TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 1**

**XÂY DỰNG CÔNG CỤ VẼ GRAPH TRÊN JAVA**

*Giảng viên hướng dẫn***: TS. PHẠM VĂN HUY**

*Sinh viên thực hiện***: NGUYỄN VĂN ĐỒNG – 51303273**

**NGUYỄN NGHĨA DINH – 51303254**

**Lớp: 13050302**

**Khóa : 17**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2016**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 1**

**XÂY DỰNG CÔNG CỤ VẼ GRAPH TRÊN JAVA**

*Giảng viên hướng dẫn***: TS. PHẠM VĂN HUY**

*Sinh viên thực hiện***: NGUYỄN VĂN ĐỒNG – 51303273**

**NGUYỄN NGHĨA DINH – 51303254**

**Lớp: 13050302**

**Khóa : 17**

**TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2016**

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ hết sức tận tình của quý Thầy Cô, gia đình và bạn bè.

Với lòng biết ơn sâu sắc nhất, chúng em xin gửi đến quý Thầy Cô ở Khoa Công Nghệ Thông Tin – Trường Đại Học Tôn Đức Thắng đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường.

Chúng em xin chân thành cảm ơn TS Phạm Văn Huy đã tận tâm hướng dẫn cho chúng em qua từng buổi báo cáo hằng tuần, thảo luận về đề tài “**Xây dựng công cụ vẽ Graph trên Java**”. Nếu như không có những lời hướng dẫn, dạy bảo của thầy thì chúng em nghĩ bài báo cáo này của em rất khó có thể hoàn thành được. Một lần nữa chúng em xin chân thành cảm ơn thầy.

Đồ án 1 này được thực hiện trong khoảng thời gian gần 3 tháng. Trong quá trình tìm hiểu kiến thức của chúng em còn nhiều hạn chế. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, chúng em rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn để kiến thức của chúng em trong lĩnh vực này được hoàn thiện hơn.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng em và được sự hướng dẫn của TS Phạm Văn Huy. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 20 tháng 12 năm 2016*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

***Nguyễn Văn Đồng***

*Nguyễn Nghĩa Dinh*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**MỤC LỤC**

[**CHƯƠNG 1. TỔNG QUAN** 9](#_Toc470111374)

[1.1. Giới Thiệu 9](#_Toc470111375)

[**1.2.** **Khảo sát hệ thống** 9](#_Toc470111376)

[1.2.1. Nguồn khảo sát: 9](#_Toc470111377)

[1.2.2. Cách thức khảo sát: 9](#_Toc470111378)

[1.2.3. Đối tượng khảo sát: 9](#_Toc470111379)

[1.2.4. Phạm vi khảo sát: 9](#_Toc470111380)

[1.2.5. Nội dung khảo sát: 10](#_Toc470111381)

[1.2.6. Phiếu phỏng vấn. 10](#_Toc470111382)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 12](#_Toc470111383)

[2.1. Tìm hiểu về Java Swing 12](#_Toc470111384)

[2.1.1. Java Swing là gì 12](#_Toc470111385)

[2.1.2. Cấu trúc của Java Swing 12](#_Toc470111386)

[2.1.3. Đặc điểm của Java Swing 12](#_Toc470111387)

[2.1.3.1. Cảm quan pluggable 12](#_Toc470111388)

[2.1.3.2. Các thành phần lightweight 13](#_Toc470111389)

[2.1.3.3. Các đặc điểm mở rộng 14](#_Toc470111390)

[2.1.4. Các gói và lớp của Swing 14](#_Toc470111391)

[2.1.4.1. javax.accessibility 14](#_Toc470111392)

[2.1.4.2. javax.swing 15](#_Toc470111393)

[2.1.4.3. javax.swing.border 15](#_Toc470111394)

[2.1.4.4. javax.swing.colorchooser 15](#_Toc470111395)

[2.1.4.5. javax.swing.event 15](#_Toc470111396)

[2.1.4.6. javax.swing.filechooser 15](#_Toc470111397)

[2.1.4.7. javax.swing.plaf 15](#_Toc470111398)

[2.1.4.8. javax.swing.table 16](#_Toc470111399)

[2.1.4.9. javax.swing.text 16](#_Toc470111400)

[2.1.4.10. javax.swing.text.html 16](#_Toc470111401)

[2.1.4.11. javax.swing.text.html.parser 16](#_Toc470111402)

[2.1.4.12. javax.swing.text.rtf 16](#_Toc470111403)

[3.1.4.13. javax.swing.tree 16](#_Toc470111404)

[3.1.4.14. javax.swing.undo 16](#_Toc470111405)

[2.1.5. Một số ví dụ minh họa 16](#_Toc470111406)

[2.2 Tìm hiểu về thuật toán Dijkstra 18](#_Toc470111407)

[2.2.1 Khái niệm 18](#_Toc470111408)

[2.2.2 Ý tưởng thuật toán 18](#_Toc470111409)

[2.2.2.1 Bài Toán 18](#_Toc470111410)

[2.2.2.2. Ý tưởng 18](#_Toc470111411)

[2.2.3. Mã giả 19](#_Toc470111412)

[2.2.4. Thuật toán 20](#_Toc470111413)

[2.2.5. Ví dụ 20](#_Toc470111414)

[2.3. Tìm hiểu về thuật toán Prim 24](#_Toc470111415)

[2.3.1. Mô tả bài toán 24](#_Toc470111416)

[2.3.2. Ý tưởng thuật toán 24](#_Toc470111417)

[2.3.3. Ví dụ 25](#_Toc470111418)

[2.4 Tìm hiểu thuật toán duyệt theo chiều sâu DFS 29](#_Toc470111419)

[2.4.1. Mô tả bài toán 29](#_Toc470111420)

[2.4.2. Mã giả 29](#_Toc470111421)

[2.4.3 Ví dụ 30](#_Toc470111422)

[2.5. Tìm hiểu thuật toán duyệt theo chiều rộng BFS 33](#_Toc470111423)

[2.5.1. Mô tả bài toán 33](#_Toc470111424)

[2.5.2. Ý tưởng thuật toán 33](#_Toc470111425)

[2.5.3. Ví dụ 33](#_Toc470111426)

[**CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG** 36](#_Toc470111427)

[**3.1. Danh sách tác nhân.** 36](#_Toc470111428)

[**3.2.** **Danh sách Usecase** 36](#_Toc470111429)

[**3.3.** **Sơ đồ usecase** 37](#_Toc470111430)

[**3.4.** **Đặc tả usecase** 37](#_Toc470111431)

[*3.4.1.* *UC001 - Đồ thị có hướng* 37](#_Toc470111432)

[*3.4.2.* *UC002 – Đồ thị vô hướng* 38](#_Toc470111433)

[*3.4.3.* *UC003 – Vẽ bằng tay* 38](#_Toc470111434)

[*3.4.4.* *UC004 – Mẫu có sẵn* 39](#_Toc470111435)

[*3.4.5.* *UC005 – Duyệt 1 lần* 39](#_Toc470111436)

[*3.4.6.* *UC006 – Duyệt từng bước* 39](#_Toc470111437)

[*3.4.7.* *UC007 – Tạo nút* 40](#_Toc470111438)

[*3.4.8.* *UC008 – Tạo cạnh nối* 40](#_Toc470111439)

[*3.4.9.* *UC009 – Di chuyển nút* 40](#_Toc470111440)

[*3.4.10.* *UC010 – Làm mới đồ thị* 41](#_Toc470111441)

[*3.4.11.* *UC011 – Mở File* 41](#_Toc470111442)

[*3.4.12.* *UC012 – Lưu File* 41](#_Toc470111443)

[**CHƯƠNG 4: THIẾT KẾ GIAO DIỆN** 43](#_Toc470111444)

[4.1 Giao diện Main 43](#_Toc470111445)

[4.2 Giao diện them nút 43](#_Toc470111446)

[4.3 Giao diện Show đồ thị 44](#_Toc470111447)

[4.4 Giao diện mở file 44](#_Toc470111448)

[4.5. Giao diện Main thực tế 45](#_Toc470111449)

[**CHƯƠNG 5: TỔNG KẾT** 47](#_Toc470111450)

[5.1. Kết quả đạt được 47](#_Toc470111451)

[5.2. Hạn chế 47](#_Toc470111452)

[5.3. Hướng phát triển: 47](#_Toc470111453)

[**TÀI LIỆU THAM KHẢO** 48](#_Toc470111454)

# **CHƯƠNG 1. TỔNG QUAN**

## Giới Thiệu

Chương trình “Vẽ Graph trên Java” được thiết kế và cài đặt nhằm giúp cho công việc vẽ đồ thị được dễ dàng, chi tiết, đỡ tốn công sức, nhanh và chính xác hơn. Thay vì phải làm công việc vẽ đồ thị một cách thủ công, rườm rà, mất thời gian, mất công mất sức mà hiệu quả mang lại không cao thì ta có thể sử dụng chương trình này. Chương trình thiết kế các chức năng cần thiết để vẽ đồ thị cũng như cách duyệt đồ thị như tạo nút,nối các nút lại với nhau, chỉnh sữa trọng số, duyệt đồ thị chạy từng bước, mở file, lưu file…. Cùng với sự hỗ trợ tối đa về mô hình và các yêu cầu, chương trình này được phát triển theo tiêu chí hỗ trợ khả năng thao tác và khai thác phần mềm của người dùng với tiêu chí giao diện thân thiện dễ sử dụng, có hướng dẫn chi tiết, rõ ràng, dễ hiểu giúp cho người dùng dễ sử dụng mà không cần khả năng về tin học, kĩ năng sử dụng máy tính ở mức độ kinh nghiệm. Tuy nhiên chương trình khi đi vào thực tế có thể sẽ vẫn còn thiếu xót, chưa phù hợp với một số loại đồ thị mức độ cao hơn, không đáp ứng được yêu cầu sử dụng của một số người dùng xong nhóm em vẫn mong ý kiến của thầy cô để ứng dụng có thể hoàn thiện hơn.

