CÁC THUẬT TOÁN THÔNG MINH

DÙNG A\* ĐỂ GIẢI QUYẾT BÀI TOÁN TACI

**GVHD: TS Vũ Thanh Nguyên**

2011

**Trần Văn Bình**

**07520026**

6/7/2011

Mục lục

[1. Giới thiệu 3](#_Toc295222447)

[2. Cấu trúc project 4](#_Toc295222448)

[3. Hướng dẫn sử dụng 12](#_Toc295222449)

[a. Cài đặt phát triển 12](#_Toc295222450)

[b. Tạo mới 13](#_Toc295222451)

[c. Run 14](#_Toc295222452)

[d. Menu Swap when equal 17](#_Toc295222453)

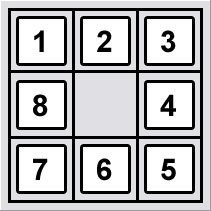
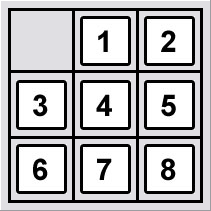
[e. Tùy chọn Swapping speed 18](#_Toc295222454)

[4. Tham khảo 19](#_Toc295222455)

# Giới thiệu

Bài toán Ta-ci đã từng là một trò chơi khá phổ biến, đôi lúc người ta còn gọi đây là bài toán 8-puzzle. Trò chơi bao gồm một hình vuông kích thước 3x3 ô. Có 8 ô có số, mỗi ô có một số từ 1 đến 8. Một ô còn trống. Mỗi lần di chuyển chỉ được di chuyển một ô nằm cạnh ô trống về phía ô trống. Vấn đề là từ một trạng thái ban đầu bất kỳ, làm sao đưa được về trạng thái cuối .

Có **hai** trạng thái cuối như sau:

*Trạng thái* ***A*** *Trạng thái* ***B***

Với mỗi trạng thái ban đầu, chỉ tìm được 1 trạng thái đích có thể đạt tới

Điều đầu tiên cần phải quan tâm để giải bài toán này là xác định trạng thái đích. Trạng thái đích được xác định dựa trên trạng thái ban đầu.

Đầu tiên hãy thử tính có bao nhiêu số bé hơn 8 ở sau ô chứa giá trị 8. Kết quả nhận được là 6 (những ô màu vàng)



Làm tương tự như vậy với ô có giá trị 6. Dễ thấy trong 3 ô (4, 7, 5) có 2 giá trị nhỏ 6 là 4, 5.



Làm như trên từ ô đầu tiên (2) tới ô cuối cùng (5) và cộng dồn các giá trị nhận được:

N = 1 + 6 + 1 + 0 + 2 + 0 + 1 + 0 = 11

Nếu N là số lẻ thì chúng ta chỉ có thể có đáp an là trạng thái A, ngược lại là trạng thái B

Chúng ta đã xác định được trạng thái đích cần đạt được, bây giờ bắt đầu tìm kiếm giải thuật để tìm ra đích.

Ở đây dùng thuật toán **A\*** để minh họa

Gọi G là số bước đã di chuyển ô trống

H là hàm heuristic, ước tính số hao tổn để tới trạng thái đích, tính bằng tổng các quãng đường của các ô ở vị trí sai để về tới vị trí đúng

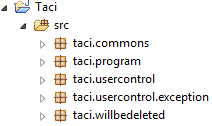
*F=G+H*

# Cấu trúc project

Chương trình minh họa được viết bằng ngôn ngữ lập trình Java, sử dụng IDE Eclipse

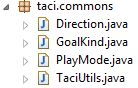
Để phát triển được chương trình cần cài đặt bộ **JDK1.6**

* 1. Cấu trúc



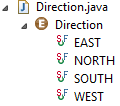
Project phát triển được chia làm 5 gói (package) như hình trên.

* 1. Gói *taci.commons*



Gói này chứa tất cả những hàm tiện ích, các enum được sử dụng nhiều lần trong chương trình.

* Enum **Direction** được dùng để xác định hướng đi của ô trống khi board chuyển từ trạng thái A sang trạng thái B. Có bốn hướng đi như hình dưới đây.



