**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO THỰC TẬP CƠ SỞ**

**Lập trình mô phỏng thuật toán Dijkstra**

**tìm đường đi ngắn nhất**

**Giảng viên hướng dẫn : Nguyễn Đình Hưng**

**Sinh viên thực hiện : Nguyễn Khánh Duy**

**Lớp : 61-CNTT-1**

**MSSV : 61133540**

**Khánh Hòa, tháng 1/2022**

**MỤC LỤC**

[**DANH MỤC HÌNH VẼ** 4](#_Toc92349006)

[**LỜI CẢM ƠN** 5](#_Toc92349007)

[**LỜI NÓI ĐẦU** 6](#_Toc92349008)

[**I)** **Giới thiệu** 7](#_Toc92349009)

[**II)** **Thuật toán** 7](#_Toc92349010)

[**1.** **Giới thiệu giải thuật** 7](#_Toc92349011)

[**2.** **Ý tưởng của thuật toán** 7](#_Toc92349012)

[**3.** **Bài toán tổng quát** 8](#_Toc92349013)

[**4.** **Mô tả thuật toán Dijkstra** 9](#_Toc92349014)

[**a.** **Các bước trong thuật toán Dijkstra** 9](#_Toc92349015)

[**b.** **Ví dụ thuật toán Dijkstra** 9](#_Toc92349016)

[**III)** **Cài đặt bài toán** 10](#_Toc92349017)

[**1.** **Thiết kế thuật toán Dijkstra** 10](#_Toc92349018)

[**2.** **Giải thuật thô** 11](#_Toc92349019)

[**3.** **Độ phức tạp thuật toán** 11](#_Toc92349020)

[**4.** **Minh họa thuật toán Dijkstra** 11](#_Toc92349021)

[**IV)** **Kết quả thực hiện** 12](#_Toc92349022)

[**1.** **Cửa sổ giao diện** 12](#_Toc92349023)

[**2.** **Source code** 15](#_Toc92349024)

[**a.** **Các thư viện khai báo** 15](#_Toc92349025)

[**b.** **Các hàm tạo giao diện** 15](#_Toc92349026)

[**c.** **Chức năng trên đồ thị** 15](#_Toc92349027)

[**d.** **Xử lý thuật toán** 16](#_Toc92349028)

[**e.** **Nhập đỉnh và trọng số** 16](#_Toc92349029)

[**f.** **Định dạng kiểu chữ và màu sắc khi xuất ra** 16](#_Toc92349030)

[**g.** **Mục menu** 17](#_Toc92349031)

[**3.** **Các hàm xử lý thuật toán** 17](#_Toc92349032)

[**4.** **Các trường hợp kết quả** 19](#_Toc92349033)

[**a.** **Không có đường đi từ đỉnh đầu đến đỉnh cuối** 19](#_Toc92349034)

[**b.** **Nhập trọng số âm** 20](#_Toc92349035)

[**c.** **Chưa có đồ thị mà kích vào các nút thao tác trên đồ thị** 21](#_Toc92349036)

[**V)** **Ngôn ngữ và môi trường phát triển:** 22](#_Toc92349037)

[**1.** **Ngôn ngữ sử dụng** 22](#_Toc92349038)

[**2.** **Môi trường phát triển** 22](#_Toc92349039)

[**VI)** **Tài liệu tham khảo** 22](#_Toc92349040)

# **DANH MỤC HÌNH VẼ**

[Hình 1: Bài toán thực tiễn](#h1)

[Hình 2: Mô tả bằng đồ thị](#h3)

[Hình 3: Ví dụ thuật toán dijkstra](#h2)

[Hình 4: Đồ thị mô tả](#h4)

[Hình 5: Bước thực hiện](#h5)

[Hình 6: Cửa sổ chính](#h6)

[Hình 7: Cửa sổ thuật toán dijkstra](#h7)

[Hình 8: Thao tác file](#h8)

[Hình 9: Cửa sổ ma trận](#h9)

[Hình 10: Các nút thao tác và đồ thị](#h10)

[Hình 11: Hiện thi giúp đỡ](#h11)

[Hình 12: Kết quả khi chạy](#h12)

[Hình 13: Thư viện](#h13)

[Hình 14: Hàm tạo cửa sổ](#h14)

[Hình 15: Các chức năng](#h15)

[Hình 16: Xử lý thuật toán dijkstra](#h16)

[Hình 17: Nhập đỉnh](#h17)

[Hình 18: Định dạng kiểu và màu](#h18)

[Hình 19: Tổng hợp menu](#h19)

[Hình 20: Khởi chạy thuật toán](#h20)

[Hình 21: Thuật toán Dijkstra](#h21)

[Hình 22: Kết quả sau khi chạy](#h22)

[Hình 23: Kết quả không có đường đi từ 1 đến 3](#h23)

[Hình 24: Kết quả không có đường đi từ 2 đến 1](#h24)

[Hình 25: Trọng số âm](#h25)

[Hình 26: Chưa có đồ thị](#h26)

# **LỜI CẢM ƠN**

Lời đầu tiên em xin chân thành cảm ơn thầy Nguyễn Đình Hưng, Khoa Công Nghệ Thông Tin Trường Đại Học Nha Trang đã giúp đỡ, củng cố kiến thức và chỉ bảo nhiệt tình cho em.

