

SIKSHA 'O' ANUSANDHAN
INSTITUTE OF TECHNICAL EDUCATION
AND RESEARCH

Algorithm Design – 1 (CSE3131)

Project report

On

**SHIP ROUTING IN THE BLACK SEA BASED ON
DIJKSTRA ALGORITHM**



SUBMITTED BY:-

Sl. No.	Name	Registration No.	Branch/Sec
1.	Bikash Kumar Dash	1941012406	CSE - D
2.	Saswat Mohanty	1941012407	CSE - D

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our mentor, **Mr. Paresh Baidya** for his unceasing support at our **Algorithm Design study and research**, unrivalled patience, motivation, enthusiasm and, colossal knowledge during the active discourse of our end semester project. His all-time charm and insightful persona has been a blessing in disguise to all of us, which has immensely catalyzed the process during all time of research and writing this piece of work. We could not have imagined having a better advisor and mentor for our project on **“Ship Routing in the Black Sea based on Dijkstra Algorithm.”**

Adding up, as fellow members, we are equally reciprocative to each one of our relentless efforts at completion of the project, under the stipulated time frame. The members include **Bikash Kumar Dash** (1941012406) and **Saswat Mohanty** (1941012407). We are immensely grateful and value all the stimulating discussions, sleepless nights, rigorous online meetings and a lot more subtleties that have been poured in, to make the project work a very effective and a completely transformative experience. Again, not to forget all the fun we have had in the last few weeks. In particular, it was a very engaging learning experience and we are proud to have worked together, as a team.

Last but not the least, we would like to thank our **family** members for showing such strong character and acting the backbone to keep the process ever energetic, lively and anchored. In short, we just can't do enough to return the selfless gestures with any better grandeur. Thank you for everything, our all-time saviors.

Input Specification

Dijkstra's Algorithm finds the shortest path between a given node and all other nodes in a graph. This algorithm uses the weights of the edges to find the path that minimizes the total distance (weight) between the source node and all other nodes.

Here we need only two input specifications: 1-vertex, 2-node.

Conditions or Constraints

- The value of the path-cost should be positive.
- The addition of edges should not form a cycle.

Goal/Objective/Desired output

In determining the best navigation route or solving SPP (Shortest Path Problem), Dijkstra algorithm managed with success for small weighted graphs with N nodes, and E edges searching the shortest path from the source node to all others until reaching the destination with the lowest cost. For Example: The case study proposed by this paper is related to optimization of the cost of transport (fuel consumption, ship time, crew), the structural safety of the ship (accident prevention), and protection of the environment (emissions of GHG, sea pollution). The study area represented by a portion of the Black Sea basin and chosen to investigate a voyage from south to north, more precisely from the touristic port of Sinop (42.019500N, 35.142500E) and Sevastopol (42.019500N, 35.142500E). The scenario uses meteorological and oceanographic data of a numerical model provided by Copernicus Marine when in the proximity of the northern littoral of the Black Sea waves with heights above 4 m were recorded. We mention that due to navigation purposes, namely port approach and departure procedures and maneuvers, the graph will consider different vertices as sources and destinations.

Brief Description

In the precursory section of the paper, we succinctly present the Black Sea basin's oceanographic features with a focus on factors that can lead to naval accidents. The Black Sea basin is characterized, especially in winter months, by sudden storms, big waves, and winds, seriously affecting the ships and crews' safety. The said basin has also confined areas with pleasure crafts and commercial ships, offshore platforms with accompanying auxiliary fleet and underwater pipes and installations or shallow waters, and wrecks all of them favoring naval accidents. The naval accidents can lead to the dramatic loss of human lives, damage or loss of property and negatively impact on the marine environment. In the paper, we establish a model based on realistic wind and wave conditions presenting the **Black Sea and analyze the output given by an adapted Dijkstra algorithm**. We set a study area, from the whole Black Sea basin, chose a specific region, split it into squares, and the algorithm found the potential waypoints of the recommended path as the vertices of the established squares. Finding the optimal route between two ports being the most efficient from an economic point of view, reduced emissions, and the safety of the crew and the ship is the main objective of the paper. Usually, the routing algorithms apply longer routes considerably. The proposed algorithm was meant to assist shipmasters' and owners' decisions and minimize fuel consumption by choosing the shortest and safest path related to shipping and sea conditions.

Mathematical formulation with Model

We step through Dijkstra's algorithm on the graph used in the algorithm above:

1. Initialize distances according to the algorithm.
2. Pick the first node and calculate distances to adjacent nodes.
3. Pick the next node with minimal distance; repeat adjacent node distance calculations.
4. Final result of shortest-path tree.

Algorithm Design Paradigm Used

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Mark all nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
2. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes. Set the initial node as current.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances through the current node. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be $6 + 2 = 8$. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, the current value will be kept.
4. When we are done considering all of the unvisited neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

6. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

When planning a route, it is actually not necessary to wait until the destination node is "visited" as above: the algorithm can stop once the destination node has the smallest tentative distance among all "unvisited" nodes (and thus could be selected as the next "current").

