CX2101 Algorithm Design and Analysis

Tutorial 2 (Graphs)

Week 6: Q1-Q3

This Tutorial

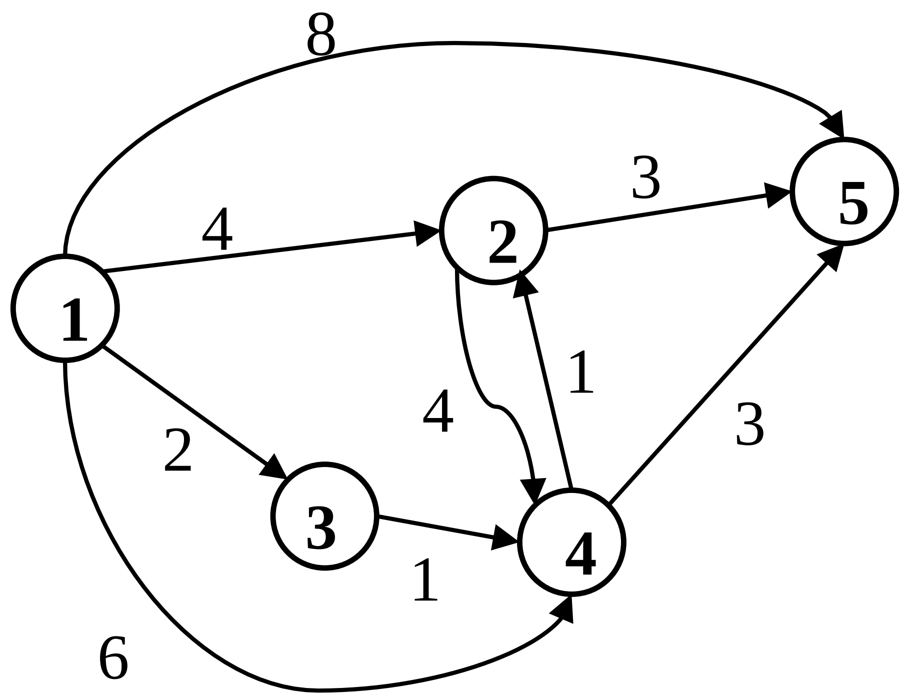
- Single-source shortest path algorithm

Question 1

Apply the Dijkstra's algorithm on the graph represented by the following adjacency matrix to find the shortest distances and the shortest paths from vertex 1 to the other vertices. Show the contents of arrays S, d and pi after each iteration of the while loop.

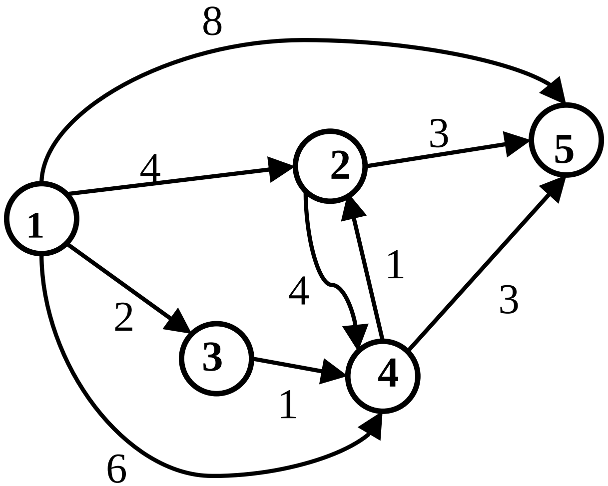
$$
\begin{array}{cc}
\text{vertex} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 4 & 2 & 6 & 8 \\
\infty & 0 & \infty & 4 & 3 \\
\infty & \infty & 0 & 1 & \infty \\
\infty & 1 & \infty & 0 & 3 \\
\infty & \infty & \infty & \infty & 0
\end{bmatrix}
\end{array}
$$

$$
\begin{array}{cc}
\text{vertex} & \begin{matrix} \;1 & \;\;2 & \;\;3 & \;\;4 & \;\;5 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} &
\begin{bmatrix}
0 & 4 & 2 & 6 & 8 \\
\infty & 0 & \infty & 4 & 3 \\
\infty & \infty & 0 & 1 & \infty \\
\infty & 1 & \infty & 0 & 3 \\
\infty & \infty & \infty & \infty & 0
\end{bmatrix}
\end{array}
$$



