Trường: **ĐH CNTP TP.HCM** Khoa: **Công nghệ thông tin** Bộ môn: **Khoa học máy tính** MH: **TH Cấu trúc rời rạc**	**BÀI 6** THUẬT TOÁN FLOYD TÌM ĐƯỜNG ĐI NGẮN NHẤT	

A. MỤC TIÊU:

B. DỤNG CỤ - THIẾT BỊ THÍ NGHIỆM CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

I. Tóm tắt lý thuyết

1. Thuật toán Floyd

Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh hoặc chỉ ra đồ thị có mạch âm. Ngoài ma trận khoảng cách D ta còn dùng ma trận $Q = (q_{ij})$, trong đó:

$$Q_{ij} = \begin{cases} j \text{ khi } ij \in E \\ 0 \text{ khi } ij \notin E \end{cases}$$

Bước 1: $D_0 = D$, $Q_0 = Q$, $k = 1$.

Bước 2: Với i = 1 đến n, với j = 1 đến n. Đặt

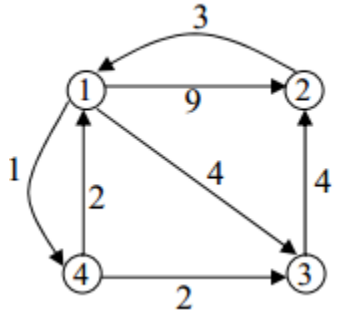
$$D_k(i,j) = \begin{cases} D_{k-1}(i,k) + D_{k-1}(k,j) & \text{nếu } D_{k-1}(i,j) > D_{k-1}(i,k) + D_{k-1}(k,j) \\ D_{k-1}(i,j) & \text{nếu } D_{k-1}(i,j) \le D_{k-1}(i,k) + D_{k-1}(k,j) \end{cases}$$

$$Q_k(i,j) = \begin{cases} Q_{k-1}(i,k) & \text{nếu } D_{k-1}(i,j) > D_{k-1}(i,k) + D_{k-1}(k,j) \\ Q_{k-1}(i,j) & \text{nếu } D_{k-1}(i,j) \le D_{k-1}(i,k) + D_{k-1}(k,j) \end{cases}$$

Bước 3: Nếu k = n thì dừng. Nếu k<n thì trở lại Bước 2 với k:=k+1

2. Ví dụ thuật toán Floyd

Ví dụ: Dùng thuật toán Floyd, hãy tìm đường đi ngắn nhất giữa tất các cặp đỉnh của đồ thị sau.



Đồ thị trên có ma trận khoảng cách D và ma trận đường đi Q như sau:

$D=$

	1	2	3	4
1	0	9	4	1
2	3	0	∞	∞
3	∞	4	0	∞
4	2	∞	2	0

$Q=$

	1	2	3	4
1	0	2	3	4
2	1	0	0	0
3	0	2	0	0
4	1	0	3	0

$D_1=$

	1	2	3	4
1	0	9	4	1
2	3	0	[7]	[4]
3	∞	4	0	∞
4	2	[11]	2	0

$Q_1=$

	1	2	3	4
1	0	2	3	4
2	1	0	[1]	[1]
3	0	2	0	0
4	1	[1]	3	0

$D_2=$

	1	2	3	4
1	0	9	4	1
2	3	0	7	4
3	[7]	4	0	[8]
4	2	11	2	0

$Q_2=$

	1	2	3	4
1	0	2	3	4
2	1	0	1	1
3	[2]	2	0	[2]
4	1	1	3	0

$D_3=$

	1	2	3	4
1	0	[8]	4	1
2	3	0	7	4
3	7	4	0	8
4	2	[6]	2	0

$Q_3=$

	1	2	3	4
1	0	[3]	3	4
2	1	0	1	1
3	2	2	0	2
4	1	[3]	3	0

$D_4=$

	1	2	3	4
1	0	[7]	[3]	1
2	3	0	[6]	4
3	7	4	0	8
4	2	6	2	0

$Q_4=$

	1	2	3	4
1	0	[4]	[4]	4
2	1	0	[1]	1
3	2	2	0	2
4	1	3	3	0

Đến đây thuật toán dừng và ta xác định được đường đi giữa 2 đỉnh bất kỳ dựa vào ma trận Q_4 và tổng trọng số trên đường đi tương ứng dựa vào ma trận D_4

