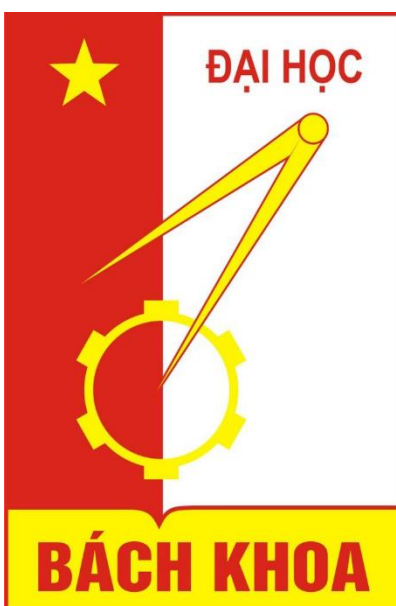


TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
-----o0o-----



BÁO CÁO

Học phần: Lý thuyết và ngôn ngữ hướng đối tượng

Mã HP: IT3102

Mã lớp: 122024

Hà Nội 2021

Thành viên nhóm:

STT	Họ và tên	MSSV
1	Trần Hữu Bình	20184048
2	Trần Quang Nam	20184161
3	Trần Đức Quân	20184178

Contents

1	Đề tài Project	3
2	Phân công công việc	3
3	Trình bày thuật toán.....	3
3.1	Thuật toán Dijkstra	3
3.1.1	Đặt vấn đề.....	3
3.1.2	Mô tả giải thuật Dijkstra.....	3
3.2	Thuật toán Bellman-Ford.....	4
3.2.1	Đặt vấn đề.....	4
3.2.2	Mô tả giải thuật Bellman-Ford	4
3.3	Thuật toán A*	5
3.3.1	Đặt vấn đề.....	5
3.3.2	Mô tả giải thuật A*	5
4	Use Case Diagram	7
5	Class Diagram.....	8
6	Giới thiệu chung về chương trình	9
6.1	Giao diện chương trình	9
6.2	Các thao tác tạo, sửa đổi đồ thị.....	9
6.3	Chạy mô phỏng thuật toán : Tìm đường đi từ đỉnh bắt đầu tới đỉnh kết thúc	11
7	Thực thi File từ dòng lệnh	12
7.1	Yêu cầu	12
7.2	Cách thực thi.....	12
7.3	Cách thực thi khác	12

1 Đề tài Project

An application to (visually) demonstrate the shortest path problem:

- a) Dijkstra
- b) Bellman-Ford
- c) A*

Tạo ứng dụng để miêu tả trực quan bài toán tìm đường ngắn nhất, sử dụng các thuật toán sau:

- a) Dijkstra
- b) Bellman-Ford
- c) A*

2 Phân công công việc

STT	Họ tên	Phân công công việc	Công việc chung
1	Trần Hữu Bình	- Code đồ thị - Code thuật toán	- Tìm hiểu thuật toán tìm đường - Phân tích đề tài và thiết kế Class Diagram
2	Trần Quang Nam	- Code đồ thị - Viết báo cáo - Tìm hiểu và tạo file executable	
3	Trần Đức Quân	- Code giao diện - Viết báo cáo	

3 Trình bày thuật toán

3.1 Thuật toán Dijkstra

3.1.1 Đặt vấn đề

Cho một đồ thị $G = (V, E)$, một hàm trọng số $w: E \rightarrow [0, \infty)$ và một đỉnh (node) nguồn S , một đỉnh đích T . Cần tìm được đường đi ngắn nhất từ đỉnh nguồn S đến đỉnh đích T của đồ thị.

Ví dụ: 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 do đó là không âm). Chúng ta cần di chuyển từ thành phố S đến thành phố T . Thuật toán Dijkstra sẽ giúp chỉ ra đường đi ngắn nhất chúng ta có thể đi.

Trọng số không âm của các cạnh của đồ thị mang tính tổng quát hơn khoảng cách hình học giữa hai đỉnh đầu mút của chúng. Ví dụ, với 3 đỉnh A, B, C đường đi $A-B-C$ có thể ngắn hơn so với đường đi trực tiếp $A-C$.