Mặc dù em đã có sự cố gắng, nhưng trong khoảng thời gian cho phép cũng như hạn chế về kiến thức, nên báo cáo thực tập cơ sở này của em không thể tránh khỏi những kiến thức thiếu sót. Chính vì vậy, em rất mong nhận được sự góp ý của các thầy cô giáo khi chấm bài của em.

Em xin chân thành cảm ơn quý thầy cô giáo!

Sinh viên thực hiện

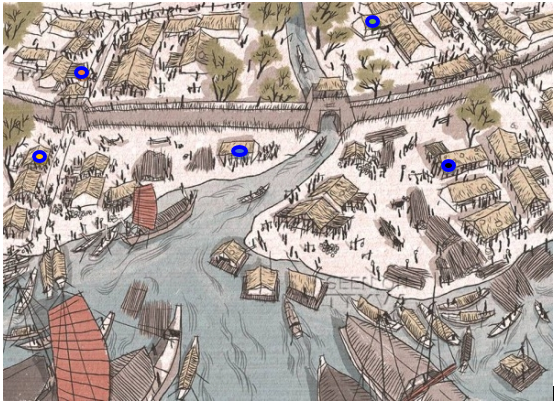
**Nguyễn Khánh Duy**

# **LỜI NÓI ĐẦU**

Trong đời sống, chúng ta thường gặp những tình huống như sau:để đi từ địa điểm A đến địa điểm B, có nhiều đường đi, nhiều cách đi; có lúc ta chọn đường đi ngắn nhất (theo nghĩa cự ly), có lúc lại cần chọn đường đi nhanh nhất ( theo nghĩa thời gian) và có lúc phải cân nhắc để chọn đường đi rẻ tiền nhất (theo nghĩa chi phí),…

Toàn bộ mã nguồn của chương trình được tải lên theo địa chỉ:

<https://github.com/NguyenKhanhDuy612/ThucTapCoSo>

 Hình 1: Bài toán thực tiễn

1. **Giới thiệu**

Bài toán tìm đường đi ngắn nhất trên đồ thị là một trong những bài toán đa dạng, có nhiều ứng dụng thực tế (như trong Google Maps, hay các bài toán networking,…).

Có ba thuật toán cơ bản của dạng bài tìm đường đi ngắn nhất:

Thuật toán Bellman - Ford.

Thuật toán Dijkstra.

Thuật toán Floyd-Warshall (còn gọi là thuật toán Floyd).

Trong đề tài này thì sẽ nói về thuật toán Dijkstra.

**Khái niệm đồ thị:**

* Đồ thị là một tập các đối tượng gọi là đỉnh, nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh).
* Đồ thị có hướng G (graph) là một cặp có thứ tự G = (V, A), trong đó:

+ V là tập các đỉnh hoặc nút.

+ A là tập các cặp có thứ tự chứa các đỉnh, được gọi là các cạnh có hướng hoặc cung. Một cạnh a = (x, y) được coi là có hướng từ x tới y; x được gọi là điểm đầu/gốc và y được gọi là điểm cuối/ngọn của cạnh.

1. **Thuật toán**
2. **Giới thiệu giải thuật**

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 từ một đỉnh đến các đỉnh còn lại của 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)”.

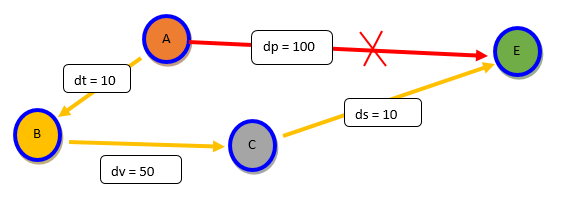
1. **Ý tưởng của thuật toán**

Ví dụ:

Với 4 đỉnh A, B, C, E.

Đường đi từ A đến E với độ dài là dp, đường đi từ A đến B với độ dài là dt, đường đi từ B đến C với độ dài là dv, đường đi từ C đến E với độ dài là ds.

Đường đi từ A đến B, từ B đến C và từ C đến E có thể chọn để đi so với đường đi trực tiếp từ A đến E vì dp > dt + dv + ds.



Hình 2: Ví dụ thuật toán dijkstra

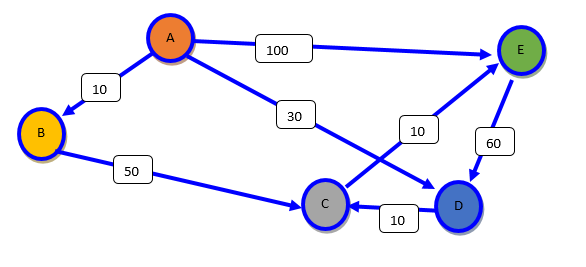
1. **Bài toán tổng quát**

Cho một đồ thị có hướng G= (V, A), một hàm trọng số w: A → [0, ∞) và một đỉnh nguồn x. Cần tính toán được đường đi ngắn nhất từ đỉnh nguồn x đến mỗi đỉnh của đồ thị.

Ví dụ:

Chúng ta có các điểm A, B, C, D, E và khoảng cách là:

Từ A đến B là 10, từ A đến E là 100, từ A đến D là 30, từ B đến C là 50, từ C đến E là 10, từ D đến C là 10, từ E đến D là 60.