* Enum **GoalKind** được dùng để chỉ định trạng thái đích của một trạng thái ban đầu (ngẫu nhiên). Có hai trạng thái đích như đã đề cập ở phần giới thiệu (phần 1.) đó là A và B.



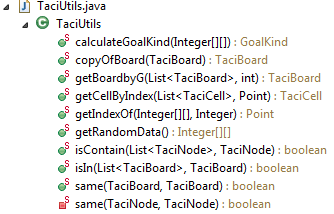
* Enum **PlayMode** được dùng để xác định chương trình đang ở mode nào. **AUTO** là mode chạy kết quả mà không hiễn thị chi tiết.

**STEP\_BY\_STEP** là mode chạy kết quả và hiễn thị chi tiết

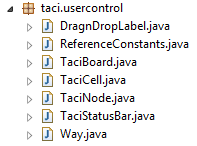
**NO\_PLAY** là mode idle



* Class **TaciUtils** đảm nhận việc cung cấp các hàm tiện ích dùng cho cả chương trình. Các hàm này là static.
* Hàm *calculateGoalKind* trả về kiểu trạng thái đích của trạng thái đầu vào. Kiểu trả về là Enum **GoalKind.**
* Hàm *copyOfBoard* trả về một **TaciBoard** là bản sao của bảng đầu vào
* Hàm *getCellByIndex* trả về một **TaciCell** trong board, được xác định bởi index. Cần chú ý, board là một danh sách các **TaciCell.**
* Hàm *getRandomData* gía trị trả về là trạng thái ban đầu ngẩu nhiên của board.
* Hàm *same* kiểm tra xem hai board hoặc hai node có giống nhau không. Ở đây **TaciCell**, và **TaciNode** có ý nghĩa tương đương nhau(đại diện cho 1 ô trên board) điểm khác nhau là **TaciCell** hỗ trợ đồ họa còn **TaciNode** chỉ đơn thuần lưu dữ liệu không hỗ trợ đồ họa.

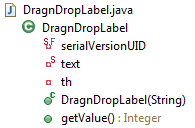


* 1. Gói *taci.usercontrol*



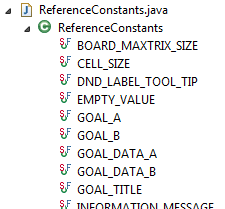
Gói này gồm các lớp chính của chương trình.

* Lớp **DragnDropLabel**



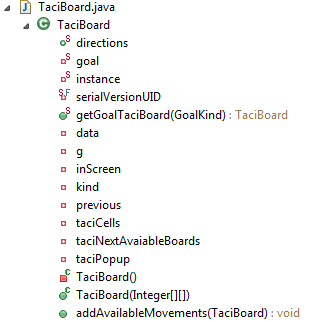
Được dùng trong tạo mới trạng thái ban đầu. Người dùng có thể tự do di chuyển và đổi chỗ các con số trên board để tạo ra trạng thái ban đầu như ý muốn. Lớp này kế thừa lớp **JLabel** của JDK6, **TransferHandler** class để quản lý việc Drag n Drop.

* Lớp **ReferenceConstants**



Chứa tất cả các hằng số được dùng trong chương trình.

* Lớp **TaciBoard**



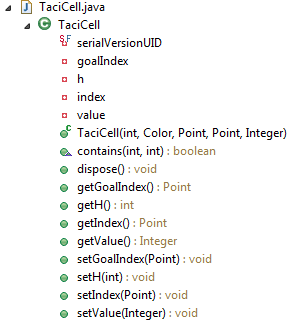
Mỗi board đại diện cho 1 trạng thái trong chuỗi trạng thái từ trạng thái ban đầu đến trạng thái đích. Mỗi board chứa một danh sách các TaciCell.

Mỗi TaciBoard cho biết giá trị H, G, và F hiện tại của board bằng tooltip dạng text HTML.

