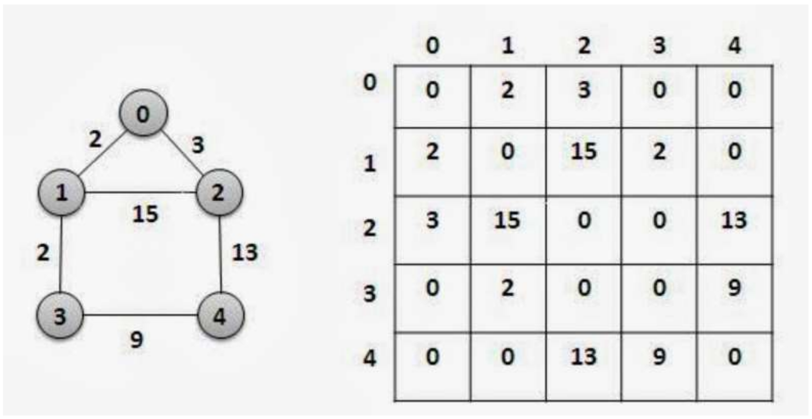
Professor Malaram Kumhar

INTRODUCTION:

Graph is a non-linear data structure used to represent the network in terms of nodes and corresponding edges connecting to nodes. For any applications, graphs can be represented in the following two types:

1. Adjacency List
2. Adjacency Matrix

Adjacency matrix method is method to represent the graph in terms of connection representation by corresponding weights and no connection represented by infinity.



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

Figure 1: Adjacency Matrix Representation of Undirected Weighted Graphs.

In above figure 1 adjacency matrix representation is shown. The values represent the cost or weight of edges connecting to corresponding node represented by labels of row and column. Graphs are prefect mapping for the non-symmetric network or map having nodes meeting each other. The best real-life example of graphs are road networks. So, finding the optimum path is also a challenging task in dense road networks. Two algorithms, namely Dijkstra's and Floyd Warshall algorithm are very popularly used for this purpose.

PROBLEM STATEMENT

Compute the shortest route for determining the nearest hospital taking in account the distance, as well as the time taken in going from the starting point to a destination.

- The user would input the number of hospitals and their names.
- Distance & time taken in travelling from one point to another would also be given by the user.

- A starting point would be given by the user.
- Compute & display the shortest path from the starting point to all the other destinations.
- Considering both the distance & time, display an overall shortest route.

METHODOLOGIES

Dijkstra's Algorithm

The Dijkastra's algorithm is called single source shortest path algorithm, based on greedy technique giving the shortest path of all vertices from the source node in non-negative weighted graph. It is usually used in industries based on modeling the networks for minimum shortest distancing.

It applies blind search, and keeps track of vertices that has been visited. It is widely used in GPS to find the shortest route in the dense road network.

The steps involved in the algorithm are as follows,

1. Create cost matrix `C[ ][ ]` from adjacency matrix `G[ ][ ]`. `C[u][v]` is the cost of going from vertex (`u`) to vertex (`v`). If there is no edge between vertices `u` and `v` then `C[u][v]` is infinity.
2. Array `visited[ ]` is initialized to zero and get cost of source node to distance vector.
```
for(i = 0; i < n; i++)
      dist[i] = cost[start][i]

      visited[i] = 0

      pred[i]=start

End for
```
3. The source vertex `visited[start]` is marked as 1.
4. `while (count<n-1) // n is number of nodes`
```
for(i = 1; i < n; i++)
```

```
        o  Choose a vertex u, such that dist[u] is minimum and visited[u]
           is 0. Mark visited[u] as 1.
        o  Recalculate the shortest distance of remaining vertices
        o  The vertices not marked as 1 in array visited[ ] will go for
           recalculation of distance.

    End for

    for each vertex (v)

            if(visited[v]==0)

            dist[v]=minimum(dist[v], dist[u]+cost[u][v])

    End for
```

The time-complexity of this algorithm would be in the order of O(n^2). This can be verified by observing the iterations in the for loop.