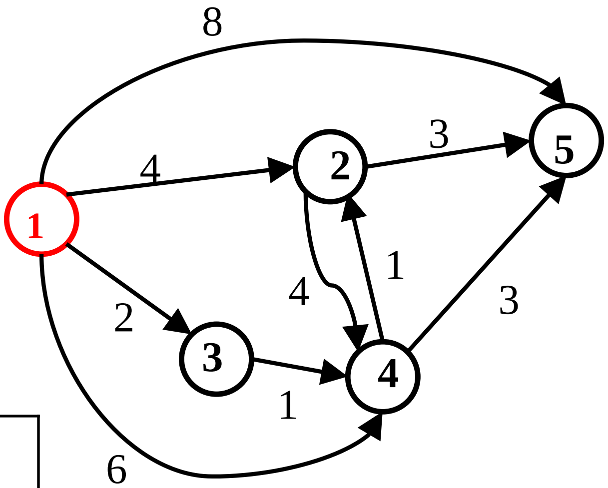
After initialization:

	1	2	3	4	5
S	**0**	**0**	**0**	**0**	**0**
d	**0**	**∞**	**∞**	**∞**	**∞**
pi	**null**	**null**	**null**	**null**	**null**



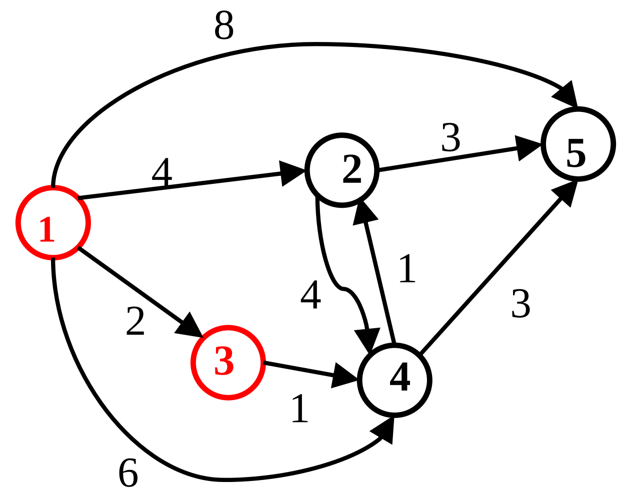
First iteration:

	1	2	3	4	5
S	**1**	**0**	**0**	**0**	**0**
d	**0**	**4**	**2**	**6**	**8**
pi	**null**	**1**	**1**	**1**	**1**



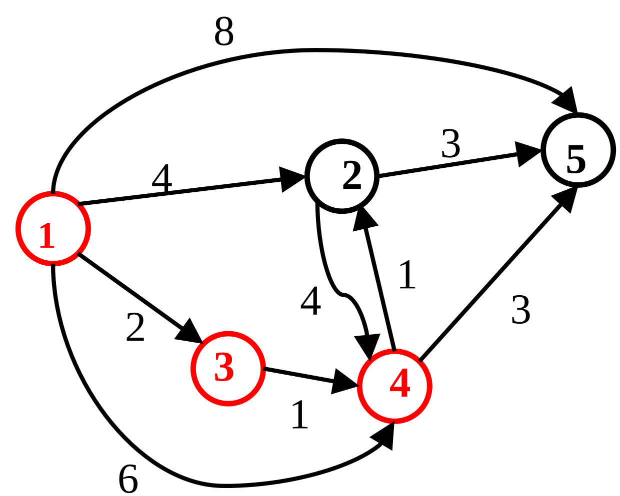
2nd iteration:

	1	2	3	4	5
S	1	0	1	0	0
d	0	4	2	3	8
pi	null	1	1	3	1



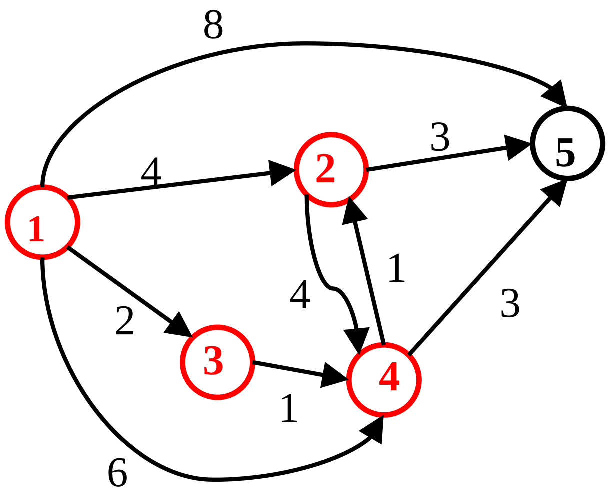
3rd iteration:

	1	2	3	4	5
S	1	0	1	1	0
d	0	4	2	3	6
pi	null	1	1	3	4



4th iteration:

	1	2	3	4	5
S	1	1	1	1	0
d	0	4	2	3	6
pi	null	1	1	3	4



5th iteration:

	1	2	3	4	5
S	1	1	1	1	1
d	0	4	2	3	6
pi	null	1	1	3	4

Shortest paths:

1 – 2

1 – 3

1 – 3 – 4

1 – 3 – 4 – 5

Question 2

- Let $G = (V, E, W)$ be a weighted graph, and let s and z be distinct vertices. In the graph, there may be more than one shortest path from s to z. Explain how to modify Dijkstra's shortest-path algorithm to determine the number of distinct shortest paths from s to z. Assume all edge weights are positive.

Question 2

```
shortest_paths( Graph g, Node source )
// The array count[] records the number of
// shortest paths from
// source to each of the vertices
{
     for each vertex v {
            d[v] = infinity;
            pi[v] = null pointer;
            S[v] = 0;
            count[v] = 0;
     }
     d[source] = 0;
     count[source] = 1;
     put all vertices in queue, Q, in d[v]'s order;
```

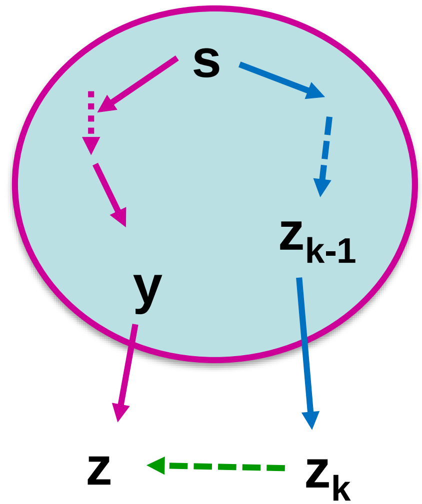
```
while not Empty(Q) {
    u = ExtractCheapest( Q );
    S[u] = 1; /* Add u to S */
    for each vertex v adjacent to u
        if (S[v] != 1 && d[v] > d[u] + w[u,v]) {
            remove v from Q;
            d[v] = d[u] + w[u,v];
            pi[v] = u;
            // the shortest paths to u extends to v, the
            // previously known shortest path(s) to v is replaced
            count[v] = count[u];
            insert v into Q according to its d[v];
        }
        else if (d[v] == d[u] + w[u, v]) {
            // additional shortest path(s) through u to v is found
            count[v] += count[u];
        }
    }
}
```

Question 3

- Dijkstra's algorithm requires that the input graph has all edges being non-negative. Give an example where Dijkstra's algorithm does not work correctly with negative weights.

Question 3

- Thoughts: Assumption of "non-negative weights" is used in the proof of Theorem D1.



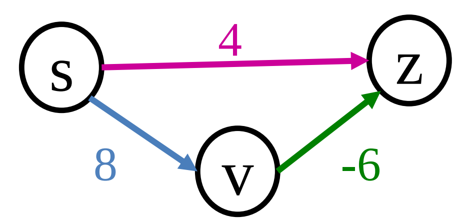
$W(P) = d[y] + W(y, z)$

$W(P') = d[z_{k-1}] + W(z_{k-1}, z_k) +$ distance from z_k to z

Note that: $d[z_{k-1}] + W(z_{k-1}, z_k) \geq d[y] + W(y, z)$

Since distance from z_k to z is non-negative, therefore, **W(P) <= W(P').**

- Design of a counter-example:
  - Look at different edges across the border
  - Design a case that violates the proof



What we have exercised

- Single-source shortest path algorithm: Dijkstra’s algorithm
  - Running of the algorithm
  - Data structure contents in each iteration
  - Modify it to capture more information
  - Design counter-example for its correctness