Mỗi board chứa những trạng thái tiếp theo cho trạng thái hiện tại

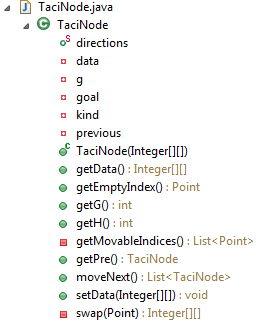
TaciBoard kế thừa từ JPanel của JDK6

* Lớp **TaciCell**



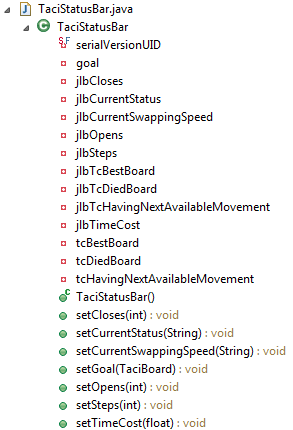
Mỗi TaciCell là một ô trên board. Mỗi TaciCell kế thừa từ JLabel. Hiễn thị đồ họa cho mỗi node của board. Mỗi cell hiễn thị thông tin về số bước còn lại để về trạng thái đích thông qua tooltip

* Lớp **TaciNode**



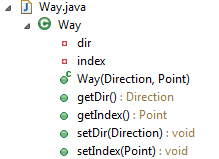
Tương tự TaciCell nhưng không hỗ trợ chế độ đồ họa.

* Lớp **TaciStatusBar**



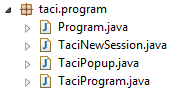
Lưu trạng thái của chương trình. Lưu thông tin về số board được mở, số board đã đóng và một vài chú thích khác.

* Lớp **Way**

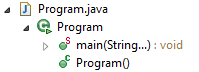


Gồm hai trường dữ liệu dir:Direction và index:Point. Cho ta biết tại vị trí (cell index) này sẽ đi hướng nào (dir).

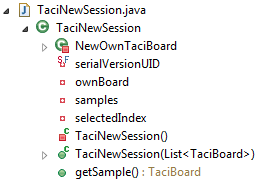
* 1. Gói *taci.program*



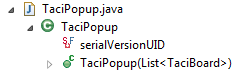
* Lớp **Program** lớp chứa hàm main chạy chương trình

****

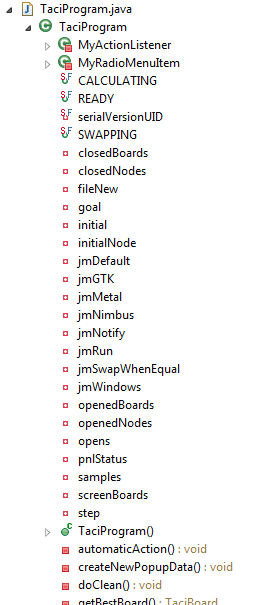
* Lớp **TaciNewSession** tạo mới một phiên làm việc



* Lớp **TaciPopup** hiễn thị chi tiết những board được mở ở bước tiếp theo



* Lớp **TaciProgram** đây là lớp trung tâm của cả chương trình



* **private** TaciBoard initial; lưu trữ board khởi tạo
* **private** TaciBoard goal; lưu trữ board đích tương ứng với board khởi tạo
* **private** List<TaciBoard> openedBoards = **new** ArrayList<TaciBoard>(); lưu trữ những board được mở, khi mở một trạng thái mới phải xét xem trạng thái đó đã mở chưa
* **private** List<TaciBoard> closedBoards = **new** ArrayList<TaciBoard>(); lưu trữ những board đã đóng khi mở một trạng thái mới phải xét xem trạng thái đó đã mở chưa.
* **private int** step = 0; lưu trữ số bước phải thực hiện để đi từ trạng thái đầu đến trạng thái đích

# Hướng dẫn sử dụng

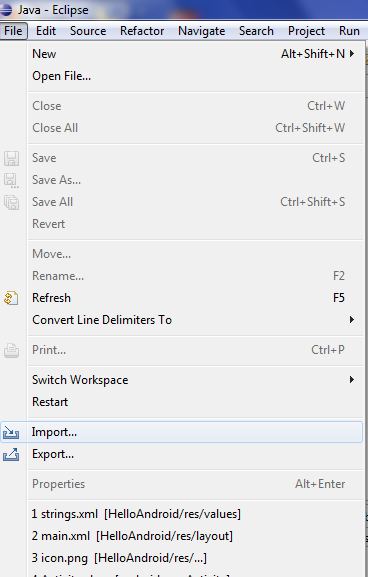
## Cài đặt phát triển

Cài bộ JDK 1.6 trở lên vào hệ thống.

