Trường đại học Bách Khoa Hà Nội

Viện Công Nghệ Thông Tin và Truyền Thông

----------o 0 o----------

Báo cáo môn học: Trí Tuệ Nhân Tạo

Đề tài: Ứng dụng giải thuật tìm kiếm A* tìm lời giải cho trò chơi nối điểm Flow Free

Giáo viên hướng dẫn: TS. Nguyễn Nhật Quang

Sinh viên thực hiện: Nhóm 4 – Lớp 104408

Trần Huy Hùng	**20164777**
Vi Thành Đạt	**20164803**
Trần Quang Huy	**20161863**
Nguyễn Việt Tiến	**20164077**

Hà Nội, ngày… tháng 12 năm 2018

Mục Lục

Nội dung

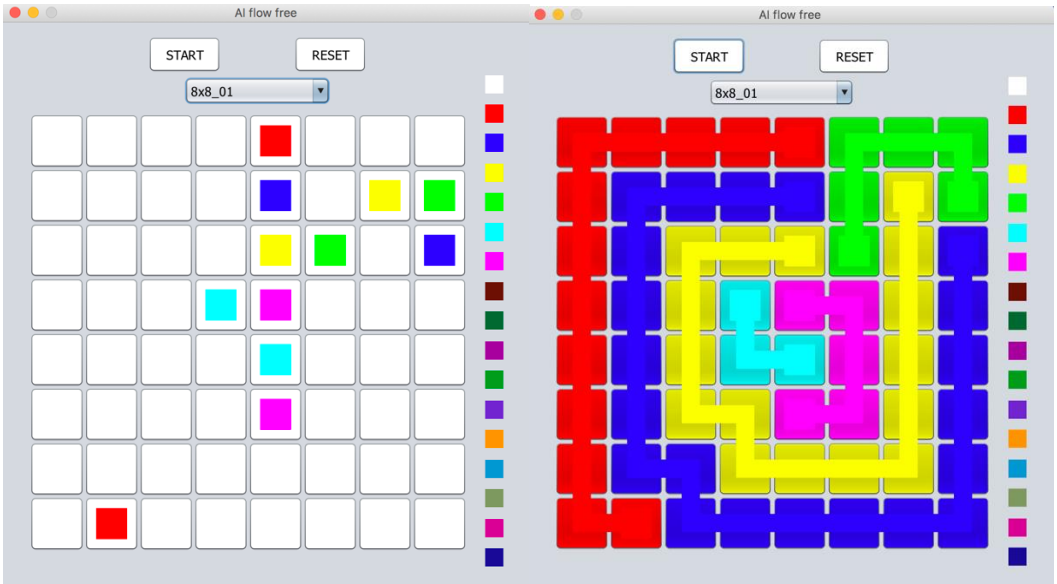
PHẦN I: MÔ TẢ BÀI TOÁN

1. Giới thiệu.

Flow Free là một loại game giải đố được phát hành bởi Big Duck Games studio trên iOS và Android vào Tháng 6 năm 2012.

- Mỗi màn chơi là một bảng vuông gồm các cặp chấm màu chiếm một số ô trong bảng.
- Flow là đường nối 2 điểm cùng màu.
- Mục tiêu của trò chơi là nối tất cả các cặp điểm sao cho trong bảng không còn ô nào trống và không có 2 flow nào cắt nhau.
- Kích thước bảng ô vuông từ 5x5 tới 15x15.

Hình ảnh minh họa:



2. Các hướng tiếp cận để giải bài toán.

Có ít nhất 2 hướng hướng để tiếp cận và giải bài toán:

- Đưa về bài toán thoả mãn ràng buộc
- Các thuật toán tìm kiếm: Greedy best first search, Thuật toán A*

Đề tài này sử dụng thuật toán A* để tiếp cận giải bài toán.

Phần II. THUẬT TOÁN A*

1.Mô tả thuật toán

Trong khoa học máy tính, A* là thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm một đường đi đơn từ nút bắt đầu tới một nút mục tiêu cho trước (hoặc tới một nút thỏa mãn điều kiện đích), sử dụng một đánh giá heuristic để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó và duyệt các nút theo thứ tự của đánh giá heuristic đó. Do đó, thuật toán A* là một ví dụ của tìm kiếm theo lựa chọn tốt nhất (best-first search).

Thuật toán A* được mô tả lần đầu vào năm 1968 bởi Peter Hart, Nils Nilsson, và Bertram Raphael.