Hình 3: Mô tả bằng đồ thị

Chúng ta dùng các đỉnh của đồ thị để mô hình các thành phố và các cạnh để mô hình các đường nối giữa chúng. Khi đó trọng số các cạnh có thể xem như độ dài của các con đường (và độ dài đó là không âm). Chúng ta cần vận chuyển từ thành phố A đến thành phố E.

1. **Mô tả thuật toán Dijkstra**
   1. **Các bước trong thuật toán Dijkstra**

Tập các đỉnh của đồ thị là V, gọi S là tập các đỉnh lấy từ V

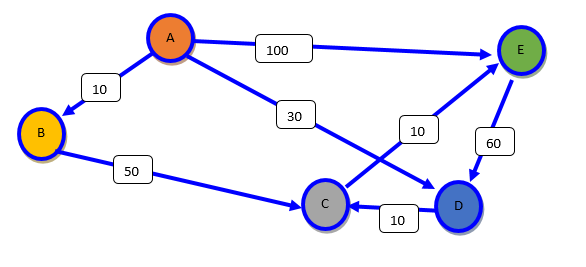
Bước 1:

Chọn một đỉnh nguồn x0 có trong tập V và thêm vào tập S (S = x0).

Bước 2:

Chọn đỉnh w thuộc tập V nhưng chưa có trong tập S sao cho độ dài đường đi ngắn nhất từ x0 đến w là ngắn nhất. Thêm w vào tập S.

Lặp lại bước 2 đến (n-1) lần và chúng ta có thể rìm được đường đi ngắn nhất từ v0 đến các đỉnh còn lại mà chỉ đi qua các đỉnh đã tồn tại trong S.

* 1. **Ví dụ thuật toán Dijkstra**

Hình 4: Đồ thị mô tả

Cho tập V gồm 5 đỉnh A, B, C, D, E và có hướng (giống như đường 1 chiều) và có các trọng số từ đỉnh A đến B là 10, đỉnh A đến D là 30, đỉnh A đến E là 100, đỉnh B đến C là 50, đỉnh C đến E là 10, đỉnh D đến C là 10, đỉnh E đến D là 60.

Còn các đỉnh không nối trực tiếp thì trọng số là vô cùng (∞).

Bước 1:

Giả sử đỉnh nguồn là A 🡪 S={A} và độ dài từ đỉnh A đến các đỉnh còn lại là {10, ∞, 30, 100}

Bước 2:

Chọn ra độ dài nhỏ nhất là 10 tương ứng là đỉnh B vào tập S

Cập nhật lại S={A,B} và độ dài từ các đỉnh còn lại là {10, 60, 30, 100} vì lúc này đỉnh A qua B và đến C nên cập nhật lại là 60.

Chọn độ dài nhỏ nhất là 30 tương ứng với đỉnh D vào tập S

Cập nhật lại S={A,B,D} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 90} vì lúc này đỉnh A qua D và đến C là 40 và A qua D đến E là 90.

Chọn độ dài nhỏ nhất là 40 tương ứng với đỉnh C vào tập S

Cập nhật lại S={A,B,D,C} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 50} vì lúc này đỉnh A qua D và qua C đến E là 50.

Chọn độ dài nhỏ nhất là 50 tương ứng với đỉnh E vào tập S

Cập nhật lại S={A,B,D,C,E} và độ dài từ A đến các đỉnh còn lại là {10, 40, 30, 50} vì đã đi hết các đỉnh nên dừng lại.

1. **Cài đặt bài toán**
   1. **Thiết kế thuật toán Dijkstra**

Tạo ma trận A lưu độ dài các cạnh (số thực), tức là A[x,y] là độ dài của cạnh (x,y), nếu không có cạnh (x,y) thì A[x,y] = ∞.

Mảng S[n] trong đó n là số nguyên lưu các số 0 và 1, đây là một vecto biểu diễn tập hợp S.

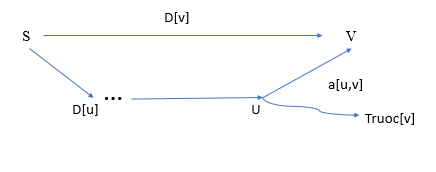
Nếu S[x]==1 là x thuộc S, nếu S[x] == 0 là thuộc tập V-S.

Khởi đầu cho S[x0]=1 nghĩa là x0 thuộc tập S, và các S[x]=0 nghĩa là các x còn lại nằm ngoại tập hợp S. Tại mỗi bước nếu w (đường đi ngắn nhất) được đưa vào S => đặt S[w]=1.

Mảng D[n] để lưu độ dài của đường đi ngắn nhất từ x0 đến mỗi đỉnh của đồ thị.

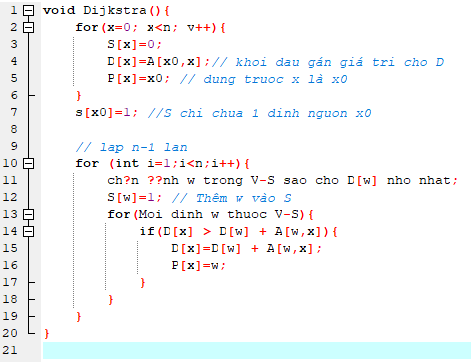
Khởi đầu cho D[x] bằng độ dài của x0 đến x thì D[x]=A[x0,x], nếu không có cạnh từ x0 đến x thì D[x]=∞ .