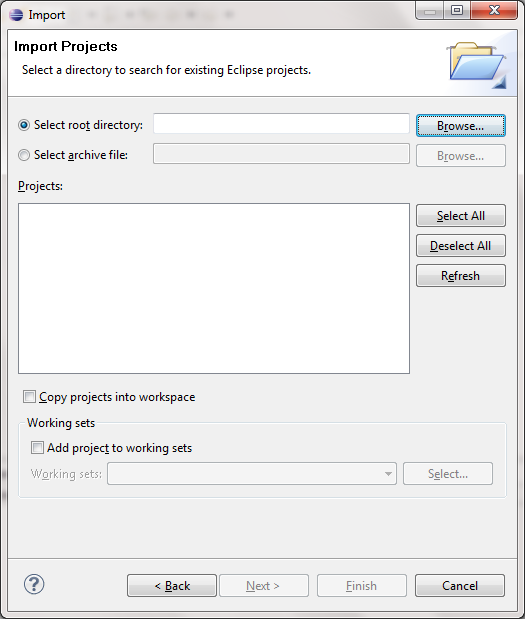
Giải nén file eclipse.rar và chạy eclipse

Import project từ source code như sau:

* Vào File 🡪 Import…



* Hộp thoại Import xuất hiện



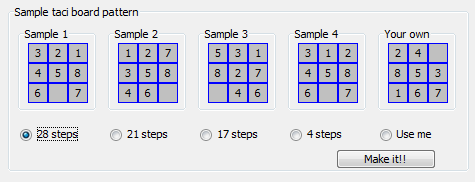
* Nhấn vào nút Browse… để browser đến thư mục chứa source code của project
* Click Finish

## Tạo mới

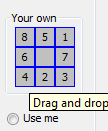
Double click vào file thực thi **Taci.jar** để khởi động chương trình

****

File 🡪 New



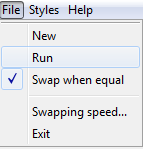
Chọn các board khởi tạo sample 1, 2, 3, 4 hoặc tạo mới (your own) bằng cách kéo thả các cell trong board.



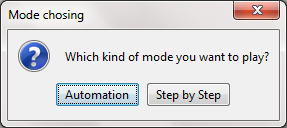
## Run

Để chạy được chương trình cần phải cài đặt Java Runtime Environment 1.6 update xx trở lên vào hệ thống. Bộ cài đặt môi trường này có thể download trên trang chủ của Oracle – Sun.

Sau khi chọn được board khởi tạo vào File 🡪 Run



Xuất hiện hộp thoại chọn kiểu run

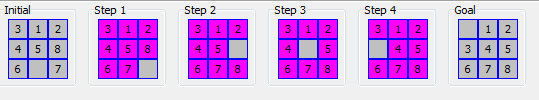


Có hai mode run

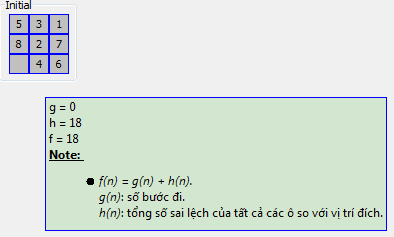
+ **Automation**: chạy chương trình mà không lưu trữ các bước trung gian thực hiện trực quan từng bước chuyển đổi trạng thái để đi đến trạng thái đích



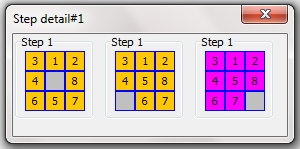
+ **Step by Step**: chay chương trình và lưu trữ chi tiết các bước trung gian (gồm những trạng thái được mở trung gian). Lưu ý khi chạy theo mode này là tốn nhiều bộ nhớ

