```
clc, clearvars,

% Change these parameter (lines 5-35) as per your problem.

Nodes = 1:6;  % This creates an array of nodes.

Source = 1;   % Source node

% Adjacency matrix denotes whether there two nodes are adjacent or not,
% i.e., whether they are connected by a directed edge or not. The i-th row
% and j-th column element of this matrix denotes whether there is a
% directed edge from i-th node to the j-th node. (1 means edge, o means no
% edge)
Adjacency_matrix = [0 1 1 1 1 0;
                    0 0 0 1 0 1;
                    0 1 0 0 1 0;
                    0 0 0 0 0 1;
                    0 0 0 0 0 1;
                    0 0 1 0 0 0;];

Weight_matrix =[inf  5   1   4   7  inf;
                inf inf inf  1  inf  5 ;
                inf  1  inf inf  2  inf;
                inf inf inf inf inf  5 ;
                inf inf inf inf inf  6 ;
                inf inf  3  inf inf inf;];


Open = Nodes;
Closed = [];

C(Nodes) = inf;
C(1) = 0;
parent(Nodes) = NaN;


%% You should not need to change the following code

while ~isempty(Open)

    [~, m] = min(C(Open));
    n = Open(m);
    Closed = [Closed, n];
    Open(m) = [];

    neighborOf_n = find(Adjacency_matrix(n,:) == 1);

    for n_prime = neighborOf_n

        if C(n_prime) >  C(n) + Weight_matrix(n,n_prime)
            C(n_prime) = C(n) + Weight_matrix(n,n_prime);
            parent(n_prime) = n;   % this extra line to store the parent node. This
will help in printing the optimal path.
        end

    end

end
```

```
% Optimal paths are stored in the Path variable below.

for node = Nodes
        p = node;
        n = node;
        while n ~=Source
            p = [parent(n), p];
            n = parent(n);
        end
        Path{node} = p;
end
```