BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO THỰC TẬP CƠ SỞ**

**THUẬT TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT GIỮA HAI ĐIỂM VỚI DANH SÁCH CẠNH**

**(THUẬT TOÁN DIJKSTRA)**

**Giảng viên hướng dẫn : Nguyễn Thủy Đoan Trang**

**Sinh viên thực hiện : Nguyễn Sơn Tùng**

**Mã số sinh viên : 60137435**

Khánh Hòa: 20…

**MỤC LỤC**

[PHẦN MỞ ĐẦU 3](#_Toc61888099)

[LỜI CẢM ƠN 4](#_Toc61888100)

[CHƯƠNG I. CƠ SỞ LÝ THUYẾT 5](#_Toc61888101)

[I.1. Lịch sử thuật toán 5](#_Toc61888102)

[I.2. Ý tưởng thuật toán 5](#_Toc61888103)

[I.2.1. Điều kiện để có thể áp dụng thuật toán 5](#_Toc61888104)

[I.2.2. Các bước thực hiện 7](#_Toc61888105)

[I.3. Giải quyết bài toán qua ví dụ 7](#_Toc61888106)

[CHƯƠNG II. THIẾT KẾ CHƯƠNG TRÌNH 9](#_Toc61888107)

[II.1. Dữ liệu đầu vào 9](#_Toc61888108)

[II.2. Tổ chức lưu trữ, các biến dữ liệu 10](#_Toc61888109)

[II.2.1. Các biến khai báo trong chương trình 10](#_Toc61888110)

[II.2.2. Định nghĩa một số giá trị mặc định 10](#_Toc61888111)

[II.2.3 Các hàm chức năng xử lý từng nhiệm vụ riêng 11](#_Toc61888112)

[II.3 . Chương trình hoàn thiện 16](#_Toc61888113)

[CHƯƠNG III. THỰC THI CHƯƠNG TRÌNH 21](#_Toc61888114)

[CHƯƠNG IV. MÔ PHỎNG CHƯƠNG TRÌNH BẰNG ĐỒ HỌA 25](#_Toc61888115)

[IV.1. Giới thiệu về Windows Forms 25](#_Toc61888116)

[IV.2. Cấu trúc về giao diện sẽ mô phỏng bài toán 25](#_Toc61888117)

[Chương V. Kết luận và hướng phát triển 29](#_Toc61888118)

[V.1. Những điểm đạt được 29](#_Toc61888119)

[V.2. Những điểm chưa đạt được 29](#_Toc61888120)

[V.3. Hướng phát triển trong tương lai 29](#_Toc61888121)

[V.4. Kết luận 29](#_Toc61888122)

[V.5. Tài liệu tham khảo sử dụng trong toàn bộ bài báo cáo 30](#_Toc61888123)

# PHẦN MỞ ĐẦU

Lý thuyết đồ thị là một lĩnh vực nghiên cứu đã có từ lâu đời và những thành quả của các công trình nghiên cứu đấy đã và đang được ứng dụng rộng rãi trong ngày nay. Một trong những kết quả đầu tiên trong lý thuyết đồ thị xuất hiện trong báo cáo của Leonhard Euler về “Bảy cây cầu ở Königsberg”, xuất bản năm 1736. Bài báo này cũng được xem như một trong những kết quả topo[1] đầu tiên trong hình học, tức là, nó không hề phụ thuộc vào bất cứ độ đo nào. Nó diễn tả mối liên hệ sâu sắc giữa lý thuyết đồ thị và topo học. Từ đó đến nay lý thuyết đồ thị không ngừng phát triển và đạt nhiều thành tựu đáng kể với nhiều các vấn đề được giải quyết với nhiều thuật toán khác nhau.

Đồ thị được sử dụng để giải quyết các bài toán trong nhiều lĩnh vực khác nhau. Tìm đường đi ngắn nhất cũng là một trong những vấn đề đã được lý thuyết đồ thị giải quyết. Hiện nay cũng có khá nhiều thuật toán giải quyết vấn đề này. Nhưng trong phần báo cáo này em sẽ tập trung tìm hiểu về thuật toán của Dijkstra. Nhờ các thuật toán này mà mà chúng ta sẽ biết được đường đi ngắn nhất khi đi đến một địa điểm nào đó, vừa ít tốn thời gian, vừa không hao phí công sức nhiều. Hay nói cách bao quát hơn, nhờ thuật toán tìm đường đi ngắn nhất mà việc tìm di chuyển đến từ vị trí này đến vị trí kia một cách hợp lí hiệu quả. Nội dung bài báo cáo sẽ trình bày rõ hơn về thuật toán Dijkstra tìm đường đi ngắn nhất một cách chi tiết nhất, về mặt lý thuyết và cả cài đặt chương trình bằng ngôn ngữ C/C++.

# LỜI CẢM ƠN

Để hoàn thành được bài báo cáo này thì em đã nhận được sự giúp đỡ của GVHD Nguyễn Thủy Đoan Trang rất nhiều. Trong thời gian thực hiện báo cáo thì cô đã luôn quan sát và giúp đỡ em, mỗi tuần cô đều kiểm tra tiến độ thực hiện công việc và thành quả sau mỗi tuần. Mỗi tuần khi em báo cáo luôn nhận được sự quan tâm và góp ý một cách chính xác những khuyết điểm và ưu điểm cách em thực hiện báo cáo. Trong thời gian ấy cô đã giúp em nắm vững thêm kiến thức bổ ích và có tính thực tế cao. Mặc dù bài báo cáo này em đã làm hết khả năng nhưng chắc chắn sẽ không tránh khỏi những khuyết điểm và thiếu xót, mong cô sẽ xem xét và góp ý cho em để hoàn thành báo cáo một cách hoàn chỉnh nhất.

Kính chúc cô sức khỏe, hạnh phúc thành công trên con đường giảng dạy của mình.

# CHƯƠNG I. CƠ SỞ LÝ THUYẾT

## I.1. Lịch sử thuật toán

* Thuật toán Dijkstra, mang tên của nhà khoa học máy tính người Hà Lan Edsger Dijkstra vào năm 1956 và ấn bản năm 1959, là một thuật toán giải quyết bài toán đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng không có cạnh mang trọng số không âm. Thuật toán thường được sử dụng trong định tuyến với một chương trình con trong các thuật toán đồ thị hay trong công nghệ Hệ thống định vị toàn cầu (GPS).
* Đâu là con đường ngắn nhất để đi từ Rotterdam đến Groningen, hay nói chung: từ thành phố này đến thành phố nhất định. Đây là thuật toán tìm đường đi ngắn nhất, mà tôi đã thiết kế trong khoảng hai mươi phút. Vào một buổi sáng đi mua sắm ở Amsterdam cùng vị hôn thê trẻ tuổi với sự mệt mỏi, chúng tôi ngồi trên sân thượng để uống một tách cà phê và tôi chỉ nghĩ liệu tôi có thể làm điều này không, và sau đó tôi đã thiết kế thuật toán tìm đường đi ngắn nhất. Như tôi đã nói, đó là một phát minh trong hai mươi phút. Trên thực tế, nó đã được xuất bản vào năm 1959, ba năm sau đó. Các ấn phẩm vẫn có thể đọc được, quả thật như vậy, và khá đẹp. Một trong những lý do khiến nó rất hay là tôi đã thiết kế nó mà không cần bút chì và giấy. Sau này tôi đã học được rằng một trong những lợi thế của việc thiết kế mà không cần bút chì và giấy là bạn gần như buộc phải tránh đi mọi sự phức tạp có thể tránh được. Cuối cùng, thuật toán đó đã trở thành một trong những nền tảng giúp tôi nổi tiếng, với sự kinh ngạc lớn của tôi. —Edsger Dijkstra trong một cuộc phỏng vấn với Philip L. Frana, Communications of the ACM, 2001.