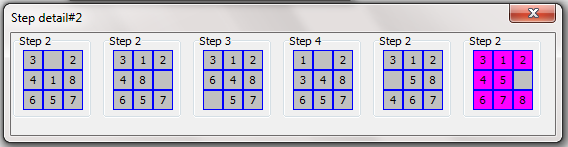
****

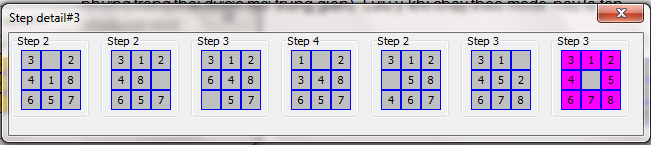
Ở mỗi bước ta có thể xem chi tiết các thông số g, h, f bằng cách rê chuột vào board và chờ khoảng 2s

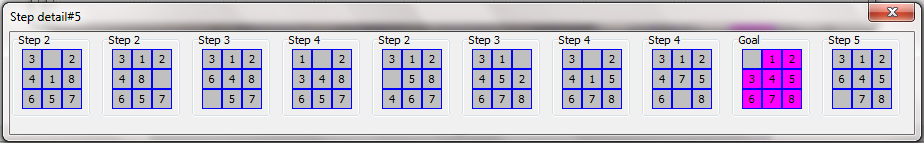
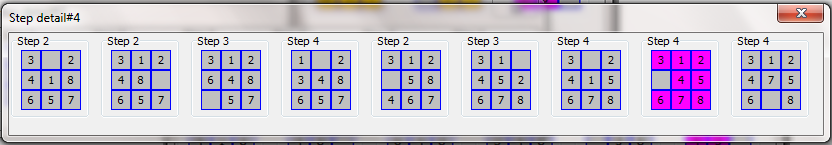


Ở mỗi bước ta có thể xem được có bao nhiêu trạng thái được mở tiếp theo ở mode này



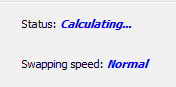




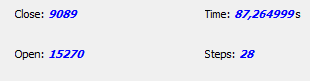


**Thanh trạng thái:**

Cho biết thông tin chương trình hiện tại



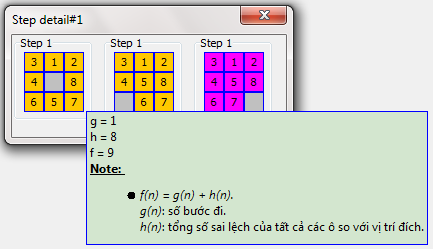
Sau khi chạy xong cho biết cần bao nhiêu bước để đi từ trạng thái ban đầu đến trạng thái đích, số trạng thái đã mở, số trạng thái đã đóng, và thời gian thực hiện



## Menu Swap when equal

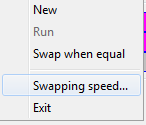
**

Đây là một tùy chọn của thuật toán A\*. Khi hai trạng thái có cùng f thì nếu như check vào menu này sẽ chọn trạng thái sau cùng. Nếu không check vào menu này sẽ chọn ngẫu nhiên trạng thái đầu tiên

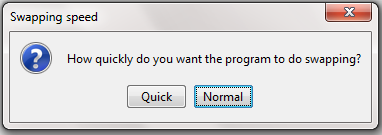


Trường hợp trên ba trạng thái đó đều có f là 9. Nếu không check vào menu *Swap when equal* chương trình sẽ chọn trạng thái đầu tiên để mở tiếp. Nếu check vào menu *Swap when equal* chương trình sẽ chọn trạng thái cuối cùng để mở

## Tùy chọn Swapping speed



Đây là tùy chọn vận tốc tráo đổi các cell trong board. Có hai mode như popup dưới đây:



Tùy chọn này chỉ có tác dụng cho mode **Automation**

# Tham khảo

[1] Bạch Hưng Khang, Hoàng Kiếm, Trí tuệ nhân tạo: Các phương pháp và ứng dụng. Nhà xuất bản Khoa học và Kỹ thuật, 1989