Dijkstra's algorithm

By Juan Baltazar Tapia

Introduction:

I've always wondered how google maps computed the shortest path between my location and the destination I put. What type of algorithm or math did google maps use to do this? It turns out, this can be solved using weighted graphs. Every location can be represented with a vertex, and each path can be represented with an edge. Each edge will have a "weight", the distance of the edge itself. This brings up the next question.

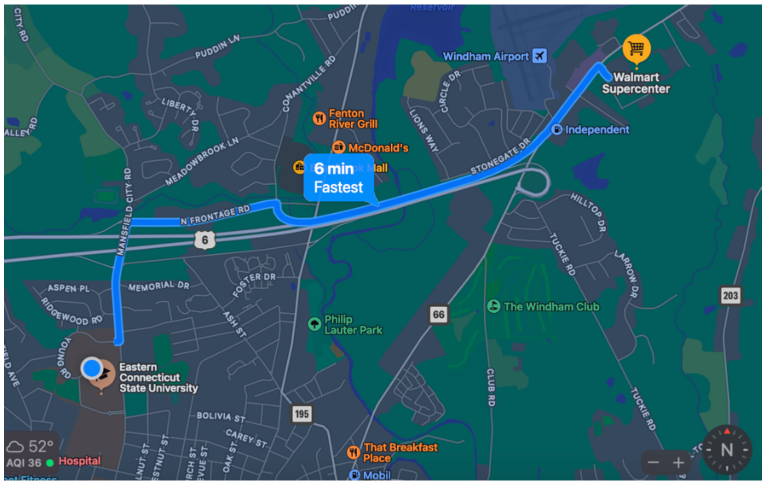


Figure 1 Example of the shortest path to Walmart from ECSU

What algorithm is used to compute the shortest algorithm?

The algorithm used is known as Dijkstra's algorithm named after Dr. Edsger W. Dijkstra, a computer scientist, and software engineer. Fun fact, he claims to have come up with it in 20 minutes after sitting down on a café terrace with a cup of coffee (Frana). Not only is the shortest path from the source vertex (the vertex you want to start at) to the destination calculated, but the shortest path from the source vertex to every other vertex is also calculated as well.

One case study used Dijkstra's algorithm to find a specialist doctor in Bandar Lampung (Gunawan). In this city, there were 19 hospitals with 229 specialist doctors. Finding the shortest path between a health service and a specialist doctor is important as the faster this process is, the better a patient's safety and health (Gunawan).

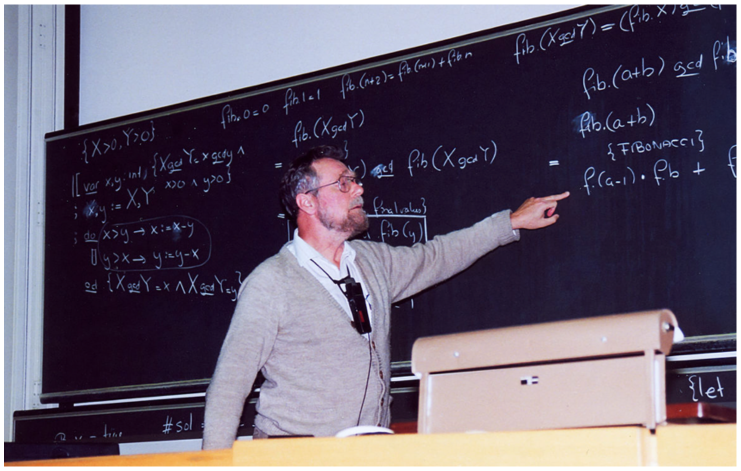


Figure 2 Dr. Edsger Dijkstra at ETH Zurich in 1994 (image by Andreas F. Borchert)(Navone)

So how do you compute this shortest path?

The graph down below, Figure 3, will be used as an example. Dijkstra's algorithm creates a table with three columns. The first column contains each vertex in the graph. The second column contains the shortest distance, or cost from the "starting" vertex to the current vertex, in our case A. The last column contains the vertex before it which will be used to calculate the shortest path.