## I.2. Ý tưởng thuật toán

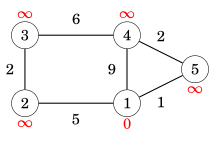
### I.2.1. Điều kiện để có thể áp dụng thuật toán

* Thuật toán Dijkstra là một trong những thuật toán cơ bản dùng để tìm đường đi ngắn nhất từ một điểm tới một điểm nào đó, và mở rộng ra là tìm đường đi ngắn nhất từ 1 điểm tới mọi điểm còn lại của đồ thị, với điều kiện các trọng số của đồ thị đó không âm.
* Ví dụ:

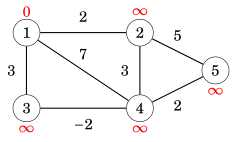
Thuật toán Bellman-Ford cũng là một thuật toán tính các đường đi ngắn nhất nguồn đơn trong một đồ thị có hướng có trọng số (trong đó một số cung có thể có trọng số âm). Thuật toán Dijkstra giải cùng bài toán này tuy nhiên Dijkstra có thời gian chạy nhanh hơn, đơn giản là đòi hỏi trọng số của các cung phải có giá trị không âm.

Hình I.2.1 là đồ thị có tất cả các cung có trọng số không âm

Hình I.2.2 là đồ thị có cung có trọng số âm



*Hình I.2.1*



*Hình I.2.2*

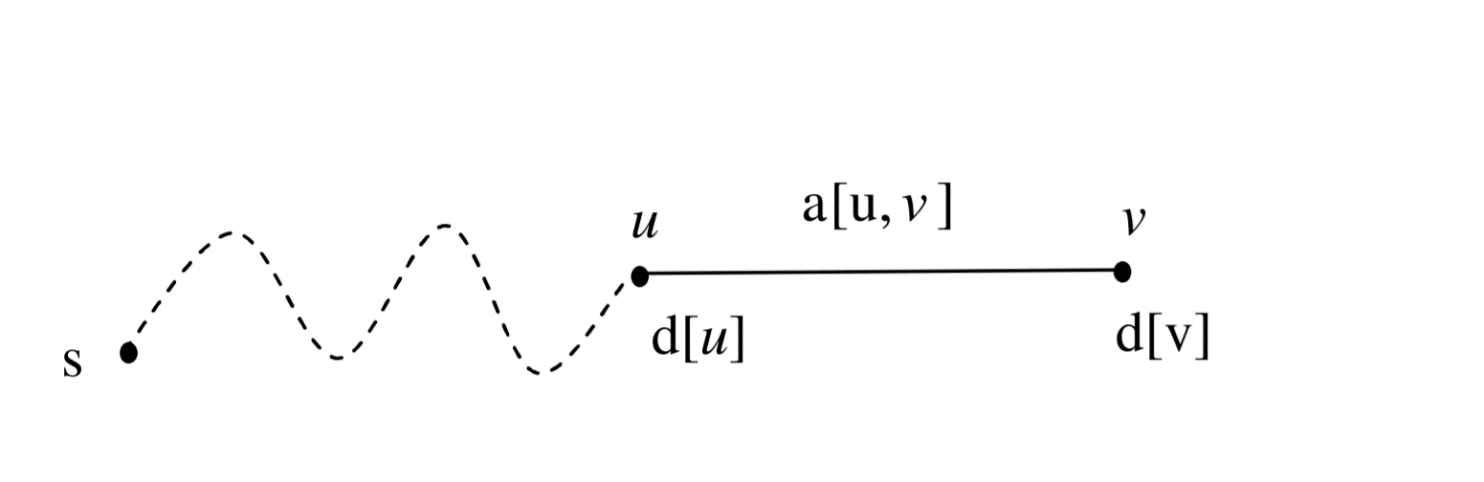
* Lí do mà thuật toán Dijkstra không thể có cung có trọng số âm
* Bởi lẻ thuật toán Dijkstra khi tiến hành duyệt thì sẽ chọn cung có trọng số nhỏ nhất để duyệt tiếp. Ví dụ ở đây nhìn vào (hình I.2.2) ta giả sử điểm bắt đầu là đỉnh **4** thì bước đầu tiên ta sẽ gán nhãn cho các đỉnh như sau với định dạng sau **[tên đỉnh, trọng số].**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Đỉnh | 1 | 2 | 3 | 4 | 5 |
| Khởi tạo | [4,7] | [4,3] | [4,-2] | [4,0] | [4,2] |

Nhìn vào bảng khởi tạo ta có thể thấy rằng nếu xét theo trọng số bé nhất sẽ được chọn để duyệt tiếp thì sẽ chọn đỉnh số **3** chứ không phải chọn đỉnh thứ **4** làm đỉnh xuất phát như mong muốn.

### I.2.2. Các bước thực hiện

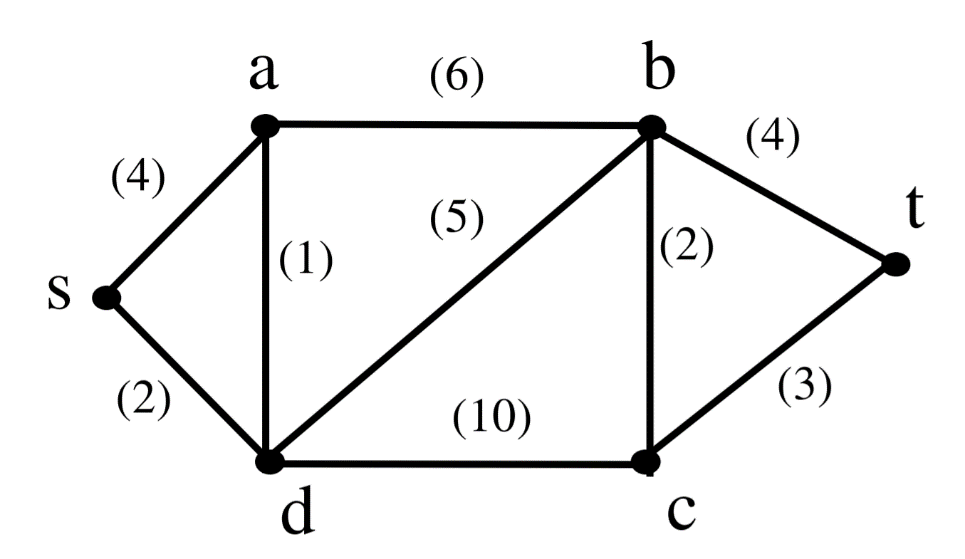
* Đầu tiên là bước khởi tạo nhãn cho các đỉnh. Nhãn của đỉnh được gán tạm thời theo qui tắc: Đỉnh xuất phát s được gắn nhãn bằng 0 (xem như đã cực tiểu hóa), các đỉnh kề với s được gán bằng trọng số tương ứng của cạnh nối đỉnh đó với s, các đỉnh còn lại được gán nhãn là  (một giá trị đủ lớn, chẳng hạn n\*max {a[u,v]: u, vV (G) }).
* Bước tiếp theo là cực tiểu hóa nhãn cho các đỉnh còn lại. Ở mỗi bước lặp, trong danh sách các đỉnh được gán nhãn, chọn đỉnh có nhãn nhỏ nhất (đỉnh này xem như đã cựu tiểu hóa và nhãn của nó chính là độ dài của đường đi ngắn nhất từ đỉnh s đến đỉnh này) để cực tiểu hóa các đỉnh còn lại theo công thức: Giả sử tại đỉnh u đã được cực tiểu hóa nhãn (Hình I.2.1), khi đó đỉnh v kề với u sẽ được gán nhãn là d[v]:= min {d[v], d[u]+a[u,v]}. Trong trường hợp này, đỉnh u được gọi là đỉnh đi trước đỉnh v trên đường đi ngắn nhất từ s đến v qua u . Thuật toán kết thúc khi mọi đỉnh đã được cực tiểu hoá nhãn.