Floyd Warshall Algorithm

The Floyd Warshall Algorithm is a dynamic algorithm which returns, in a single execution, the distance of the shortest paths for all the pairs of vertices in a weighted directed/undirected graph. Further modifications in the algorithm can be made in order to compute the shortest paths for the vertices.

The simplicity and efficiency of this algorithm allows one to implement it for extremely dense graphs. Hence, this algorithm is a good fit for the purpose of this project as road networks generally tend to be very dense.

This algorithm involves updating a solution matrix (called the distance matrix) considering, iteratively, all the nodes as intermediate nodes in a path from one node to another. The steps involved in this algorithm are as follows,

1. Initialize the distance matrix (`dist`) same as the received adjacency matrix. The distance from the node to itself would be zero, and if there’s no connection between two nodes, the distance between them would be infinity (`INF`).

2.
   ```
   For every intermediate node (k)
         For every source vertex (i)
               For every destination vertex (j)
   ```

```
                    If dist[i][k] + dist[k][j] < dist[i][j]
                         Then,
                         dist[i][j] = dist[i][k] + dist[k][j]
                    End If
               End For
          End For
     End For
```

For storing the shortest path from a node (`i`) to node (`j`), an additional matrix (`next`) can be initialized

1.
   ```
   If path exists between (i) and (j)
        next[i][j] = j
   Else
        next[i][j] = -1
   End If
   ```
2. Now, inside the if condition of the main algorithm, the change would be as follows,
   ```
   If dist[i][k] + dist[k][j] < dist[i][j]
        Then,
        dist[i][j] = dist[i][k] + dist[k][j]
        next[i][j] = next[i][k]
   End If
   ```

The time-complexity of the Floyd-Warshall algorithm would be in the order of O(n^3). This can be verified by observing the iterations in the for loop.

