Shortest-Path-from-all-vertices-to-a-destination

```
#include <bits/stdc++.h>
using namespace std;
#define INF 0x3f3f3f3f

typedef pair<int, int> iPair;

class Graph {
    int V;
    list<pair<int, int> >* adj;

    public:
    Graph(int V);
    void addEdgeRev(int u, int v, int w);
    void shortestPath(int s);
};

Graph::Graph(int V){
    this->V = V;
    adj = new list<iPair>[V];
}

void Graph::addEdgeRev(int u, int v, int w){
    adj[v].push_back(make_pair(u, w));
}

void Graph::shortestPath(int dest){
    priority_queue<iPair, vector<iPair>, greater<iPair> > pq;
    vector<int> dist(V, INF);

    pq.push(make_pair(0, dest));
    dist[dest] = 0;

    while (!pq.empty()) {
        int u = pq.top().second;
        pq.pop();

        list<pair<int, int> >::iterator i;
        for (i = adj[u].begin(); i != adj[u].end(); ++i) {
            int v = (*i).first;
            int weight = (*i).second;

            if (dist[v] > dist[u] + weight) {
                dist[v] = dist[u] + weight;
                pq.push(make_pair(dist[v], v));
            }
        }
    }

    printf("Destination Vertex Distance from all vertex\n");
    for (int i = 0; i < V; ++i)
        printf("%d \t\t %d\n", i, dist[i]);
}

int main(){
    int V = 5;
    Graph g(V);

    g.addEdgeRev(0, 2, 1);
    g.addEdgeRev(0, 4, 5);
    g.addEdgeRev(1, 4, 1);
    g.addEdgeRev(2, 0, 10);
    g.addEdgeRev(2, 3, 5);
    g.addEdgeRev(3, 1, 1);
    g.addEdgeRev(4, 0, 5);
    g.addEdgeRev(4, 2, 100);
    g.addEdgeRev(4, 3, 5);

    g.shortestPath(0);

    return 0;
}
```