Trong bài toán nối điểm Flow Free, A* xây dựng tăng dần tất cả các flow từ điểm xuất phát cho tới khi nó tìm thấy một flow đi tới điểm đích. Để biết những đường flow nào có khả năng sẽ dẫn tới đích, A* sử dụng một "đánh giá heuristic" về khoảng cách từ điểm bất kỳ cho trước tới đích.

2.Thuật toán

- Ý tưởng thuật toán: Tránh việc xét (phát triển) các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) là có chi phí cao.
- A* lưu giữ một tập các lời giải chưa hoàn chỉnh, nghĩa là các đường đi qua đồ thị, bắt đầu từ nút xuất phát. Tập lời giải này được lưu trong một hàng đợi ưu tiên (priority queue). Thứ tự ưu tiên gán cho một đường đi tại điểm n được quyết định bởi hàm đánh giá: ***f(n) = g(n) + h(n)***, trong đó:
  - g(n) = chi phí từ nút gốc cho đến nút hiện tại n
  - h(n) = chi phí ước lượng từ nút hiện tại n tới đích
  - f(n) = chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích

3.Các đặc điểm

- Nếu không gian các trạng thái là hữu hạn và có giải pháp để tránh việc xét (lặp) lại các trạng thái, thì giải thuật A* là hoàn chỉnh (tìm được lời giải) – nhưng không đảm bảo là tối ưu.

- Nếu không gian các trạng thái là hữu hạn và không có giải pháp để tránh việc (lặp) lại các trạng thái, thì giải thuật A* là không hoàn chỉnh (không đảm bảo tìm được lời giải).
- Nếu không gian các trạng thái là vô hạn, thì giải thuật A* là không hoàn chỉnh (không đảm bảo tìm được lời giải).

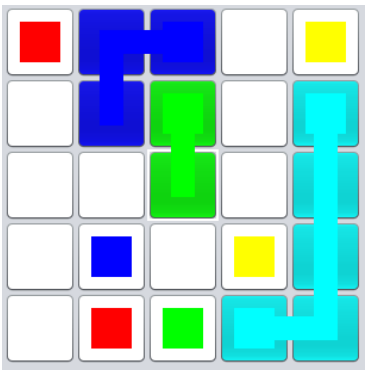
Phần III. CHI TIẾT VÀ PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN

1. Mô hình hóa bài toán

Mục tiêu của bài toán là tìm lời giải cho một **puzzle** (màn chơi) xác định. Trạng thái của puzzle hiện tại sẽ thay đổi mỗi khi kéo dài 1 flow trong puzzle thêm 1 ô. Người chơi sẽ liên tục thay đổi trạng thái puzzle theo cách đó tới khi nối hết các flow.

Như vậy, có thể mô hình hóa puzzle về bài toán tìm kiếm trên đồ thị (Graph Search):

- Mỗi **nút** (Node) của đồ thị là 1 **trạng thái** của puzzle, được xác định bởi:
  - **Bảng trạng thái** từng ô của puzzle
    - Mỗi flow sẽ được gán 1 chữ cái đại diện tương ứng màu flow đó.
      VD: R (red – đỏ), B (blue - xanh dương), G (green – xanh lục), Y (yellow – vàng), C (cyan – xanh lơ)
    - Ô nào còn trống thì gán ký tự dấu chấm (.)
    - Ô đã có flow đi qua, được gán chữ cái đại diện của flow. Nếu ô đó là điểm đầu hoặc cuối của flow thì là chữ in hoa, còn ở giữa thì là chữ in thường.
  - **Vị trí hiện tại** của từng flow.
    - Hai đầu của 1 flow được quy ước rõ là điểm đầu hay điểm cuối.
    - Từ điểm đầu, flow đang đi đến ô nào, thì tọa độ ô đó trên bảng là vị trí hiện tại của flow.
  - Ví dụ: dưới đây là 1 puzzle 5x5, được mã hóa thành 2 dữ kiện như trên



1) Bảng trạng thái puzzle:

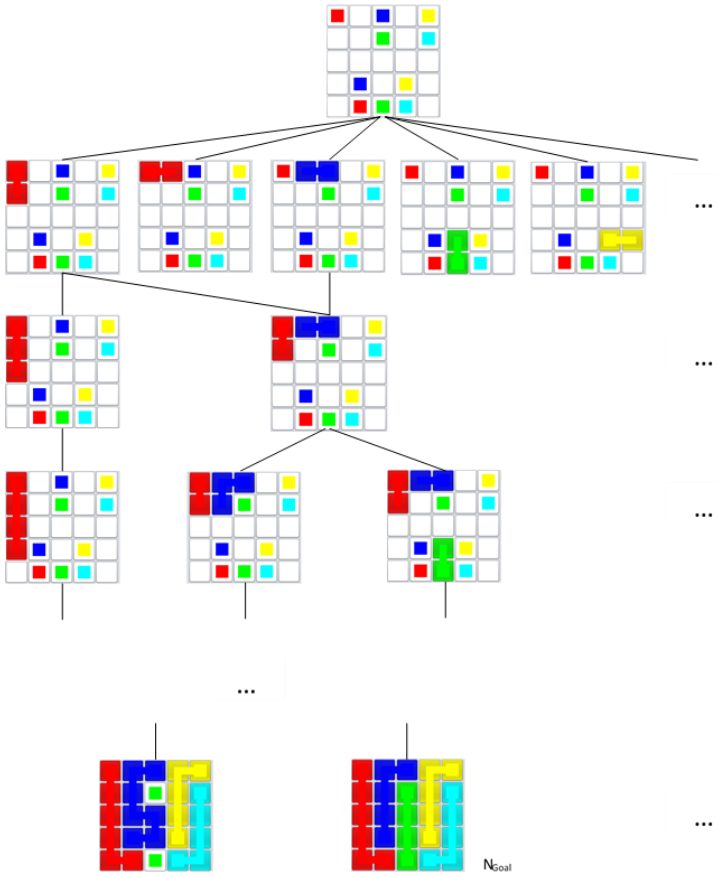
```
R b B y Y
. b G . C
. . g . c
. b . Y c
. R G C c
```

2) Vị trí hiện tại từng flow: (mặc định điểm đầu flow về phía trên cùng bên trái)

- R (đỏ): (1, 1)
- B (xanh dương): (2, 2)
- G (xanh lục): (3, 3)
- Y (vàng): (1, 5)
- C (xanh lơ): (5, 4)

- **Nút con** của 1 nút: Chính là các trạng thái tiếp theo có thể sinh ra trực tiếp từ trạng thái nút đó (bằng cách kéo dài 1 flow chưa hoàn thành thêm đúng 1 ô).
- **Nút gốc** (Initial Node): Trạng thái khi mới bắt đầu chơi (các flow đều chưa được nối).
- **Nút đích** (Goal Node): Thỏa mãn 2 điều kiện:
  - Tất cả flow đều đã nối hoàn chỉnh 2 đầu
  - Không ô nào trên bảng ở trạng thái còn trống

Ví dụ:



2. Áp dụng giải thuật A*

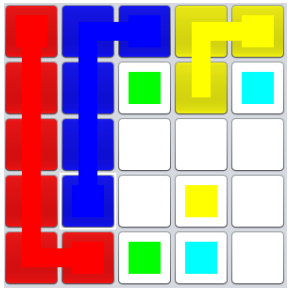
Sau khi đưa việc giải puzzle về bài toán tìm kiếm trên đồ thị, chúng ta sẽ áp dụng giải thuật A* để tìm đường đi từ nút gốc đến đích. Tuy nhiên, có thể thấy bài toán này không quá quan trọng việc đường đi tới đích là "ngắn nhất", mà mục đích chính tìm ra ít nhất một đường tới đích. Do đó, A* và các kỹ thuật khác sẽ tập trung vào việc định hướng tìm ra đích càng nhanh càng tốt (số nút phải xét để đến đích ít nhất có thể).

Do vậy, hàm đánh giá f(n) của A* sẽ được tính dựa trên g(n) và các h(n) dưới đây:

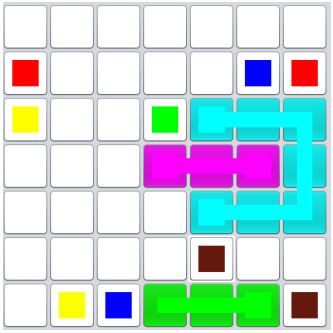
a) Chi phí tới nút hiện tại g(n)

❖ Định nghĩa **nước đi bắt buộc**: Xét 2 trường hợp:

- TH1: 1 flow bình thường có thể đi theo tối đa 4 hướng khác nhhau. Tuy nhiên, nếu 3 trong 4 hướng bị chặn (bởi tường hoặc flow khác), thì flow đó chắc chắn chỉ có thể đi theo hướng còn lại (như flow vàng hình bên).



- TH2: Ở hình bên, xét flow màu **nâu**. Từ điểm đầu có thể đi về bên trái hoặc phải. Nhưng ô kề bên phải đã bị chặn 2 hướng, vậy flow duy nhất có thể đi qua ô đó là flow nâu.
  Nếu flow nâu đi bên trái, ô bên phải sẽ không bao giờ được đi qua, trái điều kiện bài toán.
  ➔ Flow nâu bắt buộc phải đi sang phải.



