HƯỚNG DẪN CODE THUẬT TOÁN FLOYD

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 2 chiều Sau_Nut dùng để lưu vết đường đi từ bất kỳ đỉnh i đến đỉnh j nào, 1 mảng 2 chiều khác với tên là L dùng để lưu lại độ dài đường đi ngắn nhất từ đỉnh i tới đỉnh j trong đồ thị.

```
#define VOCUC 1000
int Sau_Nut[MAX][MAX]; // Sau_Nut[i][j] = đỉnh liền sau i trên đường đi từ i à j
int L[MAX][MAX]; // L[i][j] = lưu lại độ dài đường đi ngắn nhất từ đỉnh i tới đỉnh j trong đồ thị
```

Bước 2: viết hàm Floyd để tiến hành thuật toán Floyd tìm đường đi ngắn nhất từ đỉnh giữa hai đỉnh bất kỳ i và j.

```
void Floyd(DOTHI g)
{
	int i,j;
	// khởi tạo giá trị cho 2 mảng 2 chiều L và Sau_Nut như trong thuật toán
	for(i = 0; i< g.n ; i++)
	{
		for( j = 0; j < g.n ; j++)
		{
			if(g.a[i][j] > 0) // nếu có cạnh nối đỉnh i với đỉnh j
			{
				Sau_Nut[i][j] = j; // khởi tạo đỉnh liền sau i trên đường tìm đường đi từ i tới j là j.
				L[i][j] = g.a[i][j]; // lưu lại độ dài ngắn nhất tại thời điểm hiện tại từ điểm i đến đỉnh j là cạnh nối đỉnh i với đỉnh j.
			}
			else  // không có cạnh nối từ đỉnh i đến đỉnh j
			{
				Sau_Nut[i][j] = -1; // khởi tạo đỉnh liền sau i trên đường tìm đường đi từ i tới j là -1.
				L[i][j] = VOCUC; // lưu lại độ dài ngắn nhất tại thời điểm hiện tại từ điểm i đến đỉnh j là VOCUC (không có đường đi).
			}
		}
	}

	// Thi hành thuật toán Floyd
	for(int k = 0; k< g.n ; k++)
	{
		for(i = 0 ; i < g.n ; i++)
```

```
            {
                for(j = 0; j< g.n ; j++)
                {
                    if(L[i][j] > L[i][k] + L[k][j]) // nếu tồn tại một đỉnh trung gian k sao cho đường đi từ đỉnh i qua đỉnh k tới đỉnh j ngắn hơn đường đi từ đỉnh i đến đỉnh j thì tiến hành chọn đường đi đó.
                    {
                        L[i][j] = L[i][k] + L[k][j]; // cập nhập lại độ dài đường đi từ i tới j.
                        Sau_Nut[i][j] = Sau_Nut[i][k]; // lưu lại đỉnh liền sau i trên đường tìm đường đi từ i tới j là k.
                    }
                }
            }
        }

        // xuất kết quả tìm đường đi ngắn nhất từ S --> F
        int S,F;
        printf ("nhap vao dinh bat dau: ");
        scanf("%d",&S);
        printf("Nhap vao dinh ket thuc: ");
        scanf("%d",&F);
        if (Sau_Nut[S][F] == -1) // nếu giá trị Sau_Nut[S][F] là -1 thì điều này có nghĩa là đỉnh liền sau S là -1 trên đường tìm đường đi từ S tới F. Tương đương với việc không có đường đi từ đỉnh S đến F.
        {
            printf ("Khong co duong di tu dinh %d den dinh %d la :\n",S,F);
        }
        else // ngược lại có đường đi từ S tới F.
```

```
	{
		printf ("Duong di ngan nhat tu dinh %d den dinh %d la :\n",S,F);
		printf ("\t%d",S);
		int u = S;
		while(Sau_Nut[u][F] != F) // trong khi giá trị đỉnh liền sau u trên đường tìm đường đi ngắn nhất từ S tới F không phải là F thì tiếp tục theo vết đi tới đến khi nào gặp F thì dừng.
		{
			u = Sau_Nut[u][F];
			printf (" --> %d",u);
		}
		printf (" --> %d",F);
		printf ("\n\tVoi tong trong so la %d",L[S][F]);
	}
}
```

Bước 3: 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 đường đi ngắn nhất theo thuật toán Floyd ...\n\n");
```

```
            getch();
            Floyd(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