Pseudo code

In the following pseudo code algorithm, the code $v \leftarrow \text{vertex in } Q \text{ with } \min \text{dist}[u]$, searches for the vertex v in the vertex set Q that has the least $\text{dist}[u]$ value. $\text{length}(v, V)$ returns the length of the edge joining (i.e. the distance between) the two neighbor-nodes v and V . The variable $\text{adj}[u]$ is the length of the path from the root node to the neighbor node V if it were to go through v . If this path is shorter than the current shortest path recorded for v , that current path is replaced with this $\text{adj}[u]$ path. The prev array is populated with a pointer to the node on the source graph to get the shortest route to the source.

```
D[s] ← 0
For each v ∈ V-{s}
    Do d[v] ← infinity
    S ← ∅
    Q ← V
    Q is a priority queue maintaining -S
While Q ≠ ∅
    Do u ← extract-Min(Q)
    S ← S ∪ {u}
    For each v ∈ Adj[u]
        Do if d[v] > d[u] + w(u,v)
            Then d[v] ← d[u] + w(u,v)
```

Code

The code has been written using Python language.

```

def dijkstra(graph,src,dest,visited=[],distances={},predecessors={}):
    if src not in graph:
        raise TypeError('The root of the shortest path tree cannot be
found')
    if dest not in graph:
        raise TypeError('The target of the shortest path cannot be
found')
    if src == dest:
        path=[]
        pred=dest
        while pred != None:
            path.append(pred)
            pred=predecessors.get(pred,None)
            readable=path[0]
            for index in range(1,len(path)): readable = path[index]+'---
>' +readable
            print('shortest path - array: ' +str(path))
            print("path:" + readable + ", cost="+str(distances[dest]))
        else :
            if not visited:
                distances[src]=0
            for neighbor in graph[src] :
                if neighbor not in visited:
                    new_distance = distances[src] + graph[src][neighbor]
                    if new_distance <
distances.get(neighbor,float('inf')):
                        distances[neighbor] = new_distance
                        predecessors[neighbor] = src
            visited.append(src)
            unvisited={}
            for k in graph:
                if k not in visited:
                    unvisited[k] = distances.get(k,float('inf'))
            x=min(unvisited, key=unvisited.get)
            dijkstra(graph,x,dest,visited,distances,predecessors)
if __name__ == "__main__":
    graph = {'A1': {'B1': 1, 'B2': 1.41, 'A2': 1}, 'A2': {'B1': 1.41,
'B2': 1, 'B3': 1.41, 'A1': 1, 'A3': 1}
    .....}
}

```

Results Analysis

In the real situation in the time interval considered on the first path, without applying the routing algorithm, the ship will encounter high seas with a harmful impact on ship safety.

After running the code, we observed that the path chosen was A1 -- B2 -- C3 -- C4 -- D5 -- D6 -- E7 -- F8 -- G8 -- H9. The cost for this path is 11.459, corresponding to a distance of 229.2 nautical miles, with 11.8 nautical miles longer than the previous one in calm sea conditions. The model ship can navigate this distance through 28 h 30 m compared with 27 h 11 m without the high sea.

The difference between the two paths is not considerable, and our main aim was to bring the ship in safe conditions with minimum impact on the environment and with a minimum economic cost.

The difference between two paths is not considerable and our main aim was to bring the ship in safety conditions with minimum impact on the environment and with a minimum economic cost.

```
dijkstra(graph, 'A1', 'H9')
shortest path - array: ['H9', 'G8', 'F8', 'E7', 'D6', 'D5', 'C4', 'C3', 'B2', 'A1']
path: A1--->B2--->C3--->C4--->D5--->D6--->E7--->F8--->G8--->H9, cost=11.459999999999999
```

Conclusion

The model proposed proved that complicated navigation applications involving plotting and consistent volume of calculations were easily solved with programming tools with a considerable economy of time.

The algorithm chose the optimal route though at first view the path is not much longer, and compared the emissions level is higher, but the most important is the ship avoided the dangerous area prone to put the ship in danger with potential impact on property, human life and environment.

The optimum route chosen with the help of the algorithm and Python code is just inception; for the future, multiple directions of research arose. Among these opportunities of research are:

- decreasing the spatial resolution which will result in an increased number of nodes and accuracy of routes chosen
- extending the area of study for the whole basin of the Black Sea
- increasing the number of true headings from 8 to 16 or 32
- automation of the input by using predefined codes and nodes networks
- embedding forecast models and codes that will allow the crew to run different scenarios and select the one with the best performance
- engage data visualization tools and compile outputs for fast graphic analysis

Real-life Applications

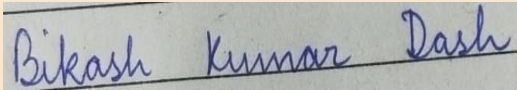
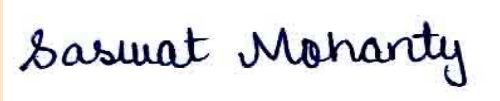
Dijkstra's algorithm is one of the most popular algorithms for solving many single-source shortest path problems having non-negative edge weight in the graphs i.e., it is to find the shortest distance between two vertices on a graph. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

Dijkstra's Algorithm has several real-world use cases, some of which are as follows:

1. Digital Mapping Services in Google Maps
2. Social Networking Applications
3. Telephone Network
4. IP routing to find Open shortest Path First
5. Flying Agenda
6. Designate file server
7. Robotic Path

Statement of Declaration for Genuinity of the Project

We, the undersigned students of B. Tech. **Computer Science And Engineering** Department hereby declare that we own the full responsibility for the information, results etc. provided in this PROJECT titled “**Ship Routing in the Black Sea based on Dijkstra Algorithm**” submitted to **Siksha ‘O’ Anusandhan Deemed to be University, Bhubaneswar** for the partial fulfillment of the subject **Algorithm Design – 1 (CSE3131)**. We have taken care in all respects to honor the intellectual property right and have acknowledged the contribution of others for using them in academic purpose and further declare that in case of any violation of intellectual property right or copyright we, as the candidate(s), will be fully responsible for the same.

NAME	REGD. NO.	SIGNATURE
Bikash Kumar Dash	1941012406	
Saswat Mohanty	1941012407	

DATE: 17th July 2021

PLACE: ITER, SOA University, Odisha.