Dijkstra algorithm

Simeon Monov

Dijkstra’s algorithm

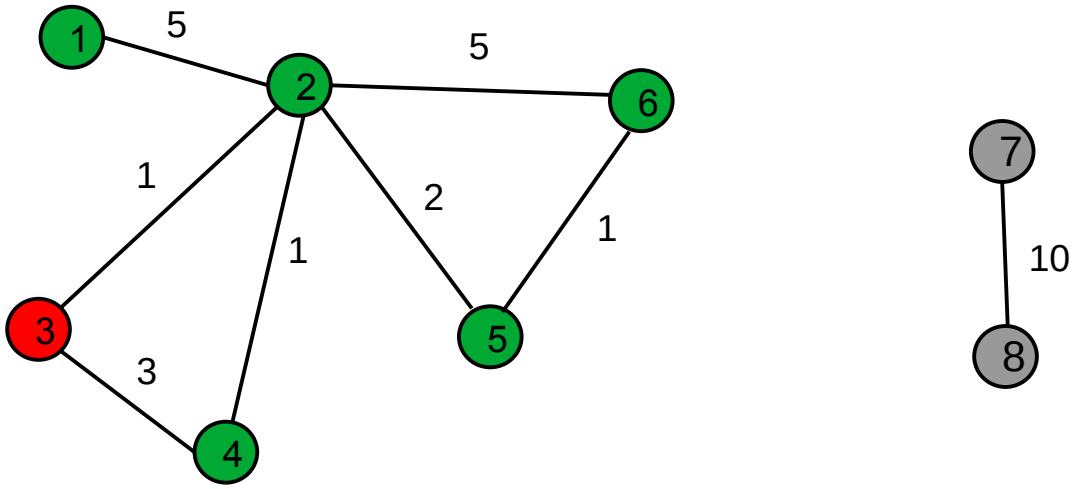
- Finds minimal path between vertex **a** and all other vertices in the graph
- Works on undirected / directed, unweighted and weighted with positive weights graphs

Algorithm:

- Matrix A is adjacency matrix of the graph
- Array D will contain min distances from start node (**s**) to all other nodes (∞ means there is no path found so far)
- Set T will contain unvisited nodes
- Initialize D with direct distances from **s** to all other nodes. If no direct path (edge connection) exists, set it to ∞. Distance from **s** to **s** is 0
- Initialize T with all nodes except start node **s**
- While T contains at least one node i, for which d[i] != ∞, repeat:
  - Choose **j** from T, such as d[j] is minimal
  - Exclude **j** from T
  - For each **i** in T: d[i] = min(d[i], d[j] + A[j,i])

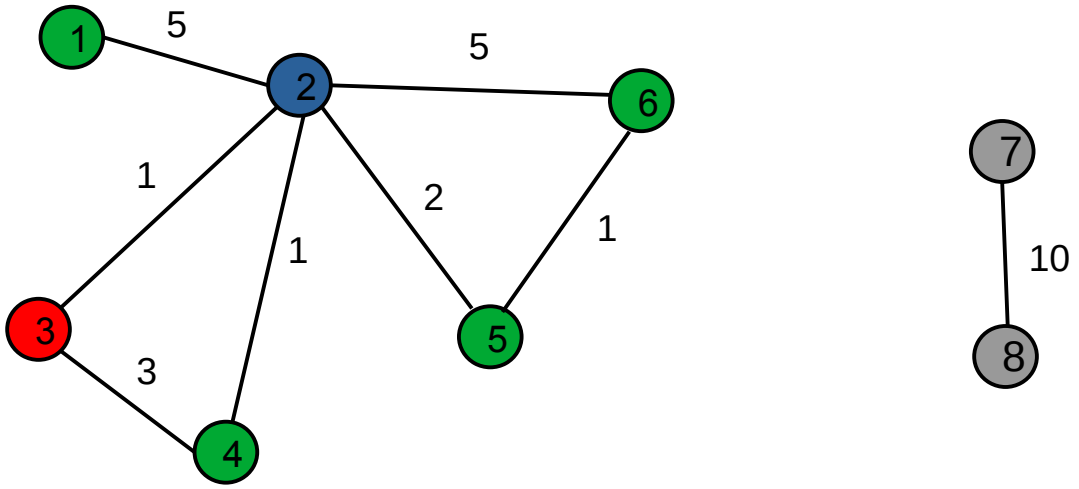
Dijkstra’s algorithm

T = { 1, 2, 4, 5, 6, 7, 8 }
D = [∞, 1, 0, 3, ∞, ∞, ∞, ∞]