Giả sử ta cần xác định đường đi ngắn nhất từ đỉnh 3 đến đỉnh 4, thực hiện như sau:

Tính Q[3,4]=2, nghĩa là đường đi ngắn nhất từ 3 đến 4 sẽ đi qua đỉnh 2. Tiếp tục tìm đường đi từ 2 đến 4

Tính Q[2,4]=1, nghĩa là đường đi ngắn nhất từ 2 đến 4 sẽ đi qua đỉnh 1, tiếp tục tìm đường đi từ 1 đến 4.

Tính Q[1,4]=4, dừng. vì đã đến đỉnh cuối (đỉnh 4)

Vậy đường đi ngắn nhất từ 3 đến 4 là: 3, 2, 1, 4

Trọng số ngắn nhất tương ứng là D[3,4]=8

II. Bài tập hướng dẫn mẫu

Bài tập 1: Áp dụng thuật toán Floyd để tìm đường đi ngắn nhất từ đỉnh bắt đầu s đến đỉnh kết thúc e.

```
#include <iostream>
#include <fstream>
#include <stack>
#define vc 100
#define vmax 100
typedef struct dothi
{
    int flag;
    int w[vmax][vmax];   // ma tran trong so
    int n;                          // so phan tu cua do thi
}Graph;
void input_matran_ke(Graph &Gr,char *path);
void input_Start_End(Graph Gr, int &start, int &end);  //nhap vao dinh dau va cuoi
int floyd (Graph Gr, int P[][vmax], int start, int end);

void main()
{
    Graph Gr;
    input_matran_ke(Gr,"input.inp");

    cout<<endl<<"-----Thuat toan Dijkstra-----"<<endl;
    int start, end;
    input_Start_End(Gr,start, end);
    int P[vmax][vmax];
    int len=floyd(Gr,P,start,end);
    // in ket qua
    cout<<endl<<"Do dai ngan nhat cua duong di tu "<<start <<" den "<<end<<" la
"<<len<<endl;
    cout<<"Qua trinh duong di: ";

    //truy vet
    char *s, *temp;
    s = new char [Gr.n*10];
    temp = new char [10];
    stack <int> S1;
    stack <int> S2;
```

```
    S1.push(start-1); //danh sach nap cac dinh vao
    S1.push(end-1); //danh sach xuat cac dinh ra
    int dich, tg;
    while (!S1.empty())
    {
        dich = S1.top(); //dich = phan tu dau tien
        S1.pop();         // dua phan tu do ra
        S2.push(dich); //cho vao danh sach xuat
        if (!S1.empty()) //trong khi S1 ko rong thi tiep tuc tim cac dinh
        {
            tg = S1.top();
            while (P[tg][dich] != -1) //tim cac dinh di tu tg den dich
            {
                S1.push(P[tg][dich]);
                tg = S1.top();
            }
        }
    }

    cout<<S2.top()+1;
    S2.pop();

    while (!S2.empty())
    {
        cout<<" --> "<<S2.top()+1;
        strcat(s,temp);
        S2.pop();
    }
    cout<<endl;
    system("pause");
}

void input_matran_ke(Graph &Gr,char *path)
{
    ifstream fileIn(path);
    if (fileIn == NULL)
    {
        cout<<"Khong tim thay file.";
        return;
    }
    fileIn >> Gr.flag;
    fileIn >> Gr.n;

    for (int i=0; i<Gr.n; i++)
    {
        for (int j=0; j<Gr.n; j++)
            fileIn >> Gr.w[i][j];
    }
    fileIn.close();
}

void input_Start_End(Graph Gr, int &start, int &end)
{
    int a,b;
    a = b = 0;
    cout<<endl<<"Cac dinh danh so tu 1 den "<<Gr.n<<endl;
    cout<<"Nhap dinh bat dau : ";
    while (a<1 || a> Gr.n)
    {
        cin>>a;
        if (a<1 || a> Gr.n)
```