Tại mỗi bước, sẽ cập nhật lại D[x] để lưu độ dài đường đi ngắn nhất từ đỉnh x0 tới đỉnh x, đường đi này chỉ đi qua các đỉnh đã có trong S.



Giả sử đỉnh U đến V là ngắn hơn đường trước đó thì sẽ xóa đường cũ đi và cập nhật lại.

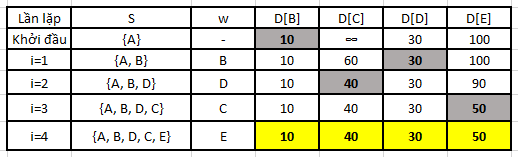
* 1. **Giải thuật thô**

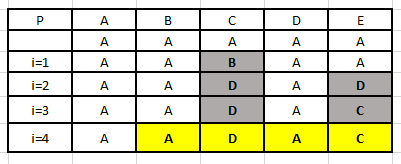


* 1. **Độ phức tạp thuật toán**

Thuật toán Dijkstra bình thường sẽ có độ phức tạp là O(n2 + m). Tuy nhiên ta có thể sử dụng kết hợp với cấu trúc heap, khi đó độ phức tạp sẽ là O((m + n)log (n)), nếu dùng Fibonacci Heap thì độ phức tạp giảm xuống còn O(m + n log (n)). Trong đó m là số cạnh, n là số đỉnh của đồ thị đang xét.

* 1. **Minh họa thuật toán Dijkstra**

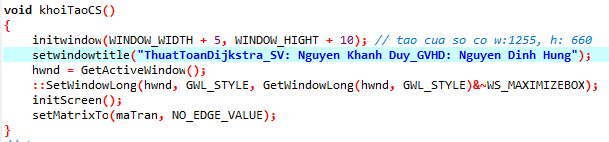


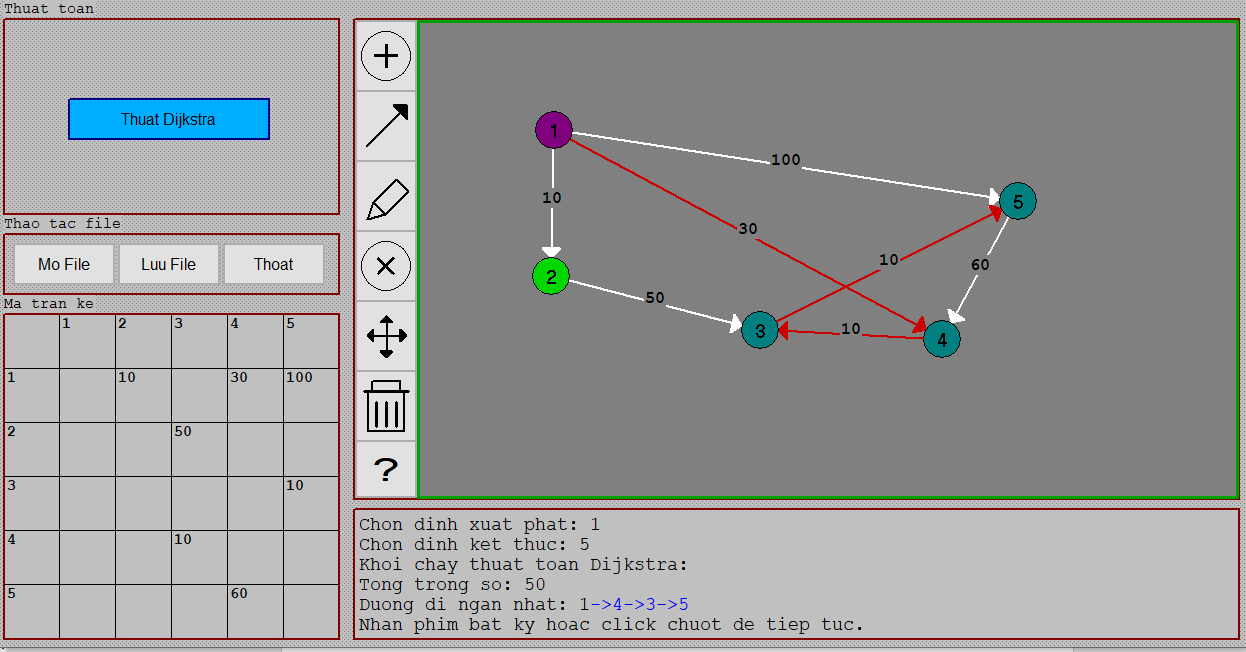


Hình 5: Bước thực hiện

1. **Kết quả thực hiện**
2. **Cửa sổ giao diện**

* Có 5 phân vùng và tổng kích thước chiều rộng là 1255, chiều cao là 660





Hình 6: Cửa sổ chính

* Phân vùng 1 chạy thuật toán khi thực hiện click chuột.



