


Trưởng: <b>ĐH CNTP TP.HCM</b> Khoa: <b>Công nghệ thông tin</b> Bộ môn: <b>Khoa học máy tính</b> MH: <b>TH Cấu trúc rời rạc</b>	<p style="text-align: center;"><b>BÀI 5</b></p> <p style="text-align: center;"><b>THUẬT TOÁN DIJKSTRA TÌM ĐƯỜNG ĐI NGẮN NHẤT</b></p>	
---	--	---

## 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. Giới thiệu thuật toán Dijkstra

Năm 1959 E. W. Dijkstra đưa ra một thuật toán rất hiệu quả để giải bài toán đường đi ngắn nhất.

**Bài toán:** Cho  $G = (V, E)$  đơn, liên thông, có trọng số dương ( $w(u, v) > 0$  với mọi  $u$  khác  $v$ ). Tìm đường đi ngắn nhất từ  $u_0$  đến  $v$  và tính khoảng cách  $d(u_0, v)$

#### Phương pháp:

Xác định tuần tự các đỉnh có khoảng cách đến  $u_0$  từ nhỏ đến lớn

1. Trước tiên đỉnh có khoảng cách nhỏ nhất đến  $u_0$  là  $u_0$ .
2. Trong  $V \setminus \{u_0\}$  tìm đỉnh có khoảng cách đến  $u_0$  nhỏ nhất (đỉnh này phải là một trong các đỉnh kề với  $u_0$ ) giả sử đó là  $u_1$
3. Trong  $V \setminus \{u_0, u_1\}$  tìm đỉnh có khoảng cách đến  $u_0$  nhỏ nhất (đỉnh này phải là một trong các đỉnh kề với  $u_0$  hoặc  $u_1$ ) giả sử đó là  $u_2$
4. Tiếp tục như trên cho đến bao giờ tìm được khoảng cách từ  $u_0$  đến mọi đỉnh.

Nếu  $G$  có  $n$  đỉnh thì:  $0 = d(u_0, u_0) < d(u_0, u_1) \leq d(u_0, u_2) \leq \dots \leq d(u_0, u_{n-1})$

#### 2. Thuật toán Dijkstra

Bước 1:  $i:=0$ ,  $S:= V \setminus \{u_0\}$ ,  $L(u_0):=0$ ,  $L(v):=\infty$  với mọi  $v \in S$  và đánh dấu đỉnh  $v$  bởi  $(\infty, -)$ . Nếu  $n=1$  thì xuất  $d(u_0, u_0)=0=L(u_0)$

Bước 2: Với mọi  $v \in S$  và kề với  $u_i$  (nếu đồ thị có hướng thì  $v$  là đỉnh sau của  $u_i$ ), đặt  $L(v):=\min\{L(v), L(u_i)+w(u_i v)\}$ . Xác định  $k=\min L(v)$ ,  $v \in S$

Nếu  $k = L(v_j)$  thì xuất  $d(u_0, v_j)=k$  và đánh dấu  $v_j$  bởi  $(L(v_j); u_i)$

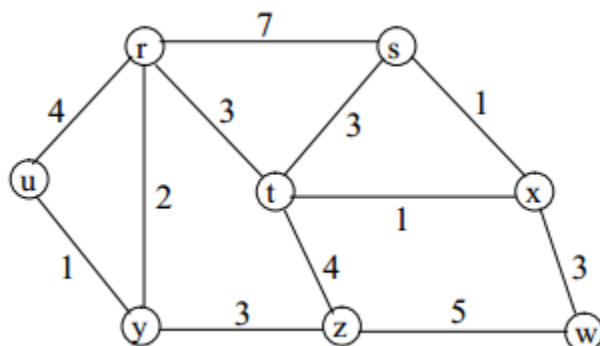
$u_{i+1}:=v_j$ ,  $S:=S \setminus \{u_{i+1}\}$

Bước 3:  $i:=i+1$

- Nếu  $i = n - 1$  thì kết thúc
- Nếu không thì quay lại bước 2.

### 3. Ví dụ thuật toán Dijkstra

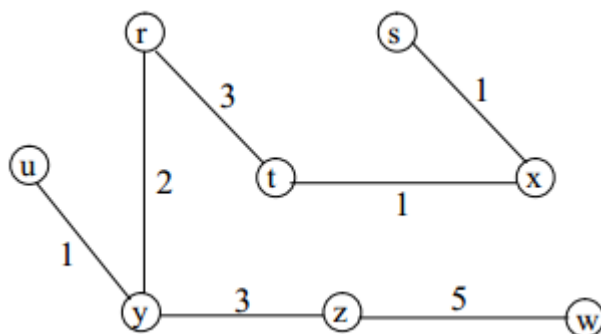
**Ví dụ:** Tìm đường đi ngắn nhất từ đỉnh u đến các đỉnh còn lại.