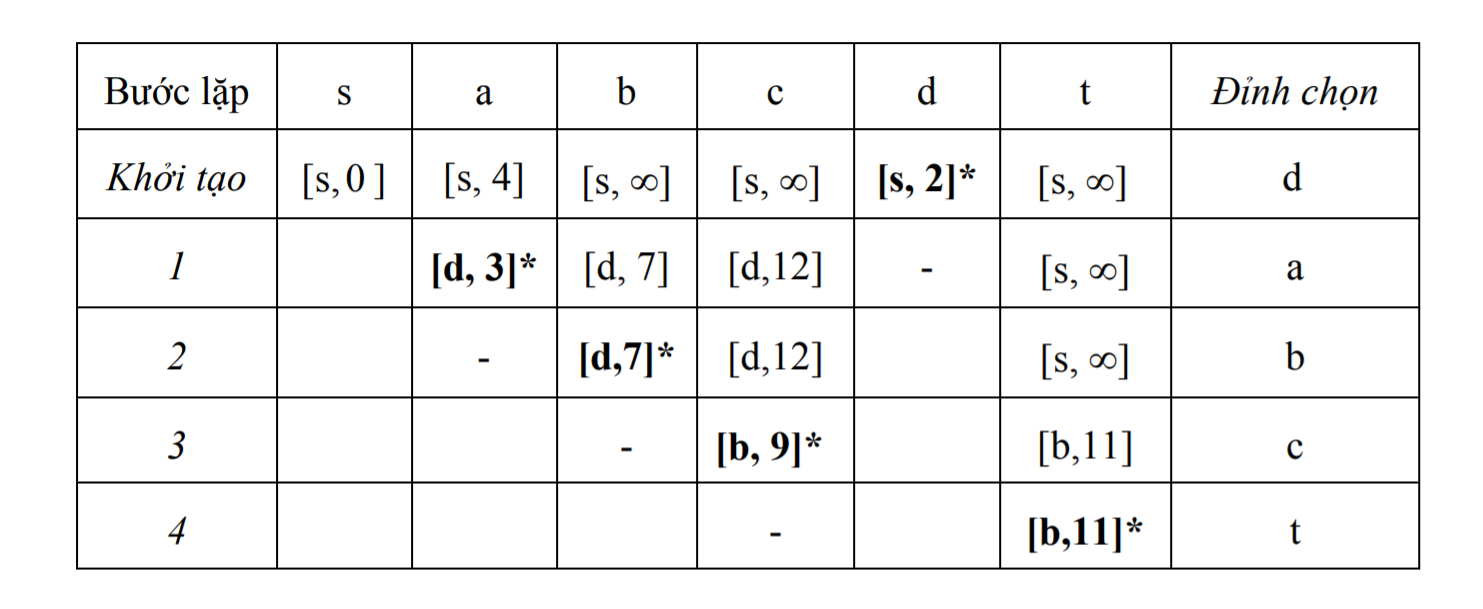


*Hình I.2.1*

## I.3. Giải quyết bài toán qua ví dụ

**Vấn đề đặt ra** : Tìm đường đi ngắn nhất từ ***s*** đến ***t*** sử dụng thuật toán Dijkstra





Trong đó, mỗi ô trong bảng gồm hai thông số, thông

số đầu chỉ tên đỉnh trung gian của đường đi từ s đến đỉnh tương ứng của ô, thông số sau là

nhãn của đỉnh đó. Ví dụ, ô [d, 7] ở bước lặp 1 có ý nghĩa là độ dài đường đi xuất phát từ s

đến b có độ dài là 3 và trước khi qua b phải đi qua đỉnh d. Ô [d, 3]\* cho biết độ dài đường

đi ngắn nhất từ s đến a là 3 và trước khi qua đỉnh a phải đi qua đỉnh d.

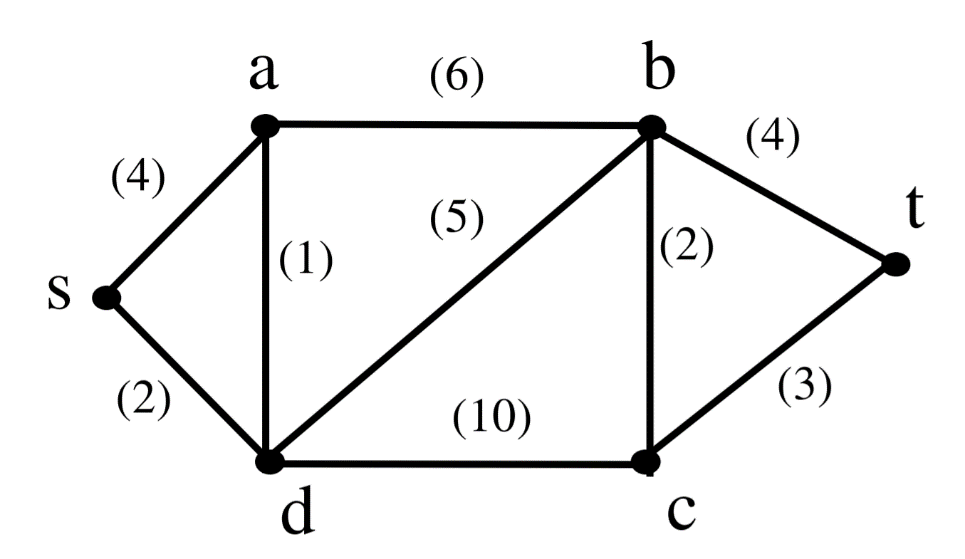
Qua bài giải trên có thể thấy rằng đường đi ngắn nhất từ s 🡪 t là : s 🡪 d 🡪 b 🡪 t

# CHƯƠNG II. THIẾT KẾ CHƯƠNG TRÌNH

## II.1. Dữ liệu đầu vào

* Dữ liệu đầu vào của chương trình sẽ danh sách các cạnh, có dạng (u,v)=w, với u,v là 2 đỉnh của cạnh và w là trọng số của cạnh.

Ví dụ :



* Với định dạng đầu vào như trên thì ta sẽ có các cạnh với trọng số sau:

+ (s,a) = 4

+ (s,d) = 2

+ (a,b) = 6

+ (a,d) = 1

+ (b,d) = 5

+ (b,c) = 2

+ (b,t) = 4

+ (c,d) = 10

+ (c,t) = 3

* Cùng với đó để tiện trong việc xử lý thì cần có thông tin thêm về tổng số cạnh, số đỉnh.
* Thay vì chúng ta sẽ nhập (s,a)=4, thì ta nên đại diện cho các kí tự đỉnh bằng những con số (s=1,a=2,b=3,c=4,d=5,t=6) để dễ dàng nhập dữ liệu và thuận tiện cho việc nhập dữ liệu sau này.
* File INPUT.IN của sẽ có dạng như sau :

+ Dòng 1 : gồm 4 chữ số đó là số đỉnh n và số cạnh m có trong đồ thị, điểm bắt đầu, điểm kết thúc.

+ m dòng tiếp theo đó là gồm thông tin các 2 đỉnh và trọng số

Ví dụ ta dùng đồ thị trên làm dữ liệu INPUT.IN

6 9 1 6

1 2 4

1 5 1

2 3 6

2 5 1

3 5 5

3 4 2

3 6 4

4 5 10

4 6 3

## II.2. Tổ chức lưu trữ, các biến dữ liệu

### II.2.1. Các biến khai báo trong chương trình

+ int n; // lưu trữ tổng số đỉnh của đồ thị

+ int m; // lưu trữ tổng số cạnh của đồ thị

+ int s; // lưu trữ đỉnh bắt đầu

+ int t; // lưu trữ đỉnh kết thúc

+ int nhan[100]; // mảng dùng để lưu trữ gán nhãn và đánh dấu đường đi

+ int kc[100]; // mảng lưu trữ khoảng cách đã đi qua

+ int xet[100]; // mảng lưu trữ để xem xét các đỉnh đã xét hay chưa

### II.2.2. Định nghĩa một số giá trị mặc định

+ #define CHUA 0 // nó dùng để làm rõ giá trị các đỉnh đã xét hay chưa (ở đây nếu chưa thì nó sẽ là 0)

+ #define ROI 1 // nó cũng để dùng làm rõ giá trị các đỉnh đã xét hay chưa (ở đây là rồi thì nó sẽ là 1), hai định nghĩa này chỉ dùng trong gán giá trị của mảng **xet[100]**.

+ #define VC 1000000 // được xem là giá trị vô cùng lúc khởi tạo.

### II.2.3 Các hàm chức năng xử lý từng nhiệm vụ riêng

* Chúng ta sẽ có tất cả 3 nhiệm chính cần giải quyết như sau:

+ Thứ nhất là đọc dữ liệu từ file INPUT.IN

+ Thứ hai là tìm đường đi ngắn nhất với thuật toán Dijkstra.

+ Thứ ba là in ra kết quả đường đi ngắn nhất vừa tìm được.