IMPLEMENTATION & RESULTS

Shown below are the graphs that are going to be used for the implementation,

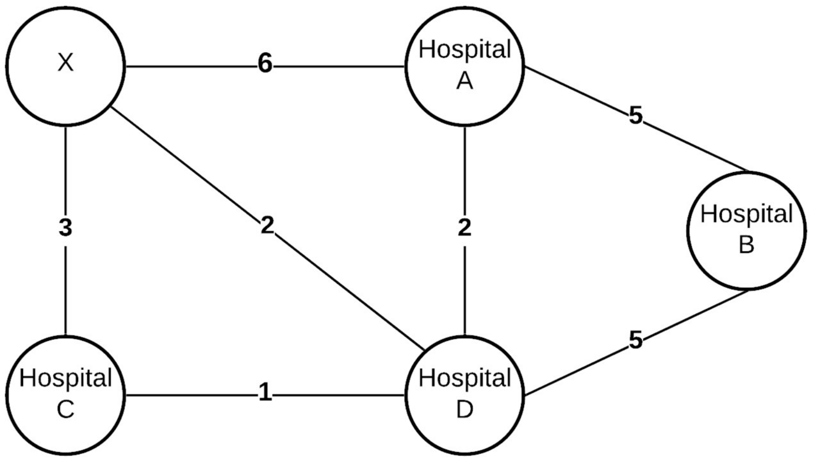


Figure 2: Distance Graph

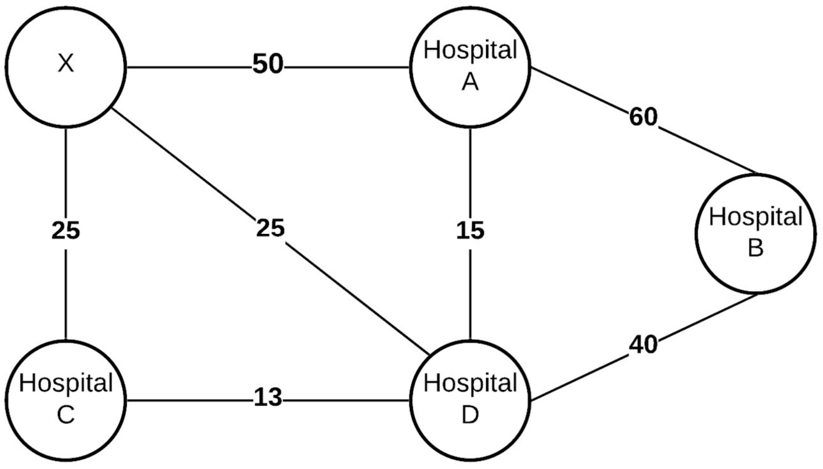
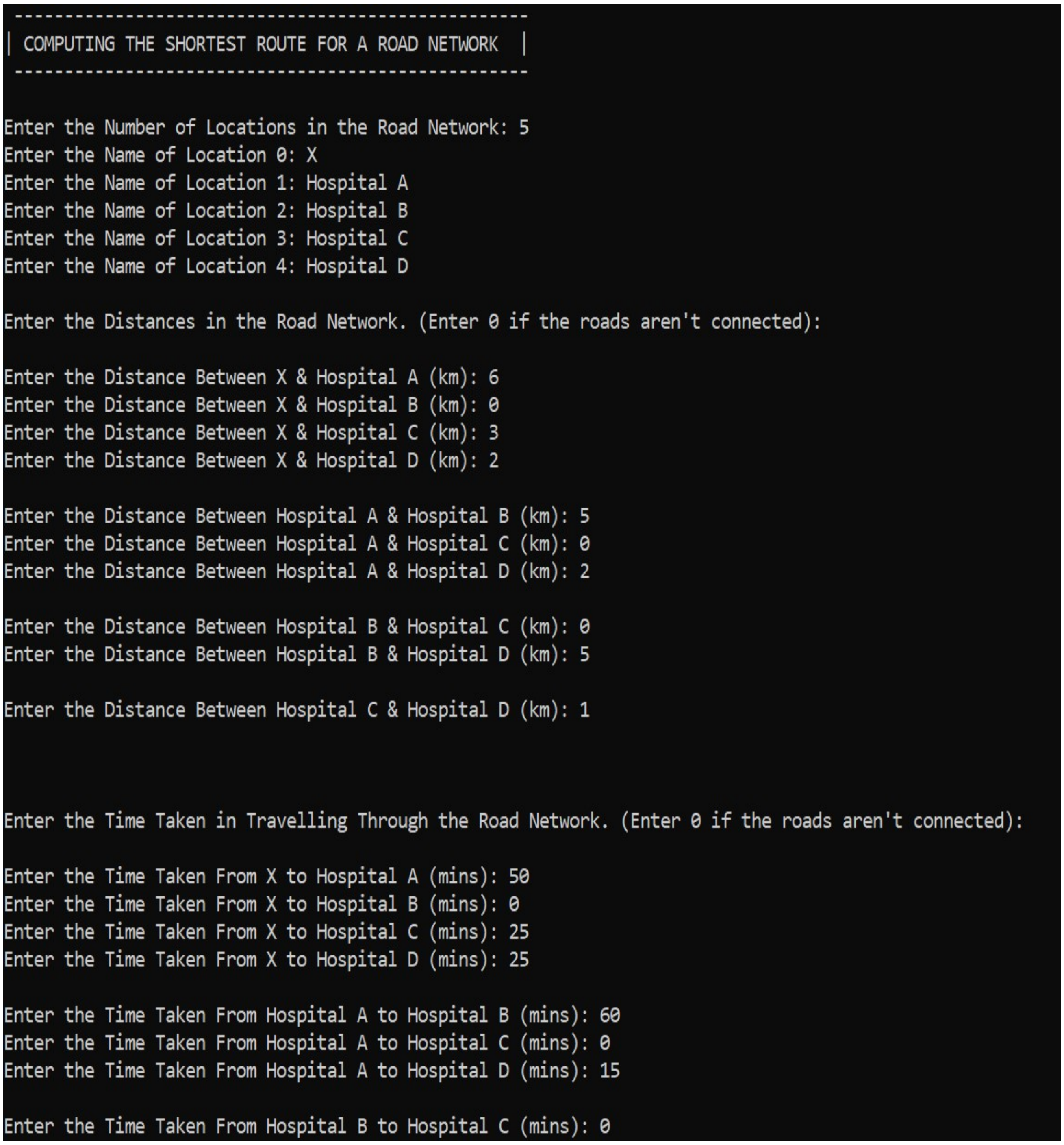