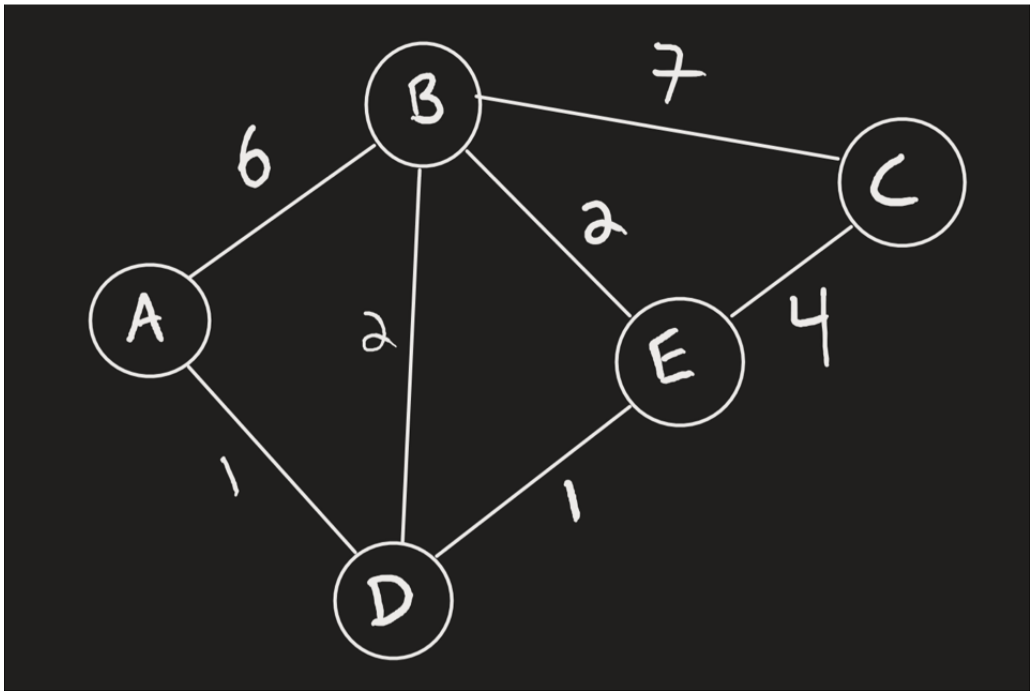
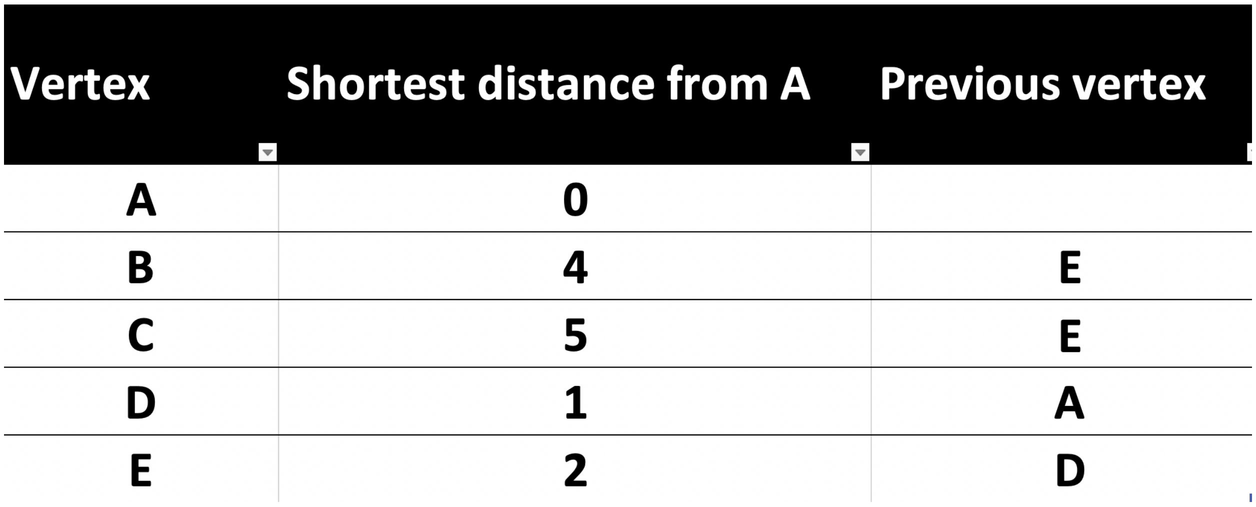


