Experiment No .8

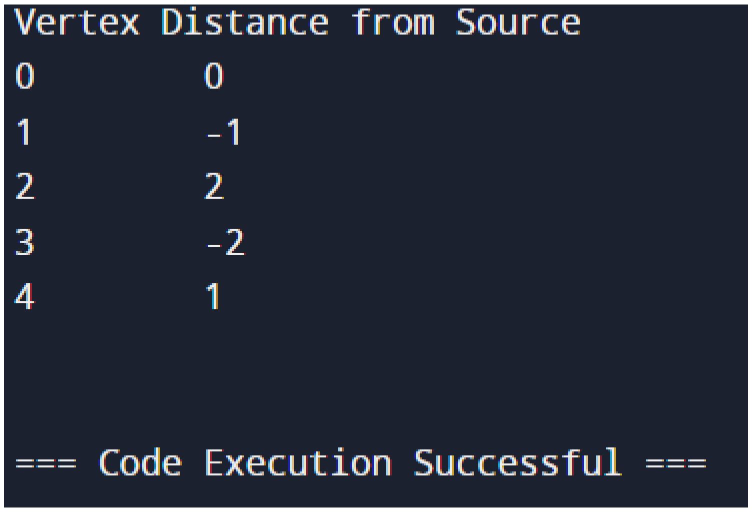
Bellman Ford Method

Program:-

```
#include <stdio.h>
#include <limits.h>
#include <malloc.h>
struct Edge {
int src, dest, weight;
};

struct Graph {
int V, E;
struct Edge* edge;
};

struct Graph* createGraph(int V, int E) {
struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
graph->V = V;
graph->E = E;
graph->edge = (struct Edge*)malloc(E * sizeof(struct Edge));
return graph;
}

void printArr(int dist[], int n) {
printf("Vertex Distance from Source\n");
for (int i = 0; i < n; ++i)
printf("%d \t\t %d\n", i, dist[i]);
}

void BellmanFord(struct Graph* graph, int src) {
int V = graph->V;
int E = graph->E;
int dist[V];
for (int i = 0; i < V; i++)
dist[i] = INT_MAX;
dist[src] = 0;
for (int i = 1; i <= V - 1; i++) {
for (int j = 0; j < E; j++) {
int u = graph->edge[j].src;
int v = graph->edge[j].dest;
int weight = graph->edge[j].weight;
if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
dist[v] = dist[u] + weight;
}
}
for (int i = 0; i < E; i++) {
int u = graph->edge[i].src;
int v = graph->edge[i].dest;
int weight = graph->edge[i].weight;
if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
printf("Graph contains negative weight cycle\n");
return;
}
}
printArr(dist, V);
}

int main() {
int V = 5;
int E = 8;
struct Graph* graph = createGraph(V, E);
graph->edge[0].src = 0;
graph->edge[0].dest = 1;
graph->edge[0].weight = -1;

graph->edge[1].src = 0;
graph->edge[1].dest = 2;
graph->edge[1].weight = 4;

graph->edge[2].src = 1;
graph->edge[2].dest = 2;
graph->edge[2].weight = 3;

graph->edge[3].src = 1;
graph->edge[3].dest = 3;
graph->edge[3].weight = 2;

graph->edge[4].src = 1;
graph->edge[4].dest = 4;
graph->edge[4].weight = 2;

graph->edge[5].src = 3;
graph->edge[5].dest = 2;
graph->edge[5].weight = 5;
```

```
graph->edge[6].src = 3;
graph->edge[6].dest = 1;
graph->edge[6].weight = 1;

graph->edge[7].src = 4;
graph->edge[7].dest = 3;
graph->edge[7].weight = -3;

BellmanFord(graph, 0);

free(graph->edge);
free(graph);

return 0;
}
```

Output:-



```
Vertex Distance from Source
0        0
1        -1
2        2
3        -2
4        1


=== Code Execution Successful ===
```