* 1. **Khảo sát hệ thống**
     1. Nguồn khảo sát:
* Khảo sát tại trường Đại học Tôn Đức Thắng
* Khảo sát, tham khảo trực tuyến.
  + 1. Cách thức khảo sát:
* Test ứng dụng
  + 1. Đối tượng khảo sát:
* Đại học Tôn Đức Thắng: Giảng viên, sinh viên.
  + 1. Phạm vi khảo sát:
* Số 19, Đường Nguyễn Hữu Thọ, Phường Tân Phong, Quận 7, TP. Hồ Chí Minh.
  + 1. Nội dung khảo sát:
* Tạo nút
* Tạo cạnh nối
* Duyệt đồ thị
* Mở File/Lưu File
* D chuyển nút
  + 1. Phiếu phỏng vấn.

|  |  |
| --- | --- |
| Phiếu phỏng vấn | |
| Dự án: Xây dựng công cụ vẽ Graph trên Java | |
| Người hỏi: Nguyễn Văn Đồng, Nguyễn Nghĩa Dinh | **Người Trả lời: Sinh viên Trường Đại học Tôn Đức Thắng** |
| Câu hỏi | **Câu trả lời** |
| Khi vẽ Graph bạn dùng gì để vẽ? | Tôi thường hay vẽ lên vỡ. |
| Khi cần chỉnh sữa thông tin gì trên Graph thì bạn làm sao? | Siêng thì vẽ lại cái đồ thị khác, còn lười thì quẹt quẹt vài đường rồi ghi đề lên thôi.(cười) |
| Lúc cần duyệt đồ thị thì bạn sẽ duyệt bằng cách nào?(cười) | Cái này thì mình tự chạy trong đầu thôi. |
| Vậy l;àm thế nào để biết được cách duyệt đồ thị của mình đã chính xác hay chưa? | Mình tìm hiểu giải thuật chạy trước rồi cứ thế chạy từng bước trong đầu rồi dự đoán điểm đích nếu nó ra như mong muốn thì là đúng. |
| Các bạn thường vẽ Graph vào môn học nào? | Lý thuyết đồ thị. |
| Bao lâu để bạn vẽ ra một đồ thị trên giấy vừa đẹp vừa chính xác? | Mình nghĩ nếu vẽ cẩn thận thì khá tốn thời gian |
| Thời hạn nộp học phí của nhà trường trong bao lâu? | Không quá 10 ngày so với lúc bắt đầu ra thông báo cho phụ huynh. |
| Bạn thấy có tốn giấy khi vẽ các đồ thị không? | Tất nhiên là có rồi. (cười) |
| Khi muốn xem lại đồ thị mà mình đã vẽ qua rồi thì bạn làm thế nào? | Có lúc nhớ vẽ chỗ nào thì tìm để xem lại, có lúc không nhớ là mình đã vẽ ở đâu nên phải mất công vẽ lại.(Cười) |
| Bạn nghĩ sao nếu có một ứng dụng giúp bạn trong việc vẽ đồ thị trong môn Lý thuyết đồ thị? | Chức năng của nó sẽ như thế nào, có dễ dàng sử dụng không? |
| Chức năng bào gồm: tạo nút, tạo cạnh nối, xóa đỉnh, xóa cạnh, duyệt đồ thị, lưu đồ thị,… | Thế thì thật tuyệt vời. Mình rất moing đợi vào ứng dụng này.(Cười) |
| Bạn có thể cho biết chuyên ngành bạn đang học không ạ? | Mình học Công Nghệ Thông Tin |
| Cảm ơn bạn đã dành ít thời gian cho chúng mình , chúc bạn nhiều sức khỏe và thành công trong học tập. | Không có gì! Tạm biệt các bạn |

Table 1: Phiếu trả lời phỏng vấn

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

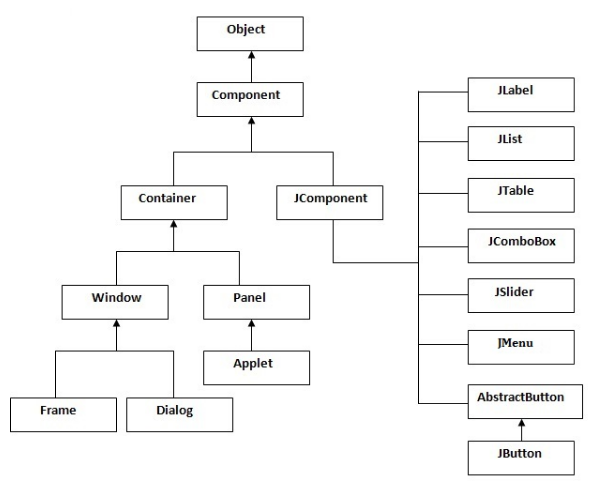
## 2.1. Tìm hiểu về Java Swing

### 2.1.1. Java Swing là gì

Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng Window-Based. Nó được xây dựng ở trên cùng của AWT (Abstract Windowing Toolkit) API và được viết hoàn toàn bằng Java.

Không giống AWT, Java Swing cung cấp các thành phần (Component) gọn nhẹ và độc lập nền tảng. Javax.swing. Package cung cấp các lớp cho Java Swing chảng hạn như JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser, …

### 2.1.2. Cấu trúc của Java Swing



### 2.1.3. Đặc điểm của Java Swing

### 2.1.3.1. **Cảm quan pluggable**

Một trong những tính thể hiện thú vị nhất ở các lớp Swing là khả năng viết những cảm quan (Look&Feels) cho mỗi thành phần, thậm chí thay đổi cảm quan vào thời điểm runtime. L&Fs trở thành một vấn đề quan trọng trong việc phát triển GUI qua 10 năm. Nhiều người dùng quen thuộc với kiểu giao diện Motif phổ biến trong Windows 3.1 và vẫn còn sử dụng rộng rãi trong platform Unix. Microsoft tạo ra L&F nhỏ gọn hơn trong các hệ điều hành 95/98/NT/2000 của họ. Ngoài ra, hệ thống máy tính Macintosh cũng có một L&F được thiết kế riêng mà hầu hết người dùng Apple đều cảm thấy thoải mái.  
Swing có khả năng thể hiện nhiều L&F khác nhau và hiện tại hỗ trợ các L&F bao gồm Windows, Unix Motif và Java Metal là L&F mặc định. Thêm vào đó, Swing cho phép người dùng chuyển đổi L&F vào thời điểm runtime mà không phải đóng ứng dụng. Theo cách này, một người dùng sẽ thử xem cảm quan nào là thể hiện tốt nhất đối với họ. Và nếu bạn cảm thấy bạn thật sự có tham vọng là một nhà phát triển, bạn hãy tạo L&F cho riêng bạn đối với mỗi thành phần Swing.

### **2.1.3.2. Các thành phần lightweight**

Hầu hết các thành phần Swing đều không nặng nề. Theo nghĩa đen, nghĩa là những thành phần này độc lập trên những L&F hiển thị chúng. Thay vào đó, chúng sử dụng môi trường đồ họa gốc đơn giản để vẽ chúng trên màn hình.  
Khả năng tạo ra những thành phần lightweight lần đầu tiên được đề cập trong JDK 1.1, mặc dù những thành phần AWT chính lại không nhận được những đặc điểm tiện lợi của nó. Trước đó, các lập trình viên không có một sự lựa chọn nào nhưng có những lớp con java.awt.Canvas hoặc java.awt.Panel để những lập trình viên có thể tạo ra những thành phần của riêng họ. Từ 2 lớp này, Java xác định một đối tượng opaque từ hệ điều hành đang chạy để thay thế thành phần này, tác động mạnh đến mỗi thành phần để tương tác nếu như bản thân nó là cửa sổ, và vì thế sẽ được đặt trên hình chữ nhật, đường viền đơn (solid). Vì thế những thành phần này mang tên gọi “heavyweight” bởi vì chúng nắm giữ những phần mở rộng ở mức độ gốc mà Java không sử dụng.  
  
Với các thành phần lightweight, mỗi thành phần thể hiện bản thân nó bằng cách sử dụng đồ họa gốc của đối tượng Graphics. Chúng luôn thể hiện bản thân chúng trên bề mặt của các thành phần heavyweight cao nhất mà chúng được chứa trong đó. Những lớp này cho phép các lập trình viên thể hiện một cách nhanh chóng ngữ cảnh đồ họa của khung chứa (container). Kết quả là tiết kiệm được vùng nhớ, chạy nhanh hơn những gì có sẵn trước đó.   
Hầu hết các thành phần Swing là lightweight, chỉ có một vài container top-level là không phải. Thiết kế này cho phép các lập trình viên vẽ và vẽ lại L&F của ứng dụng của họ vào thời điểm runtime kết hợp chúng vào cảm quan của hệ điều hành chạy ứng dụng đó. Thêm vào đó, cách thiết kế các thành phần Swing cho phép hỗ trợ dễ dàng chỉnh sửa cách hành xử của các thành phần.