Figure 3 Example Graph

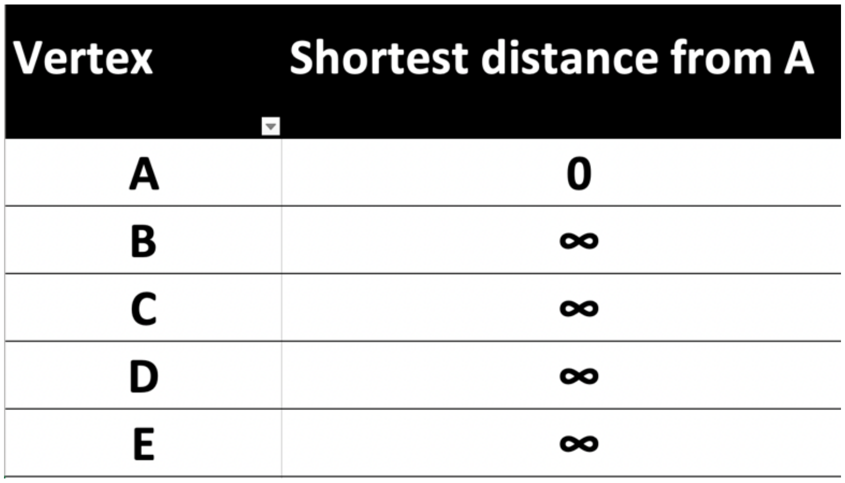


Vertex	Shortest distance from A	Previous vertex
A	0	
B	4	E
C	5	E
D	1	A
E	2	D

Figure 4 Table showing the final calculation

In-Depth:
To begin, there are two lists you use. One to keep track of visited vertices, and one to keep track of unvisited vertices.
Visited = [ ]   Unvisited = [A, B,C , D, E]

The distance from A from itself is 0, and since we don't know the distances from the rest of the vertices to A, we set them to infinity.



Vertex	Shortest distance from A
A	0
B	∞
C	∞
D	∞
E	∞

Figure 5 Table showing the first two columns

Our current vertex is A.

Step One: For the current vertex, calculate the distance of each adjacent vertex from the starting vertex that is in the unvisited list.

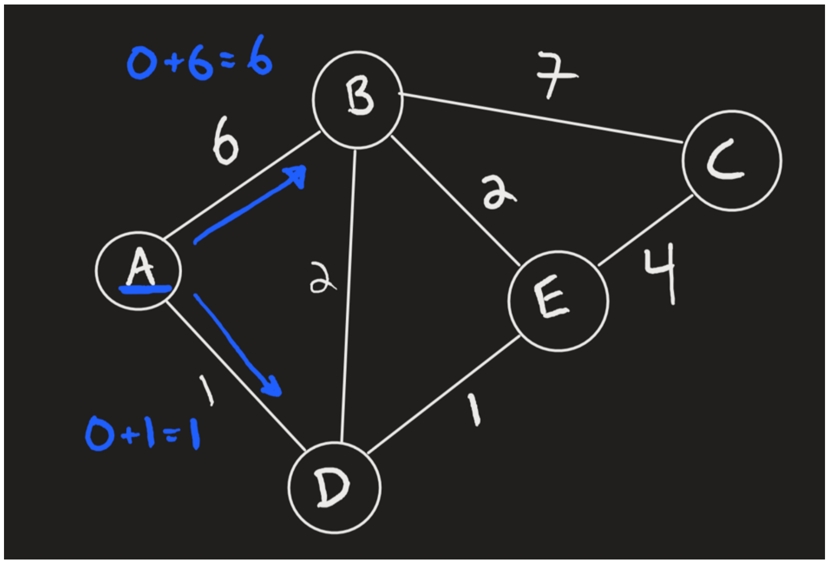


Figure 6 In our case, we are calculating the distance of B from A, and D from A

Step Two: If the calculated distance of the vertex is less than the know distance, we update the shortest distance in the table. We also update the previous vertex for each updated distance.

Vertex	Shortest distance from A	Previous vertex
A	0	
B	6	A
C	∞	
D	1	A
E	∞	

Figure 7 Table showing updated distance and previous vertex

Step Three: Add the current vertex to the list of visited vertices and repeat. Vertex A will not be visited anymore
Visited = [A]

Step Four: Visit the vertex with the smallest known distance from the starting vertex. This will be our current vertex. As seen in Figure 6, our current vertex is **D**

Next, we will repeat **step one**. Calculate the distance of each adjacent vertex from the starting vertex that is in the unvisited list.