Nội dung từng phần như sau

* **Đọc dữ liệu từ file.**
* Sau khi định dạng được dữ liệu đầu vào theo nguyên tắc đã quy ước ở trên thì ta tiến hành như sau:

void DocFile()

{

int tam[100][100];

FILE \*fp = fopen("DIJKSTRA\_IN.txt","r");

fscanf(fp,"%d%d%d%d",&n,&m,&s,&t);

for(int i=1;i<=m;i++)

for(int j=1;j<=3;j++)

fscanf(fp,"%d",&tam[i][j]);

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

mt[i][j]=VC;

for(int i=1;i<=m;i++)

{

mt[tam[i][1]][tam[i][2]]=tam[i][3];

mt[tam[i][2]][tam[i][1]]=tam[i][3];

}

fclose(fp);

}

+ Đầu tiên với các số biến **n,m,s,t** được lấy từ file và gán giá trị cho chúng.

+ Tiếp đến với khai báo danh sách biến **tam[100][100]** với biến **tam** này thì **tam[][1]** sẽ lưu trữ đỉnh đầu tiên, **tam[][2]** sẽ lưu trữ đỉnh thứ hai, **tam[][3]** để lưu trữ trọng số và các dữ liệu này được lấy từ các dòng tiếp theo trong file **INPUT.IN**

+ Công việc cuối cùng là để chuyển danh sách các cạnh thành ma trận n\*n và những cạnh không có giá trị thì sẽ được gán bằng giá trị **VC** (1000000).

+ Độ phức tạp của hàm này O(n2) với n là số đỉnh của đồ thị.

* **Tìm đường bằng thuật toán Dijkstra**

void Dijkstra()

{

int i, min;

for(int j=1;j<=n;j++)

{

kc[j]=mt[s][j];

nhan[j]=s;

xet[j]=CHUA;

}

nhan[s]=0;

kc[s]=0;

xet[s]=ROI;

+ Bước này gọi là bước khởi tạo các giá trị bắt đầu từ đỉnh **s**

+ Tất cả các đỉnh sẽ được gán nhãn bằng **s**

+ Tất nhiên là đỉnh s sẽ là đỉnh được xét đầu tiên **xet[s] = ROI**

while(!xet[t])

{

min=VC;

for(int j=1;j<=n;j++)

{

if(!xet[j] && min > kc[j])

{

i=j;

min=kc[j];

}

}

xet[i]=ROI;

if(!xet[t])

{

for(int j=1;j<=n;j++)

{

if(!xet[j] && (kc[i] + mt[i][j] < kc[j]))

{

kc[j] = kc[i] + mt[i][j];

nhan[j] = i;

}

}

}

}

}

+ Tiến hành xét những đỉnh mà chúng ta chưa xét

+ Tìm ra đỉnh có khoảng cách bé nhất đến đỉnh **s**

+ Nếu chưa xét đến đỉnh **t** thì tiến hành so sánh gán lại nhãn cho các đỉnh

+ Nếu đã xét đến **t** thì dừng vòng lặp

+ Độ phức tạp của hàm này là O(n2) với n là số đỉnh của đồ thị.

* **In ra kết quả**

void GhiFile()

{

FILE \*fp = fopen("DIJKSTRA\_OUT.txt","w");

int i=nhan[t];

int tam[100];

int sodinh=2;

while(i!=s)

{

tam[sodinh]=i;

i=nhan[i];

sodinh++;

}

fprintf(fp,"%d %d\n",sodinh,kc[t]);

fprintf(fp,"%d ",s);

for(int j=sodinh-1;j>=2;j--)

{

fprintf(fp,"%d ",tam[j]);

}

fprintf(fp,"%d",t);

fclose(fp);

}

+ Kết quả đầu ra là một file DIJKSTRA\_OUT.txt với các thành phần sau :

Dòng 1 gồm 2 số. Đầu tiên số đỉnh mà đoạn đường ngắn nhất đi qua (bao gồm cả đỉnh bắt đầu và kết thúc). Số thứ hai là độ dài đoạn đường ngắn nhất.

Dòng 2 là các đỉnh mà đoạn đường đó đi qua từ đỉnh bắt đầu đến đỉnh kết thúc.

+ Độ phức tạp của hàm này là O(n) với n là số đỉnh của đồ thị.

## II.3 . Chương trình hoàn thiện

#include <stdio.h>

#define CHUA 0

#define ROI 1

#define VC 1000000

int n;

int m;

int s,t;

int nhan[100];

int kc[100];

int mt[100][100];

int xet[100];

void DocFile()

{

int tam[100][100];

FILE \*fp = fopen("DIJKSTRA\_INPUT.txt","r");

fscanf(fp,"%d%d%d%d",&n,&m,&s,&t);

for(int i=1;i<=m;i++)

for(int j=1;j<=3;j++)

fscanf(fp,"%d",&tam[i][j]);

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

mt[i][j]=VC;

for(int i=1;i<=m;i++ )

{

mt[tam[i][1]][tam[i][2]]=tam[i][3];

mt[tam[i][2]][tam[i][1]]=tam[i][3];

}

fclose(fp);

}

void InKQ()

{

printf("\nDuong di ngan nhat tu dinh %d den dinh %d la\n",s,t);

printf("%d <= ",t);

int i = nhan[t];

while(i != s)

{

printf("%d <= ",i);

i = nhan[i];

}

printf("%d",s);

printf("\nDo dai doan duong la : %d",kc[t]);

}

void Dijkstra()

{

int i, min;

for(int j=1;j<=n;j++)

{

kc[j]=mt[s][j];

nhan[j]=s;

xet[j]=CHUA;

}

nhan[s]=0;

kc[s]=0;

xet[s]=ROI;

while(!xet[t])

{

min=VC;

for(int j=1;j<=n;j++)

{

if(!xet[j] && min > kc[j])

{

i=j;

min=kc[j];

}

}

xet[i]=ROI;

if(!xet[t])

{

for(int j=1;j<=n;j++)

{

if(!xet[j] && (kc[i] + mt[i][j] < kc[j]))

{

kc[j] = kc[i] + mt[i][j];

nhan[j] = i;

}

}

}

}

}

int main()

{

DocFile();

Dijkstra();

GhiFile();

return 0;

}

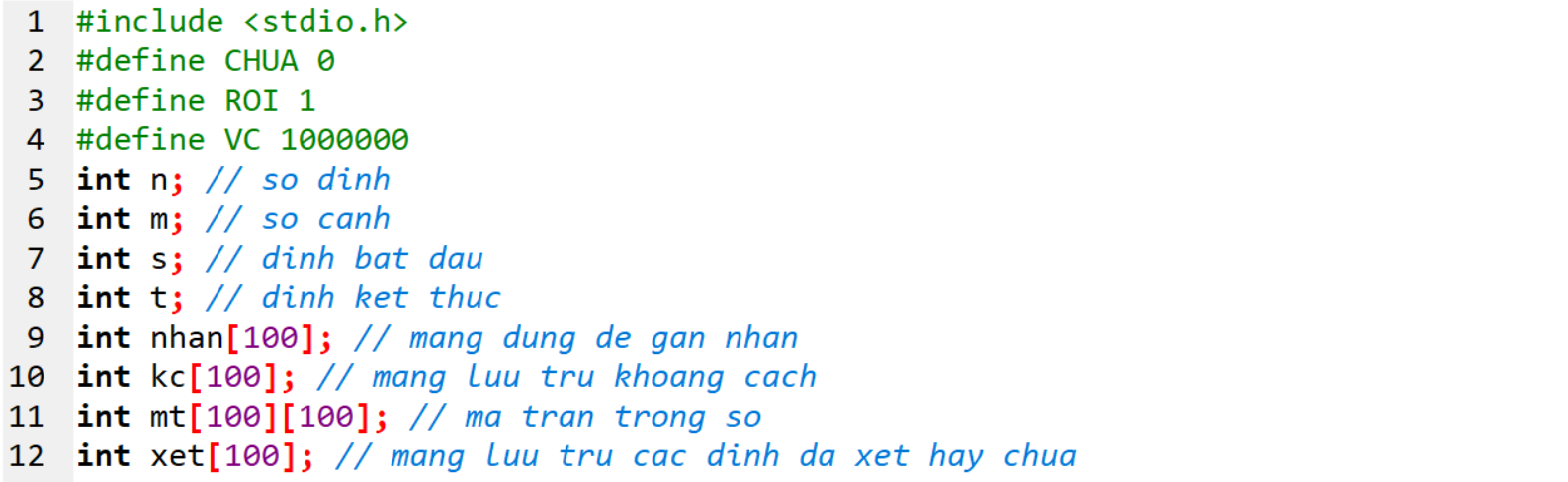
* Từ hàm main() ta có tính được độ phức tạp của đoạn chương trình như sau:

Max(DocFile(), Dijkstra(), GhiFile()) = Max(O(n2), O(n2), O(n)) = O(n2).