3.1.2 Mô tả giải thuật Dijkstra

- Bước 1:

Chọn $S = \{\}$ là tập các source_node bao gồm current_node và visited_node. Với current_node là node đang được xét đến, visited_node là các node đã được xét. current_node đầu tiên sẽ là node đích của bài toán tìm đường đi ngắn nhất.

- Bước 2:

Khởi tạo giải thuật với current_node là node đích và $\text{cost}(N)$ là giá trị của đường đi ngắn nhất từ N đến node đích.

- Bước 3:

Xét các node kề N với current_node. Gọi $d(\text{current_node}, N)$ là khoảng cách giữa node kề N và current_node.

Với $p = d(\text{current_node}, N) + \text{cost}(\text{current_node})$. Nếu $p < \text{cost}(N)$ thì $\text{cost}(N) = p$. Nếu không thì $\text{cost}(N)$ giữ nguyên giá trị.

- Bước 4:

Sau khi xét hết các node kề N , đánh dấu current_node thành visited_node.

- Bước 5:

Tìm current_node mới với 2 điều kiện: không phải visited_node và $\text{cost}(\text{current_node})$ là nhỏ nhất

- Bước 6:

Nếu tập $S = \{\}$ chứa đủ các node của đồ thị thì dừng thuật toán. Nếu không thì quay trở lại Bước 3.

3.2 Thuật toán Bellman-Ford

3.2.1 Đặt vấn đề

Giải thuật Dijkstra có thể giải quyết rất nhanh bài toán tìm đường đi ngắn nhất với đồ thị có trọng số dương. Tuy nhiên, với đồ thị trọng số âm, chúng ta sẽ chọn giải thuật Bellman-Ford.

3.2.2 Mô tả giải thuật Bellman-Ford

Ý tưởng của thuật toán như sau:

- Ta thực hiện duyệt n lần, với n là số đỉnh của đồ thị.
- Với mỗi lần duyệt, ta tìm tất cả các cạnh mà đường đi qua cạnh đó sẽ rút ngắn đường đi ngắn nhất từ đỉnh gốc tới một đỉnh khác.
- Ở lần duyệt thứ n , nếu còn bất kỳ cạnh nào có thể rút ngắn đường đi, điều đó chứng tỏ đồ thị có chu trình âm, và ta kết thúc thuật toán.

Thuật toán Bellman-Ford có 3 bước, giải thích bằng pseudocode như sau:

- Bước 1: Khởi tạo đồ thị

FOR EACH v in danh_sách_đỉnh:

IF v = nguồn THEN khoảng_cách(v) := 0

ELSE khoảng_cách(v) := ∞

đỉnh_liền_trước(v) := null

- Bước 2: Cập nhật các cạnh với n vòng lặp (n là số node) sao cho đường đi từ source_node đến node bị lặp là nhỏ nhất.

FOR i FROM 1 TO SIZE(danh_sách_đỉnh) - 1:

FOR EACH (u, v) in danh_sách_cạnh:

IF khoảng_cách(v) > khoảng_cách(u) + trọng_số(u, v):

 khoảng_cách(v) := khoảng_cách(u) + trọng_số(u,v)

 đỉnh_liền_trước(v) := u

- Bước 3: Kiểm tra xem đồ thị có chu trình âm hay không?

FOR EACH (u, v) in danh_sách_cạnh:

IF khoảng_cách(v) > khoảng_cách(u) + trọng_số(u,v):

 error "Đồ thị chứa chu trình âm"

3.3 Thuật toán A*

3.3.1 Đặt vấn đề

A* là giải thuật tìm kiếm trong đồ thị, tìm đường đi từ một đỉnh hiện tại đến đỉnh đích có sử dụng hàm để ước lượng khoảng cách hay còn gọi là hàm Heuristic.

Từ trạng thái hiện tại A* xây dựng tất cả các đường đi có thể đi dùng hàm ước lượng khoảng cách (hàm Heuristic) để đánh giá đường đi tốt nhất có thể đi. Tùy theo mỗi dạng bài khác nhau mà hàm Heuristic sẽ được đánh giá khác nhau. A* luôn tìm được đường đi ngắn nhất nếu tồn tại đường đi như thế.