➢ **Kết luận**: Tại 1 trạng thái, **nước đi bắt buộc** của 1 flow là hướng đi duy nhất flow có thể đi tiếp lúc đó. Có 2 trường hợp:
- Flow chỉ còn 1 hướng không bị chặn.
- Tồn tại một ô kề với vị trí flow đó bị chặn 2 hướng (không tính vị trí hiện tại hoặc cuối của 1 flow chưa hoàn thành khác).

❖ Hàm chi phí hiện tại g(n):

Là tổng "chi phí di chuyển" (cost) từ nút gốc tới nút hiện tại (n).

Khi đi từ nút n sang n', chi phí cost được tính: **g(n') = g(n) + cost**

- cost = 0 nếu đó là nước đi hoàn thành 1 flow, hoặc 1 nước đi bắt buộc.
- cost = 1 nếu ngược lại.

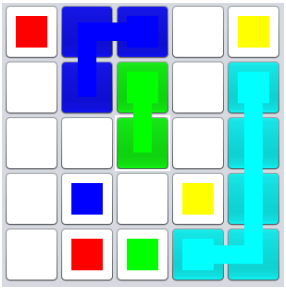
Hàm g(n) trên có ý nghĩa: coi những nước đi cần thiết hoặc bắt buộc là không tốn chi phí, còn những nước đi được lựa chọn thì tốn chi phí đánh giá.

b) Chi phí ước tính h(n)

Dưới đây là một số hàm heuristics ước lượng chi phí tới đích của nút n:

❖ **FreeCells** (Số ô còn trống)

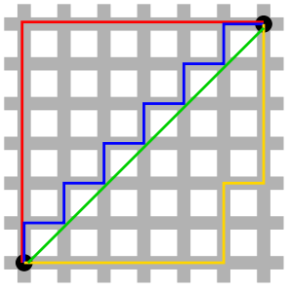
- $h_F(n)$ = Tổng số ô trống trên bảng chưa có flow nào đi qua (tính cả điểm cuối của các flow chưa hoàn thành).



- Ví dụ: Ở hình trên, số ô trống hoàn toàn là 9, cộng thêm điểm cuối của 4 flow như hoàn thành.
  - $h_F(n) = 9 + 4 = 13$
- Ý nghĩa: Xuất phát từ ràng buộc "trạng thái đích không còn ô trống". Hàm h định hướng tìm kiếm theo các nút càng ít ô trống càng tốt.

❖ **TotalManhattanDistance** (Tổng khoảng cách Manhattan)

- Khoảng cách Manhattan: Là khoảng cách giữa 2 điểm trong không gian với hệ tọa độ Descartes, bằng tổng chiều dài hình chiếu của đường thẳng nối 2 điểm này lên hệ trục tọa độ.

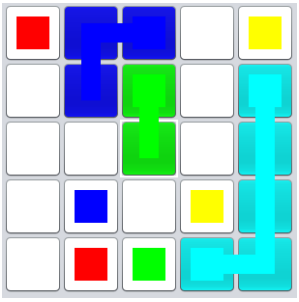
  

  Công thức khoảng cách Manhattan giữa 2 điểm tọa độ (x1, y1) và (x2, y2):

$$D = |x1-x2| + |y1-y2|$$

- Ước lượng $h_M(n)$:
  - Tại nút n, mỗi flow đã đi đến 1 vị trí hiện tại P_{cur}, và có vị trí điểm cuối P_{end} là đích cần đi đến.
  - Chi phí ước lượng tới đích của flow là khoảng cách Manhattan từ P_{cur} đến P_{end}.
  - $h_M(n)$ = Tổng chi phí tới đích của tất cả các flow

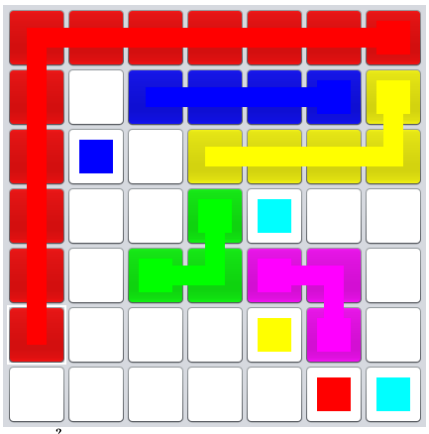
- Ví dụ:



  - Flow xanh dương có $P_{cur} = (2,2)$, $P_{end} = (4,2)$ → $D_B = |2-4| + |2-2| = 2$
  - Tương tự: $D_R = 5$, $D_G = 2$, $D_Y = 4$, $D_C = 0$
    → $h_M(n) = 2 + 5 + 2 + 4 + 0 = 13$

❖ **TotalWallDistance** (Tổng khoảng cách tới tường)

- Khoảng cách tới tường của 1 flow (Ký hiệu DW_{flow}): Là khoảng cách nhỏ nhất theo 1 trong 4 phía tường của **vị trí hiện tại** của flow đó
- Ước lượng h(n):



  - $h_W(n)$ = Tổng khoảng cách tới tường của tất cả các flow chưa hoàn thành.
- Ví dụ:
  - Flow vàng (Y): Vị trí hiện tại là (3, 4). Khoảng cách tới 4 phía tường (theo chiều kim đồng hồ, bắt đầu từ trên) lần lượt là 2, 3, 4, 3 → $DW_Y = 2$
  - Tương tự: $DW_R = 0$, $DW_B = 1$, $DW_C = 2$
  - Flow xanh lục và tím đã hoàn thành, nên bỏ qua
  - $h_W(n) = 2 + 0 + 1 + 2 = 5$
- Ý nghĩa: Hàm heuristics này dựa trên kinh nghiệm chơi Flow Free. Trong đa số puzzle, flow cạnh tường có xu hướng chạy dọc theo rìa tường, tạo thành 1 lớp tường mới bao bọc các flow bên trong (như flow đỏ ở ví dụ trên).

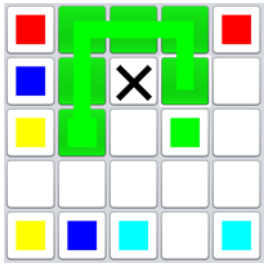
Do đó, nối flow theo thứ tự từ ngoài vào trong rất dễ xác định nước đi (chỉ cần đi dọc theo tường, các flow trong lại tiếp tục đi dọc theo tường mới tạo bởi flow ngoài).

3. Các module kiểm tra tính hợp lệ

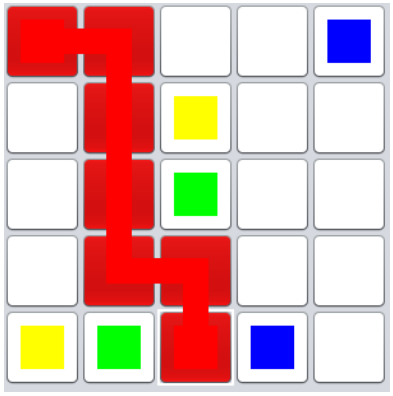
Trong quá trình tìm kiếm, sẽ có những nút chắc chắn không thể đi tới nút đích. Những nút này gọi là "nút không hợp lệ" (Invalid Node). Việc loại bỏ sớm nhánh tìm kiếm của các nút không hợp lệ sẽ giảm đáng kể chi phí giải thuật.

a) Ngõ cụt

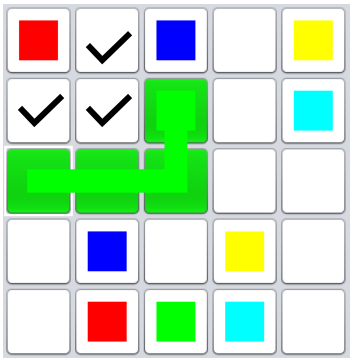
- **Ngõ cụt**: Ở hình bên, sau khi đi flow xanh lục, ô đánh dấu "X" tình cờ trở thành 1 ngõ cụt, không flow nào đi vào được, vi phạm ràng buộc "không ô nào còn trống".
- Khi di chuyển 1 flow, nếu có ít nhất 1 ô kề với nó trở thành ngõ cụt, thì nút đó là không hợp lệ.



b) Flow và vùng trống bị cô lập



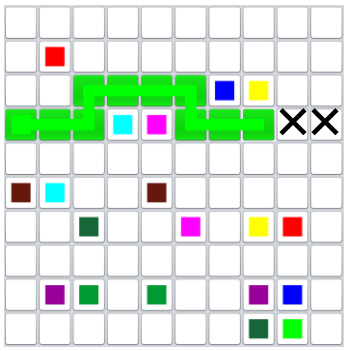
- **Flow bị cô lập**: Là flow mà 2 đầu không kề với chung bất kỳ 1 vùng trống nào.
  - Nếu 2 đầu kề với cùng 1 vùng trống, thì có thể vẽ đường nối qua vùng trống đó. Nếu không, flow đó bị cô lập (hay chia cắt).
  - Ví dụ: Ở hình bên, 2 đầu của flow màu vàng kề với 2 vùng trống khác nhau, nên không thể nối với nhau.



