HƯỚNG DẪN THUẬT TOÁN DIJKSTRA

Ứng dụng giải thuật Dijkstra, hãy tìm đường đi ngắn nhất từ đỉnh *s* đến đỉnh *t* trong đồ thị *G* có *N* đỉnh.

1. Hướng dẫn

Dữ liệu vào: tập tin văn bản DIJKSTRA.INP gồm:

- Dòng đầu chứa số nguyên *N* ($N \leq 100$).
- *N* dòng tiếp theo, mỗi dòng chứa *N* số nguyên dương biểu diễn ma trận trọng lượng của đồ thị *G* theo qui ước sau (các số trên cùng dòng cách nhau bởi khoảng trắng)

Dữ liệu xuất: hiển thị ra màn hình

- Nếu tìm được đường đi ngắn nhất: Dòng đầu chứa số nguyên *k* là chiều dài của đường đi từ *s* đến *t*. Dòng tiếp theo là các đỉnh thuộc đường đi.
- Nếu không tìm được: Thông báo không tìm được đường đi từ *s* đến *t*.

2. Thuật toán

Cho đồ thị $G = (X, E)$ có *N* đỉnh được biểu diễn bằng một ma trận trọng lượng (*N* dòng, *N* cột).

Các biến được sử dụng:

- *L*: ma trận trọng lượng
- *n*: số đỉnh của đồ thị
- LABEL: biến có cấu trúc gồm 2 trường WEIGHT VÀ LAST.
- D[*n*]: mảng các NHAN, lưu đường đi từ ngắn nhất từ s đến các đỉnh.
- T[*n*]: mảng đánh dấu các đỉnh đã biết đường đi ngắn nhất từ s đến nó.
- *s*, *t*: đỉnh bắt đầu và đỉnh kết thúc

Bước 1

Khởi tạo T[*i*]=0 và gắn nhãn D(*s*) = 0, D(*i*) = -1 $\forall$ i $\neq$ s

Bước 2

Nếu t $\notin$ T thì dừng và D(t) chính là độ dài đường đi ngắn nhất từ s đến t

Bước 3

Chọn đỉnh v $\in$ T sao cho D(v) nhỏ nhất và gán T = T \ {v}

Bước 4

Với mỗi đỉnh k $\in$ T mà có cạnh nối từ v đến k, gán

$$D(k) = \min \{D(k), D(v) + L(v,k)\}$$

Tiếp B2

3. ***Chương trình***

```
#define MAX 100

typedef struct {
	int weight;
	int last;
}LABEL;

typedef struct {
	int n;
	int L[MAX][MAX];
}GRAPH;

LABEL D[MAX];
int nD;
int T[MAX];
int nT;
int s, t;

void Init(GRAPH g)
{
	nT=g.n; nD=g.n;
	for(int i=0; i<nT; i++)
		T[i] = 0;
	for(int j=0; j<nD; j++)
	{
		D[j].weight=-1;
		D[j].last=-1;
	}
}

int Min()
{
	int min=-1, ind = -1;
	for(int i=0; i<nD; i++)
		if(D[i].weight!=-1 && T[i]!=1)
			if(min==-1 || min>D[i].weight) {
				min=D[i].weight;
				ind=i;
			}
	return ind;
}
```

```
void PrintResult()
{
	int tmp, x=30, y=15;
	if(D[t].weight != -1) {
		printf("\nDuong di ngan nhat la: %d\n", D[t].weight);
		gotoxy(x, y);
		printf("%d", t+1);
		tmp=D[t].last;
		do {
			x-=5;
			gotoxy(x, y);
			printf("%d -> ", tmp+1);
			tmp=D[tmp].last;
		} while(tmp!=-1);
	}
	else
		printf("\nKhong co duong di tu s den t");
}

void Dijkstra(GRAPH g)
{
	int v;
	printf("Nhap dinh bat dau s: ");
	scanf("%d", &s);
	printf("Nhap dinh ket thuc t: ");
	scanf("%d", &t);
	s--; t--;
	Init(g);
	D[s].weight=0;
	T[s]=1; //T=T\{s}
	v=s;
	do {
		for(int k=0; k<g.n; k++)
			if(g.L[v][k]>0 && T[k]!=1)
				if(D[k].weight==-1 ||
				D[k].weight>D[v].weight+g.L[v][k]) {
					D[k].weight=D[v].weight+g.L[v][k];
					D[k].last=v;
				}
		v=Min();
		if(v!=-1) T[v]=1; // T=T\{v}
	} while(T[t]!=-1 && v!=-1);
}
```

```
void main()
{
	GRAPH g;
	clrscr();
	ReadData("Dijkstra.inp",g);
	Dijkstra(g);
	PrintResult();
	getch();
}
```

4. ***Bài tập***

Bài tập 1:

Dùng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh u đến đỉnh v trên đơn đồ thị G có n đỉnh được đánh số từ 0 đến n-1. Trọng số của các cung trên G đều là số dương.

Dữ liệu vào từ tập tin văn bản **DIJKSTRA.IN** gồm

- Dòng đầu chứa 3 số nguyên dương n u v (n<=100)
- N dòng tiếp theo, mỗi dòng chứa n số thực là các số trong ma trận trọng lượng của đồ thị G. Qui ước: trọng lượng ghi là 0 nếu không có đường nối trực tiếp.

Dữ liệu xuất ra tập tin văn bản **DIJKSTRA.OUT** gồm:

- Dòng đầu chứa 2 số m T với m là số đỉnh có trong đường đi ngắn nhất tìm được (-1 nếu không tìm được), T là trọng số của đường đi tương ứng.
- Dòng thứ 2 chứa m số là chỉ số của các đỉnh trên đường đi tương ứng theo đúng trình tự.

Lưu ý: Các số trên cùng dòng cách nhau khoảng trắng.

Bài tập 2: Bài tập ứng dụng - Tài xế Taxi.

Thành phố A có n nút giao thông được đánh số từ 0 đến n-1. Hãng Taxi T đã khảo sát tất cả các đường nối các nút giao thông và ghi lại độ dài của chúng trong một bảng 2 chiều A gồm n x n ô. A[i][j] là độ dài đường nối từ nút giao thông i đến nút j, được ghi = 0 nếu không có. Bạn là tài xế đang ở tại nút giao thông S cần đón 2 người khách tại các nút V_1 và V_2 sau đó trả khách tại nút E. Hãy tìm cách thực hiện công việc trên với quãng đường phải đi là ít nhất.

Dữ liệu được cho trong tập tin văn bản **TAXI.IN** gồm:

- Dòng đầu chứa 5 số nguyên dương n S V_1 V_2 E (n≤100)
- N dòng tiếp theo, mỗi dòng chứa n số thực là các số trong bảng A.

Dữ liệu xuất ra tập tin văn bản **TAXI.OUT** gồm:

- Dòng đầu chứa 2 số m T với m là số nút giao thông có trong đường đi tìm được (-1 nếu không tìm được), T là độ dài của đường đi tương ứng.
- Dòng thứ 2 chứa m số là chỉ số của các nút giao thông trên đường đi tương ứng theo đúng trình tự.

Lưu ý: Các số trên cùng dòng cách nhau khoảng trắng.

Ví dụ:

TAXI.IN	**TAXI.OUT**
4 0 1 2 3 0 1 0 0 0 0 2 0 0 0 0 3 0 0 0 0	4 6 0 1 2 3