A* lưu giữ một tập các đường đi qua đồ thị, từ đỉnh bắt đầu đến đỉnh kết thúc, tập các đỉnh có thể đi tiếp được lưu trong tập Open.

Thứ tự ưu tiên cho một đường đi được quyết định bởi hàm Heuristic được đánh giá

$$f(x) = g(x) + h(x)$$

- $g(x)$ là chi phí của đường đi từ điểm xuất phát cho đến thời điểm hiện tại.
- $h(x)$ là hàm ước lượng chi phí từ đỉnh hiện tại đến đỉnh đích $f(x)$ thường có giá trị càng thấp thì độ ưu tiên càng cao.

3.3.2 Mô tả giải thuật A*

Dữ liệu:

- Open: tập các trạng thái đã được sinh ra nhưng chưa được xét đến.
- Close: tập các trạng thái đã được xét đến.
- Cost(p, q): là khoảng cách giữa p, q.
- $g(p)$: khoảng cách từ trạng thái đầu đến trạng thái hiện tại p.
- $h(p)$: giá trị được ước lượng từ trạng thái hiện tại đến trạng thái đích.
- $f(p) = g(p) + h(p)$

Các bước thực hiện giải thuật:

- Bước 1:

Open: = {s}

Close: = {}

- Bước 2:

while (Open != {})

 Chọn trạng thái (đỉnh) tốt nhất p trong Open (xóa p khỏi Open).

 Nếu p là trạng thái kết thúc thì thoát.

Chuyển p qua Close và tạo ra các trạng thái kế tiếp q sau p.

Nếu q đã có trong Open

Nếu $g(q) > g(p) + \text{Cost}(p, q)$

$g(q) = g(p) + \text{Cost}(p, q)$

$f(q) = g(q) + h(q)$

$\text{prev}(q) = p$ (đỉnh cha của q là p)