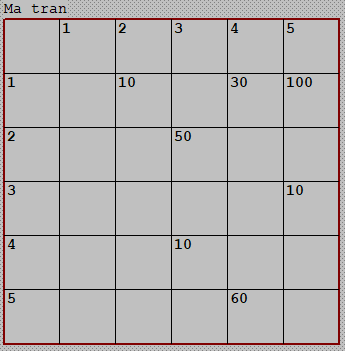
Hình 7: Cửa sổ thuật toán dijkstra

* Phân vùng 2 thao tác với file có các chức năng như mở, lưu và thoát từ tập tin đã tạo sẵn lên cửa sổ và ngược lại. (chưa thực hiện được đang phát triển thêm)



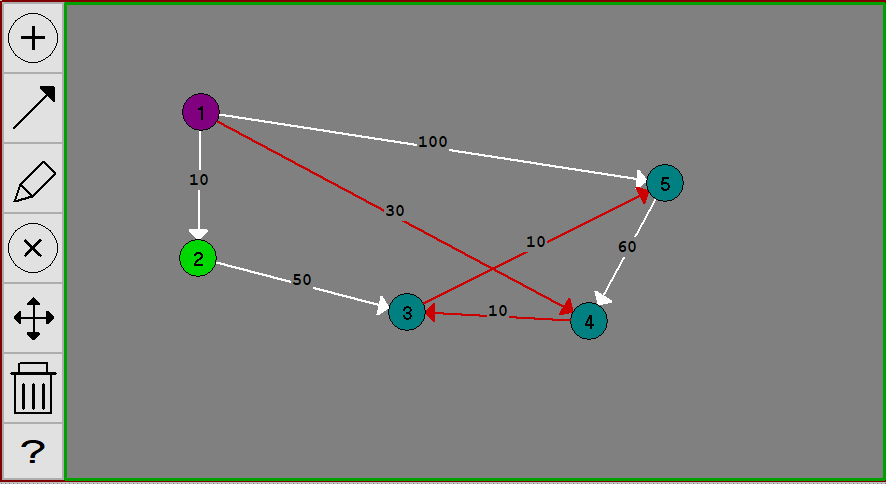
Hình 8: Thao tác file

* Phân vùng 3 là ma trận hiển thị các đỉnh và các trọng số từ đỉnh này đến đỉnh khác của đồ thị.



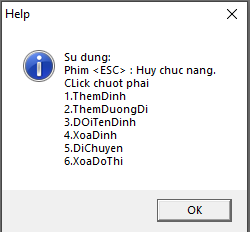
Hình 9: Cửa sổ ma trận

* Phân vùng 4 là hiện thị các nút để thao tác với đồ thị. Có các nút như thêm đỉnh, thêm trọng số, thay đổi tên đỉnh, xóa đỉnh, di chuyển đồ thị, xóa tất cả đồ thị, hiển thị giúp đỡ, và vùng hiển thị đồ thị mô tả những đỉnh và trọng số đã nhập.



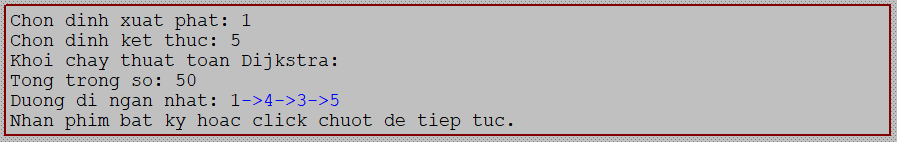
Hình 10: Các nút thao tác và đồ thị

* Khi chọn đỉnh đầu và cuối thì hai đỉnh này sẽ đổi màu và bắt đầu chạy thuật toán in ra kết quả và các đường đi ngắn nhất từ đỉnh đầu đến đỉnh cuối sẽ đổi màu đỏ, và các đỉnh sẽ đổi màu xanh.
* Kích vào nút thêm đỉnh thì sẽ tạo đỉnh mới và mặc định đỉnh đầu là số 1.
* Kích vào nút thêm cạnh thì sẽ thêm cạnh từ đỉnh này đến đỉnh khác và nhập trọng số vào. Nếu muốn thay đổi trọng số thì ta làm tương tự.
* Thay đổi đỉnh thì chúng ta đổi tên đỉnh đã tạo.
* Xóa đỉnh thì sẽ xóa đỉnh và các cạnh nối đến đỉnh đó.
* Di chuyển thì chúng ta sẽ di chuyển đỉnh hoặc dữ ctrl để di chuyển hết đồ thị.
* Xóa là xóa tất cả đồ thị.
* Help sẽ giúp chúng ta các thao tác trên.



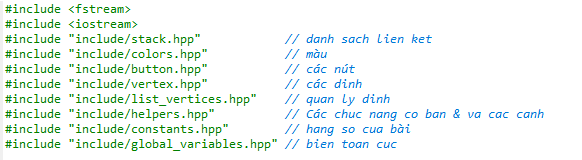
Hình 11: Hiện thi giúp đỡ

* Phân vùng 5 là hiển thị kết quả sau khi chọn đỉnh đầu và cuối.