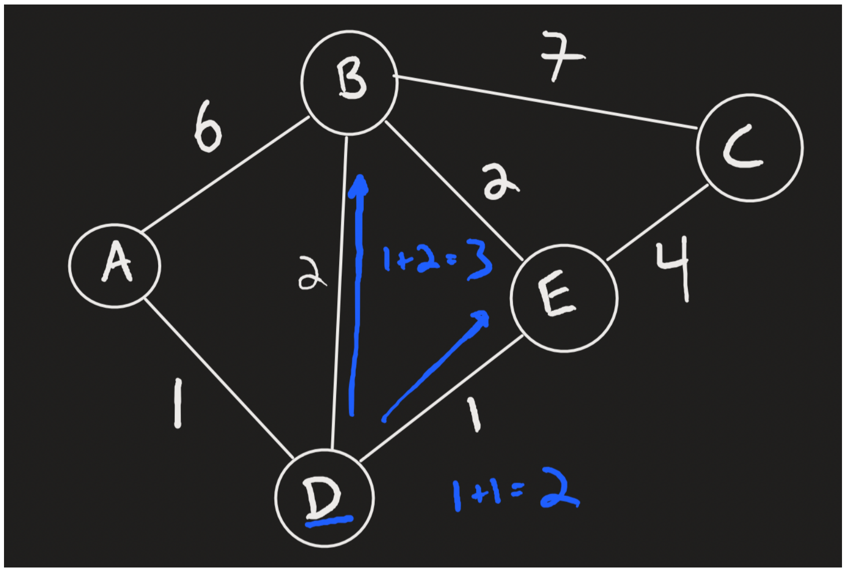
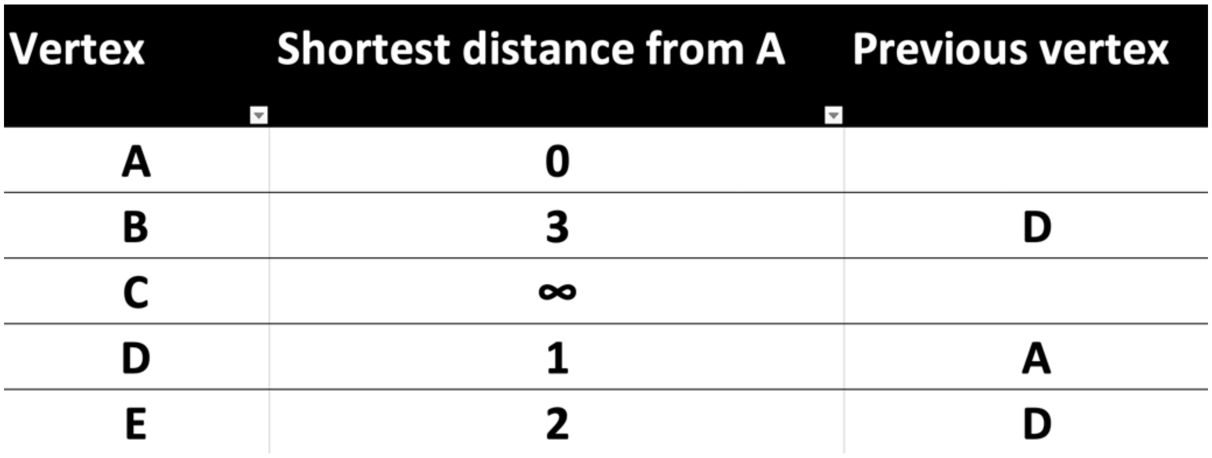


Figure 8 Showing Calculations

Then we will repeat **step two.** If the calculated distance of the vertex is less than the know distance, we update the shortest distance in the table. We also update the previous vertex for each updated distance.



Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	∞	
D	1	A
E	2	D

Figure 9 Table showing updated calculations

Finally, **step three,** add the current vertex to the list of visited vertices and repeat. Vertex A and B will not be visited anymore.

Visited = [A, B]

These three steps will be repeated until there are no more entries in the unvisited list.

Here is the rest of the calculations in images only:

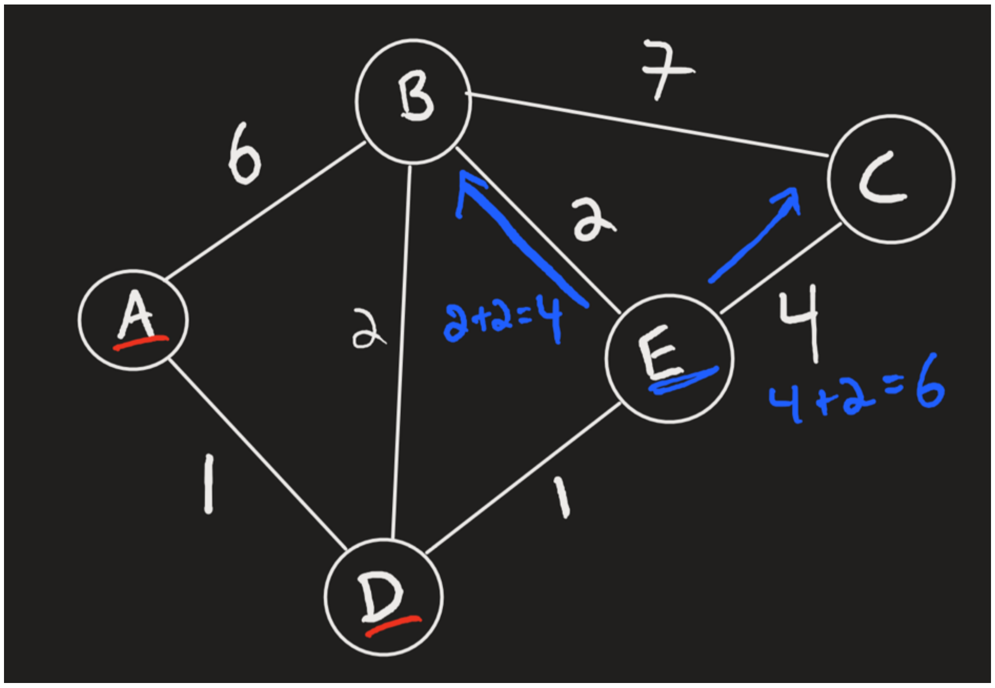


Figure 10 E is our current vertex

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	6	E
D	1	A
E	2	D

Figure 11 Update shortest distance

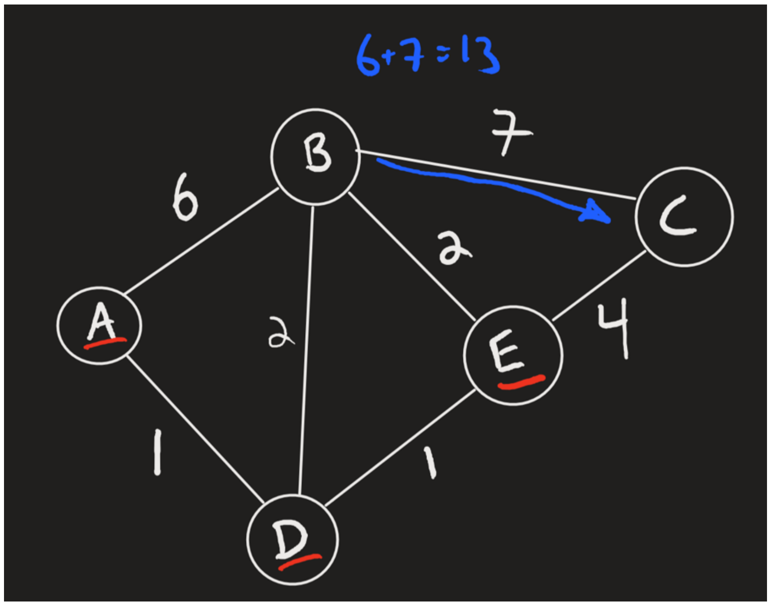


Figure 12 B is the current vertex

Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	6	E
D	1	A
E	2	D

Figure 13 There were no update values this time

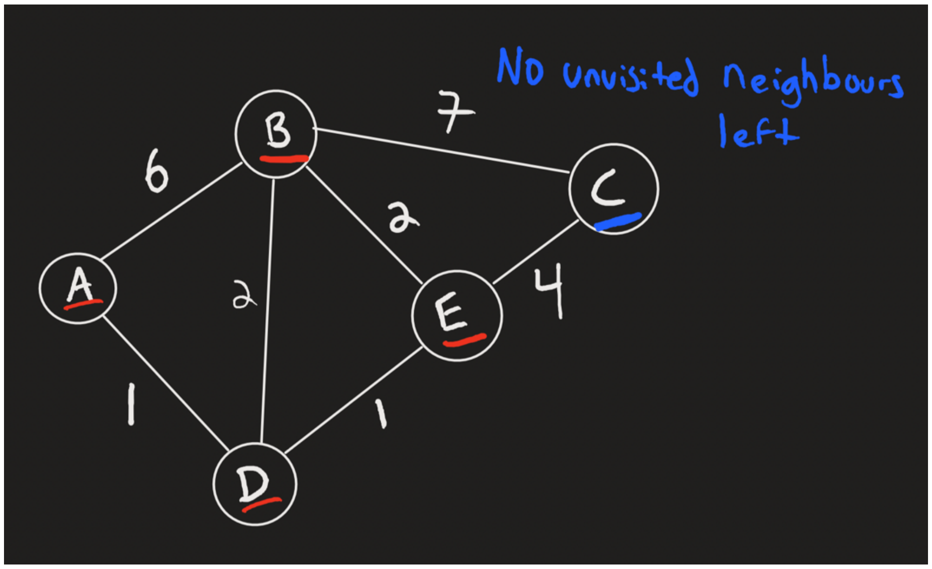


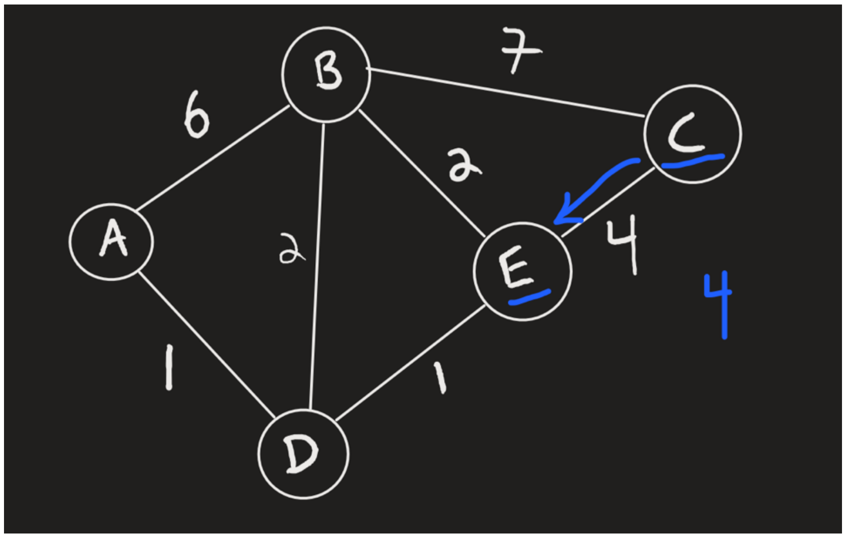
Figure 14 C has no unvisited neighbors, the algorithm ends

Now imagine we want to know the shortest distance and path from A to C.