Nếu q chưa có trong Open

$g(q) = g(p) + \text{cost}(p, q)$

$f(q) = g(q) + h(q)$

$\text{prev}(q) = p$

Thêm q vào Open

Nếu q có trong Close

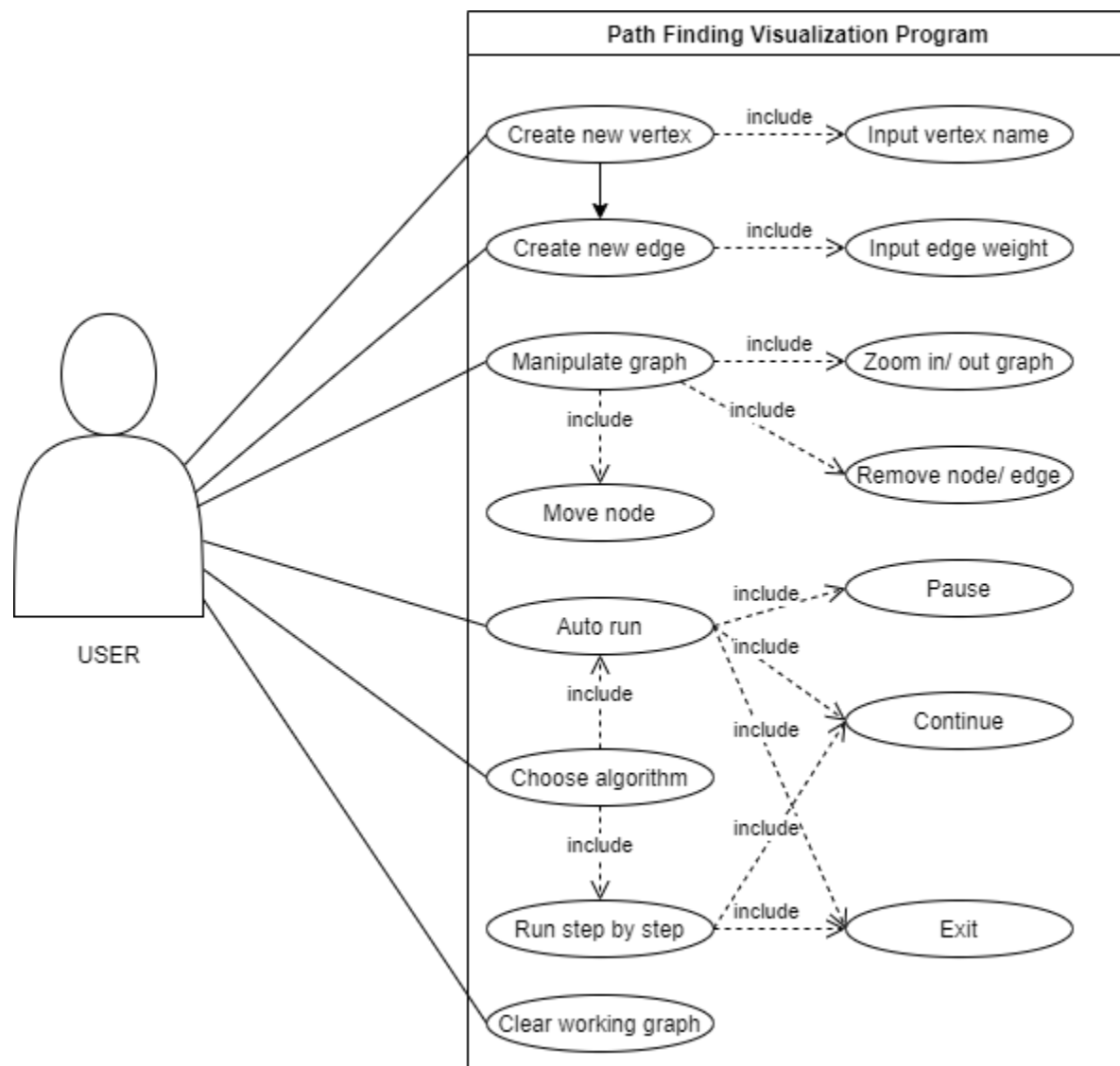
Nếu $g(q) > g(p) + \text{Cost}(p, q)$

Bỏ q khỏi Close

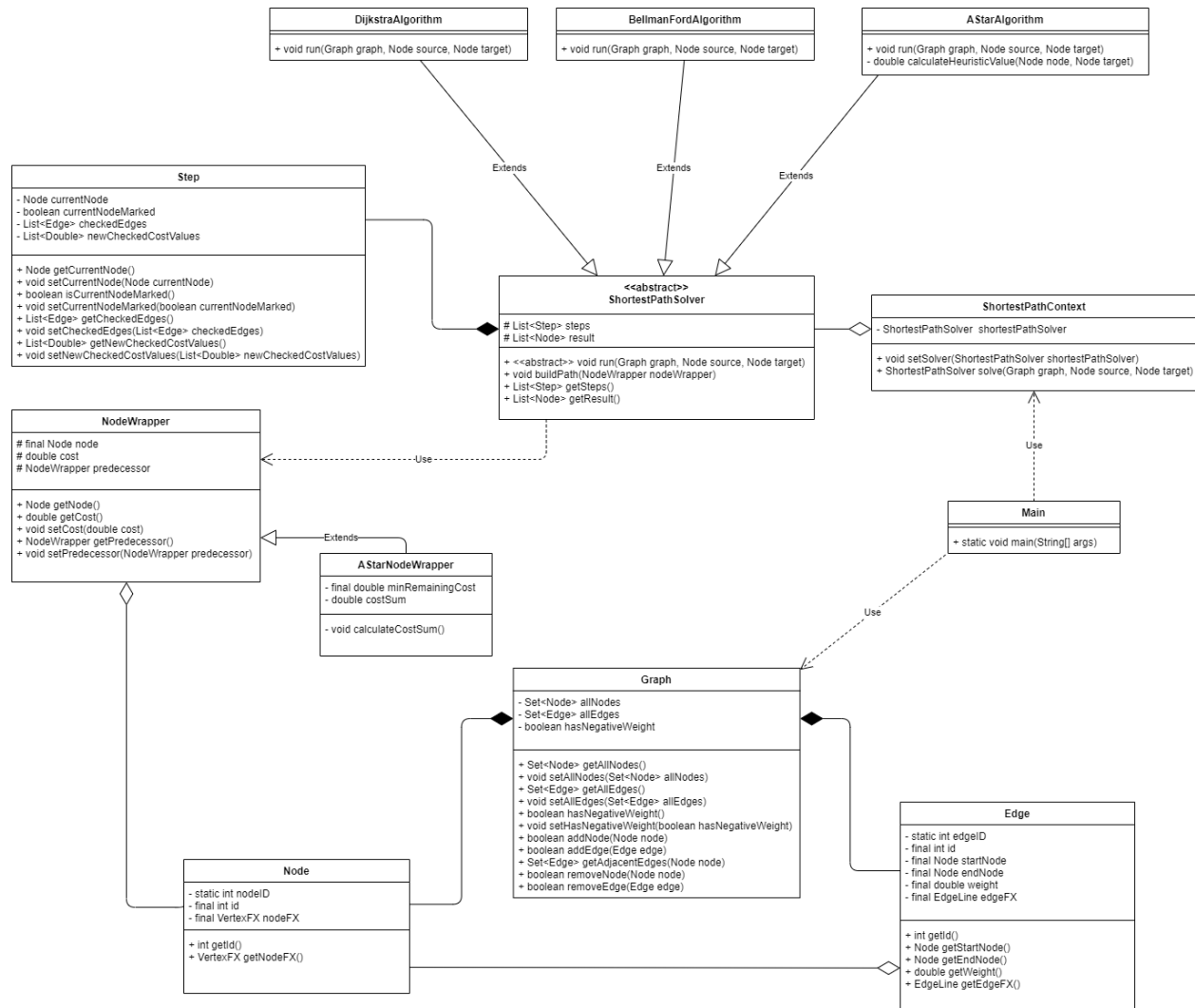
Thêm q vào Open

- Bước 3: Không tìm được.