### **2.1.3.3. Các đặc điểm mở rộng**

Những đặc điểm riêng khác phân biệt Swing với những thành phần AWT cũ  
Swing có rất nhiều những thành phần mới như table, tree, slider, spinner, progress bar, internal frame và text. Những thành phần Swing hỗ trợ việc thay thế các inset của chúng bằng một số lượng tùy ý các border lồng bên trong.  
Các thành phần Swing có các Tooltip đặt bên trên chúng. Bạn có thể tùy chọn các sự kiện bàn phím cho các thành phần, định nghĩa chúng hoạt động như thế nào với những phím nóng đã cho.  
Ngoài ra còn có hỗ trợ debug cho việc hiển thị những thành phần Swing

### 2.1.4. Các gói và lớp của Swing

### 2.1.4.1. **javax.accessibility**

Chứa các lớp và các giao tiếp mà có thể được sử dụng cho phép công nghệ trợ giúp để tương tác với các thành phần của Swing. Các công nghệ trợ giúp đề cập một dãy các item, từ bộ đọc văn bản đến phóng đại màn hình. Mặc dù các lớp accessibility là những công nghệ không thuộc các thành phần Swing nhưng chúng được sử dụng mở rộng thông qua các thành phần Swing.

### 2.1.4.2. **javax.swing**

Chứa các thành phần cốt lõi của Swing, bao gốm hầu hết các mô hình giao tiếp và các lớp hỗ trợ

### 2.1.4.3. **javax.swing.border**

Chứa định nghĩa các lớp đường viền trừu tượng cũng như 8 đường viền được định nghĩa. Border không phải là những thành phần, thay vì thế, chúng là những yếu tố đồ họa đặc biệt mà Swing sử dụng như những thuộc tính và đặt chúng xung quanh các thành phần. Nếu như bạn muốn tạo ra border của riêng bạn, bạn có thể tạo lớp con từ những border có sẵn trong gói này, hoặc bạn có thể viết mã cho một lớpmới.

### 2.1.4.4. **javax.swing.colorchooser**

Hỗ trợ cho thành phần JColorChooser

### 2.1.4.5. **javax.swing.event**

Định nghĩa những listener mới và những event mà những thành phần Swing sử dụng để giao tiếp thông tin với các class.

### 2.1.4.6. **javax.swing.filechooser**

Hỗ trợ cho thành phần JFileChooser

### 2.1.4.7. **javax.swing.plaf**

Định nghĩa những yếu tố duy nhất tạo nên L&F cho mỗi thành phần Swing. Trong gói này chứa các gói con là javax.swing.plaf.basic, javax.swing.plaf.metal và javax.swing.plaf.multi

### 2.1.4.8. **javax.swing.table**

Cung cấp các mô hình và quan sát cho thành phần table, cho phép bạn sắp xếp thông riêng biệt trong định dạng lưới với sự xuất hiện tương tự như bảng tính.

### 2.1.4.9. **javax.swing.text**

Cung cấp các điểm của các lớp và giao tiếp dựa trên văn bản hỗ trợ thiết kế thông dụng được biết đến như document/view.

### 2.1.4.10. **javax.swing.text.html**

Sử dụng kỹ thuật đọc và định dang văn bản HTML

### 2.1.4.11. **javax.swing.text.html.parser**

Hỗ trợ việc phân tích HTML

### 2.1.4.12. **javax.swing.text.rtf**

Sử dụng kỹ thuật đọc và định dạng Rich Text Format

### 3.1.4.13. **javax.swing.tree**

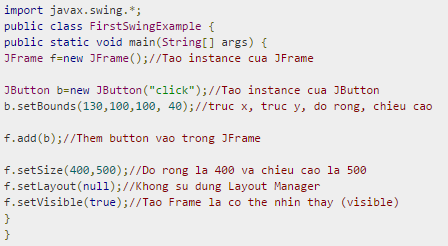
Định nghĩa các mô hình và quan sát thành phần một cây phân cấp , có thể thay thế một cấu trúc file hoặc một dãy các thuộc tính.

### 3.1.4.14. **javax.swing.undo**

Chứa những chức năng cần thiết cho việc bổ sung chức năng undo.

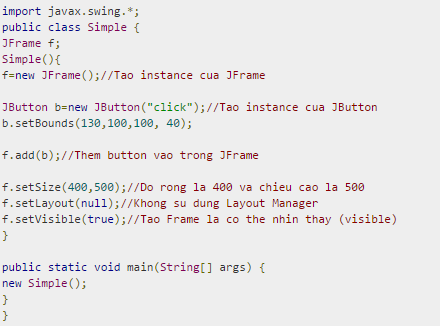
### 2.1.5. Một số ví dụ minh họa

Trong chương trình ví dụ về Java Swing sau, chúng ta tạo một button và thêm nó trên đối tượng JFrame bên trong phương thức main().



Ví dụ về Swing bởi quan hệ kết hợp bên trong Constructor

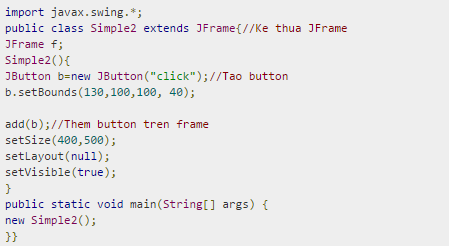
Chúng ta có thể viết tất cả code để tạo JFrame, JButton và lời gọi phương thức bên trong Java Constructor.



Phương thức setBounds(int xaxis, int yaxis, int width, int height) được sử dụng trong ví dụ trên để thiết lập vị trí của button.

Ví dụ về Swing bởi quan hệ kế thừa

Bạn cũng có thể kế thừa lớp JFrame, vì thế bạn không cần tạo instance của lớp JFrame một cách tường minh.



## 2.2 Tìm hiểu về thuật toán Dijkstra

### 2.2.1 Khái niệm

**Giải thuật Dijkstra**, mang tên của 1 nhà khoa học máy tính người Hà Lan **Edsger W. Dijkstra**, là một thuật toán giải quyết bài toán đường đi ngắn nhất trong một đồ thị có hướng **không có cạnh trọng số âm**. Ứng dụng lớn nhất của thuật toán này là trong công nghệ Hệ thống định vị toàn cầu (GPS).

### 2.2.2 Ý tưởng thuật toán

### 2.2.2.1 Bài Toán

Cho 1 đồ thị có hướng **G = (V, E)** với các cạnh có trọng số không âm, có dữ liệu nhập vào là ma trận trọng số **L** và 2 đỉnh **x, y** cho trước. Việc ta cần làm là tìm đường đi ngắn nhất từ **x** đến **y**trong đồ thị **G**.

### 2.2.2.2. Ý tưởng

Dò tìm bằng cách thử đi qua các nút trung gian

Nếu phát hiện đường đi qua đỉnh trung gian ngắn hơn đường đi hiện tại thì sẽ cập nhật đường đi mới, đồng thời chỉnh sữa các thông tin liên quan

Sử dụng 2 mảng để lưu trữ tạm thời

* Lưu trữ đường đi ngắn nhất hiện tại từ x đến y
* Lưu trữ đỉnh nằm trước y trên đường đi ngắn nhất hiện tại

### 2.2.3. Mã giả

Gán *T* = ø*; p(v) = NULL với mọi đỉnh v*

*d(a)=0; /\* a là đỉnh xuất phát*

*Với mỗi đỉnh v còn lại thì d(v) = ∞;*

Repeat

u =(u∉T | d(u) là bé nhất);

T = T ∪ {u};

for ((v là đỉnh kề của u) và v∉T)

if d(v) > d(u) + w(u,v) then

d(v) = d(u) + w(u,v)

p(v) = u

Until ( đỉnh đích chứa trong tập T)

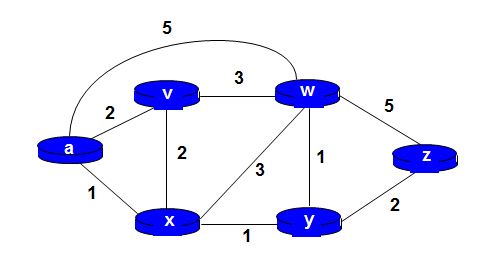
### 2.2.4. Thuật toán

Ta gọi **length[v]** là giá trị đường đi ngắn nhất từ **x -> v**, có thể hiểu **length[v]** là giá trị đường đi ngắn nhất trong các đường đi từ đỉnh **x** qua các đỉnh trong tập hợp S (nếu có) rồi đến **v**.