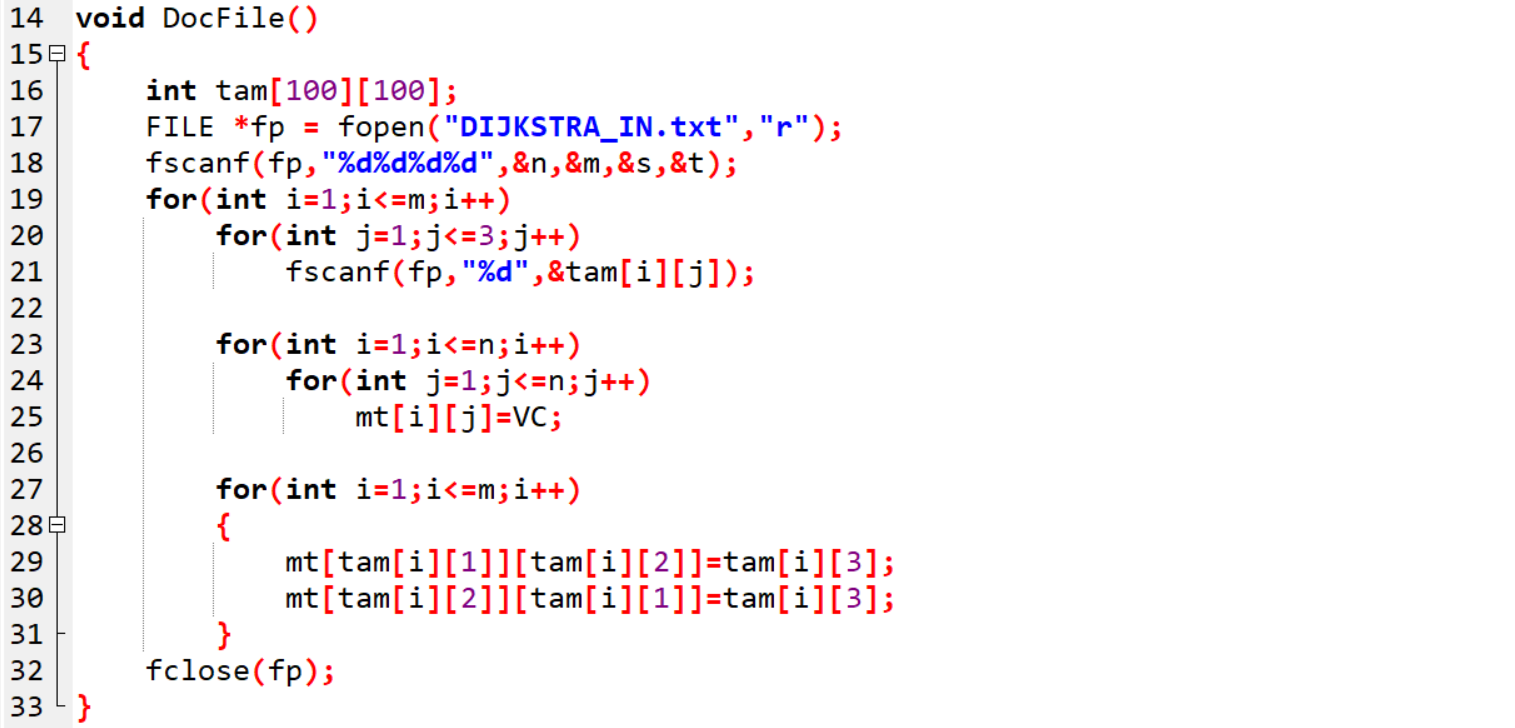
# CHƯƠNG III. THỰC THI CHƯƠNG TRÌNH

Chương trình dưới đây sẽ được cài đặt trên DevC/C++.

**Bước 1 : Khai báo các thư viện cần thiết, biến toàn cục và các định nghĩa được dùng trong chương trình.**

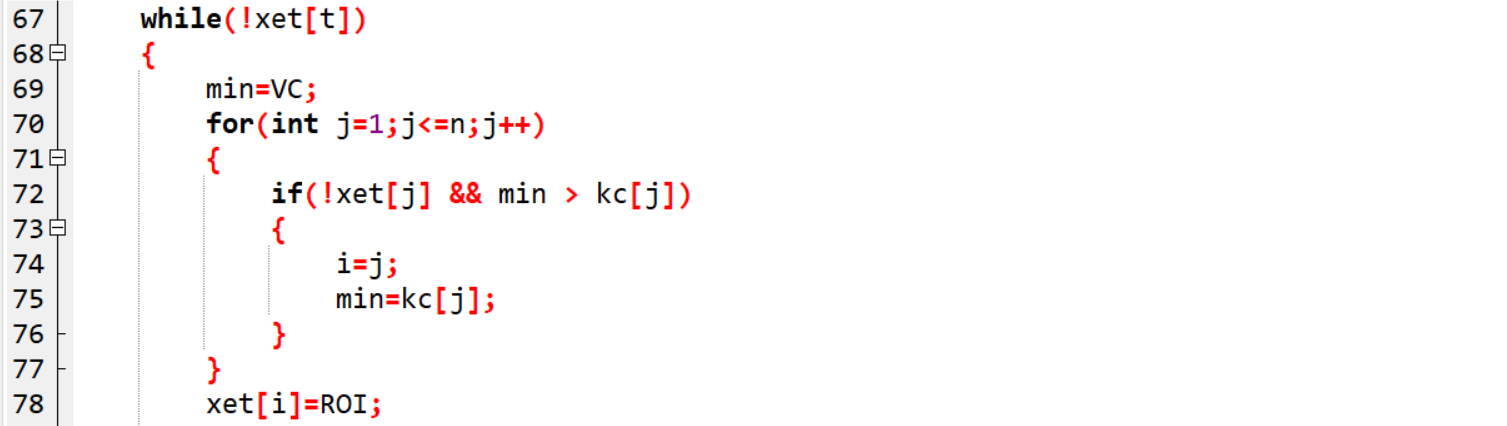


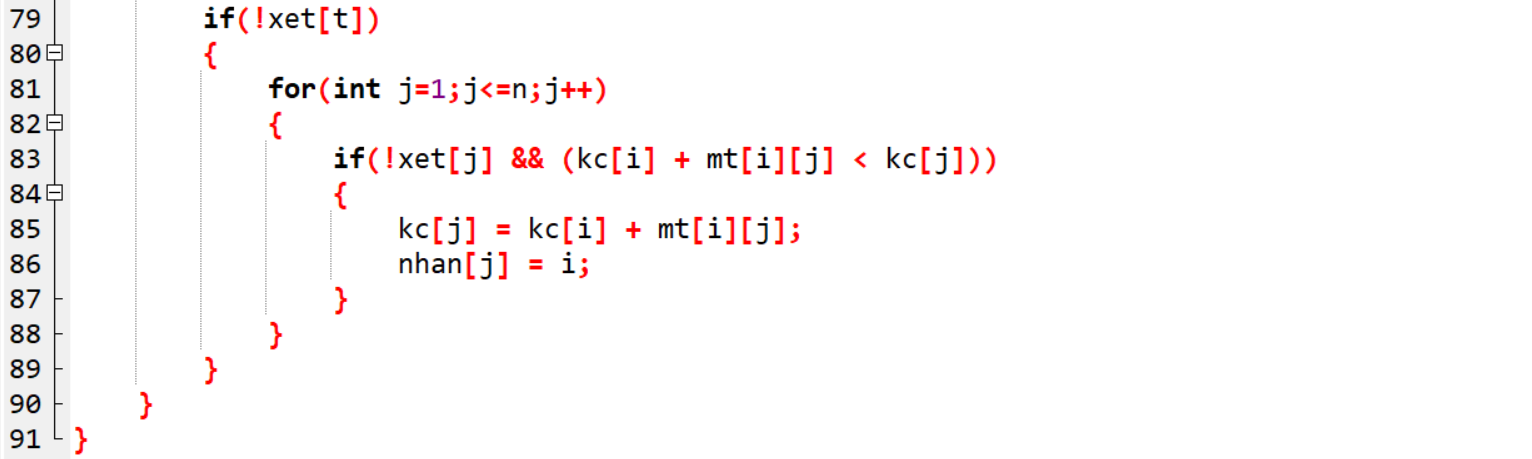
**Bước 2 : Tạo hàm có chức năng đọc file và khởi tạo giá trị cho các biến đã khởi tạo ở trên.**