Dijkstra’s algorithm

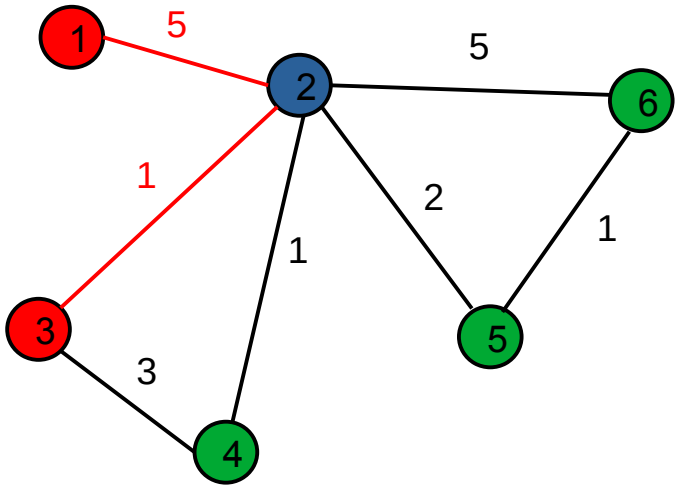
T = { 1, ~~2,~~ 4, 5, 6, 7, 8 }
D = [∞, 1, 0, 3, ∞, ∞, ∞, ∞]
Shortest distance not visited = 2 (distance 1)

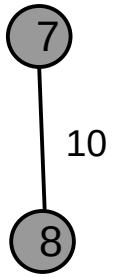


Dijkstra’s algorithm

T = { **1**, 4, 5, 6, 7, 8 }
D = [**6**, 1, 0, 3, ∞, ∞, ∞, ∞]
Shortest distance not visited = 2 (distance 1)

Checking **1**:
1 + 5 = **6** < ∞

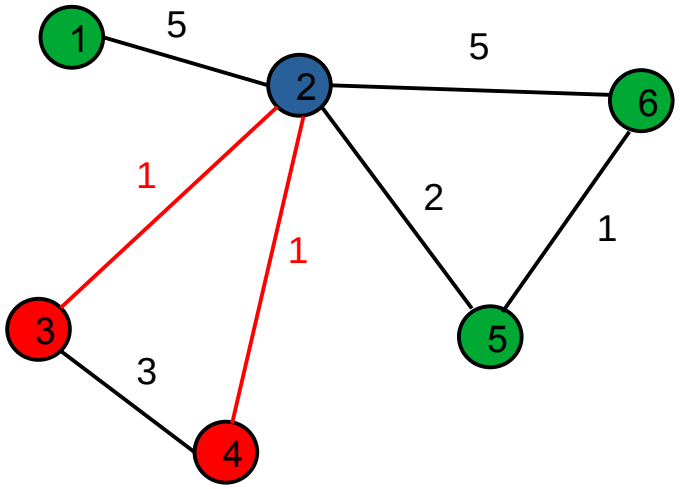


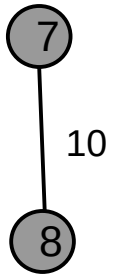


Dijkstra’s algorithm

T = { 1, **4**, 5, 6, 7, 8 }
D = [**6**, 1, 0, **2**, ∞, ∞, ∞, ∞]
Shortest distance not visited = 2 (distance 1)

Checking **4**:
1 + 1 = **2** < **3**

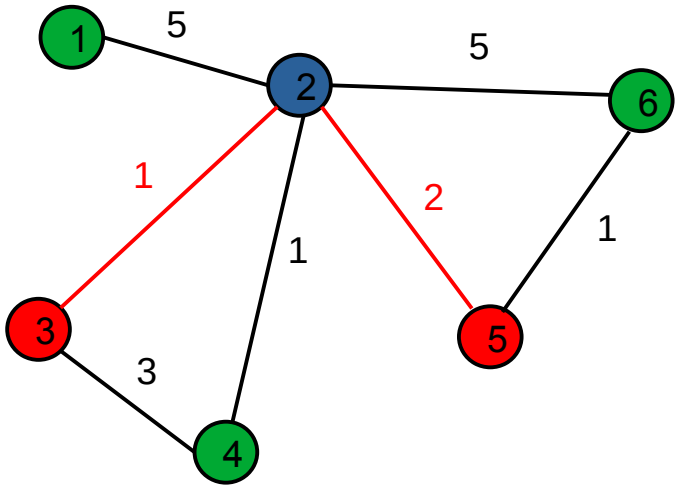


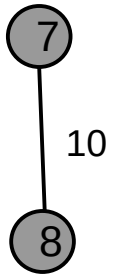


Dijkstra’s algorithm

T = { 1, 4, **5**, 6, 7, 8 }
D = [**6**, 1, 0, **2**, **3**, ∞, ∞, ∞]
Shortest distance not visited = 2 (distance 1)