- **Vùng trống bị cô lập**: Là vùng trống không kề với cùng lúc hai đầu của bất kỳ flow nào chưa hoàn thành.
  - Do không có bất kỳ flow nào vẽ đường nối qua được, vùng này luôn bị bỏ trống.
  - Ví dụ: Vùng trống được đánh dấu tích sẽ không bao giờ được lấp đầy.

❖ Các nút tồn tại flow hay vùng trống bị cô lập đều là nút không hợp lệ.

c) Nút thắt cổ chai



- Ví dụ: Ở hình bên, sau khi đi flow xanh lục, xuất hiện 1 đoạn "thắt cổ chai" rộng 2 ô được đánh dấu chéo. Tuy nhiên, vùng trống phía trên có tới 3 đầu của 3 flow cần nối (3 đầu còn lại ở nửa dưới). Nút thắt chỉ rộng 2 ô, mà cần ít nhất 3 ô để 3 flow đi qua, puzzle không thể hoàn thành.
- Nút nào xuất hiện **nút thắt cổ chai** có độ rộng nhỏ hơn số flow cần nối giữa 2 phía nút thắt, thì nút đó là nút không hợp lệ.

❖ Trong giải thuật, mỗi trường hợp trên là 1 module kiểm tra nút không hợp lệ. Bất cứ nút nào được tạo ra bị 1 module phát hiện không hợp lệ sẽ không được cho vào tập open.

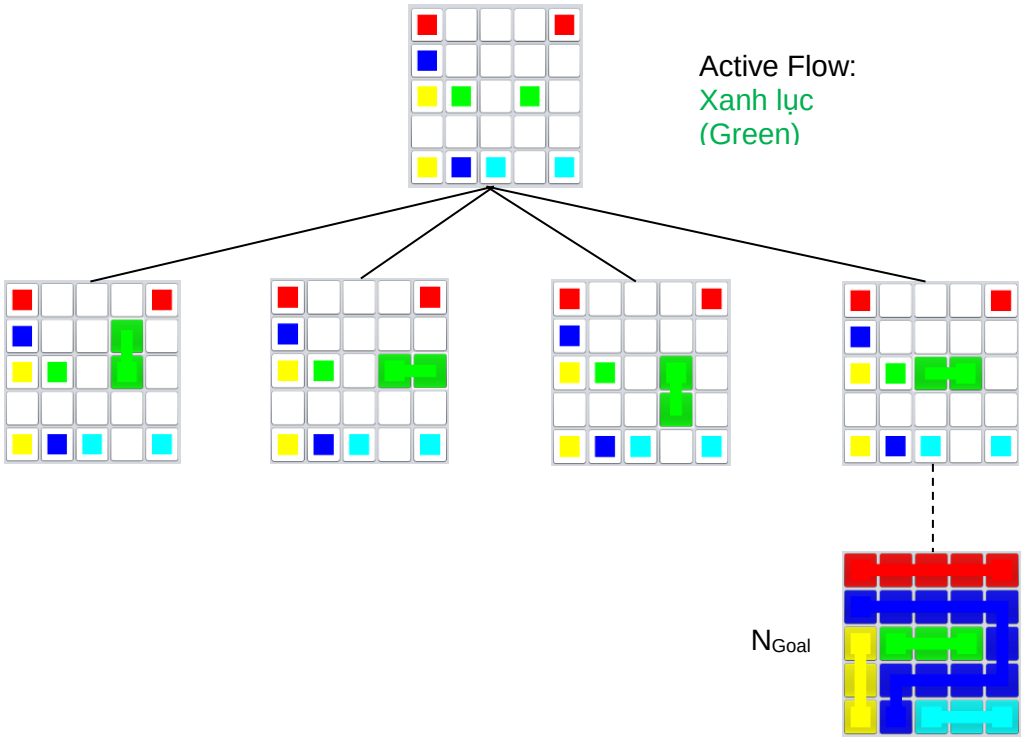
4. Cải tiến giải thuật

a) Giới hạn active flow

Nhận xét: Từ 1 nút, số nhánh sinh ra là rất lớn. Giả sử puzzle có 5 flow, mỗi flow đi được tối đa 4 hướng, thì số nút con được sinh ra tối đa là 20. Cứ như vậy, tổng số nút phải xét sẽ tăng theo hàm mũ của 20, tạo ra 1 không gian tìm kiếm khổng lồ. Do đó, hạn chế số nhánh sinh ra để tiết kiệm thời gian và bộ nhớ là rất quan trọng.