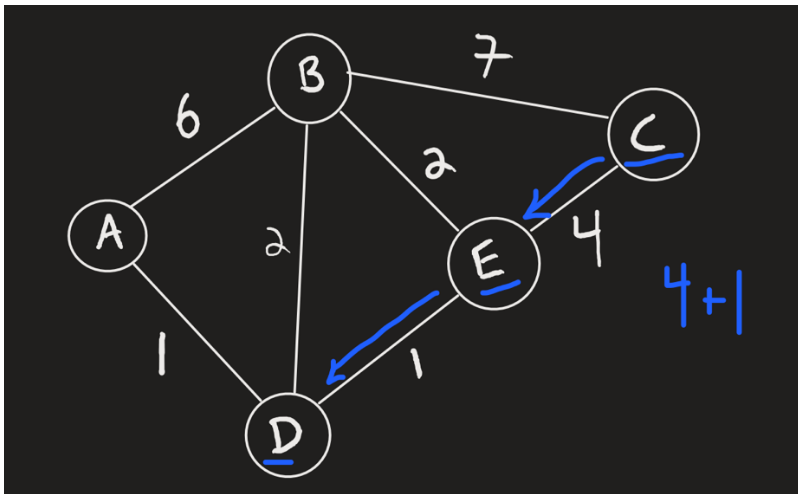
Vertex	Shortest distance from A	Previous vertex
A	0	
B	3	D
C	6	E
D	1	A
E	2	D

Figure 15 The final table

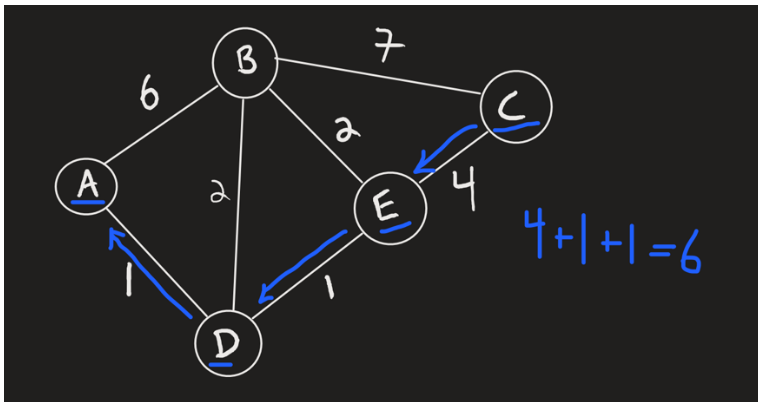
Looking at row C, we conclude that the shortest distance from A is 6. To find the path, we use the previous vertex column. To get to C, the previous vertex is E



Looking at row E, we see the previous vertex is D



Looking at row D, we see the previous vertex is A



We have arrived at the source vertex.
Adding up all the weights, we can re-affirm that the shortest distance is 6.

