

Dijkstra's Algorithm by Jaeson Han

Dijkstra's algorithm is a systematic method of finding the fastest way to get from point A to point B in a weighted graph. It was first introduced in 1959. First, we describe the algorithm and then we give an example using a map of the Penn State University Park campus.

Find or create a map with various places, or "nodes" and roads, or "edges" connecting them. (See Figures 1 and 2.) Place a number on each edge that represents the distance, time, cost or any type of appropriate weight. Make a table with a row for each node. The number of columns depends on the map. Each column represents an iteration of the algorithm. In each column we will put the distance we think it is to the starting place during that iteration. We will put a * on the entry when we are certain that the distance is correct. Here is a simple example, and its table. In the steps below, we describe how to fill out the table using Dijkstra's algorithm.

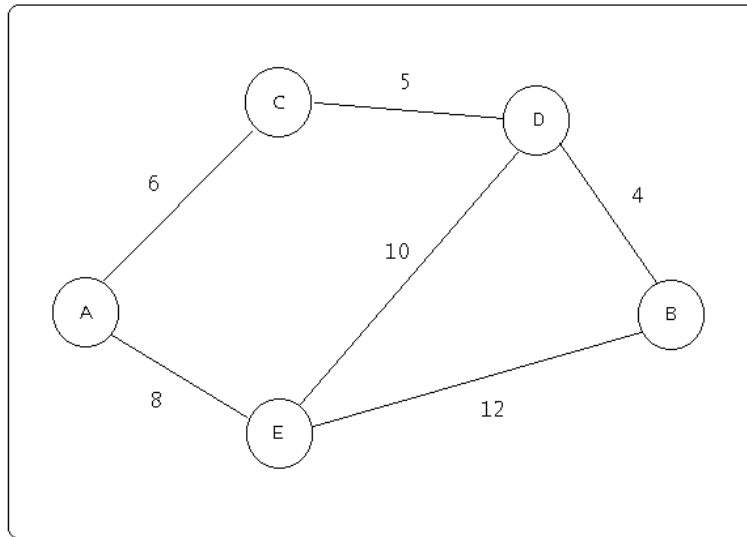


Figure 1: A simple map.

A finished table for the simple map looks like this (the actual distances from A are in the far right column):

NODES	Table of Values				
A					0*
C	6*				6*
E	8	8*			8*
D	Unknown	11	11*		11*
B	Unknown	Unknown	20	15*	15*

How to fill out the table:

1.) Choose a starting place, and find all distances to adjacent nodes, (adjacent nodes are those connected by roads to the starting place). Put these distances in the appropriate box of the table. If a node has not been reached, then it is labeled “unknown,” or left blank. For example, in the map in Figure 1, the starting place is A and the adjacent nodes are C and E.

2.) Put a star on the lowest number in the column. A node with a star in its row is called settled; the final and shortest value for that node’s distance from the beginning is the starred value. The starting point itself is always settled at a distance 0. For example, in the map in Figure 1, we settle C at a distance 6 because any other route to C would go through E, and going to E (a distance 8 away) is already farther than the direct route to C.

3.) Now fill out the next column. Start by finding all nodes that are adjacent to the settled nodes in your current column. In the example, in the first column, A and C are settled, so the adjacent nodes are D (adjacent to C) and E (adjacent to A). Calculate the distance from these nodes to the starting place through the settled node(s) to which they are adjacent, and place the values in the appropriate box of the table. For example, the distance from A to C is 6 and from C to D is 5, so the value 11 appears in for D in the second column. If an unsettled node is adjacent to more than one settled node, the least distance gets put into the table.

4.) Repeat Step 3 and 4 until all nodes have been settled.

In the completed table, notice that node B has a value of 20 in the third column of values, but it is changed to 15 in the next box. This is because the value of 15 was found after the value of 20, and since 15 is less than 20, it took 20’s place in the table. This table shows all of the shortest values from point A. From Points A to point B, the shortest distance is 15.

Here is an example using a map of the Penn State campus. The weights on the edges of the graph in this example are distances estimated using the scale on the map. An activity for the classroom using this algorithm could include choosing a map, choosing which nodes and edges (roads) to include and what weights to put on the edges, as well as using the algorithm to find the shortest path. The web page in the references includes a java applet that implements this algorithm. A very advanced student might be asked to write such a program (maybe for a science fair project?).

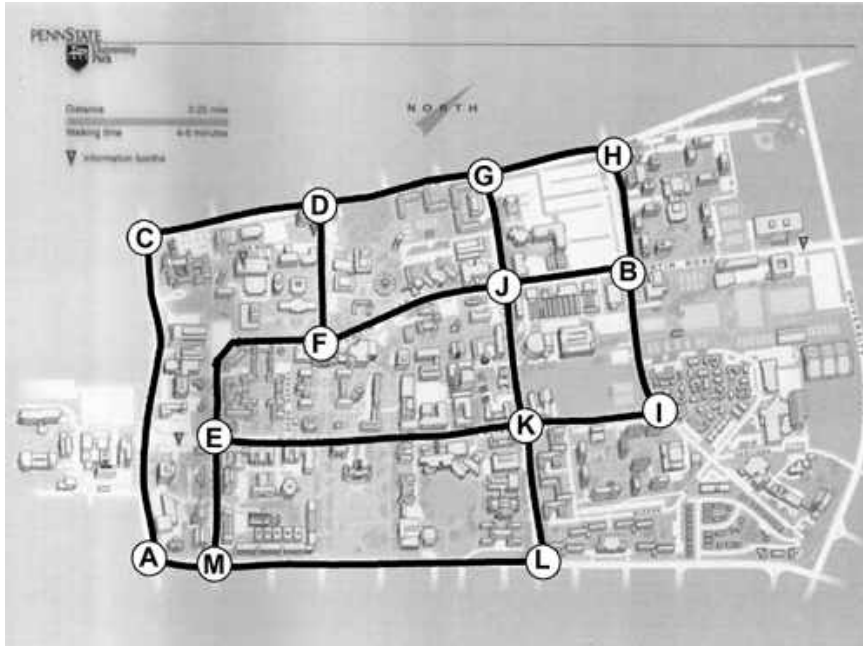


Figure 2: A map of Penn State Campus

The table for Figure 2 looks like this.

A	0*													.00*
B									1.13	1.13	1.13	1.13	1.13*	1.13*
C	.56	.56	.56	.56*										.56*
D				.85	.81	.81	.81	.81*						.81*
E		.31	.31	.31*										.31*
F				.60	.60*									.60*
G								1.09	1.09	1.09	1.09*			1.09*
H											1.30	1.30	1.30	1.30*
I							1.01	1.01	1.01	1.01*				1.01*
J					.92	.92	.92	.92	.92*					.92*
K			.8	.8	.8	.8	.8*							.80*
L		.63	.63	.63	.63	.63*								.63*
M	.10	.10*												.10*

From this we see that the fastest route from A to B is 1.13 miles (the labels in this example are distances in miles). The actual route would be A – M – E – F – J – B.

References

Dijkstra's Algorithm, by Simon Street,

<http://www.deakin.edu.au/~agoodman/graph/dijkstra1.htm>

This site contains a java applet demonstrating this algorithm.

Algorithms and Data Structures: An Approach in C, Charles Bowman, Oxford University Press, 1994.