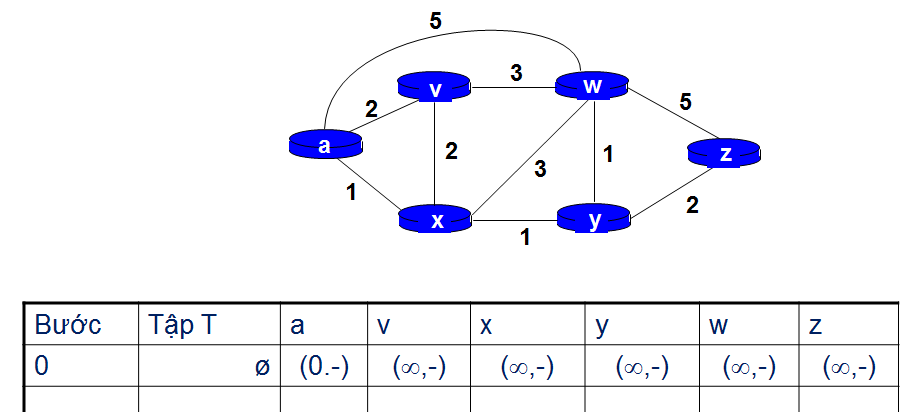
* Khởi tạo các mảng **n**phần tử: **label, length, prev**. Gán **label[k] = 1, length[k] = -1 (inf), prev[k] = -1** với **k** chạy từ **0 -> n – 1**. Gán **length[first] = 0**
* Chọn đỉnh **v** trong mảng sao cho **length[k]** là nhỏ nhất. Sau đó gán **label[k] = 0**(Đã đánh dấu)
* Tạo vòng lặp với biến chạy **k**, xét nếu **label[k] = 1**(Chưa đánh dấu) và có đường đi từ **v -> k**: Nếu **length[k] > length[v] +**trọng số từ **v -> k** hoặc **length[k] = inf**, có nghĩa là nếu ta tìm được 1 đường từ **v -> k** là nhỏ nhất, hoặc là chưa tìm được đường nào ngắn nhất (inf) => Gán **length[k] = length[v] +**trọng số **v -> k**, **prev[k] = v**(Tạo vết chân đỉnh trước đó).
* Nếu **label[last] = 0** (Đã đánh dấu đỉnh đến), kết thúc vòng lặp. Nếu không thì quay lại bước 2.

### 2.2.5. Ví dụ

Tìm đường đi ngắn nhất từ a đến w trong đồ thị sau:

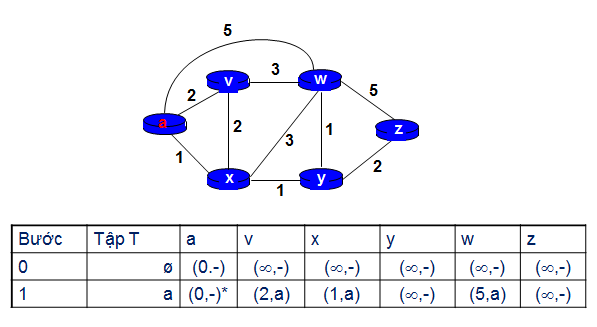


**Bước 0:** Ta đánh đấu đỉnh xuất phát bằng 0, các đỉnh còn lại bằng vô cực (a)



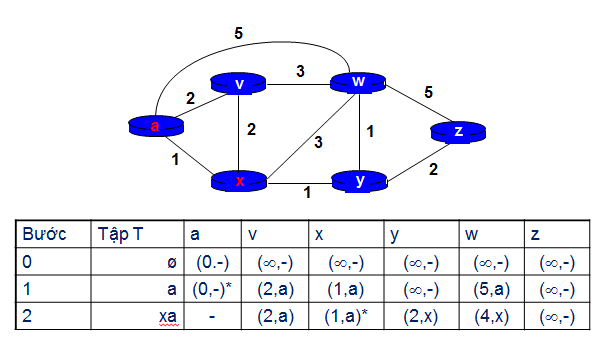
**Bước 1:**

* ại đỉnh a có x đi vào, d(a) = 1 < ∞. Cập nhật lại đỉnh x
* Tại đỉnh a có v đi vào, d(a) = 2 < ∞. Cập nhật lại đỉnh v
* Tại đỉnh a có w đi vào, d(a) = 5 < ∞. Cập nhật lại đỉnh w



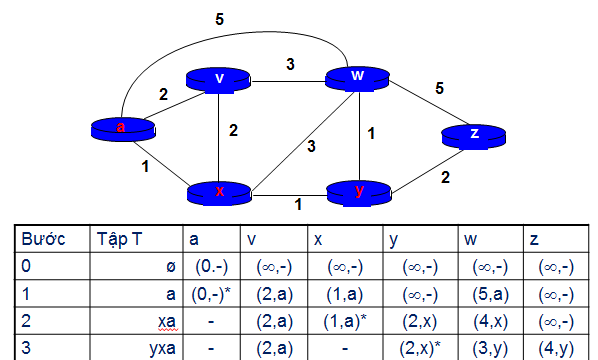
**Bước 2:** Tiếp tục chọn x làm gốc

* + Tại đỉnh x có y đi vào, d(x) = 2 < ∞. Cập nhật lại đỉnh y
  + Tại đỉnh x có w đi vào, d(x) = 4 < ∞. Cập nhật lại đỉnh w



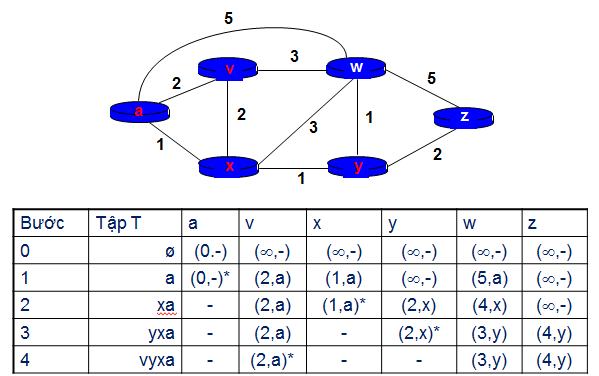
**Bước 3:** Tiếp tục chọn y làm gốc

* + Tại đỉnh y có z đi vào, d(y) = 4 < ∞. Cập nhật lại đỉnh z
  + Tại đỉnh y có w đi vào, d(y) = 3 < ∞. Cập nhật lại đỉnh w



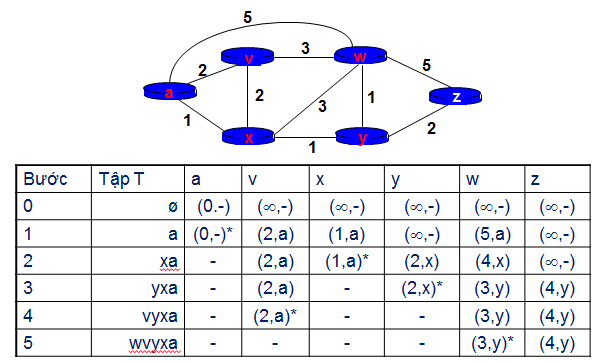
**Bước 4:** Tiếp tục chọn v làm gốc

* + Vì d(a) = 2 < d(y) = 3
  + Tại v không còn đỉnh nào kề nó mà chưa xét nên ta cập nhật lại

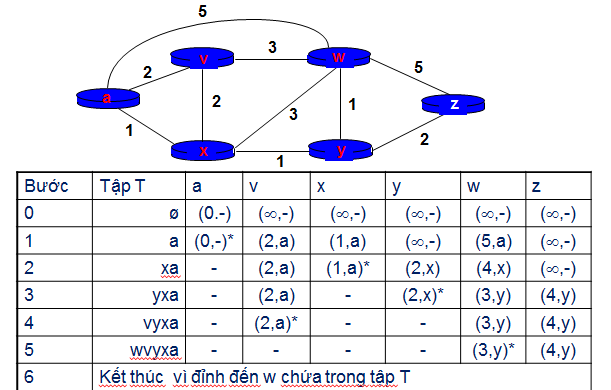


**Bước 5:** Tiếp tục chọn w làm gốc

* Lại tập đỉnh T do z đã được xét trước đó



**Bước 6:** Kết thúc. Vì đã có w trong tập T



## 2.3. Tìm hiểu về thuật toán Prim

### 2.3.1. Mô tả bài toán

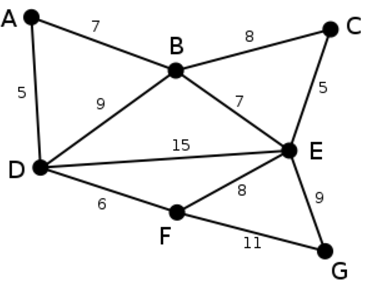
Cho đồ thị vô hướng có trọng số G=(V,E) hãy tìm đường đi sao cho tất cả các đỉnh điều có đường đi với nhau và tổng trọng số của đường đi là nhỏ nhất. Tức là tìm đồ thị con liên thông G' của G sao cho tổng trọng số của G’ là nhỏ nhất.

