```
#include<iostream>
#include<set>
#include<vector>
using namespace std;
typedef vector <int > vi;
typedef pair <int, int > ii;
typedef vector <ii > vii;
typedef vector <vii > vvii;
const int MAX = 1001;
const int MAXINT = 1000000000;
int n;
vvii G(MAX);
vi D(MAX, MAXINT);
void Dijkstra(int s) {
         //usaremos un set para obtener el menor
                set <ii > Q;
         D[s] = 0;
         Q.insert(ii(0, s));
         while (!Q.empty()) {
                 ii top = *Q.begin();
                 Q.erase(Q.begin());
                 int v = top.second;
                 int d = top.first;
                 vii::const_iterator it;
                 for (it = G[v].begin(); it != G[v].end(); it++) {
                         int v2 = it->first;
                         int cost = it->second;
                         if (D[v2] > D[v] + cost) {
                                  if(D[v2] != 1000000000) {
                                         Q.erase(Q.find(ii(D[v2], v2)));
                                 }
                                 D[v2] = D[v] + cost;
                                  Q.insert(ii(D[v2], v2));
                         }
                 }
         }
}

int main() {
        int m, ini, fin = 0;
        cout << "\nIngresar nodo: ";
        cin >> n;
        cout << "\nIngresar arcos: ";
        cin >> m;
        cout << "\nIngresar nodo de inicio: ";
        cin >> ini;
```

```
	cout << "\nIngresar nodo de fin: ";
	cin >> fin;
	// nodos, arcos, inicio, destino
	for (int i = 0; i < m; i++) {
		int a, b, p = 0;

		cout << "\nIngresar a: ";
		cin >> a;
		cout << "\nIngresar b: ";
		cin >> b;
		cout << "\nIngresar p: ";
		cin >> p;
		G[a].push_back(ii(b, p));
		G[b].push_back(ii(a, p));

	}
	Dijkstra(ini);
	printf("%d\n", D[fin]);
	cin.get();
	cin.ignore();
	return 0;
}
```