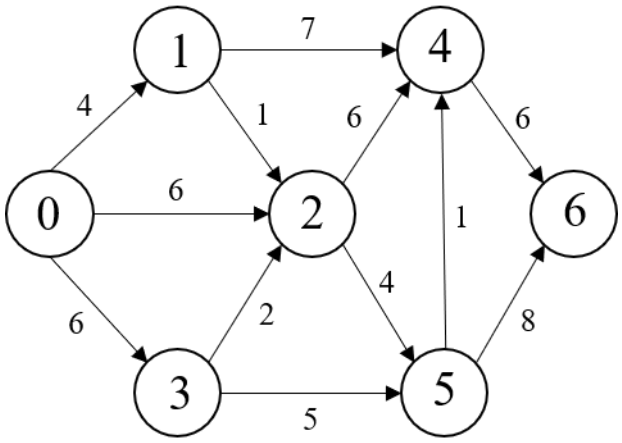
Question 1. Write codes for Dijkstra's algorithm using unsorted array for priority Q.




```cpp
#include <vector>
#include <iostream>
#include <queue>
#include <cstdio>

using namespace std;

#define LAG 20000
#define INF 20000

struct vertex{
    int p,k;
    bool operator < (const vertex & r) const{
        if(k == r.k) return p < r.p;
        else return k > r.k;
    }
};

int vertex_number = 7;
int length[LAG];
int visit[LAG];
vector<vertex> record[LAG];

void D_Algorithm(int start_vertex){
    for(int i = 1; i <= vertex_number; i++){
        length[i] = INF;
        visit[i] = 0;
    }
    length[start_vertex] = 0;
    visit[start_vertex] = 0;
```

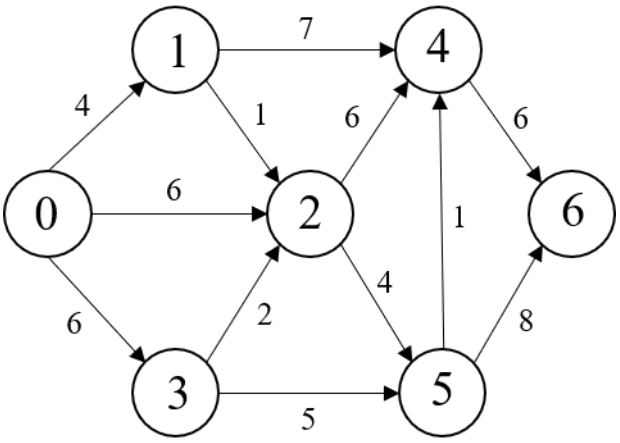
```
    priority_queue<vertex> Q;
    Q.push(vertex{start_vertex,length[start_vertex]});
    while(!Q.empty()){
        vertex node = Q.top();
        Q.pop();
        if(visit[node.p]){
            continue;
        }
        visit[node.p] = 1;
        for(int i = 0; i < record[node.p].size(); i++){
            vertex tmp = record[node.p][i];
            if(length[tmp.p] > node.k + tmp.k){
                length[tmp.p] = node.k + tmp.k;
                Q.push(vertex{tmp.p,length[tmp.p]});
            }
        }
    }
}

int main(){

    int edge_number = 12;
    int start_vertex = 0;
    int source_vertex;
    int end_vertex;
    int porperty;

    for(int i = 0; i <= vertex_number; i++){
        record[i].clear();
    }
    printf("Format: start_vertex end_vertex property.\n");
    while(edge_number--)
    {
        scanf("%d %d %d", &source_vertex, &end_vertex, &porperty);
        record[source_vertex].push_back(vertex{end_vertex,porperty});
    }
    D_Algorithm(start_vertex);
    printf("Vertex vertex number    Shortest length between this vertex
and start vertex\n");
    printf("0                       0\n");
    for (int i = 1; i < 7; i++){
        printf("%d                       %d\n", i, length[i]);
    }
```

```
    return 0;
}
```

Results:

```
PS D:\Code> cd "c:\Users\11099\Desktop\" ; if ($?) { g++ Untitled-3.cpp -o Untitled-3 } ; if ($?) { .\Untitled-3 }
Format: start_vertex end_vertex property.
0 1 4
0 2 6
0 3 6
1 2 1
3 2 2
1 4 7
3 5 5
2 4 6
2 5 4
5 4 1
4 6 6
5 6 8
Vertex vertex number    Shortest length between this vertex and start vertex
0                       0
1                       4
2                       5
3                       6
4                       10
5                       9
6                       16
```

Question 2. Write codes for Bellman-Ford algorithm.




```
#include <stdio.h>
#include <stdlib.h>

struct Edge
{
    int start_vertex, end_vertex, property;
};

struct Graph
{
    int Vertex, Edge;
    struct Edge * edge;
```

```
};

struct Graph * Graph_Initialization(int Vertex, int Edge)
{
    struct Graph * graph = (struct Graph *) malloc( sizeof(struct
Graph));
    graph->Vertex = Vertex;
    graph->Edge = Edge;
    graph->edge = (struct Edge *) malloc( 12 * sizeof( struct Edge ) );

    return graph;
};

void BF_Algorithm(struct Graph* graph, int start_vertex)
{
    int Record[7];

    for (int i = 0; i < 7; i++){
        Record[i] = 1000;
    }
    Record[start_vertex] = 0;

    for (int i = 1; i <= 6; i++)
    {
        for (int k = 0; k < 12; k++)
        {
            int property = graph->edge[k].property;
            if (Record[graph->edge[k].start_vertex] + property <
Record[graph->edge[k].end_vertex]){
                Record[graph->edge[k].end_vertex] =
Record[graph->edge[k].start_vertex] + property;
            }
        }
    }

    printf("\nVertex node number    Shortest length between this vertex
and start vertex\n");
    for (int i = 0; i < 7; i++){
        printf("%d                    %d\n", i, Record[i]);
    }

    return;
}
```

```c
int main()
{
    int Vertex = 7;
    int Edge = 12;
    int Start_Vertex = 0;

    struct Graph* graph = Graph_Initialization(Vertex, Edge);

    printf("Format: start_vertex end_vertex property.\n");
    for(int i = 0; i < 12; i++){
        scanf("%d",&graph->edge[i].start_vertex);
        scanf("%d",&graph->edge[i].end_vertex);
        scanf("%d",&graph->edge[i].property);
    }

    BF_Algorithm(graph, Start_Vertex);

    return 0;
}
```

Results:

```
PS D:\Code> cd "c:\Users\11099\Desktop\" ; if ($?) { g++ Untitled-3.cpp -o Untitled-3 } ; if ($?) { .\Untitled-3 }
Format: start_vertex end_vertex property.
0 1 4
0 2 6
0 3 6
1 2 1
3 2 2
1 4 7
3 5 5
2 4 6
2 5 4
5 4 1
4 6 6
5 6 8

Vertex node number    Shortest length between this vertex and start vertex
0                     0
1                     4
2                     5
3                     6
4                     10
5                     9
6                     16
```