Checking **5**:
1 + 2 = **3** < **∞**

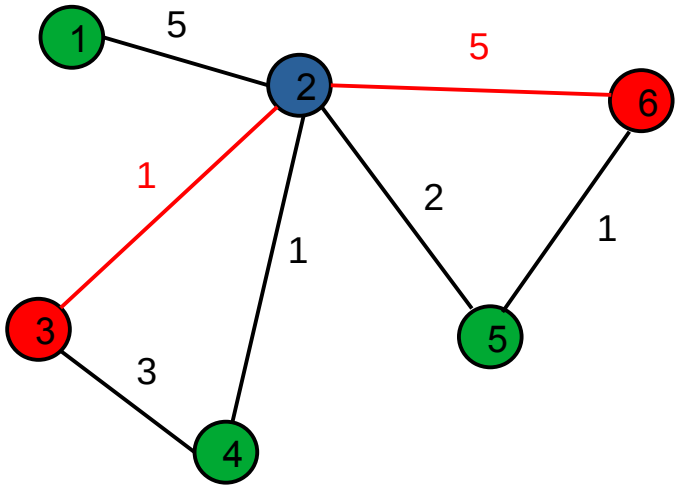


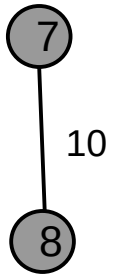


Dijkstra’s algorithm

T = { 1, 4, 5, **6**, 7, 8 }
D = [**6**, 1, 0, **2**, **3**, **6**, ∞, ∞]
Shortest distance not visited = 2 (distance 1)

Checking **6**:
1 + 5 = **6** < ∞

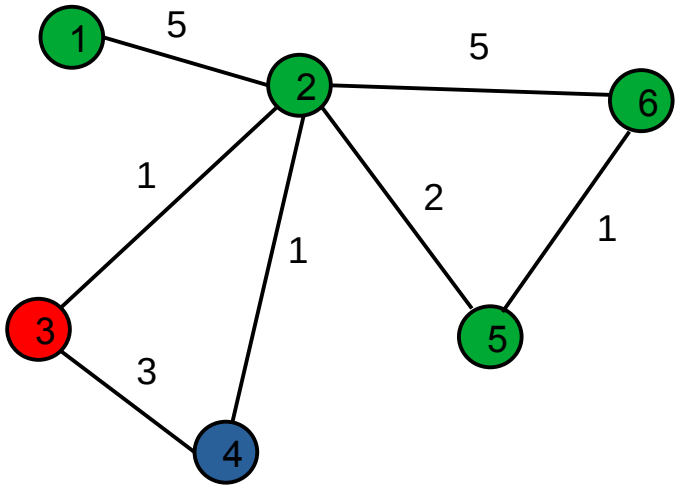




Dijkstra’s algorithm

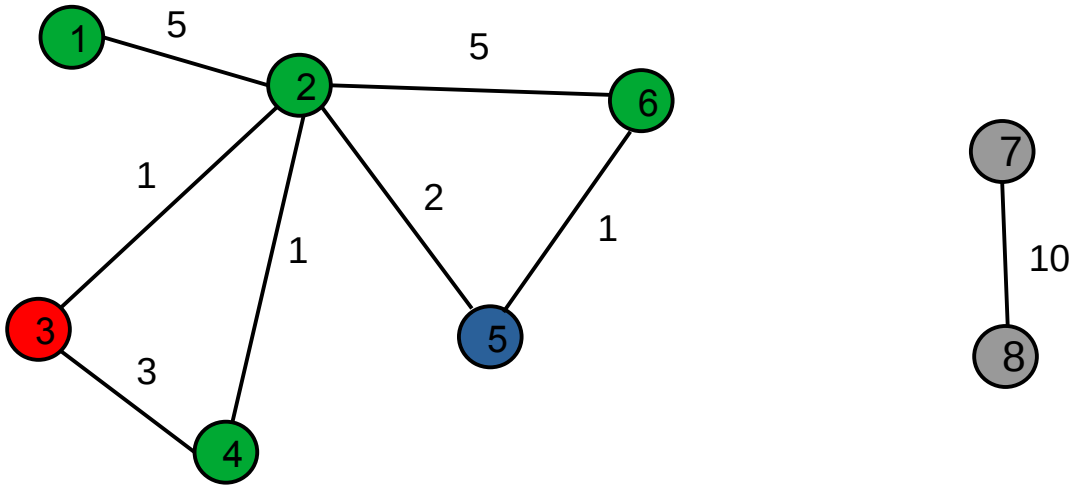
T = { 1, ~~4,~~ 5, 6, 7, 8 }
D = [6, 1, 0, 2, 3, 6, ∞, ∞]
Shortest distance not visited = 4 (distance 2)