4 Use Case Diagram

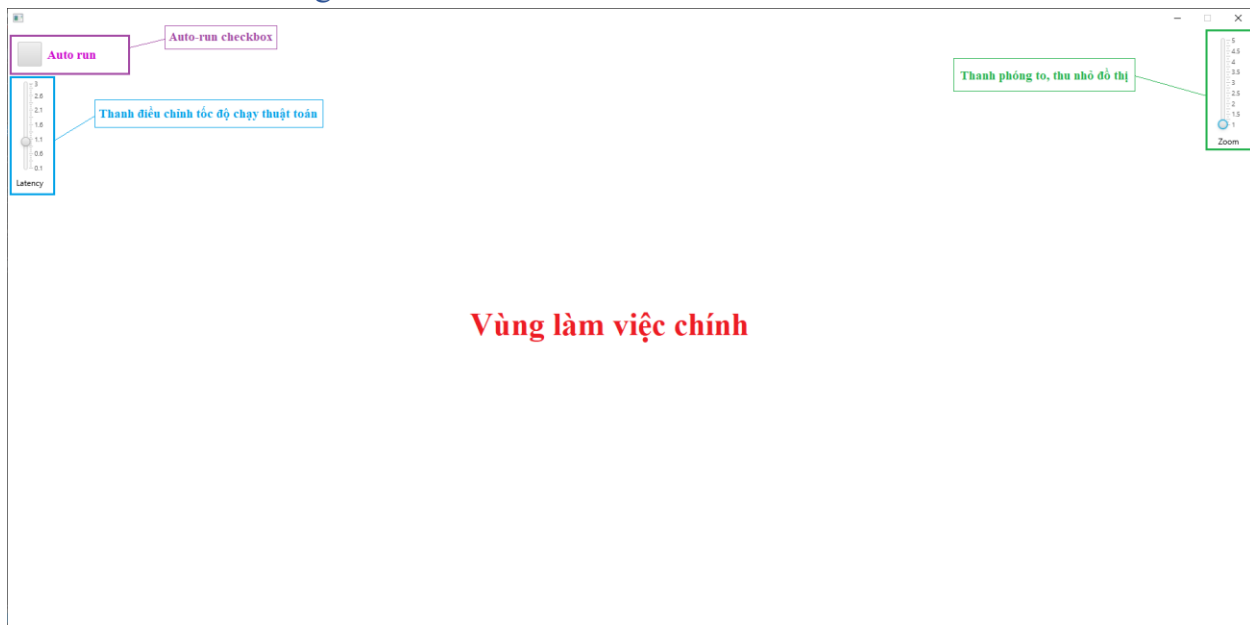


5 Class Diagram



6 Giới thiệu chung về chương trình

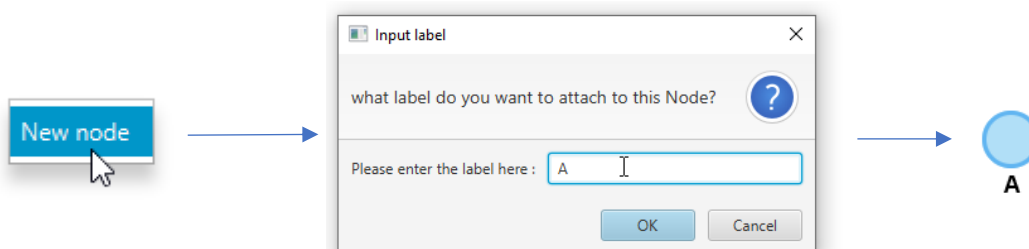
6.1 Giao diện chương trình



- Vùng làm việc chính :
 - Người dùng sẽ thực hiện vẽ đồ thị, mô phỏng và theo dõi quá trình chạy thuật toán trên vùng này.
 - Kéo chuột phải để di chuyển đồ thị trên vùng làm việc.
- Thanh điều chỉnh tốc độ chạy thuật toán (s/bước):
 - Điều chỉnh tốc độ mô phỏng thuật toán, tối thiểu là 0.1s/bước, tối đa là 3s/bước
 - Chỉ có thể điều chỉnh tốc độ trước khi bắt đầu mô phỏng thuật toán.
- Thanh phóng to, thu nhỏ :
 - Cuộn chuột lên/xuống để thực hiện phóng to, thu nhỏ đồ thị

6.2 Các thao tác tạo, sửa đổi đồ thị

- Tạo một đỉnh :
 - Nháy chuột phải tại vị trí bất kỳ trên **vùng làm việc chính**.
 - Chọn **New node**.
 - Nhập tên của đỉnh muốn tạo, rồi nhấn OK.

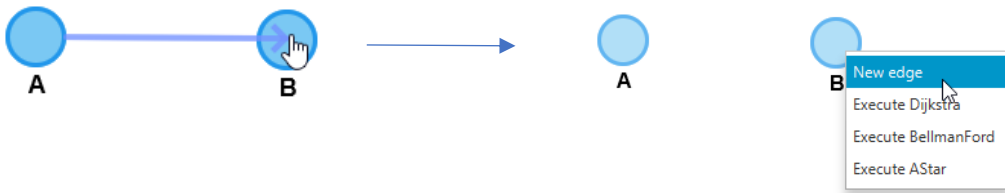


➔ Sau khi đỉnh được khởi tạo, có thể kéo thả chuột trái để di chuyển đỉnh trên khung làm việc, định dạng đồ thị cho dễ nhìn.

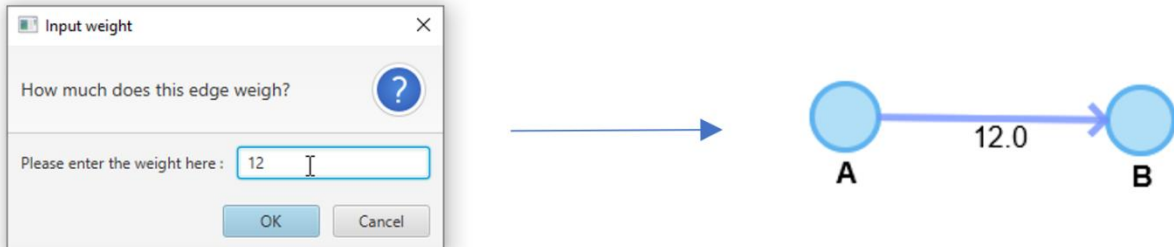
- Xoá một đỉnh :
 - Nháy chuột phải vào đỉnh muốn xoá.
 - Chọn **Delete**



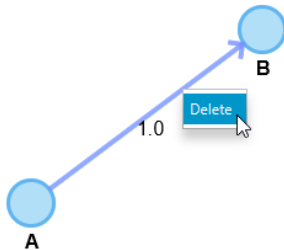
- Tạo một cạnh :
 - Kéo chuột phải từ đỉnh bắt đầu tới đỉnh kết thúc của cạnh muốn tạo.
 - Chọn **New edge**.



