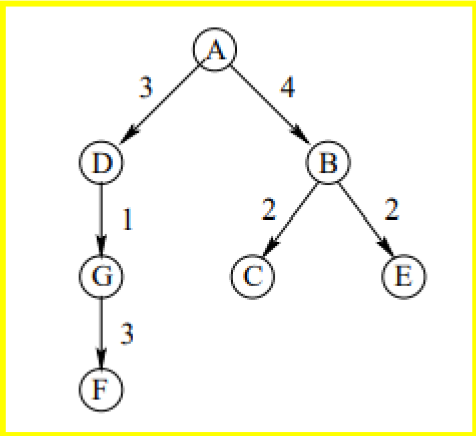
Exercise Sheet: Dijkstra

Data Structures and Algorithms (X_400614)

1. Execute Dijkstra's algorithm on the graph pictured in Figure 1, starting at the vertex A. In case of ties, the vertex with the lower letter is handled first.
   a. List the vertices in the order in which they are deleted from the priority queue and for each the shortest distance from A to the vertex.
      A,0 D,3 B,4 G,4 C,6 E,6 F,7
   b. Draw the shortest path tree that results.



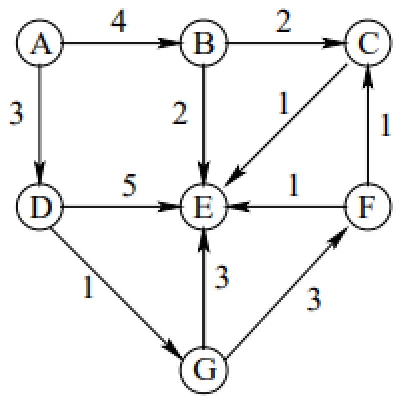


Figure 1.

2. Consider the graph in Figure 2.
   a. Execute Dijkstra's algorithm on this graph, starting from *s*. After each step, note down:
      i. The upper bounds *d[u]* (which denote what is currently believed to be the shortest distance to each node), for $u \in V$, between *s* and each node *u* computed so far,
      ii. the set *M* of all nodes for which the minimal distance has been correctly computed so far,
      iii. and the predecessor *p(u)* for each node in *M*.
      When we choose s: d[s] = 0, d[x] = d[v] = d[w] = d[y] = d[t] = ∞, M = {s}, there is no p(s).
      When we choose x: d[s] = 0, d[x] = 2, d[v] = 20, d[w] = d[y] = d[t] = ∞,

M = {s, x}, there is no p(s), p(x) = s.
When we choose y: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 3, d[w] = d[t] = ∞, M = {s, x, v, y}, there is no p(s), p(x) = s, p(v) = x, p(y) = x.
When we choose t: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 3, d[w] = 20, d[t] = 6, M = {s, x, v, y, t}, there is no p(s), p(x) = s, p(v) = x, p(y) = x, p(t) = y.
When we choose v: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 3, d[w] = 20, d[t] = 6, M = {s, x, v, y, t}, there is no p(s), p(x) = s, p(v) = x, p(y) = x, p(t) = y.
When we choose w: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 3, d[w] = 15, d[t] = 6, M = {s, x, v, y, t, w}, there is no p(s), p(x) = s, p(v) = x, p(y) = x, p(t) = y, p(w) = v.

b. Replace the weight of edge (x,y) with -1 and redo the algorithm. Does the algorithm still compute correctly? If not, where does it break?
The algorithm works correctly.
When we choose s: d[s] = 0, d[x] = d[v] = d[w] = d[y] = d[t] = ∞.
When we choose x: d[s] = 0, d[x] = 2, d[v] = 20, d[w] = d[y] = d[t] = ∞.
When we choose y: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 1, d[w] = d[t] = ∞.
When we choose t: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 1, d[t] = 4, d[w] = 18.
When we choose v: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 1, d[t] = 4, d[w] = 18.
When we choose w: d[s] = 0, d[x] = 2, d[v] = 10, d[y] = 1, d[t] = 4, d[w] = 15.

c. Now, additionally change the weight of edge (v,y) to -10. Show that in this case the algorithm doesn't work correctly. That is, show that there is a vertex $u \in V$ such that *d[u]* is not equal to minimum distance from *s* to *u* after the execution of the algorithm.
The algorithm doesn't work correctly, for example, the distance from s to y is 0, but the algorithm computes exactly the same values of d[·] as in part b), so d[y] = 1.

