Mandalay Division Data Analysis by Dijkstra Shortest Path Algorithm with Python

Aung Cho[1], Aung Si Thu[2]
[1]ITSM, [2]FCST,UCS,Maubin,Myanmar
[1]am6244052@gmail.com,[2]htinlutt2016@gmail.com

Abstract

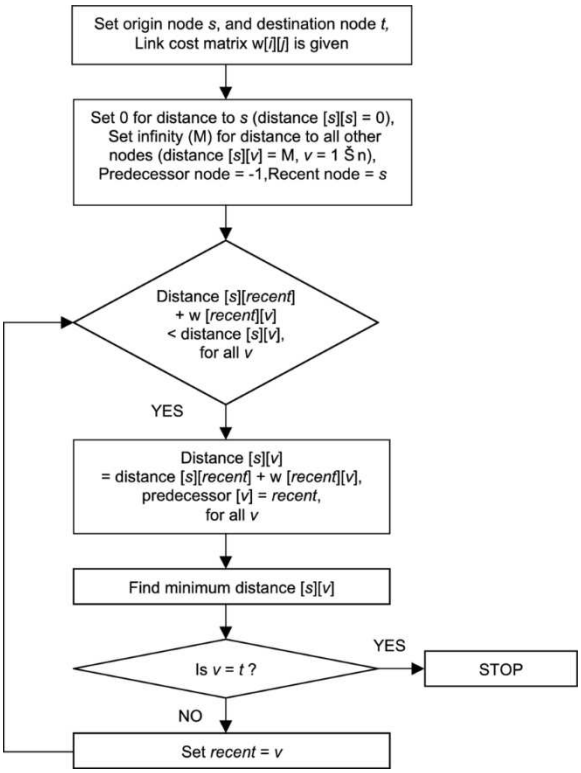
Todays every business is competing with each other and they always choose the shortest path to distribute their products and services to their customers. Also tourists, ambulance, fireman and network connection company always choose the shortest path to reach their goals. If they go through in Mandalay division, Myanmar, this paper can help them to get the shortest path of their trip. In this paper the data of Mandalay was analyzed with Dijkstra shortest path algorithm and python code supported to implement the data analyzing.

Keywords: Dijkstra shortest path algorithm, python, Mandalay

1.Introduction

Nowadays all people in the world need the shortest path to arrive their goal in fast. Shorter trip is, cost and time more save and profit is more. Dijkstra algorithm makes the trip shortest. In this paper Mandalay division data is used and python program implemented the data to get output. In this paper includes Dijkstra background algorithm, system flow chart, Mandalay map, original graph, output graph, data analytical view and manual check table.

2.Background Algorithm



Graph-1(Flowchart for-the-Shortest-Path-Algorithm)[1]

Dijkstra's Algorithm[2]

Steps of Dijkstra Shortest path algorithm

Step1:Source vertex is set value of zero and others are set values of infinity.

Step2:Fill stack array with distance and vertex pair in each room of the array.

Step3:Minimum distance vertex is taken from the stack.

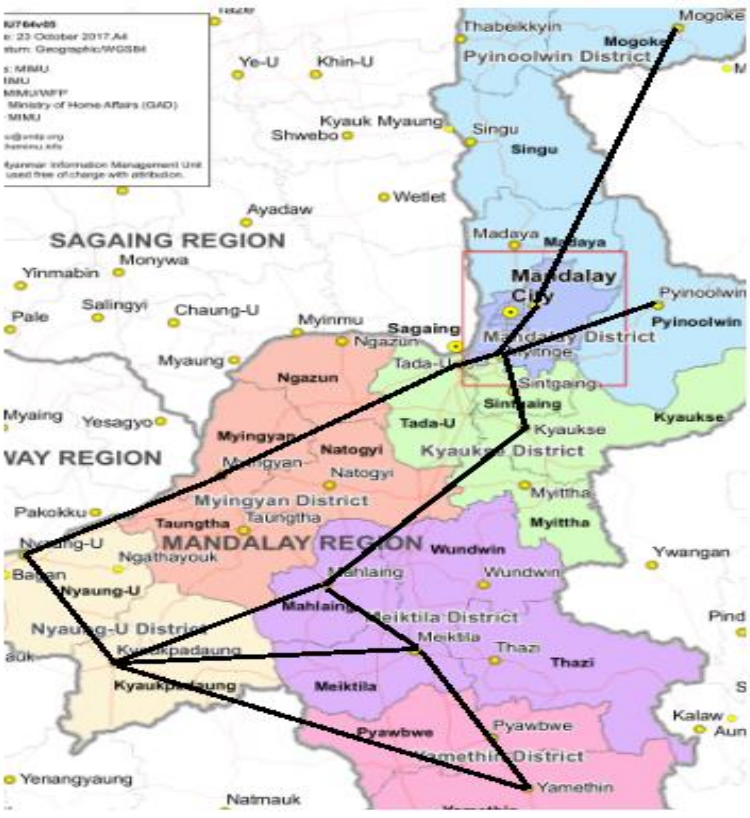
Step4:For the vertexes direct connected with the chosen vertex, their distance values are updated.

Step5:If the distance of present vertex + weight of edge is less than the distance of next vertex, then the value of the vertex is updated with new value.

Step6:If the vertex popped from the stack is visited in the previous time, not to use it and continue next process.