Nothing to check as 4 does not connect with any non visited nodes



Dijkstra’s algorithm

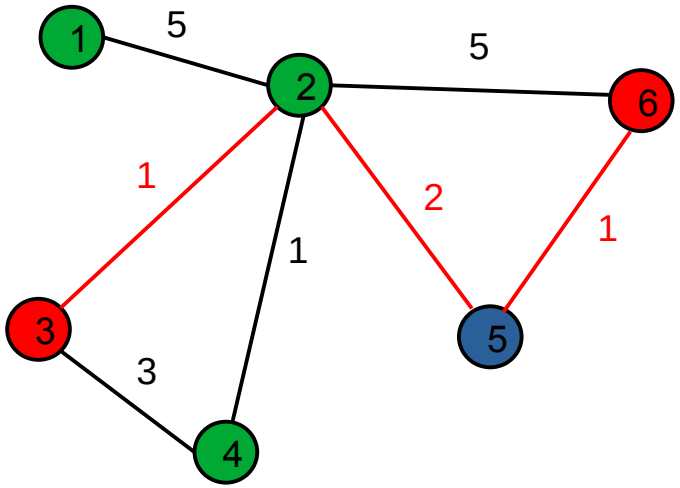
T = { 1, ~~5,~~ 6, 7, 8 }
D = [6, 1, 0, 2, 3, 6, ∞, ∞]
Shortest distance not visited = 5 (distance 3)

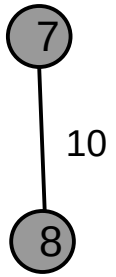


Dijkstra’s algorithm

T = { 1, **6**, 7, 8 }
D = [6, 1, 0, 2, 3, **4**, ∞, ∞]
Shortest distance not visited = 5 (distance 3)

Checking **6**:
3 + 1 = **4** < **6**

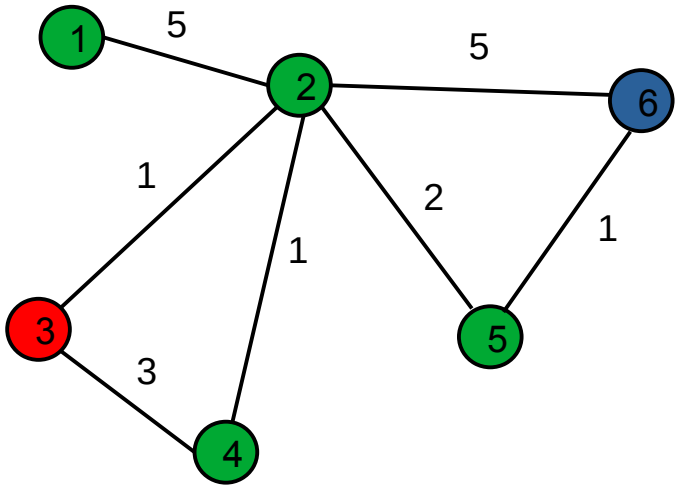




Dijkstra’s algorithm

T = { 1, ~~6~~, 7, 8 }
D = [6, 1, 0, 2, 3, 4, ∞, ∞]
Shortest distance not visited = 6 (distance 4)

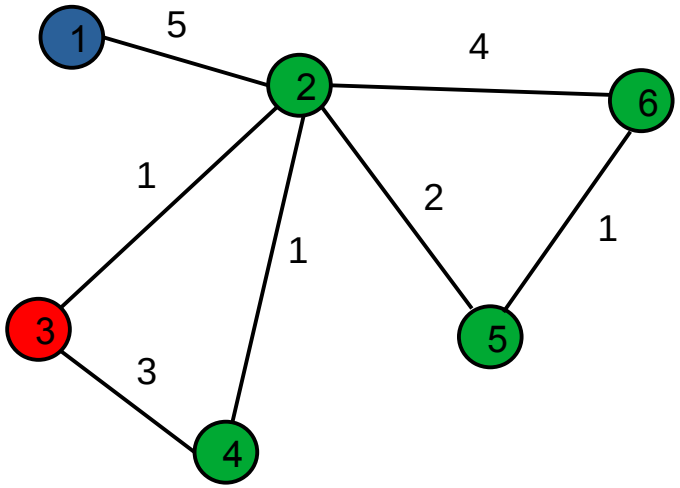
Nothing to check as 6 does not connect with any non visited nodes

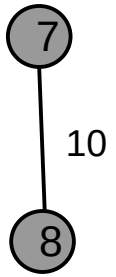


Dijkstra’s algorithm

T = { ~~1~~, 7, 8 }
D = [6, 1, 0, 2, 3, 4, ∞, ∞]
Shortest distance not visited = 1 (distance 6)

Nothing to check as 1 does not connect with any non visited nodes





Dijkstra’s algorithm

Minimal paths are :
D = [6, 1, 0, 2, 3, 4, ∞, ∞]
3 → 1 = 6
3 → 2 = 1
3 → 3 = 0
3 → 4 = 2
3 → 5 = 3
3 → 6 = 4