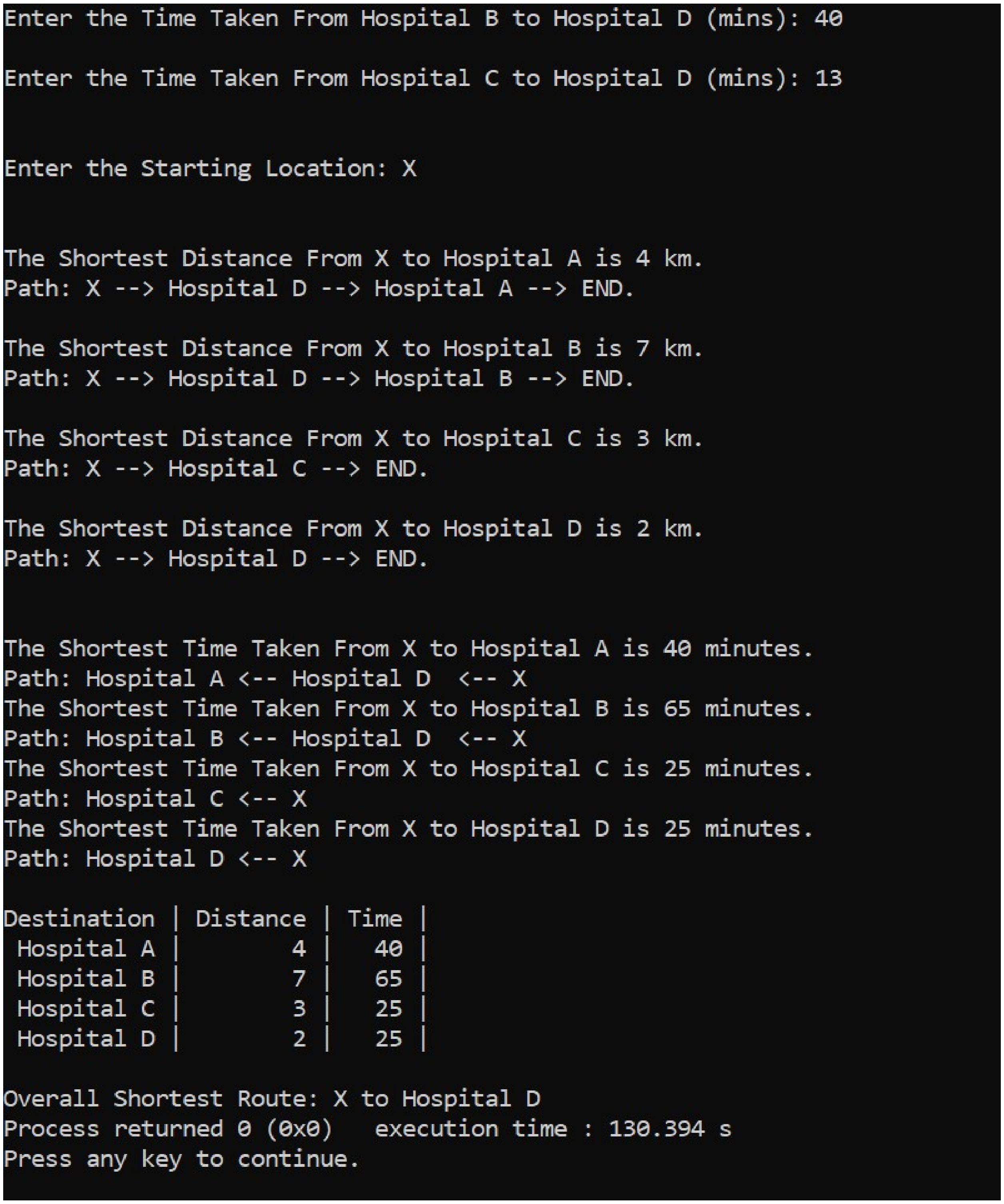
Figure 3: Time Graph

Both the graphs used are undirected and weighted.