- **Active Flow**: Để giới hạn số nhánh sinh ra, tại 1 nút, ta chỉ chọn đúng 1 flow được phép đi tiếp, các flow khác đứng yên. Flow này được gọi là "active flow".

  ➔ Mỗi nút chỉ sinh ra tối đa 4 nút con.



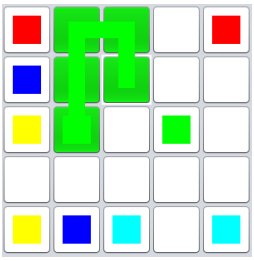
- Chứng minh tính đúng đắn: Xét ví dụ trên:
  - Puzzle tồn tại 1 nút đích là N_{Goal}. Từ nút gốc, hướng nối đúng của flow xanh lục là bên trái. Tương tự, các flow khác cũng có 1 hướng nối đúng để sinh ra nhánh có chứa N_{Goal}.
  - Chọn flow xanh lục là active flow, cắt bỏ nhánh của tất cả các flow khác. Nhưng xanh lục vẫn luôn có 1 nước đi tạo nhánh có thể dẫn đến N_{Goal}.

- Chứng minh tương tự với các flow khác và với các nút sau bằng quy nạp, có thể khẳng định việc chọn active flow luôn giữ lại ít nhất 1 con đường đến đích (nếu có đích).

- Tiêu chí chọn Active Flow:
  - Flow có số hướng đi được (không bị chặn) ít nhất.
    - Sinh ít nhánh nhất, giảm không gian tìm kiếm
  - Flow có vị trí hiện tại gần tường nhất.
    - Theo quy tắc "nối từ ngoài vào" ở phần **TotalWallDistance**

b) Ràng buộc flow đi vòng

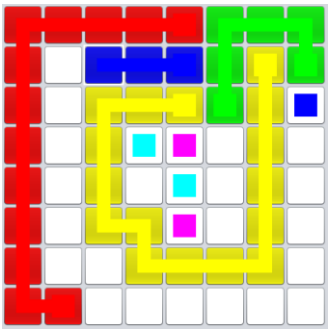
- Cơ sở: Ràng buộc này dựa trên "luật ẩn" của trò chơi Flow Free gốc:
  - Đích của trò chơi không tồn tại bất kỳ flow nào mà đường nối có đoạn đi vòng.
  - Ví dụ, flow xanh lục ở hình bên đã đi vòng và chạm vào chính nó. Lời giải của Flow Free không bao giờ tồn tại đường nối như vậy.



- Áp dụng: Trong quá trình tạo nút kế tiếp bằng cách di chuyển 1 flow, ta sẽ bỏ qua các hướng khiến flow chạm lại chính nó.

c) Tăng tốc chuỗi nước đi bắt buộc

- Ở hình bên, flow xanh dương bị kẹp giữa đỏ và vàng. Dễ thấy từ vị trí hiện tại, 1 chuỗi các nước đi của xanh dương sẽ là nước đi bắt buộc.
- Khi có 1 chuỗi nước đi bắt buộc, thay vì liên tục đưa chúng vào tập open rồi lại lấy ra sinh nút con, ta có thể đệ quy truy hết chuỗi nước đi bắt buộc rồi mới cho nút cuối vào open.



5. Thực nghiệm

- Bộ test: Kích thước từ 6x6 đến 14x14, chọn ngẫu nhiên từ trò chơi Flow Free.
- Các hàm heuristics được đánh giá:
  - Ba hàm heuristics cơ bản: h_F, h_W, h_M
  - $h_F + h_W$
  - $h_F + 5 * h_W + h_M$

Sau một số thực nghiệm, chúng em thu được kết quả sau:

❖ **Kịch bản 1:** Không giới hạn Active Flow + Không ràng buộc flow đi vòng

Test	hF	hW	hM	hF+hW	hF+5*hW+hM
6x6	7,67	7,33	7,67	4,33	6,33
7x7	13,67	657,33	13,33	10,33	11,33
8x8	1358	11292	94	166	21