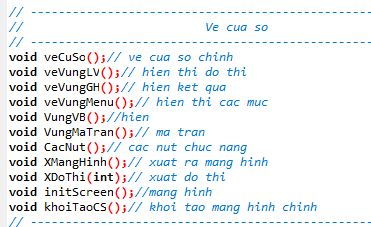
Hình 12: Kết quả khi chạy

1. **Source code**
   1. **Các thư viện khai báo**



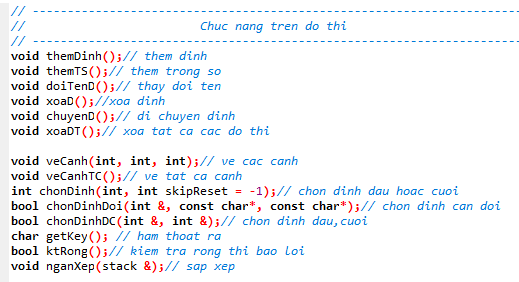
Hình 13: Thư viện

* 1. **Các hàm tạo giao diện**



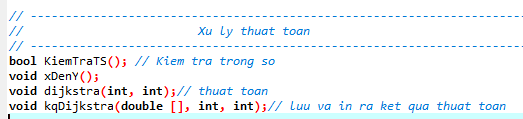
Hình 14: Hàm tạo cửa sổ

* 1. **Chức năng trên đồ thị**



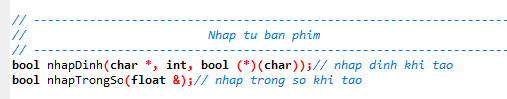
Hình 15: Các chức năng

* 1. **Xử lý thuật toán**



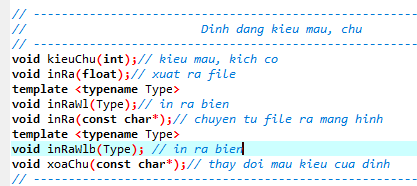
Hình 16: Xử lý thuật toán dijkstra

* 1. **Nhập đỉnh và trọng số**



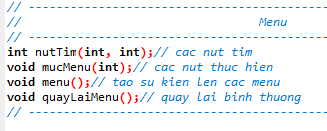
Hình 17: Nhập đỉnh

* 1. **Định dạng kiểu chữ và màu sắc khi xuất ra**



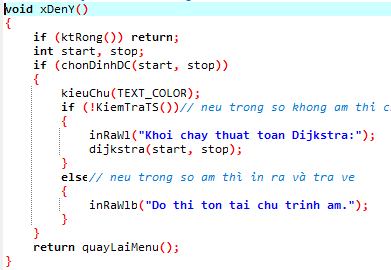
Hình 18: Định dạng kiểu và màu

* 1. **Mục menu**



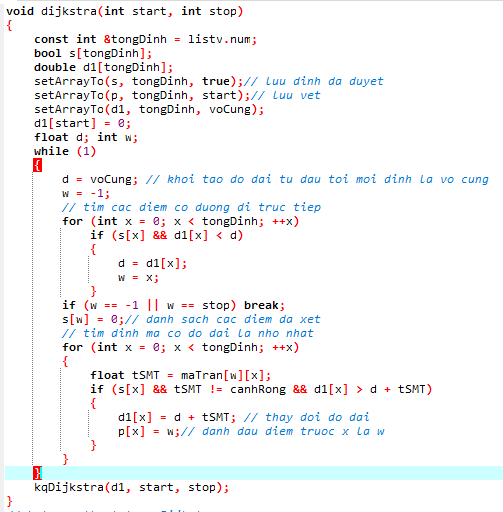
Hình 19: Tổng hợp menu

1. **Các hàm xử lý thuật toán**



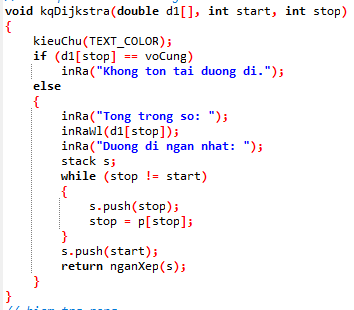
Hình 20: Khởi chạy thuật toán

* Hàm xDenY()
  + Sẽ kiểm tra nếu không có các đỉnh thì không thực hiện chạy thuật toán.
  + Sau khi chọn xong thì nó sẽ khởi chạy thuật toán Dijkstra và dừng lại.
  + Nếu có trọng số âm thì sẽ in ra thông báo và dừng lại.
* Hàm dijkstra() sẽ lưu vết các trọng số vào ma trận, và các đỉnh.