### 2.3.2. Ý tưởng thuật toán

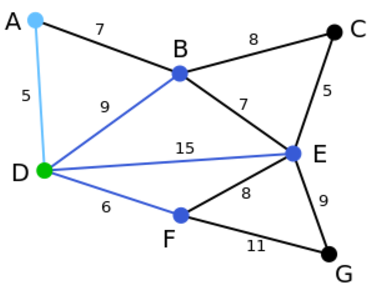
**Bước 1:** Xuất phát từ đỉnh k bất kỳ (thông thường chọn đỉnh đầu tiên) chọn một cạnh có trọng số nhỏ nhất liền kề với đỉnh k (min{A[k][j]}j=1..n) ta đánh dấu 2 đỉnh đi qua cạnh đó và số cạnh tìm được là 1. Chuyển sang Bước 2.[1]  
  
**Bước 2:** Tìm cạnh nhỏ nhất của đồ thị với điều kiện cạnh tìm được phải có 1 đỉnh chưa đánh dấu và 1 đỉnh đã đánh dấu. Nghĩa là, ta tìm min{A[i][j]}j=1..n, i=1..n sao cho i đánh đấu và j chưa đánh dấu để tránh trường hợp tạo thành chu trình con. Ta tăng số cạnh tìm được lên 1 và chuyển sang Bước 3. [1]  
  
**Bước 3:** Nếu số cạnh tìm được bằng n-1 kết thúc thuật toán, ngược lại quay về Bước 2. [1]

### 2.3.3. Ví dụ

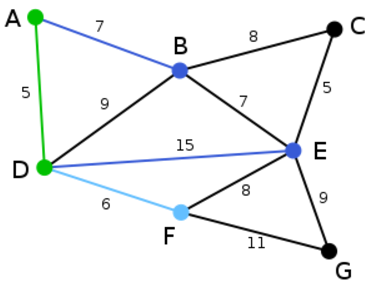
***Hình 1***: Đây là đồ thị có trọng số ban đầu. Các số là các trọng số của các cạnh. [1]



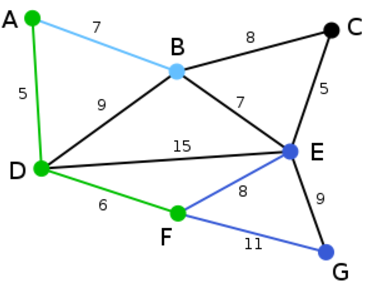
***Hình 2***: Chọn một cách tùy ý đỉnh **D** là đỉnh bắt đầu. Các đỉnh **A**, **B**, **E** và **F** đều được nối trực tiếp tới **D**bằng cạnh của đồ thị. **A** là đỉnh gần **D** nhất nên ta chọn **A** là đỉnh thứ hai của cây và thêm cạnh **AD** vào cây. [1]



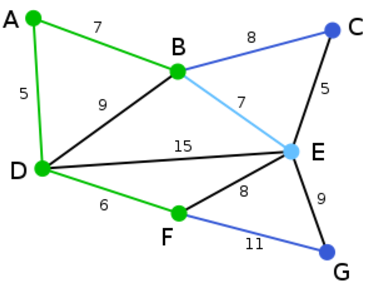
***Hình 3***: Đỉnh được chọn tiếp theo là đỉnh gần **D** hoặc **A**nhất. **B** có khoảng cách tới **D** bằng 9 và tới **A**bằng 7, **E** có khoảng cách tới cây hiện tại bằng 15, và **F** có khoảng cách bằng 6. **F** là đỉnh gần cây hiện tại nhất nên chọn đỉnh **F** và cạnh **DF**. [1]



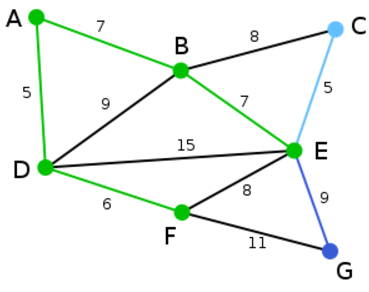
***Hình 4***: Thuật toán tiếp tục tương tự như bước trước. Chọn đỉnh **B** có khoảng cách tới **A** bằng [1]



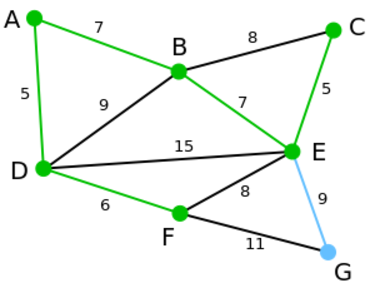
***Hình 5***: Ở bước này ta chọn giữa **C**, **E**, và **G**. **C** có khoảng cách tới **B** bằng 8, **E** có khoảng cách tới **B** bằng 7, và **G** có khoảng cách tới **F** bằng 11. **E**là đỉnh gần nhất, nên chọn đỉnh **E** và cạnh**BE**. [1]



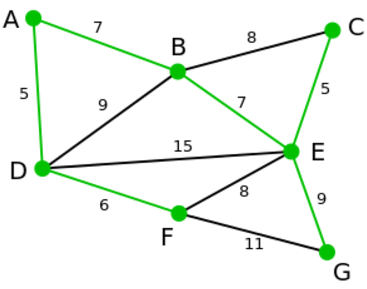
***Hình 6***: Ở bước này ta chọn giữa **C** và **G**. **C** có khoảng cách tới **E** bằng 5, và **G** có khoảng cách tới **E** bằng 9. Chọn **C** và cạnh **EC**. [1]



***Hình 7***: Quá trình tìm đỉnh kế tiếp không được tạo ra một chu trình so với ban đầu, để biết được đỉnh mới được chọn có trở thành chu trình hay không, ta cần kiểm tra đỉnh đó đã được đi qua hay chưa, nếu đã đi qua thì không thể đi tới, trong trường hợp này là đỉnh B và đỉnh F đều có trọng số mới là 8 (nhỏ nhất) nhưng ta không chọn. Như vậy, đỉnh **G** là đỉnh còn lại duy nhất. Nó có khoảng cách tới **F** bằng 11, và khoảng cách tới **E** bằng 9. **E** ở gần hơn nên chọn đỉnh **G** và cạnh **EG**. [1]



***Hình 8***: Hiện giờ tất cả các đỉnh đã nằm trong cây và cây bao trùm nhỏ nhất được tô màu xanh lá cây. Tổng trọng số của cây là 39. [1]



## 2.4 Tìm hiểu thuật toán duyệt theo chiều sâu DFS

### 2.4.1. Mô tả bài toán

Cho G = (V,E) là đồ thị có tập các đỉnh V và tập các cạnh E. v là một đỉnh trong V va u là đỉnh kề của v, sao cho u cũng thuộc V. Khi đó ta dán nhãn cho tất cả các đỉnh của đồ thị là 0. Chọn một đỉnh v thuộc tập V để bắt đầu duyệt. Gán nhãn đỉnh v này la 1-v đã được duyệt. Chọn đỉnh u trong tập V kề với đỉnh v mà nhãn là 0. Duyệt qua đỉnh u và gán nhãn u là 1. Tiếp tục quá trình duyệt đến khi tất cả các đỉnh đồ thị có nhãn là 1.

### 2.4.2. Mã giả

Void DFS (int v)

{

Gắn nhãn v đã duyệt;

For (u = 1; u <= n; u++)

If(u tồn tại trong danh sách kề V)

If(u có nhãn là 0)

{

Xử lí đỉnh u; //Gắn nhãn 1

DFS (u);

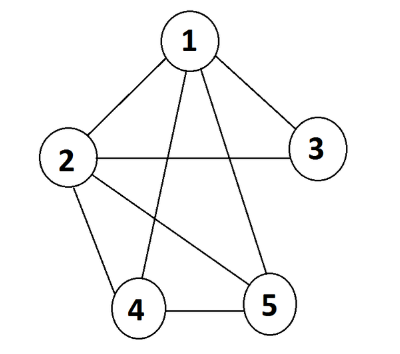
}

}

### 2.4.3 Ví dụ

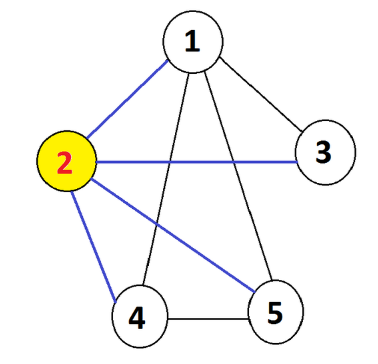
* Ban đầu ta có danh sách các đỉnh kề của đồ thị G=(V,E) có V={1,2,3,4,5} được lưu vào bảng như sau:[2]

|  |  |
| --- | --- |
| Đỉnh | Các đỉnh kề |
| 1 | 2,3,4,5 |
| **2** | 1,3,4,5 |
| 3 | 1,2 |
| 4 | 1,2,5 |
| 5 | 1,2,4 |



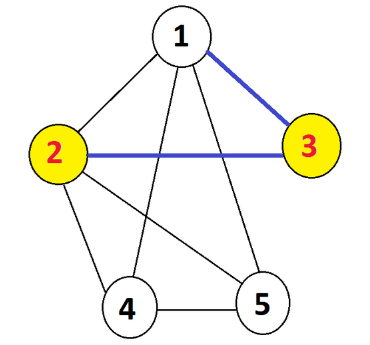
* Trước tiên, ta gán nhãn tất cả các đỉnh là 0 (chưa duyệt). Khi đỉnh nào được duyệt qua thì ta cập nhật lại nhãn cho đỉnh đó là 1. Chọn một đỉnh từ danh sách các đỉnh của G. Ở đây ta chọn đỉnh 2 làm đỉnh đầu tiên để bắt đầu duyệt. [2]

|  |  |  |
| --- | --- | --- |
| Đỉnh | Các đỉnh kề | Nhãn |
| 1 | 2,3,4,5 | 0 |
| **2** | 1,3,4,5 | 1 |
| 3 | 1,2 | 0 |
| 4 | 1,2,5 | 0 |
| 5 | 1,2,4 | 0 |