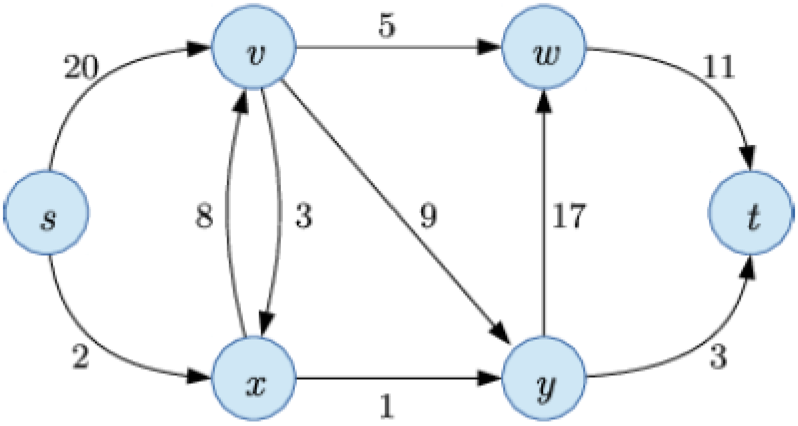


Figure 2.

3. A robot moves in the environment in Figure 3. It starts from the node labeled *Start* and needs to reach the node labeled *End*. The environment is continuous and the scale is supplied on the figure.

a. Considering the robot as a point, what is the shortest path from *Start* to *End?* Compute the Euclidean distance between the nodes. Draw the graph and apply Dijkstra's algorithm to find the shortest path from the node *Start* to the node *End*.

Initially only Start has distance 0. In the first step we compute the distance from Start to 1, which is $\sqrt{5}$, and from Start to 2, which is $\sqrt{29}$. These are set as the distances for 1 and 2.

In the next step we choose 1 and see that the distance from 1 to 3 is $\sqrt{26}$. Thus, the distance from Start to 3 is set to $\sqrt{5} + \sqrt{26}$.

The next step will consider 2, and set the distance from 2 to end to $\sqrt{29} + \sqrt{17}$. The distance from Start to 3 is unchanged as Start - 1 - 3 is shorter than Start - 2 - 3.

Finally, considering 3 doesn't change any distances as $\sqrt{29} + \sqrt{17} < 2 * \sqrt{5} + \sqrt{26}$.

Shortest path	Start	1	2	3	End
Init	0	inf	inf	inf	inf
Start	–	sqrt 5	sqrt 29	inf	inf
1	–	–	sqrt 29	sqrt 5+ sqrt 26	inf
2	–	–	–	sqrt 5+ sqrt 26	sqrt 29 + sqrt 17
3	–	–	–	–	sqrt 29 + sqrt 17

The shortest path is Start-2-End.

b. Can you find the same path faster using another algorithm? If so, which algorithm is better suited for this problem, and why?

Yes, A* is better. It will start exploring in the right direction, whereas Dijkstra is not aware of the directionality of the shortest path.

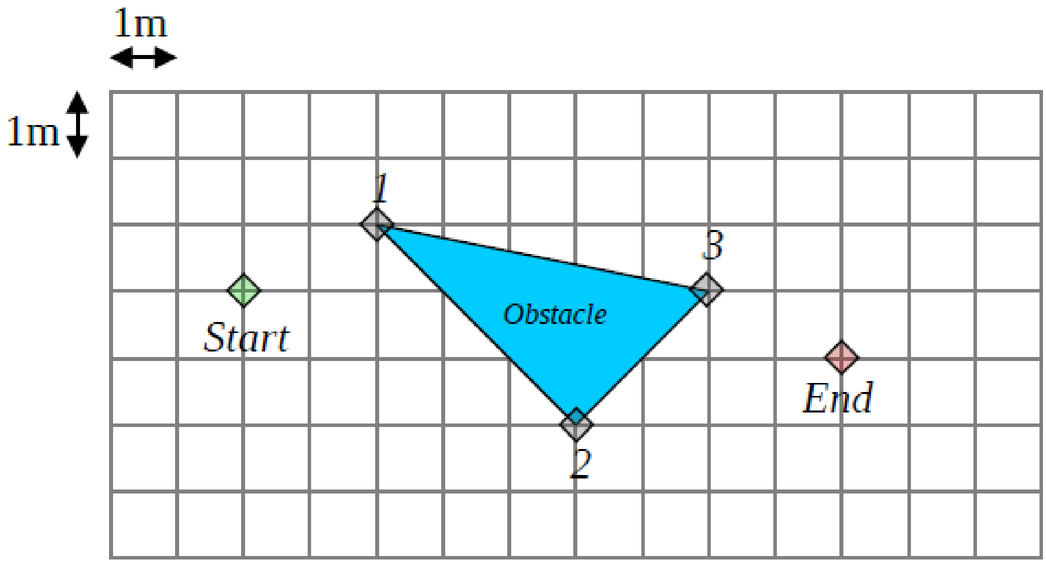


Figure 3.