Hình 21: Thuật toán Dijkstra

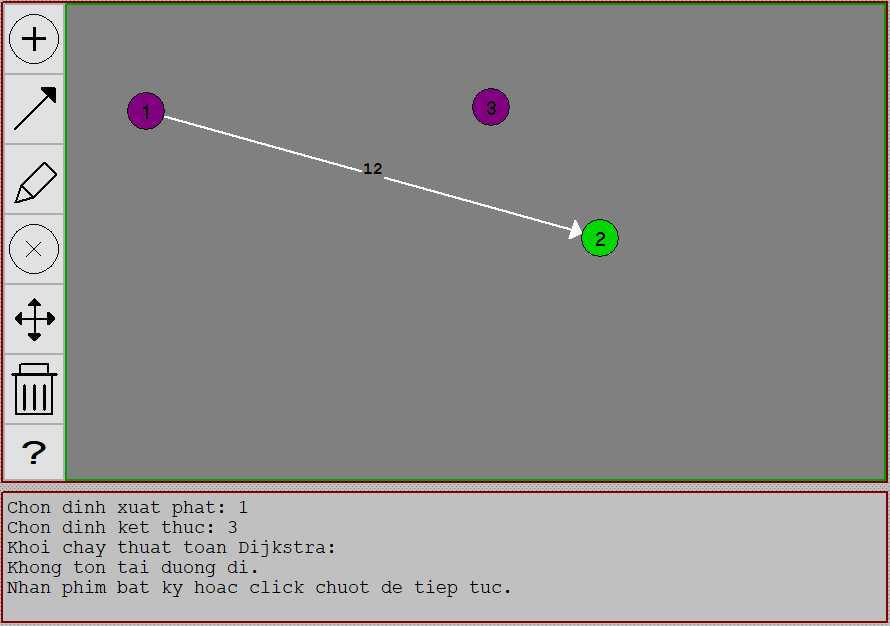
* Sau khi có được đỉnh đầu và đỉnh cuối, tạo mảng lưu các đỉnh và trọng số ngắn nhất từ đỉnh đầu đến các đỉnh còn lại.
* Cho trọng số các đỉnh chưa duyệt là vô cùng.
* Vòng lặp for đầu tiên để duyệt các đỉnh kề với đỉnh đầu được chọn.
* Vòng lặp for thứ hai là kiểm tra xem các đỉnh không kề có đường đi đến hay không. Nếu không có thì vẫn là vô cùng, nếu có đường đi ngắn hơn thì cập nhật lại.
* Cứ như thế khi nào đến hết thì thoát khỏi vòng lặp while.
* Hàm kqDijkstra() sẽ tính và in ra kết quả.



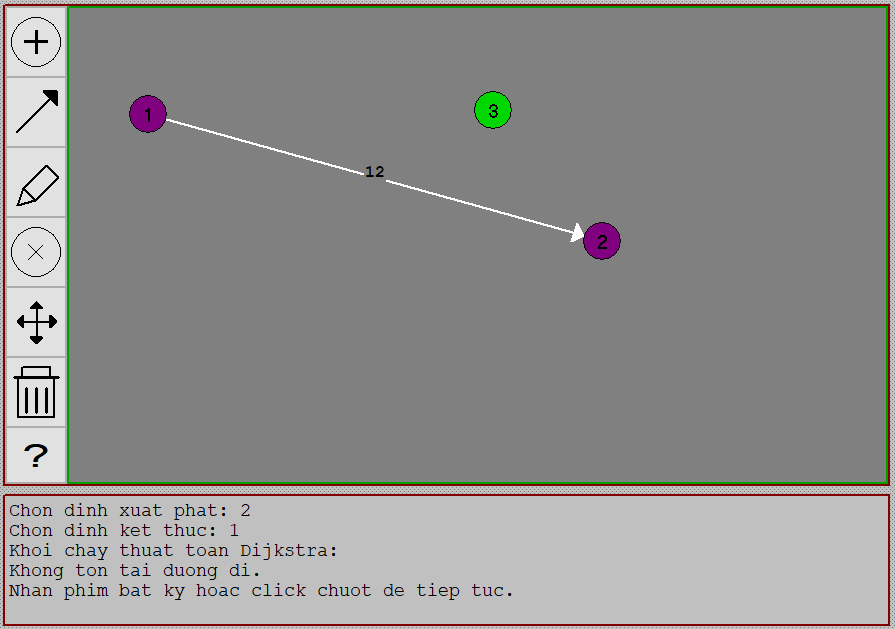
Hình 22: Kết quả sau khi chạy

* Sau khi thực hiện xong thuật toán thì in ra kết quả. Sẽ có hai trường hợp là không có đường đi đến đỉnh khác và có đường đi đến.
* Nếu không có thì ta sẽ in ra kết quả “Khong ton tai duong di”
* Nếu có thì tính tổng trọng số lại. Rồi mượn ngăn xếp stack để đảo ngược các đỉnh từ đầu thành cuối.

1. **Các trường hợp kết quả**
   1. **Không có đường đi từ đỉnh đầu đến đỉnh cuối**