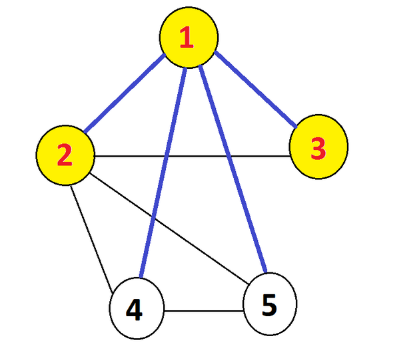
* Gán nhãn cho đỉnh 2 là 1 khi duyệt qua nó, tìm thấy các đỉnh kề của đỉnh 2 có đỉnh 3 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 3. [2]

|  |  |  |
| --- | --- | --- |
| Đỉnh | Các đỉnh kề | Nhãn |
| 1 | 2,3,4,5 | 0 |
| 2 | 3,4,5 | 1 |
| **3** | 1,2 | 1 |
| 4 | 1,2,5 | 0 |
| 5 | 1,2,4 | 0 |



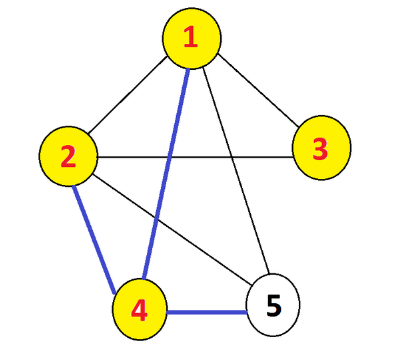
* Gán nhãn cho đỉnh 3 là 1 khi duyệt qua nó, tìm thấy các đỉnh kề của đỉnh 3 có đỉnh 1 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 1. [2]

|  |  |  |
| --- | --- | --- |
| Đỉnh | Các đỉnh kề | Nhãn |
| **1** | 2,3,4,5 | 1 |
| 2 | 3,4,5 | 1 |
| 3 | 1,2 | 1 |
| 4 | 1,2,5 | 0 |
| 5 | 1,2,4 | 0 |



* Gán nhãn cho đỉnh 1 là 1 khi duyệt qua nó, tìm thấy các đỉnh kề của đỉnh 1 có đỉnh 4 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 4. [2]

|  |  |  |
| --- | --- | --- |
| Đỉnh | Các đỉnh kề | Nhãn |
| 1 | 2,3,4,5 | 1 |
| 2 | 3,4,5 | 1 |
| 3 | 1,2 | 1 |
| **4** | 1,2,5 | 1 |
| 5 | 1,2,4 | 0 |



* Gán nhãn cho đỉnh 4 là 1 khi duyệt qua nó, tìm thấy các đỉnh kề của đỉnh 4 có đỉnh 5 có nhãn là 0(chưa được duyệt).Lặp lại quá trình với đỉnh 5. Đến đây, tất cả các đỉnh đã duyệt qua nên ta dừng thuật toán. Xuất ra dãy các đỉnh đã duyệt như sau: 2,3,1,4,5. [2]

## 2.5. Tìm hiểu thuật toán duyệt theo chiều rộng BFS

### 2.5.1. Mô tả bài toán

Giả sử ta có đồ thị G với các đỉnh ban đầu được đánh dấu là chưa duyệt (unvisited). Từ một đỉnh v nào đó ta bắt đầu duyệt như sau: đánh dấu v đã được duyệt, kế đến là duyệt tất cả các đỉnh kề với v. Khi ta duyệt một đỉnh v rồi đến đỉnh w thì các đỉnh kề của v được duyệt trước các đỉnh kề của w, vì vậy ta dùng một hàng để lưu trữ các nút theo thứ tự được duyệt để có thể duyệt các đỉnh kề với chúng. Ta cũng dùng mảng một chiều mark để đánh dấu một nút là đã duyệt hay chưa, tương tự như duyệt theo chiều sâu.

### 2.5.2. Ý tưởng thuật toán

Từ một đỉnh (nút) gốc ban đầu là đỉnh đang xét, xác định và lần lượt duyệt các đỉnh kề xung quanh đỉnh gốc vừa xét. Tiếp tục quá trình duyệt qua các đỉnh kề đỉnh vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua tất cả các đỉnh.

**Bước 1**: Tập Open chứa đỉnh gốc s chờ được xét.

**Bước 2**: Kiểm tra tập Open có rỗng không.

Nếu tập Open không rỗng, lấy một đỉnh ra khỏi tập Open làm đỉnh đang xét p. Nếu p là đỉnh g cần tìm, kết thúc tìm kiếm.

Nếu tập Open rỗng, tiến đến bước 4.

**Bước 3**: Đưa đỉnh p vào tập Close, sau đó xác định các đỉnh kề với đỉnh p vừa xét. Nếu các đỉnh kề không thuộc tập Close, đưa chúng vào cuối tập Open. Quay lại bước 2.

**Bước 4**: Kết luận không tìm ra đỉnh đích cần tìm.

### 2.5.3. Ví dụ

Ví dụ duyệt theo chiều rộng đồ thị hình trên. Giả sử bắt đầu duyệt từ A. Duyệt A, kế đến duyệt tất cả các đỉnh kề với A; đó là B, C, D theo thứ tự đó. Kế tiếp là duyệt các đỉnh kề của B, C, D theo thứ tự đó. Vậy các nút được duyệt tiếp theo là F, E,G. Có thể minh hoạ hoạt động của hàng trong phép duyệt trên như sau:

* Duyệt A nghĩa là đánh dấu visited và đưa nó vào hàng:

|  |  |  |
| --- | --- | --- |
|  | A |  |

* Kế đến duyệt tất cả các đỉnh kề với đỉnh đầu hàng mà chưa được duyệt; tức là ta loại A khỏi hàng, duyệt B, C, D và đưa chúng vào hàng, bây giờ hàng chứa các đỉnh B, C, D.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | B | C | D |  |

* Kế đến B được lấy ra khỏi hàng và các đỉnh kề với B mà chưa được duyệt, đó là F, sẽ được duyệt, và F được đưa vào hàng đợi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | C | D | F |  |

* Kế đến thì C được lấy ra khỏi hàng và các đỉnh kề với C mà chưa được duyệt sẽ được duyệt. Không có đỉnh nào như vậy, nên bước này không có thêm đỉnh nào được duyệt.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | D | F |  |  |

* Kế đến thì D được lấy ra khỏi hàng và duyệt các đỉnh kề chưa duyệt của D, tức là E, G được duyệt. E, G được đưa vào hàng đợi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | F | E | G |  |

* Tiếp tục, F được lấy ra khỏi hàng. Không có đỉnh nào kề với F mà chưa được duyệt. Vậy không duyệt thêm đỉnh nào.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E | G | D |  |

* Tương tự như F, E rồi đến G được lấy ra khỏi hàng. Hàng trở thành rỗng và giải thuật kết thúc

# **CHƯƠNG 3: PHÂN TÍCH HỆ THỐNG**

**3.1. Danh sách tác nhân.**

|  |  |  |
| --- | --- | --- |
| STT | Tên tác nhân | Mô tả |
| 01 | User | Có thể tạo được các nút, nối cạnh, duyệt đồ thị, mở file, lưu file,… |

Table 2: Danh sách tác nhân

* 1. **Danh sách Usecase**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Code | Tên Usecase | Mô tả chức năng |
| 01 | UC001 | Đồ thị có hướng | Cho phép người dùng chọn đồ thị có hướng |
| 02 | UC002 | Đồ thị vô hướng | Cho phép người dùng chọn đồ thị vô hướng |
| 03 | UC003 | Vẽ bằng tay | Cho phép người dùng vẽ đồ thị thủ công bằng tay |
| 04 | UC004 | Mẫu có sẵn | Cho phép người dùng tạo ra các đồ thị bằng các mẫu có sẵn |
| 05 | UC005 | Duyệt 1 lần | Cho phép người dùng thấy được kết quả cuối cùng khi duyệt đồ thị |
| 06 | UC006 | Duyệt từng bước | Cho phép người dùng thấy kết quả duyệt từng bước trong đồ thị |
| 07 | UC007 | Tạo nút | Cho phép người dùng tạo các nút của đồ thị |
| 08 | UC008 | Tạo cạnh nối | Cho phép người dùng nối các nút lại với nhau |
| 09 | UC009 | Di chuyển nút | Cho phép người dùng di chuyển các nút sao cho cảm thấy ưng ý |
| 10 | UC010 | Làm mới đồ thị | Làm mới các bước duyệt đồ thị, để người dùng có thể còn điểm duyệt khác |
| 11 | UC011 | Mở File | Cho phép người dùng mở file Graph lưu ở dạng .dat |
| 12 | UC012 | Lưu File | Cho phép người dùng lưu file Graph ở dạng .dat |

Table 3Danh sách usecase

* 1. **Sơ đồ usecase**



Hình 1. Sơ đồ Usecase

* 1. **Đặc tả usecase**
     1. *UC001 - Đồ thị có hướng*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Đồ thị có hướng | |
| Mô tả | Cho phép người dùng chọn loại đồ thị có hướng | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào Radio Button đồ thị có hướng | |
| Hậu điều kiện | Cạnh nối giữa các nút sẽ có hướng | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Chọn loại đồ thị** | |
| Người dùng nhấn vào Radio Button Đồ thị có hướng | Sẽ xác nhận dữ liệu rồi show lên TextField |
| **Luồng phụ: Không show ra được đồ thị** | |
|  | Không show ra được đồ thị |
| Thông điệp |  | |