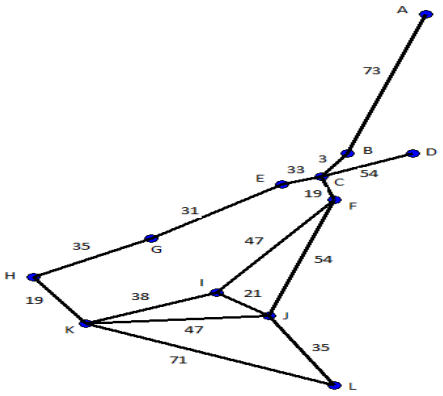
Step7:While the stack includes empty, continue do the process 2 to 6.

3.Implementation



Graph-2(Mandalay Division Map)[3]

Original Path Graph



Graph-3(Original path)

Some Python Codes:

Some Python Codes:

- dijkstra(self, source, dest) method was used.

- distances = {vertex: inf for vertex in self.vertices}means source vertex is set zero and others set infitity.

- previous, current vertexes is changes for minimum distance.

- minimum distance vertex is remove from queue.

-stop processes in the time queue is empty.

When create graph with pair of vertex and distance, two directions values was used as the following:

Line1:graph = Graph([

Line2:("a", "b",73), ("b", "c", 3), ("c", "d", 54), ("c", "e", 33),

Line3:("c", "f", 19), ("e", "g", 31), ("f", "i", 47), ("f", "j", 54),

Line4:("g", "h", 35), ("h","k",19),("i", "j", 21), ("i", "k", 38),

Line5:("j", "k", 47), ("j", "l", 35), ("k", "l", 71),

Line6:("b", "a",73), ("c", "b", 3), ("d", "c", 54), ("e", "c", 33),

Line7:("f", "c", 19), ("g", "e", 31), ("i", "f", 47), ("j", "f", 54),

Line8:("h", "g", 35), ("k","h",19),("j", "i", 21), ("k", "i", 38),

Line9:("k", "j", 47), ("l", "j", 35), ("l", "k", 71)

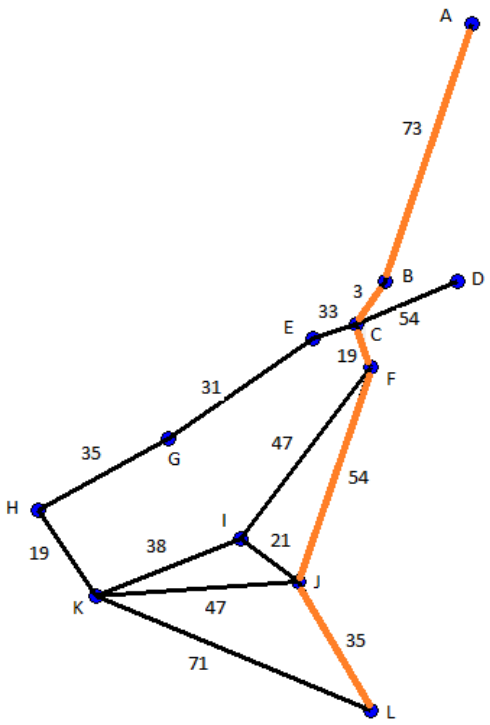
Line10:])

For input data, source is a and end is l.

Line11:graph.dijkstra("a", "l")

output

```
deque(['l'])
184
deque(['j', 'l'])
149
deque(['f', 'j', 'l'])
95
deque(['c', 'f', 'j', 'l'])
76
deque(['b', 'c', 'f', 'j', 'l'])
73
deque(['a', 'b', 'c', 'f', 'j', 'l'])
0
```



Graph-4(Output shortest path)

Marked Node	A	B	C	D	E	F	G	H	I	J	K	L
A	0	73,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A
B	0	73,A	76,B	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A
C	0	73,A	76,B	130,C	109,C	95,C	∞,A	∞,A	∞,A	∞,A	∞,A	∞,A
F	0	73,A	3,A	130,C	109,C	95,C	∞,A	∞,A	142,F	149,F	∞,A	∞,A
E	0	73,A	3,A	130,C	109,C	95,C	140,E	∞,A	142,F	149,F	∞,A	∞,A
D	0	73,A	3,A	130,C	109,C	95,C	140,E	∞,A	142,F	149,F	∞,A	∞,A
G	0	73,A	3,A	130,C	109,C	95,C	140,E	175,G	142,F	149,F	∞,A	∞,A
I	0	73,A	3,A	130,C	109,C	95,C	140,E	175,G	142,F	149,F	180,I	∞,A
J	1	73,A	3,A	130,C	109,C	95,C	140,E	175,G	142,F	149,F	180,I	184,J

Table-1 (Manual Checking Output with Dijkstra's algorithm)

Shortest path(A to L) = L to J + J to F + F to C + C to A

184 = 35 + 54 + 19 + 76

3.1Data analytical view

Any source to any end can be tested by two directions to get the shortest path.

Example: A to L is same distant as L to A.

B to J is same distant as J to B.

4.Conclusion

Todays people in the world needs the shortest path to reach their goals of trips for various cases such as business social, education and so on. If the trip is shorter, cost and time more save and profit is more. Dijkstra shortest path algorithm can solve their needs. In this paper data of Mandalay division, Myanmar is used to implement the data analysis by shortest path algorithm. Moreover, other maps in the world can be tested with Dijkstra algorithm. In this paper used python program to implement the data analysis for producing of shortest path output.

References

1. https://images.search.yahoo.com/...

2. https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/

3. https://www.Google.com

4. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

5. https://www.codespeedy.com/how-to-implement-dijkstras-shortest-path-algorithm-in-python/