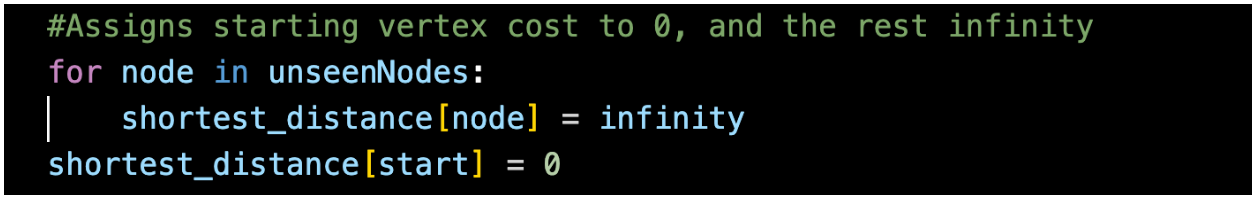
The first step in implementing Dijkstra's algorithm:
Dijkstra's algorithm can be summarized in pseudocode. Now, what is pseudocode? In computer science, pseudocode refers to a plain language description of the logic you want to implement. As seen above Dijkstra's algorithm can be written in pseudocode as follows:

```
Let the distance of the start vertex = 0
Let the distance of all other vertices = ∞

While the visited list still has vertices:
        Visit the unvisited vertex with the smallest know distance from the start vertex
        For the current vertex, examine its unvisited neighbors
        For the current vertex, calculate the distance of each adjacent vertex from the start
        vertex
        If the calculated distance of a vertex is less than the know distance, update the shortest
        distance, and update the previous vertex
        Add the current vertex to the list of visited vertices
Repeat until the visited list has no vertices left
```

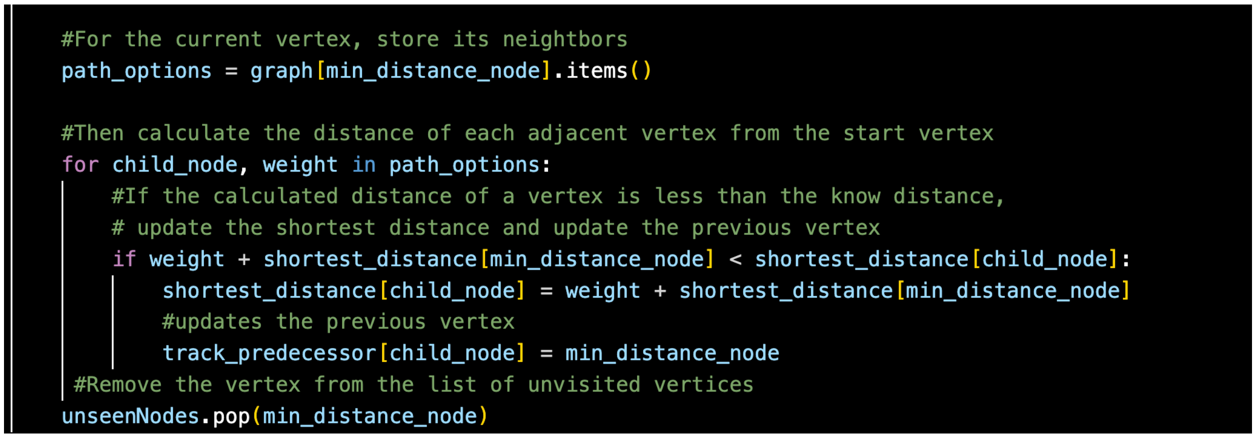
Dijkstra's algorithm in Python3:
I will give a brief overview of the implementation of Dijkstra's Algorithm in Python3. The full code will be attached alongside the blog post.