* + 1. *UC002 – Đồ thị vô hướng*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Đồ thị vô hướng | |
| Mô tả | Cho phép người dùng chọn loại đồ thị vô hướng | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào Radio Button đồ thị vô hướng | |
| Hậu điều kiện | Cạnh nối giữa các nút sẽ vô hướng | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Chọn loại đồ thị** | |
| Người dùng nhấn vào Radio Button Đồ thị vô hướng | Sẽ xác nhận dữ liệu rồi show lên TextField |
| **Luồng phụ: Không show ra được đồ thị** | |
|  | Không show ra được đồ thị |
| Thông điệp |  | |

* + 1. *UC003 – Vẽ bằng tay*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Vẽ bằng tay | |
| Mô tả | Cho phép người dùng vẽ đồ thị bằng tay | |
| Tác nhân | User | |
| Ràng buộc | Nhấn nút Radio Button Vẽ bằng tay | |
| Hậu điều kiện | Tạo ra được đồ thị theo ý mình | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Show ra đồ thị** | |
| Người dùng chọn vào radio button vẽ bằng tay và tiến hành vẽ đồ thị | Đồ thị sẽ được show ra bảng TextField theo ý muốn người dùng |
| **Luồng phụ: Không show ra được đồ thị** | |
|  | Không show ra được đồ thị |
| Thông điệp |  | |

* + 1. *UC004 – Mẫu có sẵn*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Mẫu có sẵn | |
| Mô tả | Cho phép người dùng vẽ đồ thị bằng mẫu có sẵn | |
| Tác nhân | User | |
| Ràng buộc | Nhấn nút Radio Button mẫu có sẵn | |
| Hậu điều kiện | Tạo ra được đồ thị bằng mẫu có sẵn | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Show ra đồ thị** | |
| Người dùng chọn vào radio button mẫu có sẵn | Đồ thị sẽ được show ra bảng TextField |
| **Luồng phụ: Không show ra được đồ thị** | |
|  | Không show ra được đồ thị |
| Thông điệp |  | |

* + 1. *UC005 – Duyệt 1 lần*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Quản lý trẻ | |
| Mô tả | Cho phép người dùng thấy được kết quả cuối cùng khi duyệt đồ thị | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào nút Duyệt 1 lần | |
| Tiền điều kiện | Phải vẽ được đồ thị và chọn điểm bắt đầu và kết thúc | |
| Hậu điều kiện | Hiển thị kết quả khi duyệt đồ thị | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Duyệt thành công** | |
| Người dùng nhấn vào button Duyệt 1 lần | Sẽ show ra kết quả duyệt đồ thị từ điểm bắt đầu đến điểm mong muốn |
| **Luồng phụ: Duyệt không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC006 – Duyệt từng bước*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Duyệt từng bước | |
| Mô tả | Cho phép người dùng thấy được kết quả từng bước duyệt | |
| Tác nhân | User | |
| Ràng buộc | Nhấn nút Duyệt từng bước | |
| Tiền điều kiện | Phải vẽ được đồ thị và chọn điểm bắt đầu và điểm kết thúc | |
| Hậu điều kiện | Hiển thị kết quả khi duyệt đồ thị | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Duyệt thành công** | |
| Người dùng nhấn vào button Duyệt từng bước | Sẽ show ra được kết quả từng bước chạy của giải thuật |
| **Luồng phụ: Duyệt không thành công** | |
|  | Không show ra được kết quả |

* + 1. *UC007 – Tạo nút*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Tạo nút | |
| Mô tả | Cho phép người dùng tạo các nút của đồ thị | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào tạo nút | |
| Hậu điều kiện | Hiển thị các nút ra TextField | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Tạo thành công** | |
| Người dùng nhấn vào tạo nút | Xác nhận thông tin và show ra nút khi người dùng vẽ nút ra TextField |
| **Luồng phụ: Tạo không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC008 – Tạo cạnh nối*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Tạo cạnh nối | |
| Mô tả | Cho phép người dùng tạo cạnh nối giữa các nút với nhau | |
| Tác nhân | User | |
| Ràng buộc | Nhấn nút Tạo cạnh nối | |
| Tiền điều kiện | Có các nút trên TextField | |
| Hậu điều kiện | Hiển thị ra Graph | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Nối thành công** | |
| Người dùng chọn tạo cạnh nối và nối giữa các nút với nhau | Sẽ show ra Graph mà người dugn2 đã vẽ |
| **Luồng phụ: Nối không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC009 – Di chuyển nút*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Di chuyển nút | |
| Mô tả | Cho phép người dùng di chuyển nút ở vị trí nào tùy thích | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào Nút Di Chuyển | |
| Tiền điều kiện | Đã có nút trên TextField | |
| Hậu điều kiện | Di chuyển các nút đến vị trí tùy thích | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Di chuyển thành công** | |
| Người dùng nút di chuyển và tiến hành di chuyển các nút | Sẽ xác nhận dữ liệu và di chuyễn các nút theo ý muốn của người dùng |
| **Luồng phụ: Di chuyển không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC010 – Làm mới đồ thị*

|  |  |  |
| --- | --- | --- |
| Tên Use case | Làm mới đồ thị | |
| Mô tả | Cho phép người dùng làm mới đồ thị để duyệt lại đồ thị theo hướng khác | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào nút làm mới đồ thị | |
| Tiền điều kiện | Đã có Graph trên TextField | |
| Hậu điều kiện | Trả lại đồ thị lại như ban đầu chưa duyệt | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Làm mới thành công** | |
| Người dùng nhấn vào nút làm mới | Hệ thống sẽ xác nhận và làm mới đồ thị |
| **Luồng phụ: làm mới không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC011 – Mở File*

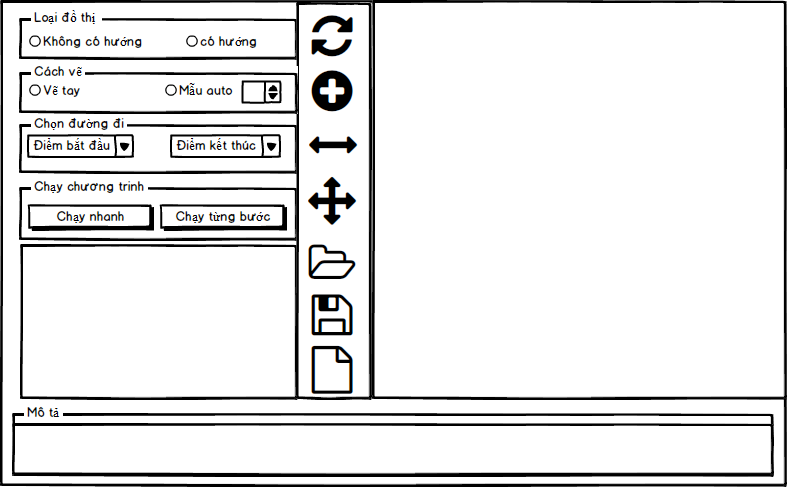
|  |  |  |
| --- | --- | --- |
| Tên Use case | Mở File | |
| Mô tả | Cho phép người dùng mở File Graph ở dạng .txt | |
| Tác nhân | User | |
| Ràng buộc | Nhấn vào nút mở File | |
| Tiền điều kiện | Phải có sẵn File dạng .txt | |
| Hậu điều kiện | Show đồ thị ra TextField | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Mở thành công** | |
| Người dùng nhấn vào nút Mở File | Sẽ xác nhận thông tin và show khung dữ liệu trên máy ra cho người dùng chọn |
| **Luồng phụ: Mở không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

* + 1. *UC012 – Lưu File*

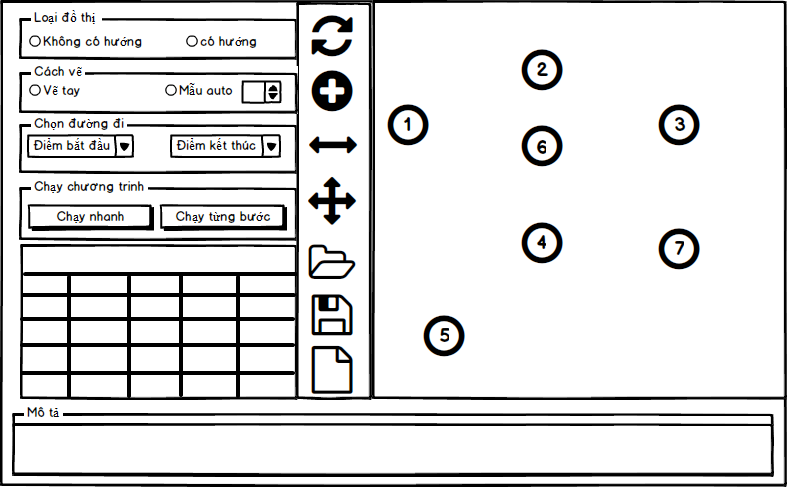
|  |  |  |
| --- | --- | --- |
| Tên Use case | Lưu File | |
| Mô tả | Cho phép người dùng Lưu File Graph ở dạng .txt | |
| Tác nhân | User | |
| Ràng buộc | Nhấn nút Lưu File | |
| Hậu điều kiện | Lưu file ở dạng .txt | |
| Hoạt động | **Tác nhân** | **Hệ thống** |
| **Luồng chính: Lưu thành công** | |
| Người dùng nhấp vào nút lưu file | Sẽ lấy thông tin Graph trên TextField và lưu vào máy ở dạng .txt |
| **Luồng phụ:Lưu không thành công** | |
|  | Không show ra được kết quả |
| Thông điệp |  | |