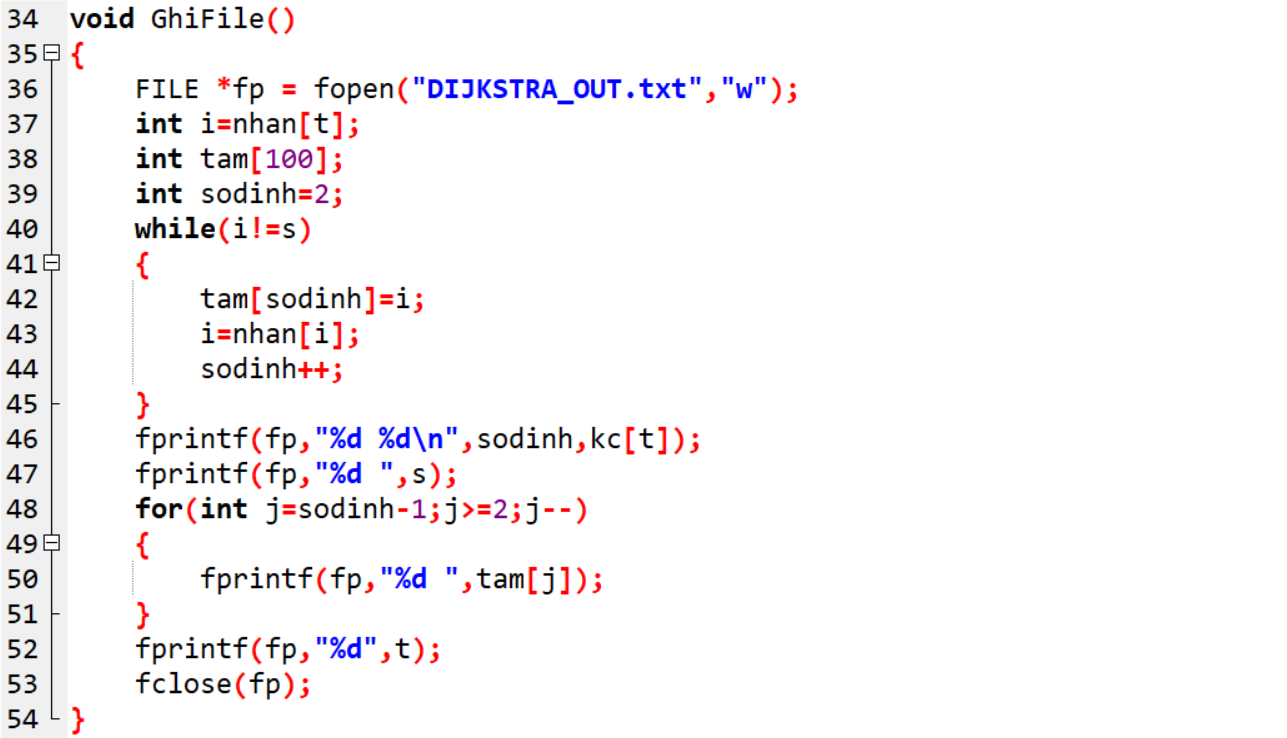
**Bước 3 : Tạo hàm có chức năng tìm ra con đường ngắn nhất dựa trên thuật toán Dijkstra.**







**Bước 4 : Tạo hàm có chức năng ghi file - xuất kết quả ra file output**

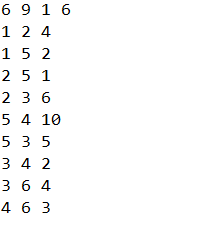


**Bước 5 : Tạo hàm int main() – hàm thực thi chính**

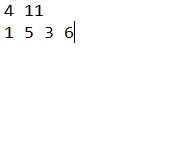


**Bước 6 : Xem kết quả sau khi thực hiện chương trình**

* Với file đầu vào như sau



* Thì kết quả đầu ra sau khi chạy chương trình như sau



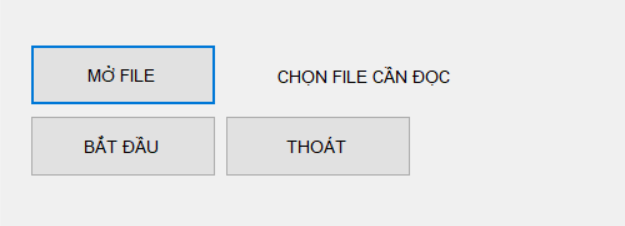
# CHƯƠNG IV. MÔ PHỎNG CHƯƠNG TRÌNH BẰNG ĐỒ HỌA

## IV.1. Giới thiệu về Windows Forms

* Windows Forms là cách cơ bản để cung cấp các thành phần giao diện (GUI components) cho môi trường .NET Framework.
* Windows Forms được xây dựng trên thư viện Windows API.
* Windows Forms cơ bản bao gồm.
* Một Form là khung dùng hiển thị thông tin đến người dùng..
* Các Control được đặt trong form và được lập trình để đáp ứng sự kiện
* Windows Forms được viết bằng ngôn ngữ C#.

## IV.2. Cấu trúc về giao diện sẽ mô phỏng bài toán

* Chức năng :
* Cho phép người dùng mở file chứa dữ liệu dưới dạng .txt
* Cho phép người dùng bắt đầu khi đã có file dữ liệu.
* Hiển thị đồ thị tượng trưng khi bắt hoàn thành xong các bước giải.
* Làm mới đồ thị sau khi hiển thị ra.
* Các thành phần trong giao diện
* Các nút chức năng



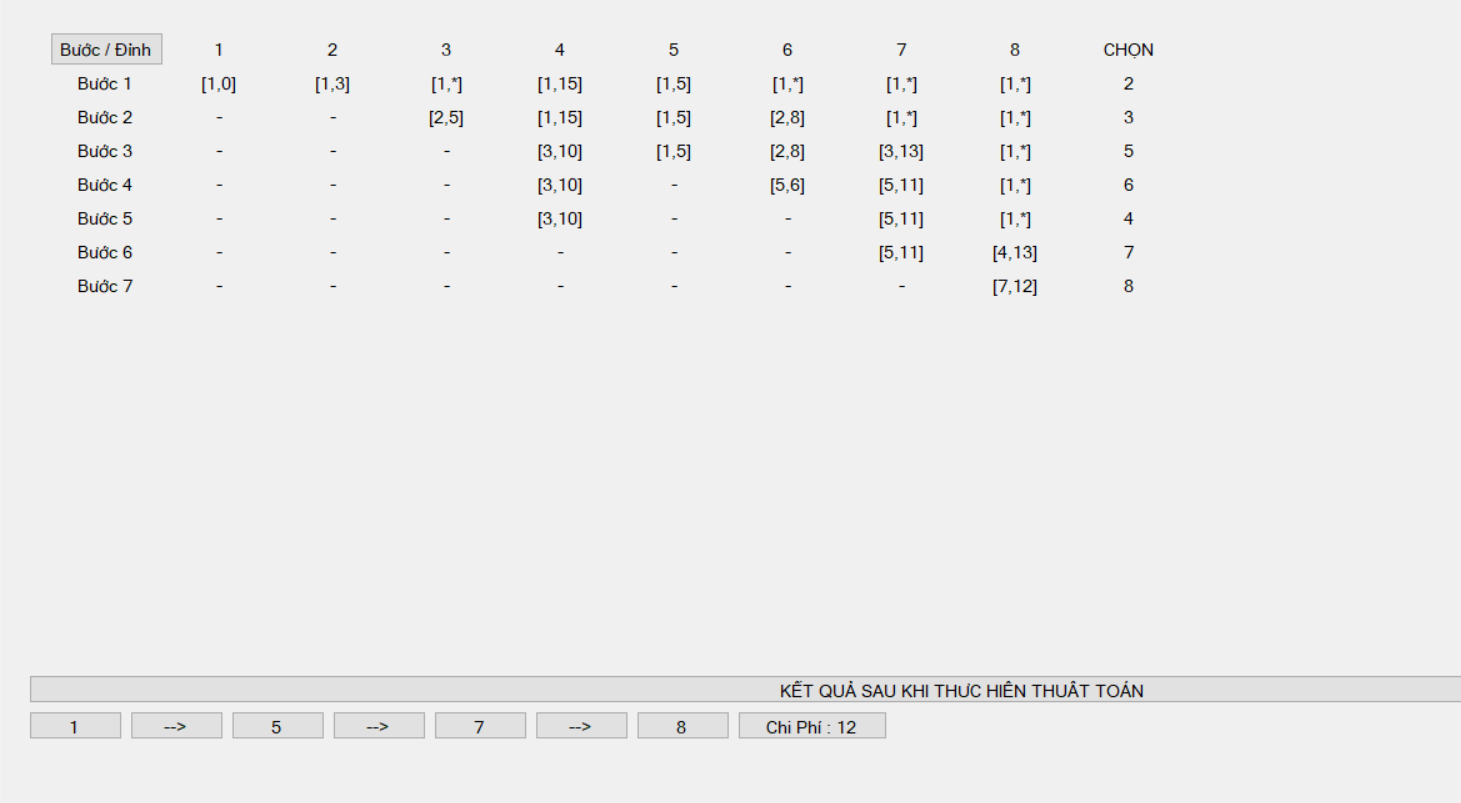
Gồm các button :