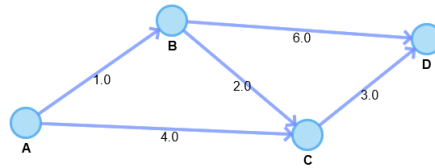
- Nhập trọng số cho cạnh muốn tạo, nhấn OK.



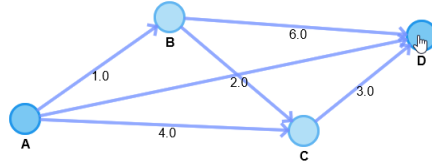
- Xoá một cạnh:
 - Nháy chuột phải vào cạnh muốn xoá
 - Chọn **Delete**.



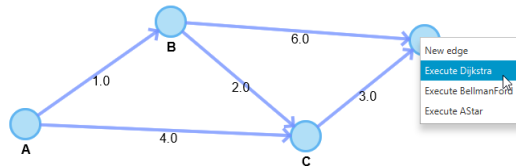
6.3 Chạy mô phỏng thuật toán : Tìm đường đi từ đỉnh **bắt đầu** tới đỉnh **kết thúc** (Ví dụ : Tìm đường đi từ đỉnh A tới đỉnh D trong đồ thị dưới đây)



- Kéo chuột phải từ đỉnh **bắt đầu** tới đỉnh **kết thúc** mà ta muốn tìm đường.

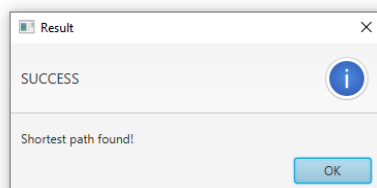


- Trong Menu hiện ra, chọn thuật toán muốn chạy mô phỏng.

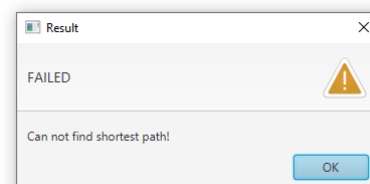


➔ Chương trình sẽ bắt đầu thực hiện mô phỏng thuật toán.

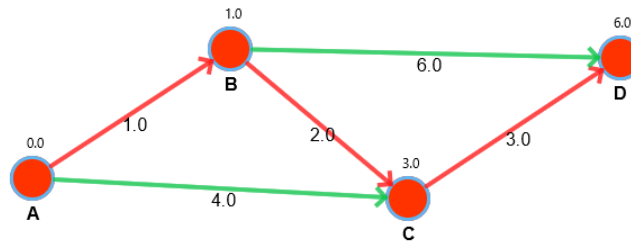
- Quan sát quá trình mô phỏng thuật toán, có thể sử dụng các chức năng sau ở góc trên-trái **vùng làm việc chính**:
 - Tạm dừng mô phỏng :  (Chỉ có ở chế độ mô phỏng Auto-run)
 - Tiếp tục/Bước kế tiếp : 
 - Thoát mô phỏng :  (Chỉ có ở chế độ mô phỏng Auto-run)
- Sau khi kết thúc và có kết quả của thuật toán, sẽ có một cửa sổ hiện lên thông báo có thể tìm đường đi yêu cầu hay không:




Hoặc



➔ Đường đi ngắn nhất (nếu tìm thấy) sẽ được làm nổi bật trên đồ thị kết quả



- Lúc này, muốn thoát khỏi mô phỏng, bấm nút **Reset**  để đưa đồ thị đang làm việc về trạng thái bình thường, sau đó có thể tiếp tục làm việc với đồ thị

7 Thực thi File từ dòng lệnh

7.1 Yêu cầu

Để có thể thực thi file từ dòng lệnh thì cần có:

- Java
- Java SE Runtime Environment
- Java FX runtime

7.2 Cách thực thi

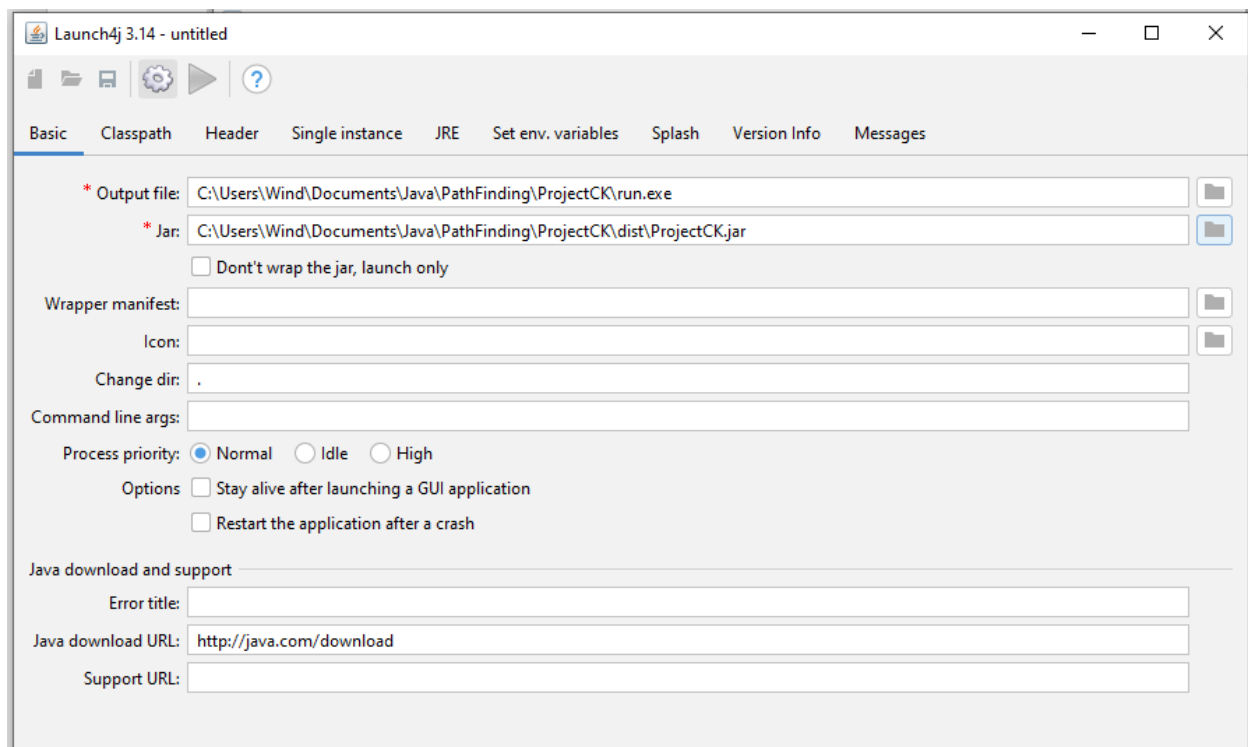
- Bước 1: Build file <file_name>.jar, có thể sử dụng các IDE hỗ trợ như NetBeans
- Bước 2: Chạy file <file_name>.jar từ dòng lệnh như sau:

```
java --module-path "path_to_javafx_sdk_lib" --add-modules javafx.controls, javafx.fxml  
-jar "path_to_file"/<file_name>.jar
```

7.3 Cách thực thi khác

Ngoài ra, có thể sử dụng 1 số phần mềm khác để tạo file executable đuôi .exe từ file .jar

Ví dụ, có thể sử dụng phần mềm Launch4j



Trong đó, 2 trường chính cần lưu ý:

- Output file: nơi lưu trữ file thực thi đuôi .exe
- Jar: nơi lưu trữ file .jar để chuyển đổi thành file thực thi .exe

Sau khi chạy thì ta sẽ có được file thực thi với đuôi .exe và có thể chạy trực tiếp chương trình từ file này.