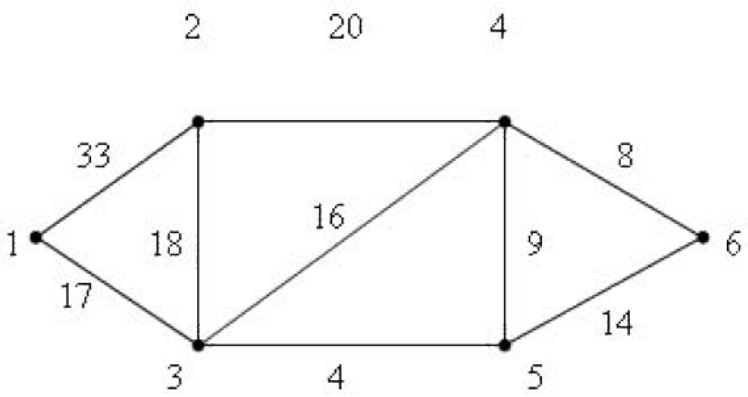
Các đồ thị ví dụ:



Hình 1: Đơn đồ thị có trọng số.

Chương trình cài đặt thuật toán tìm đường đi ngắn nhất bằng thuật toán **Bellman Ford** với có trọng số được biểu diễn bằng ma trận kề. Với đồ thị trên, ma trận kề tương ứng sẽ là:

```
6
0 33 17 0 0 0
33 0 18 30 0 0
17 18 0 16 4 0
0 20 16 0 9 8
0 0 4 9 0 14
0 0 0 8 14 0
```

Sau đây là chương trình (**graph_bellman1.cpp**) đọc đồ thị biểu diễn ở dạng ma trận kề (từ bàn phím hoặc từ file), sau đó tìm và in ra đường đi ngắn nhất giữa 2 đỉnh (nhập từ bàn phím) của đồ thị bằng thuật toán **Bellman Ford**.

```
// Single source shortest paths using Bellman Ford algorithm
// Author: Huu-Tuan Nguyen
// Lecturer at the FIT, Vietnam Maritime University
// Date: 24/11/2015
// Require: C++ 11 supported compiler

#include <iostream> // for cin, cout
#include <fstream> // for file stream
#include <vector> // for vector

using namespace std;

void print_path(int v, int u, vector<int> & prev);

struct Edge {
   int start;
   int end;
   int weight;
};

int main(int argc, char * argv[]) // argc: argument count, argv: argument values
{
```

```
   int n;
   vector<vector<int>> vvi(n, vector<int>(n, 0));
   int i, j;
   if (argc>1)
   {
      if (atoi(argv[1]) == 0)
      {
         cout << "Nhap so dinh cua do thi n=";
         cin >> n;
         vvi = vector<vector<int>>(n, vector<int>(n, 0)); // khai bao n vector, moi vector co n
phan tu
         for (i = 0; i<n; i++)
            for (j = i + 1; j<n; j++)
            {
               cout << "Nhap a[" << i << "][" << j << "]=";
               cin >> vvi[i][j];
               vvi[j][i] = vvi[i][j];
            }
      }
      else if (argc == 3)
      {
         ifstream fin(argv[2]); // mo file
         fin >> n;
         vvi = vector<vector<int>>(n, vector<int>(n, 0)); // khai bao n vector, moi vector co n
phan tu
         for (i = 0; i<n; i++)
            for (j = 0; j<n; j++)
            {
               fin >> vvi[i][j];
               vvi[j][i] = vvi[i][j];
            }
         fin.close(); // dong file
      }

      vector<Edge> edges;
      for (i = 0; i<n; i++)
         for (j = i + 1; j<n; j++)
            if (vvi[i][j]>0)
               edges.emplace_back((Edge) { i, j, vvi[i][j] });

      int INF = int(10e8); // infinitive value
      int startV, endV;
      cout << "Nhap dinh xuat phat:";
      cin >> startV;
      cout << "Nhap dinh ket thuc:";
      cin >> endV;
      vector<int> prev(n, startV); // make all the vertices to have their previous one to be
startV
      vector<int> dist(n, INF); // initialize all vertices with infinitive distances
      dist[startV] = 0;
      for (i = 0; i<n; i++)
         for (auto & edge : edges)
         {
            int u = edge.start; // first component: the vertex u
```

```
            int v = edge.end; // second component: the vertex v
            int w = edge.weight; // third component: the weight of edge[u,v]
            if (dist[v]>dist[u] + w)
            {
               dist[v] = dist[u] + w;
               prev[v] = u;
            }
         }
      // recheck whether there exists a negative circuit in the graph
      bool negCirFlag = false;
      for (auto & edge : edges)
      {
         int u = edge.start; // first component: the vertex u
         int v = edge.end; // second component: the vertex v
         int w = edge.weight; // third component: the weight of edge[u,v]
         if (dist[v]>dist[u] + w)
         {
            cerr << "Do thi co chu trinh am";
            negCirFlag = true;
            break;
         }
      }
      if (!negCirFlag)
      {
         cout << "Duong di ngan nhat tu dinh " << startV << " toi dinh " << endV << " la " <<
dist[endV] << endl;
         print_path(startV, endV, prev);
      }
   }
   else
      cout << "Chay chuong trinh nhu sau:<ten chuong trinh> <0:nhap tu ban phim,1: nhap tu file>
<ten file>";

   return 0;
}

void print_path(int v, int u, vector<int> & prev)
{
   if (prev[u] == v)
      cout << v << "-->" << u;
   else
   {
      print_path(v, prev[u], prev);
      cout << "-->" << u;
   }
}
```