The simulation results were as follows,



```
 ---------------------------------------------------
| COMPUTING THE SHORTEST ROUTE FOR A ROAD NETWORK  |
 ---------------------------------------------------

Enter the Number of Locations in the Road Network: 5
Enter the Name of Location 0: X
Enter the Name of Location 1: Hospital A
Enter the Name of Location 2: Hospital B
Enter the Name of Location 3: Hospital C
Enter the Name of Location 4: Hospital D

Enter the Distances in the Road Network. (Enter 0 if the roads aren't connected):

Enter the Distance Between X & Hospital A (km): 6
Enter the Distance Between X & Hospital B (km): 0
Enter the Distance Between X & Hospital C (km): 3
Enter the Distance Between X & Hospital D (km): 2

Enter the Distance Between Hospital A & Hospital B (km): 5
Enter the Distance Between Hospital A & Hospital C (km): 0
Enter the Distance Between Hospital A & Hospital D (km): 2

Enter the Distance Between Hospital B & Hospital C (km): 0
Enter the Distance Between Hospital B & Hospital D (km): 5

Enter the Distance Between Hospital C & Hospital D (km): 1



Enter the Time Taken in Travelling Through the Road Network. (Enter 0 if the roads aren't connected):

Enter the Time Taken From X to Hospital A (mins): 50
Enter the Time Taken From X to Hospital B (mins): 0
Enter the Time Taken From X to Hospital C (mins): 25
Enter the Time Taken From X to Hospital D (mins): 25

Enter the Time Taken From Hospital A to Hospital B (mins): 60
Enter the Time Taken From Hospital A to Hospital C (mins): 0
Enter the Time Taken From Hospital A to Hospital D (mins): 15

Enter the Time Taken From Hospital B to Hospital C (mins): 0
```



```
Enter the Time Taken From Hospital B to Hospital D (mins): 40

Enter the Time Taken From Hospital C to Hospital D (mins): 13


Enter the Starting Location: X


The Shortest Distance From X to Hospital A is 4 km.
Path: X --> Hospital D --> Hospital A --> END.

The Shortest Distance From X to Hospital B is 7 km.
Path: X --> Hospital D --> Hospital B --> END.

The Shortest Distance From X to Hospital C is 3 km.
Path: X --> Hospital C --> END.

The Shortest Distance From X to Hospital D is 2 km.
Path: X --> Hospital D --> END.


The Shortest Time Taken From X to Hospital A is 40 minutes.
Path: Hospital A <-- Hospital D  <-- X
The Shortest Time Taken From X to Hospital B is 65 minutes.
Path: Hospital B <-- Hospital D  <-- X
The Shortest Time Taken From X to Hospital C is 25 minutes.
Path: Hospital C <-- X
The Shortest Time Taken From X to Hospital D is 25 minutes.
Path: Hospital D <-- X

Destination | Distance | Time |
 Hospital A |        4 |   40 |
 Hospital B |        7 |   65 |
 Hospital C |        3 |   25 |
 Hospital D |        2 |   25 |

Overall Shortest Route: X to Hospital D
Process returned 0 (0x0)   execution time : 130.394 s
Press any key to continue.
```

COMPARING BOTH THE ALGORITHMS

Parameters	**Dijkstra's**	**Floyd Warshall**
No. of nodes	10	10
Time Complexity	O(100)	O(1000)
Time Taken (seconds)	0.011	0.022

From table it can be observed that time taken by of Dijkstra algorithm is lesser than Floyd Warshall algorithm as Dijkstra is having O(n^2) and Floyd Warshall is having complexity of O(n^3). So for the timing critical calculation it is advisable to use Dijkstra over Floyd with the disadvantage of getting distances only from one source node in Dijkstra.

Floyd warshall algorithm has also been proved to work with the negative edges (not negative cycles). Where as Dijkstra's algorithm is applicable only for non-negative weights of the graphs.

REFERENCES

[1] Risald, A. E. Mirino and Suyoto, "Best routes selection using Dijkstra and Floyd-Warshall algorithm," *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, 2017, pp. 155-158, doi: 10.1109/ICTS.2017.8265662.