**Lời giải:**

u	r	s	t	x	y	z	w
0*	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)
-	(4, u)	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	(1, u)*	( $\infty$ , -)	( $\infty$ , -)
-	(3, y)*	( $\infty$ , -)	( $\infty$ , -)	( $\infty$ , -)	-	(4, y)	( $\infty$ , -)
-	-	(10, r)	(6, r)	( $\infty$ , -)	-	(4, y)*	( $\infty$ , -)
-	-	(10, r)	(6, r)*	( $\infty$ , -)	-	-	(9, z)
-	-	(9, t)	-	(7, t)*	-	-	(9, z)
-	-	(8, x)*	-	-	-	-	(9, z)
-	-	-	-	-	-	-	(9, z)*

Cây đường đi ngắn nhất từ đỉnh u đến các đỉnh còn lại.



## II. Bài tập hướng dẫn mẫu

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

```
#include <iostream>
#include <fstream>
```

```

#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 Dijkstra(Graph Gr, int *P, 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;
    int len=Dijkstra(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: ";
    int i = end-1;
    cout<<i+1;
    while (i != start-1)
    {
        cout<<" <-- ";
        cout<<P[i] +1;
        i = P[i];
    }
    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<<"Nhập đỉnh bắt đầu : ";
while (a<1 || a> Gr.n)
{
    cin>>a;
    if (a<1 || a> Gr.n)
        cout<<"Không hợp lệ ! \nNhập lại đỉnh bắt đầu : ";
}

cout<<"Nhập đỉnh kết thúc : ";
while (b<1 || b> Gr.n)
{
    cin>>b;
    if (b<1 || b> Gr.n)
        cout<<"Không hợp lệ ! \nNhập lại đỉnh kết thúc : ";
}
start=a;
end=b;
}

int Dijkstra(Graph Gr, int *&P, int s, int e)
{
    int a=s-1, b=e-1;
    // Len[i] - Giá trị nhỏ nhất từ a -> i. Len1 đánh dấu độ dài.
    int *Len=new int [Gr.n];
    int *S=new int [Gr.n]; //Đánh dấu đỉnh thuộc danh sách đặc biệt
    P=new int [Gr.n]; //truy vết;

    fill (Len,Len+Gr.n,vc); //Gán dương vô hạn ban đầu = vô cùng
    fill (P,P+Gr.n,a);
    fill (S,S+Gr.n,0); //Danh sách đặc biệt
    Len[a] = 0; // khởi tạo độ dài từ a->a = 0
    int i = a;

    //while S<>V
    for (int k=0; k<Gr.n; k++)
    {
        //tìm độ dài ngắn nhất trong các đỉnh
        for (i=0; i<Gr.n; i++) // tìm v thuộc (V-S) và Len[v] < vô cùng
            if (!S[i] && Len[i] != vc)
                break;
        for (int j = i+1 ; j<Gr.n; j++) // tìm đỉnh có Len min
            if (!S[j] && Len[j] < Len[i])
                i = j;
        S[i] = 1;

        //-----Tính độ dài từ đỉnh đang xét tới các đỉnh tiếp

        for (int j = 0; j<Gr.n; j++) //thay đổi độ dài nếu có
        {
            if (!S[j] && Gr.w[i][j])
                if (Len[i] + Gr.w[i][j] < Len[j])
                {
                    Len[j] = Len[i] + Gr.w[i][j];
                    P[j] = i; //truy vết
                }
        }
    }
    return Len[b];
}

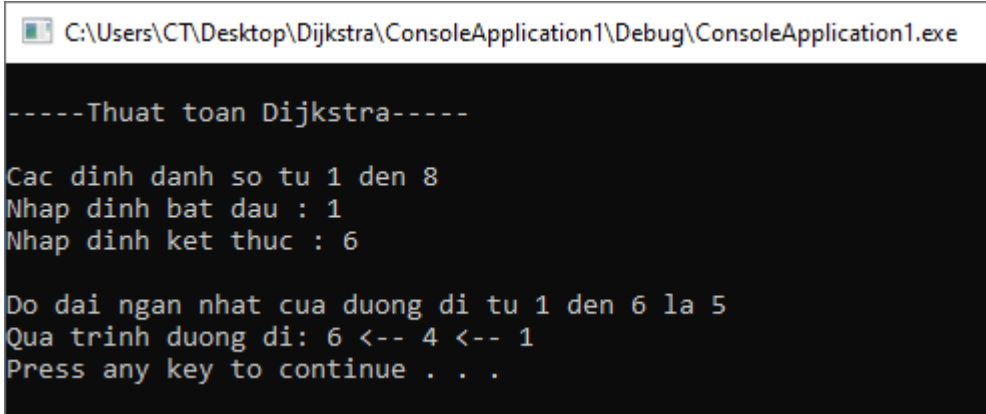
```

