HƯỚNG DẪN CODE THUẬT TOÁN DIJKSTRA

Chú ý: Trong hướng dẫn này, chỗ nào có cụm từ "bạn viết code" hay đại loại thế. Thì bạn phải viết code chỗ đó hén.

Khi bạn làm tới phần này, thì bạn **đã làm được việc đọc thông tin** của đồ thị từ một file nào đó vào chương trình của bạn rồi hén. Nếu bạn vẫn chưa làm được điều này thì đề nghị bạn mở lại file **"HƯỚNG DẪN CODE NHẬP XUẤT MA TRẬN KỀ TỪ FILE"** đọc và làm nhé. Còn nếu bạn đã làm được rồi thì chúng ta tiếp tục hén **J**

Nhắc lại: Thông tin đồ thị của bạn sẽ được lưu trữ trong chương trình thông qua một cấu trúc như sau đúng không?

```
#define MAX 20 // định nghĩa giá trị MAX
#define  inputfile  "C:/test.txt" // định nghĩa đường dẫn tuyệt đối đến file chứa thông tin của đồ thị
typedef struct GRAPH {
        int n; // số đỉnh của đồ thị
        int a[MAX][MAX]; // ma trận kề của đồ thị
}DOTHI;
```

Bước 1: Định nghĩa VOCUC, tạo 1 mảng 1 chiều LuuVet dùng để lưu vết đường đi từ S **à** F, 1 mảng 1 chiều khác với tên là ChuaXet dùng để đánh dấu đỉnh nào trong đồ thị đã xét rồi, đỉnh nào chưa xét trong quá trình tìm đường đi từ S **à** F, 1 mảng DoDaiDuongDiToi để lưu lại độ dài nhỏ nhất trong quá trình tìm đường đi từ S **à** F.

```
#define VOCUC 1000
int LuuVet[MAX]; // LuuVet[i] = đỉnh liền trước i trên đường đi từ S  à  i
int ChuaXet[MAX]; // ChuaXet[i] = 0 là đỉnh i chưa được xét đến trong quá trình tìm đường đi, còn ChuaXet[i] = 1 là đỉnh I được xét đến rồi trong quá trình tìm đường đi.
 int DoDaiDuongDiToi[MAX]; // Lưu lại độ dài nhỏ nhất tới đỉnh i
```

Bước 2: viết hàm TimDuongDiNhoNhat để tìm đỉnh chưa xét có giá trị đường đi nhỏ nhất, rồi tiếp tục thi hành thuật toán, căn cứ vào mảng DoDaiDuongDiToi.

```
int TimDuongDiNhoNhat(DOTHI g)
{
	int li = -1; // nếu không tìm thấy đỉnh nào thỏa điều kiện thì trả về -1
	float p = VOCUC;
	for(int i = 0 ; i< g.n ; i++)
	{
		if(ChuaXet[i] == 0 && DoDaiDuongDiToi[i] < p)
		{
			p = DoDaiDuongDiToi[i];
			li = i;
		}
	}
	return li; // trả về đỉnh chưa xét có giá trị đường đi nhỏ nhất
}
```

Bước 3: viết hàm CapNhatDuongDi để tiến hành cập nhập lại giá trị đường đi trong quá trình thi hành thuật toán tại đỉnh u.

```
void CapNhatDuongDi(int u, DOTHI g)
{
	ChuaXet[u] = 1;// đỉnh u đã được chọn nên phải gán giá trị ChuaXet của nó = 1
	for(int v = 0; v < g.n ; v++)
	{
		if(ChuaXet[v]==0 && g.a[u][v] >0 && g.a[u][v] < VOCUC) // tìm một đỉnh v chưa xét và có cạnh nối từ u à v
			if(DoDaiDuongDiToi[v] > DoDaiDuongDiToi[u] + g.a[u][v]) //nếu như độ dài đường đi tới đỉnh v từ đỉnh khác mà lớn hơn độ dài đường đi tới đỉnh u + cạnh (u,v) thì tiến hành cập nhập lại
```

```
                {
                        DoDaiDuongDiToi[v] = DoDaiDuongDiToi[u] + g.a[u][v];
                        LuuVet[v] = u;
                }
        }
}
```

Bước 4: viết hàm Dijkstra để tiến hành thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh S **à** đỉnh F.

```
void Dijkstra (int S, int F, DOTHI g)
{
        //Khởi tạo các giá trị cần thiết cho thuật toán
        int i;
        for (i = 0; i< g.n ; i++)
        {
                ChuaXet[i] = 0;
                DoDaiDuongDiToi[i] = VOCUC;
                LuuVet[i] = -1;
        }
        DoDaiDuongDiToi[S] = 0;

        //Thi hành thuật toán Dijkstra
        while(ChuaXet[F] == 0) // trong khi thuật toán tìm đường đi vẫn chưa xét đến đỉnh F thì tiếp tục
        {
                int u = TimDuongDiNhoNhat(g); // tìm đỉnh mà có độ dài đường đi nhỏ nhất ở bước hiện tại
                if(u == -1) break; // nếu như không tìm được đỉnh nào thì dừng thuật toán và kết quả không tìm thấy đường đi. Ngược lại tiến hành cập nhập lại độ dài đường đi.
```

```
            CapNhatDuongDi(u,g);  // cập nhập lại độ dài đường đi
        }

        if (ChuaXet[F] == 1) //Xuất kết quả đường đi nếu quá trình thi hành thuật toán đã tới điểm F
        {
            printf("Duong di ngan nhat tu dinh %d den dinh %d la: \n\t",S,F);
            i = F;
            printf("%d ", F);
            while (LuuVet[i] != S)
            {
                printf ("<-%d", LuuVet[i]);
                i = LuuVet[i];
            }
            printf ("<-%d\n", LuuVet[i]);
            printf("\tVoi do dai la %d\n",DoDaiDuongDiToi[F]);
        }
        else // ngược lại thì không có đường đi từ S à F
        {
            printf("Khong co duong di tu dinh %d den dinh %d \n",S,F);
        }
}
```

Bước 5: Code trong hàm main để gọi hàm các hàm tương ứng và chạy. Có thể làm như sau:

```
void main()
{
    DOTHI g;
    clrscr();
```

```
	if (DocMaTranKe(inputfile, g) == 1)
	{
		printf("Da lay thong tin do thi tu file thanh cong.\n\n");
		XuatMaTranKe(g);
		printf("Bam 1 phim bat ki de bat dau tim cay khung nho nhat theo thuật
toán Kruskal ...\n\n");
		getch();
		int S, F;
		printf ("Nhap vao dinh bat dau: ");
		scanf("%d",&S);
		printf ("Nhap vao dinh ket thuc: ");
		scanf("%d",&F);
		Dijkstra(S,F,g);
	}
	getch();
}
```

HƯỚNG DẪN CHI TIẾT TỚI CỠ NÀY RỒI MÀ BẠN VẪN KHÔNG LÀM ĐƯỢC NỮA THÌ BẠN CHUẨN BỊ TINH THẦN ĐI HÉN J

Chúc các bạn may mắn và học tốt môn này

GOOD LUCK TO U