Name:Vishal Indradev Chauhan

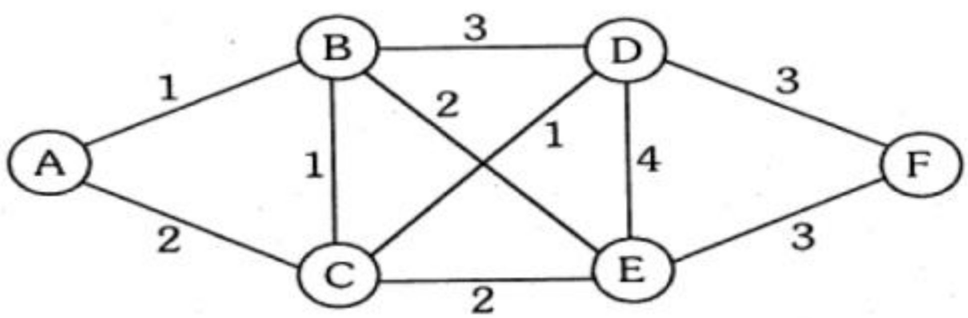
Prn:2020BTECS00090

Title:Greedy method

Sub:DAA

Ass no:8

1) From a given vertex in a weighted connected graph, implement shortest path finding Dijkstra's algorithm.



Q) Show that Dijkstra's algorithm doesn't work for graphs with negative weight edges

Q) Modify the Dijkstra's algorithm to find shortest path.

1. Algorithm: (Pseudocode)

   ……………………………………………….

   STEP 1 : initialize a distance array as positive infinity to all the vertices

   STEP 2 : put the distance from source to itself as 0 dis[source]=0

   STEP 3 : take a min heap and insert {src,0}

   STEP 4:LOOP while heap is not empty

   Temp|<-heap top

   Pop top of heap

   Traverse each neighbour of temp in graph

   Check if (dis[Child] > (wt + dis[node]))

   Then relax this node and insert into the min heap

STEP 5 : Print distance array .

→Code Implementation:

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 100005
#define pii pair<int, int>

// adjency list
vector<pii> adj[N];
// distance array
int dis[N];
//  nodes and edges
int n, m;

// Dijkistra's algorithm for shortest path
void bfs(int src)
{
    // min heap
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, src});

    // distance from source  to itself
    dis[src] = 0;

    while (!pq.empty())
    {
        pii temp = pq.top();
        pq.pop();
        int node = temp.second;
        int wt = temp.first;

        // traversing each neighbour of node temp
        for (auto child : adj[node])
        {
            int nChild = child.first;
            int nwt = child.second;

            // relaxing nodes

            if (dis[nChild] > (nwt + dis[node]))
            {
                dis[nChild] = nwt + dis[node];
                pq.push({dis[nChild], nChild});
            }
        }
```

```
    }
    for (int i = 1; i <= n; i++)
        cout << "A"
             << "  to " << char(i + 64) << "  ->  " << dis[i] << "\n";
}

signed main()
{
    cout << "Enter the number of nodes and the number of edges: ";

    cin >> n >> m;

    for (int i = 0; i < N; i++)
    {
        adj[i].clear();
        dis[i] = INT_MAX;
    }
    cout << "Edges " << endl;
    // storing graph
    for (int i = 1; i <= m; i++)
    {
        char a, b;
        cin >> a >> b;
        int u = a - 64, v = b - 64, wt;
        cin >> wt;
        adj[u].push_back({v, wt});
        adj[v].push_back({u, wt});
    }
    bfs(1);
}
```

OUTPUT:→

```
PS C:\vis-clg+ass+all\DAA\All_assignments\ass8> cd "c:\vis-clg+ass+all\DAA\All_assignments\ass8\" ; if ($?) { g++ q1.cpp -o q1 } ; if ($?) { .\q1 }
Enter the number of nodes and the number of edges: 6 10
Edges
A B 1
A C 2
B D 3
B E 2
B C 1
C E 2
D E 4
```

```
C D 1
D F 3
E F 3
A  to A  ->  0
A  to B  ->  1
A  to C  ->  2
A  to D  ->  3
A  to E  ->  3
A  to F  ->  6
------------------------------------------
```

Q2:Ans→

For negative cyclic graph while loop went in infinite loop as due to negative cycle value from node a to b is reducing continuously so here Dijkstra algorithm fails for negative cyclic graph .

```
PS C:\vis-clg+ass+all\DAA\All_assignments\ass8> cd "c:\vis-clg+ass+
Enter the number of nodes and the number of edges: 4 4
Edges
A B 5
B C -6
C D 2
D A 8
```

Here on above of –ve weight this algorithm is Stucked.

Q3:Ans→

If there is no negative cycle then it is Working properly:

```
PS C:\vis-clg+ass+all\DAA\All_assignments\ass8> cd "c:\vis-clg+ass+all\DAA\All_assignments\ass8\" ; if ($?) { g++ q1.cpp -o q1 } ; if ($?) { .\q1 }
Enter the number of nodes and the number of edges: 4 4
Edges
A B 5
B C 2
C D 1
D A 3
```

```
A  to A  ->  0
A  to B  ->  5
A  to C  ->  4
A  to D  ->  3
PS C:\vis-clg+ass+all\DAA\All_assignments\ass8>
```

2. Complexity of proposed algorithm (Time & Space)

   <u>Space Complexity :</u>

   As here I have used minimum priority queue so fetching is done in logV time and we are doing for all Edges E so time complexity is O(E log V)

   <u>Time Complexity :</u>

   Only distance array is extra space for all vertex So space complexity is O(V).

3. Your comment (How your solution is optimal?)

   In first naiive implementation where complexity is O(V2) but after using min heap we reduced time complexity to O(ElogV).

   ****