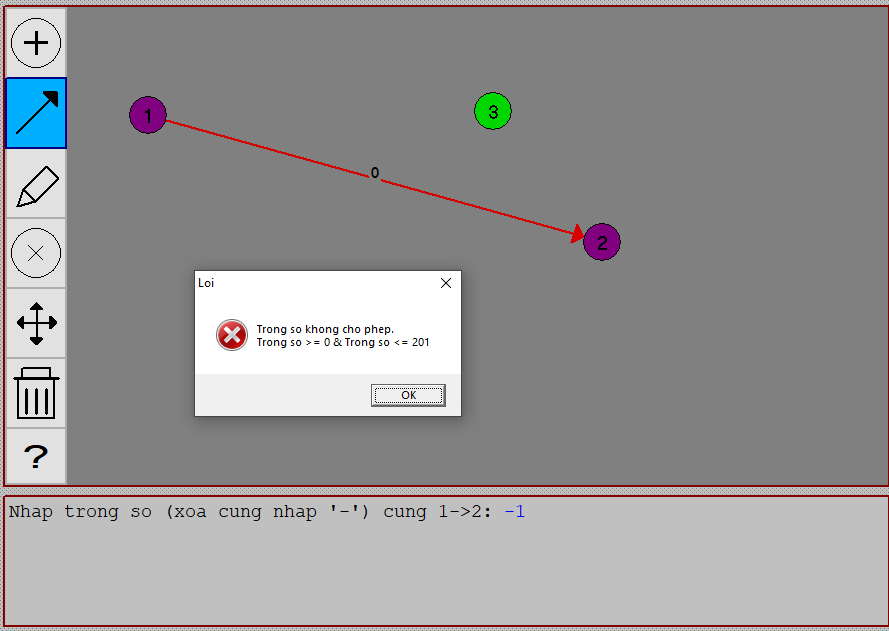


Hình 23: Kết quả không có đường đi từ 1 đến 3



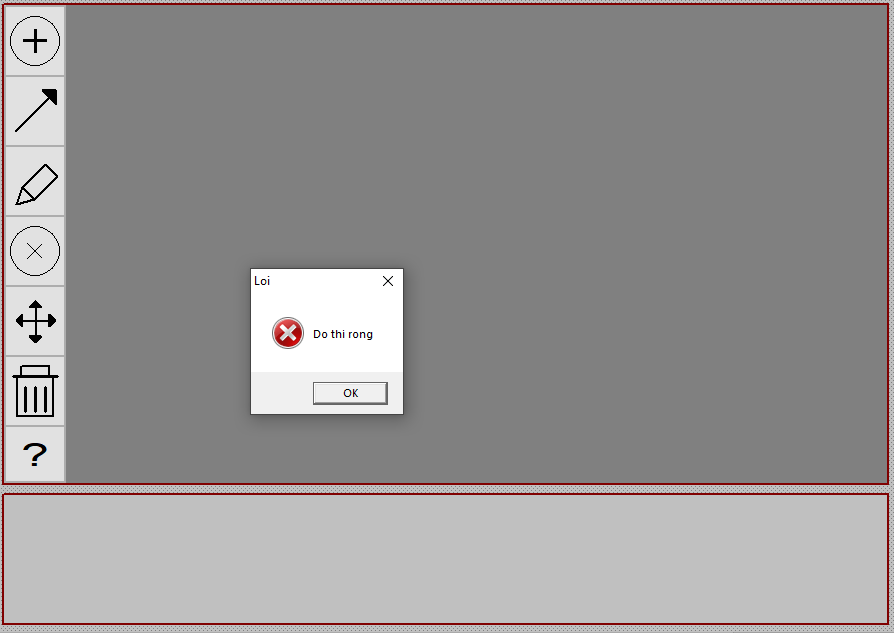
Hình 24: Kết quả không có đường đi từ 2 đến 1

* 1. **Nhập trọng số âm**



Hình 25: Trọng số âm

* 1. **Chưa có đồ thị mà kích vào các nút thao tác trên đồ thị**



Hình 26: Chưa có đồ thị

1. **Ngôn ngữ và môi trường phát triển:**
2. **Ngôn ngữ sử dụng**

C++ (C Plus Plus) là một loại ngôn ngữ lập trình bậc trung. Đây là ngôn

ngữ lập trình đa năng được tạo ra bởi Bjarne Stroustrup như một phần mở rộng của ngôn ngữ lập trình C. Ngôn ngữ đã được mở rộng đáng kể theo thời gian và C ++ hiện đại có các tính năng: lập trình tổng quát, lập trình hướng đối tượng, lập trình thủ tục, ngôn ngữ đa mẫu hình tự do có kiểu tĩnh, dữ liệu trừu tượng, và lập trình đa hình, ngoài ra còn có thêm các tính năng, công cụ để thao tác với bộ nhớ cấp thấp. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ thương mại ưa thích và phổ biến của lập trình viên.

1. **Môi trường phát triển**

Dev-C++ là một môi trường phát triển tích hợp tự do (IDE) được phân phối dưới hình thức giấy phép Công cộng GNU hỗ trợ việc lập trình bằng C/C++. Nó cũng nằm trong bộ trình dịch mã nguồn mở MinGW. Chương trình IDE này được viết bằng ngôn ngữ Delphi. Ngoài ra thêm thư viện đồ họa graphics/ winbgim dùng để mô phỏng thuật toán các trực quan hơn.

1. **Tài liệu tham khảo**

|  |  |
| --- | --- |
| [1] | Thư viện stack, [Online]. Available: https://nguyenvanhieu.vn/ngan-xep-stack/. |
| [2] | Tạo sự kiện cho chuột, [Online]. Available: https://code24h.com/c-c-click-chuot-trong-dev-c-8211-mouse-event-in-dev-c-d24172.htm. |
| [3] | Cài thêm thư viện đồ họa, [Online]. Available: https://tuhoclaptrinh.cachhoc.net/2017/02/25/bai-1-cai-dat-devc-va-them-thu-vien-do-hoa-graphics-winbgim/. |
| [4] | Đồ thị có hướng, [Online]. Available: https://123docz.net/document/20110-thuat-toan-do-thi-co-huong-va-chu-trinh.htm. |
| [5] | win32, [Online]. Available: https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-messagebox. |
| [6] | Thuật toán Dijsktra, [Online]. Available: https://cachhoc.net/2013/10/13/thuat-toan-tim-duong-di-ngan-nhat-dijkstra-floyd/. |