Here I assign the starting vertex distance to 0 and the rest to infinity



```
#Assigns starting vertex cost to 0, and the rest infinity
for node in unseenNodes:
    shortest_distance[node] = infinity
shortest_distance[start] = 0
```

I set the current vertex to the starting vertex then start looking at its neighbors and calculating costs. In the green comments, the pseudocode from the previous section demonstrates what code is doing which step.



```
#For the current vertex, store its neighbors
path_options = graph[min_distance_node].items()

#Then calculate the distance of each adjacent vertex from the start vertex
for child_node, weight in path_options:
    #If the calculated distance of a vertex is less than the know distance,
    # update the shortest distance and update the previous vertex
    if weight + shortest_distance[min_distance_node] < shortest_distance[child_node]:
        shortest_distance[child_node] = weight + shortest_distance[min_distance_node]
        #updates the previous vertex
        track_predecessor[child_node] = min_distance_node
 #Remove the vertex from the list of unvisited vertices
unseenNodes.pop(min_distance_node)
```

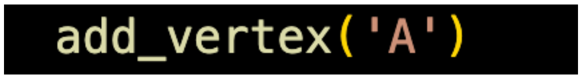
Figure 16

The code in Figure 16 will repeat until every vertex has been visited. After it is completed, the shortest distance value will be printed along with the path from the starting vertex to the selected destination.

Conclusion:

In conclusion, Dijkstra's algorithm is very important in calculating the shortest path between a source vertex and any other vertex in a graph. It is already used by google maps (Mitchell) and has applications for school bus routes, public transportation, and production in factories.

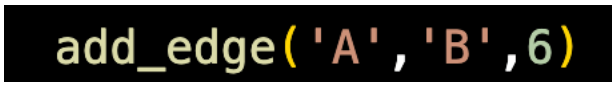
I have attached a code.txt file in which you can copy and paste the code into https://www.online-python.com/ to play around with your own graphs. At the bottom of the code, you can add vertices with this command.



```
add_vertex('A')
```

Figure 17 Make sure to add quotes around the vertex name

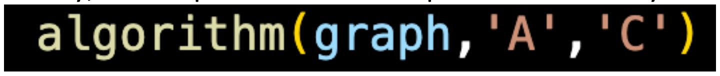
And to add edges, you add the two vertices the edge connects and the weight/cost using this command



```
add_edge('A','B',6)
```

Figure 18 Vertex, vertex, weight

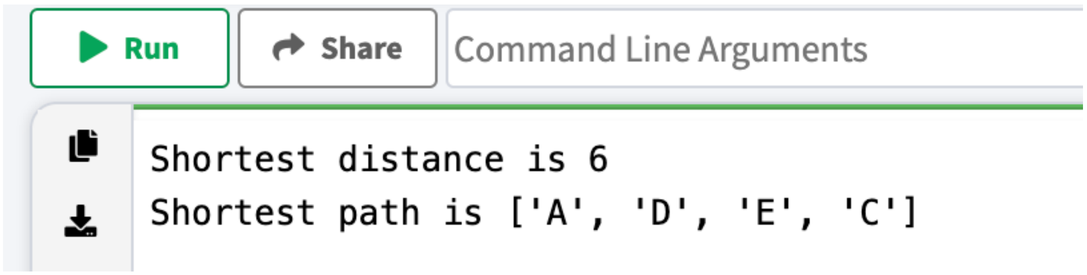
Finally, to compute the shortest path between any two vertices, use the following command



```
algorithm(graph,'A','C')
```

Figure 19 graph, source vertex, goal vertex

Then hit the play button and the results will print!



```
Shortest distance is 6
Shortest path is ['A', 'D', 'E', 'C']
```

Citations

Frana, Philip L., and Thomas J. Misa. “An Interview with Edsger W. Dijkstra.” *Communications of the ACM*, vol. 53, no. 8, 2010, pp. 41–47., https://doi.org/10.1145/1787234.1787249.

Gunawan, Rakhmat Dedi, et al. “Implementation of Dijkstra's Algorithm in Determining the Shortest Path (Case Study: Specialist Doctor Search in Bandar Lampung).” *IJISCS (International Journal of Information System and Computer Science)*, vol. 3, no. 3, 2019, p. 98., https://doi.org/10.56327/ijiscs.v3i3.768.

Mitchell, Ryan. “The Algorithms behind the Working of Google Maps.” *CodeChef*, 21 Oct. 2021, https://blog.codechef.com/2021/08/30/the-algorithms-behind-the-working-of-google-maps-dijkstras-and-a-star-algorithm/.

Navone, Estefania Cassingena. “Dijkstra's Shortest Path Algorithm - a Detailed and Visual Introduction.” *FreeCodeCamp.org*, FreeCodeCamp.org, 3 Feb. 2022, https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/.