# **CHƯƠNG 4: THIẾT KẾ GIAO DIỆN**

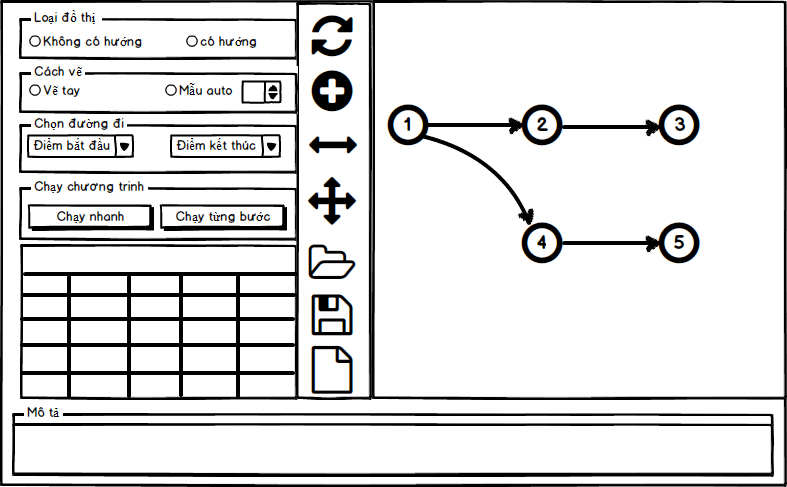
## 4.1 Giao diện Main



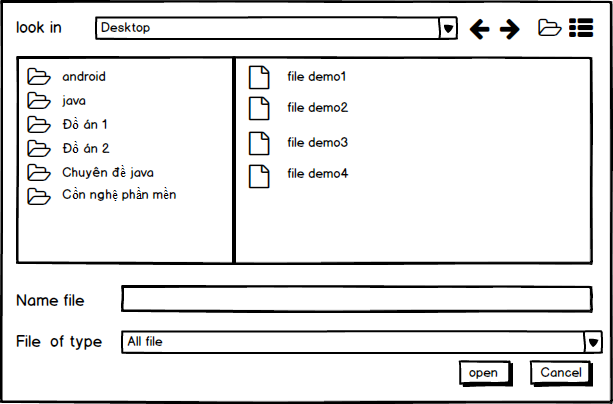
## 4.2 Giao diện them nút



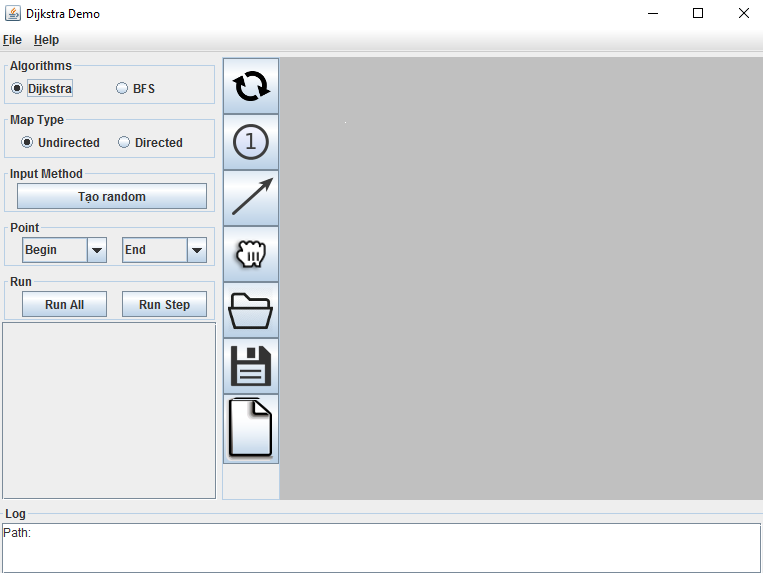
## 4.3 Giao diện Show đồ thị



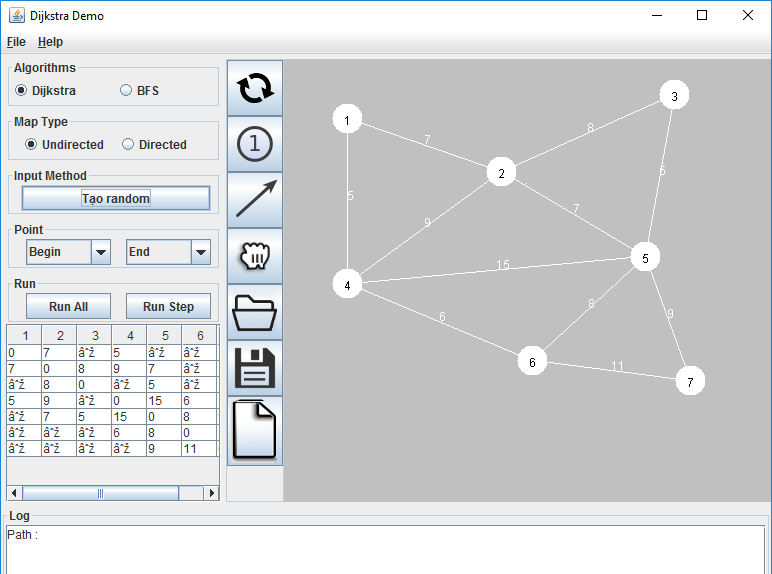
## 4.4 Giao diện mở file



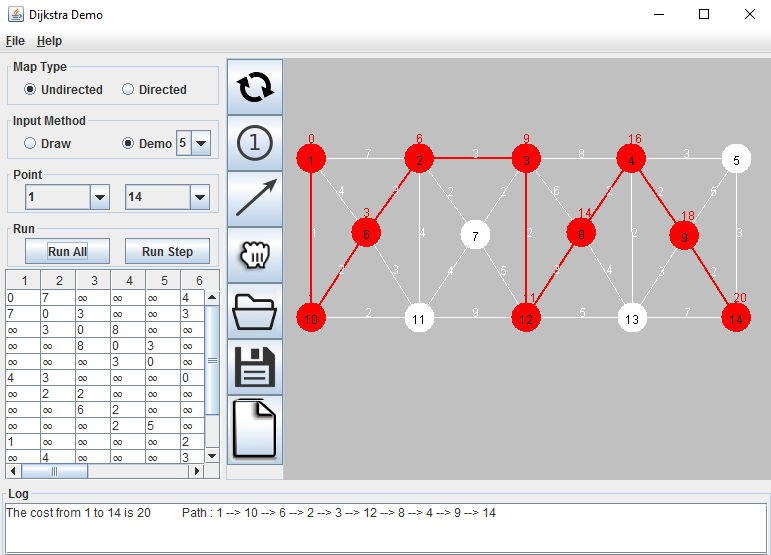
## 4.5. Giao diện Main thực tế



**4.6. Giao diện tạo đồ thị thực tế**



**4.7. Giao diện duyệt đồ thị thực tế**



# 

# **CHƯƠNG 5: TỔNG KẾT**

### 5.1. Kết quả đạt được

* Hiểu và cài đặt ứng dụng bằng ngôn ngữ Java. Sử dụng thuật toán Dijkstra, Prim, DFS, BFS. Chương trình chạy ổn định, kết quả thực hiện thuật toán sau khi cài đặt và chạy chương trình đúng với kết quả của phần lý thuyết
* Giao diện được thiết kế thân thiện với người dùng và dễ dàng sử dụng
* Chương trình được thiết kế dễ dàng cho việc sửa chữa, bảo trì khi có yêu cầu

### 5.2. Hạn chế

* Mặc dù có nhiều cố gắng hoàn thiện Đồ Án 1, nhưng đây là lần đầu tiên viết một chương trình hoàn chỉnh nên vẫn còn vài thiếu sót trong kỹ thuật lập trình, tổ chức dữ liệu, cũng như chương trình nguồn còn dài. Mặc khác do thời gian còn hạn chế nên ứng dụng làm ra còn nhiều thiếu sót
* Do kiến thức còn hạn chế nên một số thuật toán được cài đặt và áp dụng vào phần mềm sử dụng có lúc thiếu hiệu quả

### 5.3. Hướng phát triển:

Giao diện hướng đến người dùng với đặc tính dễ sử dụng, cần tối ưu chương trình ngắn hơn và chạy tốt hơn

# **TÀI LIỆU THAM KHẢO**

1. Phan Tán Tài

Bài giảng: QUY HOẠCH TUYẾN TÍNH VÀ QUY HOẠCH ĐỘNG

Khoa Công Nghệ Thông Tin – Đại Học Cần Thơ, 1998

1. Nguyễn Đức Nghĩa – Nguyễn Tô Thành

TOÁN RỜI RẠC

Nhà Xuất bản giáo dục 1998

1. Đinh Mạnh Tường

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Nhà Xuất bản Khoa Học và Kỹ Thuật, Hà Nội 2000

1. Nguyễn Tuấn Anh

LÝ THUYẾT ĐỒ THỊ VÀ ỨNG DỤNG

Nhà xuất bản Giáo dục Việt Nam 2012

1. Dương Anh Đức

NHẬP MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Đại Học Khoa Học Tự Nhiên

1. Một số website tham khảo:

[1] [2]- <https://vi.wikipedia.org/>

<https://google.com/>

<https://vietjack.com/>