+ MỞ FILE : khi click vào sẽ tiến hành truy cập đến nguồn tài nguyên (các file dữ liệu) trong máy của bạn. Lưu ý nó chỉ nhận các file text (có đuôi .txt)

+ BẮT ĐẦU : khi click vào chương trình sẽ bắt đầu thực thi, nhưng nó chỉ có tác dụng khi đã có file dữ liệu.

+ THOÁT : dùng để thoát chương trình.

* Panel hiển thị thông tin



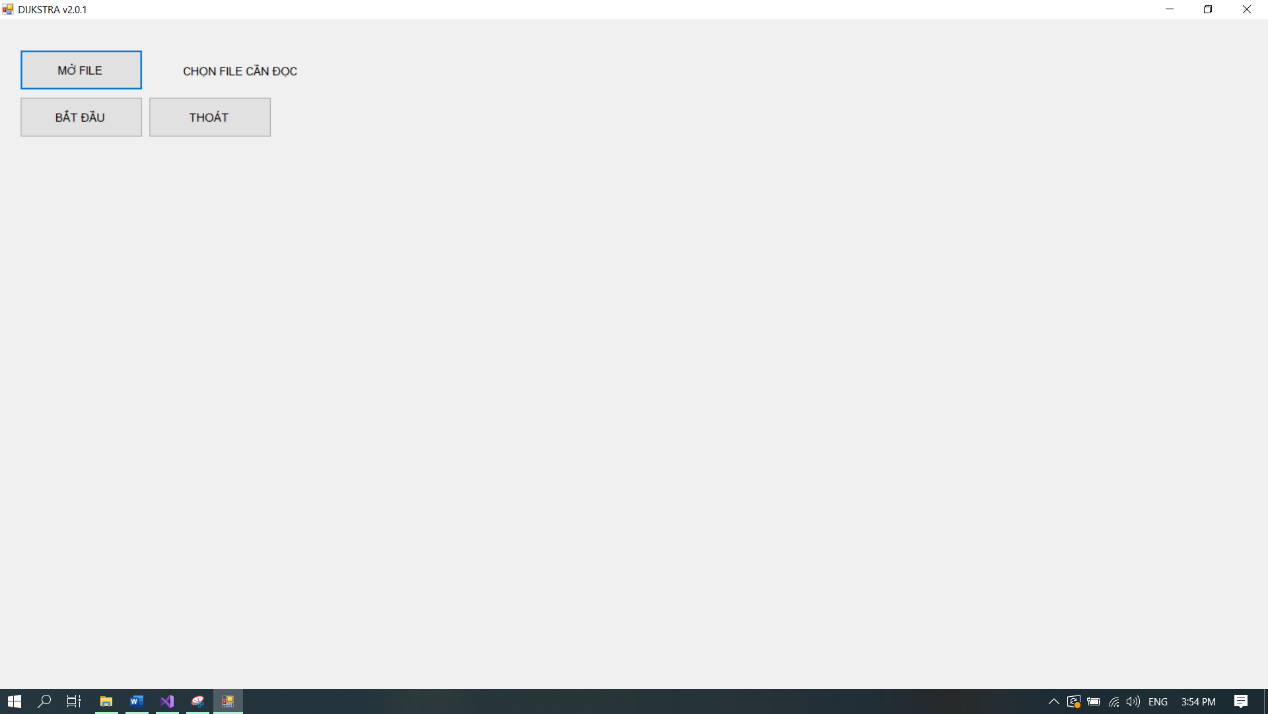
+ Bên dưới các button chức năng sẽ là 2 panel hiển thị thông tin của thuật toán.

+ Panel hiển thị các bước tìm đường ngắn nhất : nằm bên dưới các button chức năng, nó chỉ hiển thị sau khi click vào button bắt đầu. Nó sẽ hiển thị ra một bảng gồm các bước và dữ liệu lưu trữ trong từng bước mà thuật toán làm.

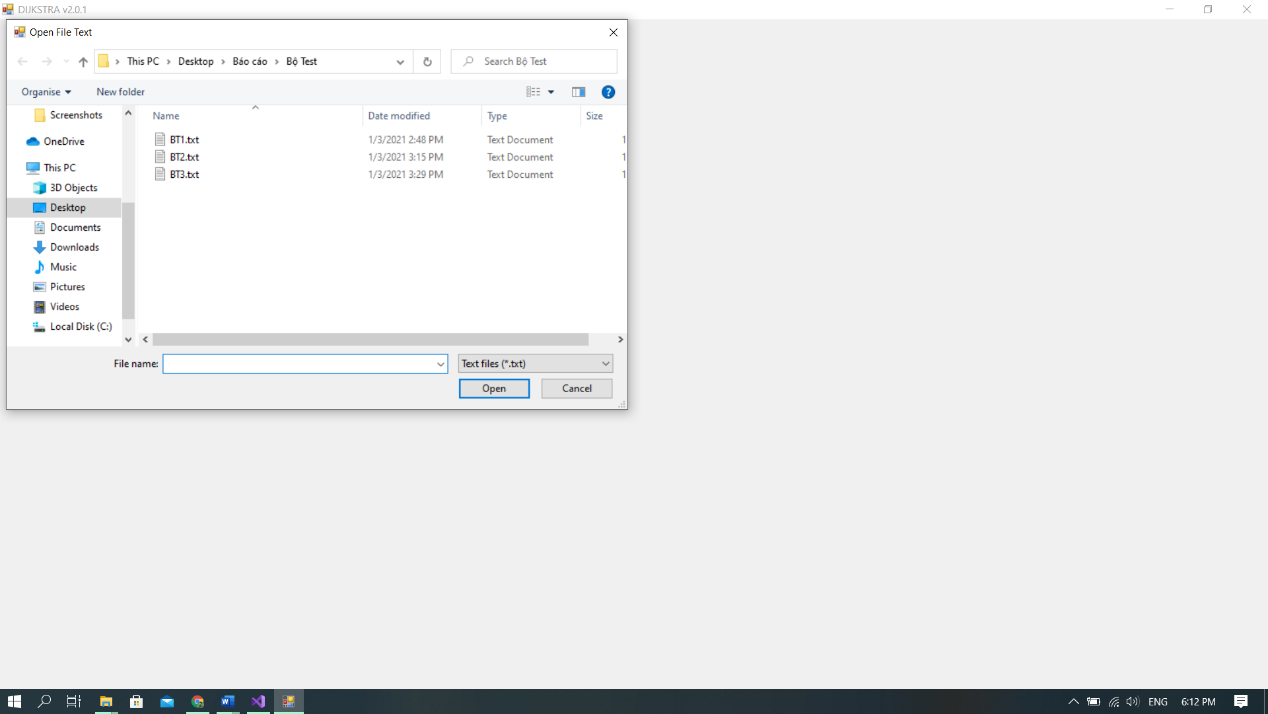
+ Panel hiển thị kết quả : nằm dưới cùng của màn hình ứng dụng, nó xuất hiện sau khi click vào button bắt đầu. Nó sẽ hiển thị thứ tự xuất phát và tổng chi phí bé nhất để đi đến điểm yêu cầu.