### 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 Dijkstra:



```
C:\Users\CT\Desktop\Dijkstra\ConsoleApplication1\Debug\ConsoleApplication1.exe

-----Thuat toan Dijkstra-----

Cac dinh danh so tu 1 den 8
Nhap dinh bat dau : 1
Nhap dinh ket thuc : 6

Do dai ngan nhac cua duong di tu 1 den 6 la 5
Qua trinh duong di: 6 <-- 4 <-- 1
Press any key to continue . . .
```

### III. Bài tập

#### Bài tập 2: Ông Ngâu và bà Ngâu.

Hẳn các bạn đã biết ngày "ông Ngâu bà Ngâu" hàng năm, đó là một ngày đầy mưa và nước mắt. Tuy nhiên, một ngày trước đó, nhà Trời cho phép 2 "ông bà" được đoàn tụ. Trong vũ trụ vùng thiên hà nơi ông Ngâu bà Ngâu ngự trị có N hành tinh đánh số từ 1 đến N, ông ở hành tinh Adam (có số hiệu là S) và bà ở hành tinh Eva (có số hiệu là T). Họ cần tìm đến gặp nhau.

N hành tinh được nối với nhau bởi một hệ thống cầu vồng. Hai hành tinh bất kỳ chỉ có thể không có hoặc duy nhất một cầu vồng (hai chiều) nối giữa chúng. Họ luôn đi tới mục tiêu theo con đường ngắn nhất. Họ đi với tốc độ không đổi và nhanh hơn tốc độ ánh sáng. Điểm gặp mặt của họ chỉ có thể là tại một hành tinh thứ 3 nào đó.

**Yêu cầu:** Hãy tìm một hành tinh sao cho ông Ngâu và bà Ngâu cùng đến đó một lúc và thời gian đến là sớm nhất. Biết rằng, hai người có thể cùng đi qua một hành tinh nếu như họ đến hành tinh đó vào những thời điểm khác nhau

**Dữ liệu:** vào từ file văn bản ONGBANGAU.INP:

- Dòng đầu là 4 số N, M, S, T ( $N \leq 100$ ,  $1 \leq S \neq T \leq N$ ), M là số cầu vồng.
- M dòng tiếp, mỗi dòng gồm ba số nguyên I, J, L thể hiện có cầu vồng nối giữa hai hành tinh i và j có độ dài là L ( $1 \leq I \neq J \leq N$ ,  $0 < L \leq 200$ ).

**Kết quả:** ghi ra file văn bản ONGBANGAU.OUT: do tính chất cầu vồng, mỗi năm một khác, nên nếu như không tồn tại hành tinh nào thỏa mãn yêu cầu thì ghi ra một dòng chữ CRY. Nếu có nhiều hành tinh thỏa mãn thì ghi ra hành tinh có chỉ số nhỏ nhất.

**Ví dụ:**

ONGBANGAU . INP	ONGBANGAU . OUT
4 4 1 4 1 2 1 2 4 1 1 3 2 3 4 2	2

**Thuật toán:**

Ta có nhận xét:

- + Hai hành tinh bất kỳ chỉ được nối đến nhau bởi nhiều nhất một cầu vồng
- + Ông Ngâu và bà Ngâu luôn đi tới mục tiêu theo con đường ngắn nhất
- + Họ đi với vận tốc không đổi và nhanh hơn vận tốc ánh sáng

Thực chất đây là một bài toán đồ thị, ta có thuật toán như sau:

Từ hành tinh S (nơi ông Ngâu ở) ta xây dựng bảng SP, trong đó SP[i] là đường đi ngắn nhất từ hành tinh S đến hành tinh i (do ông Ngâu luôn đi tới mục tiêu theo con đường ngắn nhất). SP[i] = 0 tức là không có đường đi từ hành tinh S đến hành tinh i.

Tương tự ta sẽ xây dựng bảng TP, trong đó TP[i] là đường đi ngắn nhất từ hành tinh T đến hành tinh i. Và TP[i] = 0 tức là không có đường đi từ hành tinh T đến hành tinh i.

Do yêu cầu của bài toán là tìm hành tinh khác S và T mà 2 ông bà Ngâu cùng đến một lúc và trong thời gian nhanh nhất. Tức là ta sẽ tìm hành tinh h sao cho (h khác S và T) và (SP[h] = TP[h]) đạt giá trị nhỏ nhất khác 0. Nếu không có hành tinh h nào thỏa mãn thì ta thông báo CRY

Để xây dựng mảng SP và TP ta chọn giải thuật Dijkstra tìm đường đi ngắn nhất giữa 2 đỉnh đồ thị.



