```
#include<iostream>
#include<queue>
#include<list>
#include<vector>
#include<fstream>
using namespace std;
class Dijsktra {
	int V; // nr de varfuri
	list<pair<int, int>>* adj;
public:
	Dijsktra(const char* file) {
		ifstream ifs(file);
		ifs >> V; //citesc din fisier numarul de varfuri
		adj = new list<pair<int, int>>[V];
		while (!ifs.eof()) {
			int u, v, w;
			ifs >> u >> v >> w;
			adj[u].push_back(make_pair(v, w));
		}
	}
	void print() {
		for (int i = 0; i < V; i++)
		{
			cout << i << "->";
			list<pair<int, int>> ::iterator j;
			for (j = adj[i].begin(); j != adj[i].end(); ++j)
				cout << "(" << (*j).first << "," << (*j).second << ")";
			cout << endl;
		}
	}
	void shortestPath(int src) {
		//initializam vectorul distantelor cu infinit
		vector<int> dist(V, INT_MAX);
		vector<int> parent(V);
		for (int i = 0; i < V; i++)
			parent[i]=i;
		//cream coada de prioritati unde se vor aduga nodurile neprocesate
		priority_queue< pair<int,int>, vector <pair<int, int>>, greater<pair<int, int>>> pq;

		pq.push(make_pair(0, src));
		dist[src] = 0;
		while (!pq.empty()) {
			int u,v;// nodul sursa, nodul destinatie
```

```
                int w; //ponderea muchiei [uv]
                u = pq.top().second;
                w = pq.top().first;
                pq.pop();
                list<pair<int, int>> ::iterator i;
                for (i = adj[u].begin(); i != adj[u].end(); ++i) {
                        v = (*i).first;
                        w = (*i).second;
                        if (dist[u] + w < dist[v])
                        {
                                dist[v] = dist[u] + w;
                                pq.push(make_pair(dist[v], v));
                                parent[v] = u;
                        }

                }


        }
        for (int i = 0; i < V; i++)
                cout << dist[i] << ",";
        cout << endl << "PAths:" << endl;
        for (int i = 0; i < V; i++) {
                showPath(parent, i);
                cout << endl;
        }
    }
    void showPath(vector<int> parent, int v) {
        if(v != parent[v])
                showPath(parent,parent[v]);
        cout << v << "->";
    }
};
void main() {
    Dijsktra d("graf.txt");
    d.print();
    d.shortestPath(0);
}
```