* Cách thức sử dụng ứng dụng

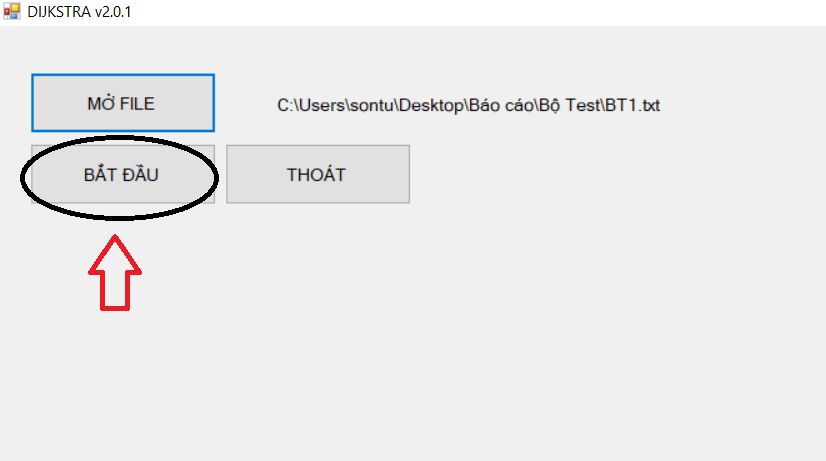
Bước 1 : Mở ứng dụng và chờ nó hiển thị lên giao diện



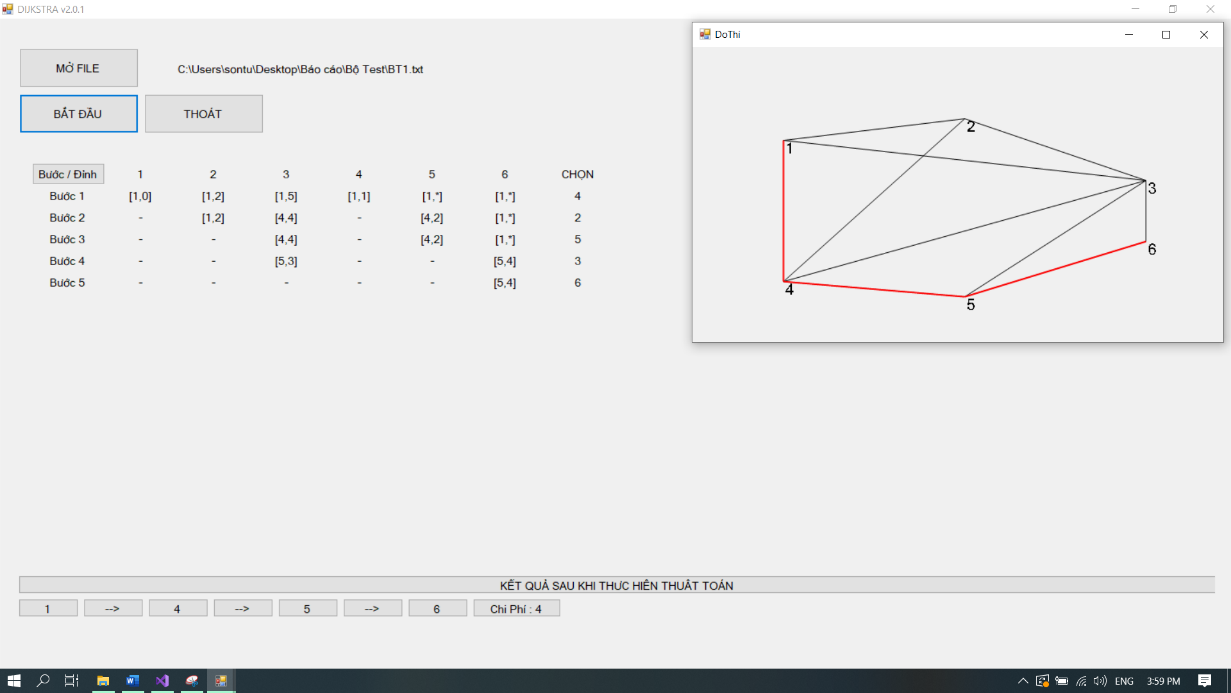
Bước 2 : Click vào chọn button **MỞ FILE** và chọn file dữ liệu đã chuẩn bị sẵn (phải đúng cấu trúc đầu vào quy định như ở trên)



Bước 3 : Sau khi chọn file dữ liệu hợp lệ thì bên cạnh button **MỞ FILE** sẽ hiển thị đường dẫn của file dữ liệu bạn đang dùng. Tiếp đến thì click vào button **BẮT ĐẦU** để thực thi chương trình và chờ kết quả.



Bước 4 : Kết quả sẽ hiển thị ra thêm một cửa sổ mới có chức năng hiển thị đồ thị tượng trưng và nếu gặp trường hợp đồ thị hiển thị ra khó nhìn thì chúng ta nhấn nút R trên bàn phím để thay đổi lại đồ thị cho dễ nhìn.



Chúng ta có thể thay đổi file dữ liệu khác bằng cách click vào button **MỞ FILE** và làm lại như bước 2. Sau khi đã thực hiên xong ta click vào button thoát để thoát chương trình.

# Chương V. Kết luận và hướng phát triển

## V.1. Những điểm đạt được

* Kiến thức được nâng cao.
* Kinh nghiệm xây dựng và giải quyết một bài toán.
* Tiếp cận được môi trường phát triển ứng dụng trên windows.
* Hiểu rõ được thuật toán và có thể áp dụng trong thực tế.

## V.2. Những điểm chưa đạt được

* Chương trình mô phỏng đồ họa còn khá sơ sài.

## V.3. Hướng phát triển trong tương lai

* Thiết kế lại giao diện hoàn thiện hơn, đầy đủ chức năng và thân thiện với người dùng.
* Đưa nó vào giải quyết thực tế trên bản đồ số hoặc phần mềm quản lý thông tin địa lý (ví dụ : ArcGis, GisOnline,…).
* Lập trình thuật toán theo hướng đối tượng.

## V.4. Kết luận

Trong suốt quá trình thực hiện thực tập cơ sở thì em nhận thấy rằng khả năng làm việc, kiến thức được phát triển và được bổ sung rất nhiều. Trong những tuần qua em được trải nghiệm được những kiến thức mới, nâng cao khả năng tự học, kích thích tinh thần muốn khám phá, tìm hiểu và vượt qua khó khăn. Bài toán tìm đuòng đi ngắn nhất là bài toán khá phổ biến hiện nay, do cùng với sự phát triển mạnh mẽ của GIS (Hệ thống thông tin địa lý) trong mọi lĩnh vực. Qua khoảng thời gian tìm hiểu trên, đối với em nó không còn là vấn đề nửa. Ngoài ra khi tiếp cận với một bài toán thực tế như vậy, làm cho em nhận ra được những điểm yếu trong quá trình thực hiện, đó cũng chính là kinh nghiệm quý báu trên con đường sau này của em. Một lần nữa em xin cảm ơn chân thành đến giảng viên hướng dẫn đã hướng dẫn em rất nhiệt tình và sự giúp đở em rất nhiều.

## V.5. Tài liệu tham khảo sử dụng trong toàn bộ bài báo cáo

* Toán rời rạc – TS. Đỗ Như An
* Cấu trúc dữ liệu và giải thuật – TS. Nguyễn Đức Thuần
* https://drive.google.com/drive/folders/1N0Y7sKAufaYNw-XlnHLu0kUkW5KUlVFK?usp=sharing