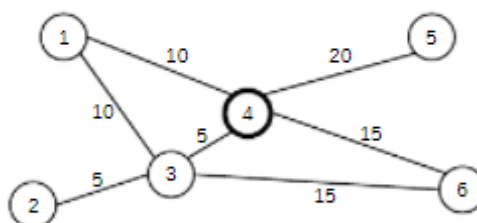
### Bài tập 3: Đôi bạn.

**Kết quả :** ghi ra file văn bản FRIEND.OUT

- Dòng 1: Ghi từ YES hay NO tùy theo có phương án giúp cho hai bạn gặp nhau hay không. Trong trường hợp có phương án:
- Dòng 2: Ghi thời gian ít nhất để Tuấn tới trường
- Dòng 3: Ghi các nút giao thông theo thứ tự Tuấn đi qua
- Dòng 4: Ghi thời gian ít nhất để Mai tới trường
- Dòng 5: Ghi các nút giao thông theo thứ tự Mai đi qua
- Dòng 6: Ghi số hiệu nút giao thông mà hai bạn gặp nhau
- Dòng 7: Thời gian sớm nhất tính bằng giây kể từ 6 giờ sáng mà hai bạn có thể gặp nhau.

**Ví dụ :** Với sơ đồ giao thông sau: (N=6,M=7, Ha=1, Sa=6, Hb=2, Sb=5)

Dòng	FRIEND . INP	FRIEND . OUT
1	6 7	YES
2	1 6 2 5	25
3	1 3 10	1 4 6
4	1 4 10	30
5	2 3 5	2 3 4 5
6	3 4 5	4
7	3 6 15	10
8	4 5 20	
9	4 6 15	



#### Thuật toán:

Sử dụng thuật toán Dijkstra, xây dựng thủ tục: Dijkstra(start:integer, var d: mảng\_nhân); để xây dựng mảng nhân d cho đường đi ngắn nhất từ điểm xuất phát start đến mọi đỉnh (có thể tới từ xuất phát). Sau đó gọi thủ tục này 4 lần bằng các lời gọi:

Trước kia Tuấn và Mai là hai bạn cùng lớp còn bây giờ hai bạn học khác trường nhau. Cứ mỗi sáng, đúng 6 giờ cả hai đều đi từ nhà tới trường của mình theo con đường mất ít thời gian nhất (có thể có nhiều con đường đi mất thời gian bằng nhau và đều ít nhất). Nhưng hôm nay, hai bạn muốn gặp nhau để bàn việc họp lớp cũ nhân ngày 20-11.

Cho biết sơ đồ giao thông của thành phố gồm N nút giao thông được đánh số từ 1 đến N và M tuyến đường phố (mỗi đường phố nối 2 nút giao thông). Vị trí nhà của Mai và Tuấn cũng như trường của hai bạn đều nằm ở các nút giao thông. Cần xác định xem Mai và Tuấn có cách nào đi thỏa mãn yêu cầu nêu ở trên, đồng thời họ lại có thể gặp nhau ở nút giao thông nào đó trên con đường tới trường hay không ? (Ta nói Tuấn và Mai có thể gặp nhau tại một nút giao thông nào đó nếu họ đến nút giao thông này tại cùng một thời điểm). Nếu có nhiều phương án thì hãy chỉ ra phương án để Mai và Tuấn gặp nhau sớm nhất.

**Dữ liệu:** vào từ file văn bản FRIEND.INP

- Dòng đầu tiên chứa 2 số nguyên dương N, M ( $1 \leq N \leq 100$ );
- Dòng tiếp theo chứa 4 số nguyên dương Ha, Sa, Hb, Sb lần lượt là số hiệu các nút giao thông tương ứng với: Nhà Tuấn, trường của Tuấn, nhà Mai, trường của Mai.
- Dòng thứ i trong số M dòng tiếp theo chứa 3 số nguyên dương A, B, T. Trong đó A và B là hai đầu của tuyến đường phố i. Còn T là thời gian (tính bằng giây  $\leq 1000$ ) cần thiết để Tuấn (hoặc Mai) đi từ A đến B cũng như từ B đến A.

Giả thiết là sơ đồ giao thông trong thành phố đảm bảo để có thể đi từ một nút giao thông bất kỳ đến tất cả các nút còn lại.

**Bài tập 4:** Song song hóa thuật toán dijkstra tìm đường đi ngắn nhất từ một đỉnh đến tất cả các đỉnh.