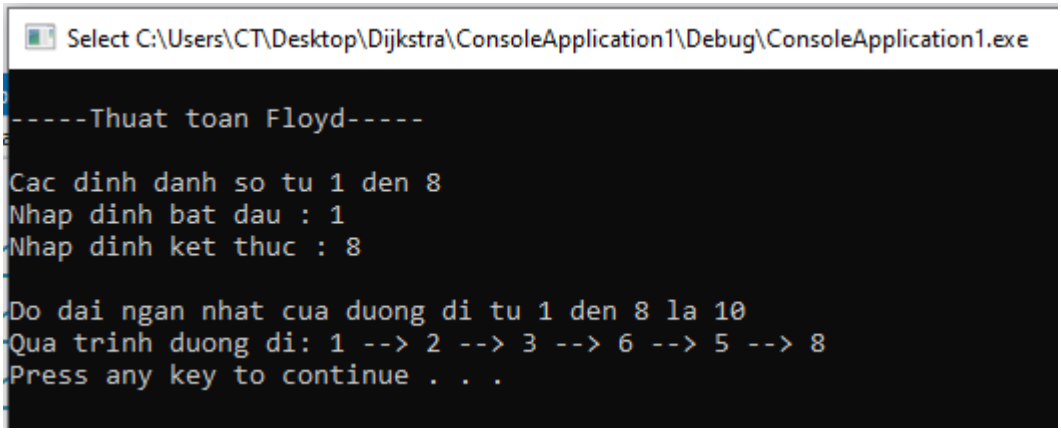
```
            cout<<"Khong hop le ! \nNhap lai dinh bat dau : ";
    }

    cout<<"Nhap dinh ket thuc : ";
    while (b<1 || b> Gr.n)
    {
        cin>>b;
        if (b<1 || b> Gr.n)
            cout<<"Khong hop le ! \nNhap lai dinh ket thuc : ";
    }
    start=a;
    end=b;
}

int floyd (Graph Gr, int P[][vmax], int start, int end)
{
    int a=start-1, b=end-1;
    int A[vmax][vmax];

    for (int i=0; i<Gr.n; i++)
    {
        for (int j=0; j<Gr.n; j++)
        {
            if (Gr.w[i][j])
                A[i][j] = Gr.w[i][j];
            else A[i][j] = vc;
            P[i][j] = -1;
        }
    }

    for (int k=0; k<Gr.n; k++)
    {
        for (int i=0; i<Gr.n; i++)
            for (int j=0; j<Gr.n; j++)
                if (A[i][j] > A[i][k] + A[k][j])
                {
                    A[i][j] = A[i][k] + A[k][j];
                    P[i][j] = k ;
                }
    }
    return A[a][b];
}
```

Ví dụ thực thi thuật toán:

Dữ liệu đầu vào

```
1
8
0 3 5 2 0 0 0 0
3 0 1 0 7 0 0 0
5 1 0 4 0 1 0 0
2 0 4 0 0 3 6 0
0 7 0 0 0 2 0 3
0 0 1 3 2 0 4 6
0 0 0 6 0 4 0 5
0 0 0 0 3 6 5 0
```

Kết quả thực hiện thuật toán Floyd:



III. Bài tập

Bài tập 2: Hãy sử dụng thuật toán Floyd để giải quyết bài toán Ông Ngâu và bà Ngâu được trình bày trong buổi 5.

Bài tập 3: Hãy sử dụng thuật toán Floyd để giải quyết bài toán Đôi bạn được trình bày trong buổi 5.

Bài tập 4: Tìm đường đi ngắn nhất bằng Floyd và xây dựng ứng dụng Game Pikachu.

Bài tập 5: Tìm đường đi ngắn nhất bằng Floyd và xây dựng ứng dụng tìm đường đi trong mê cung.

Bài tập 6: Tìm đường đi ngắn nhất bằng Floyd và xây dựng ứng dụng game rắn săn mồi.