Số nút trong Closed - Thực nghiệm 1

❖ **Kịch bản 2:** Có giới hạn Acive Flow + Không ràng buộc flow đi vòng

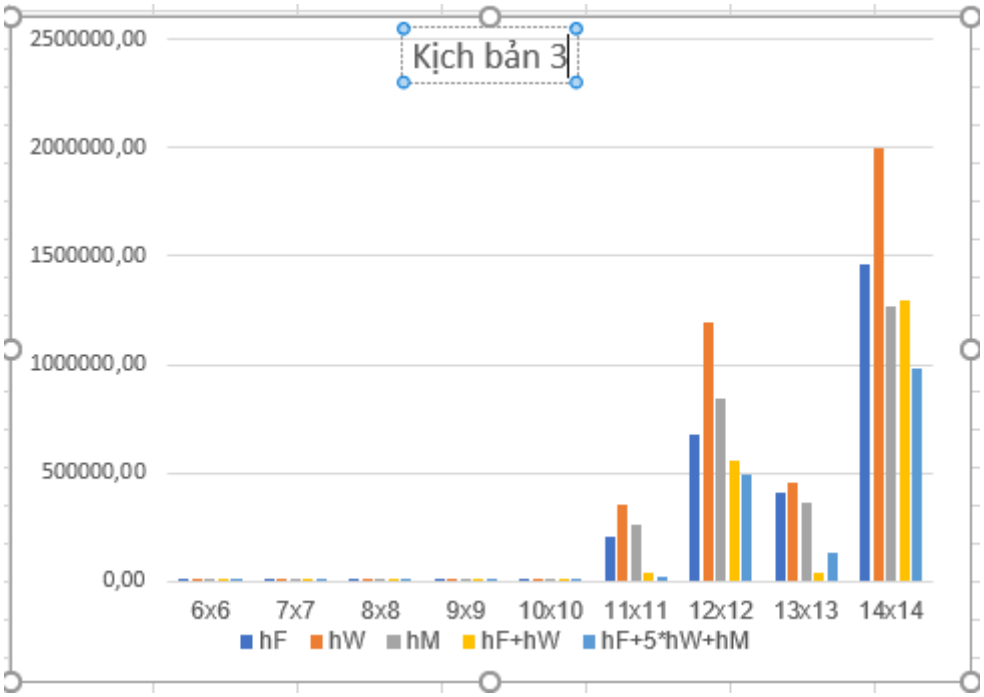
Test	hF	hW	hM	hF+hW	hF+5*hW+hM
6x6	150	158,75	149,75	150	150
7x7	6	6	6	6	6
8x8	12	15	13	11,33	13,33
9x9	2195,33	2717	2051	438,33	2159
10x10	394,67	6340,33	2662,33	2002,67	508,00

Số nút trong Closed - Thực nghiệm 2

❖ **Kịch bản 3:** Có giới hạn Active Flow + Có ràng buộc flow đi vòng

Test	hF	hW	hM	hF+hW	hF+5*hW+hM
6x6	5,67	5,67	5,67	5,67	5,67
7x7	9,67	10,00	10,00	9,33	9,67
8x8	234,33	259,00	177,33	99,00	187,00
9x9	410,50	6962,75	4012,50	4693,50	618,75
10x10	2097	4701	4335,75	1910,25	1420,25
11x11	205365,00	351250,71	263368,43	41779,14	19373,71
12x12	680038,8	1195675,7	846930,0	557746,3	491292,3
13x13	408443	451465	366753	35944	128662
14x14	1458303	2000001	1268780	1300494	984700

Số nút trong Closed - Thực nghiệm 3



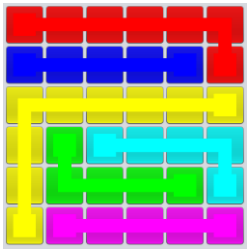
Biểu đồ cột - Thực nghiệm 3

➢ **Đánh giá:**

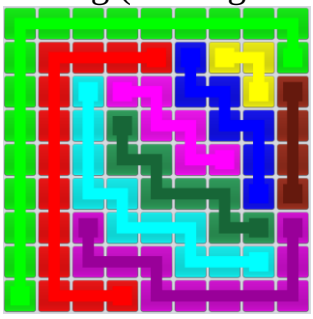
- Kỹ thuật Active Flow và ràng buộc flow vòng hiệu quả rõ rệt.
- Trong phần lớn trường hợp, 2 hàm $h_F + h_W$ và $h_F + 5 * h_W + h_M$ cho hiệu quả tốt nhất.

❖ **Đánh giá các hàm heurisctic:**

- h_M (Tổng khoảng cách Manhattan): Hiệu quả với những puzzle mà dạng của flow hầu hết là nối thẳng, không ngoằng ngoèo (độ dài rất gần với khoảng cách Manhattan giữa 2 đầu).



- h_W (tổng khoảng cách tới tường): Hiệu quả với những puzzle có dạng xoắn trôn ốc từ ngoài vào trong (flow ngoài bọc flow trong).

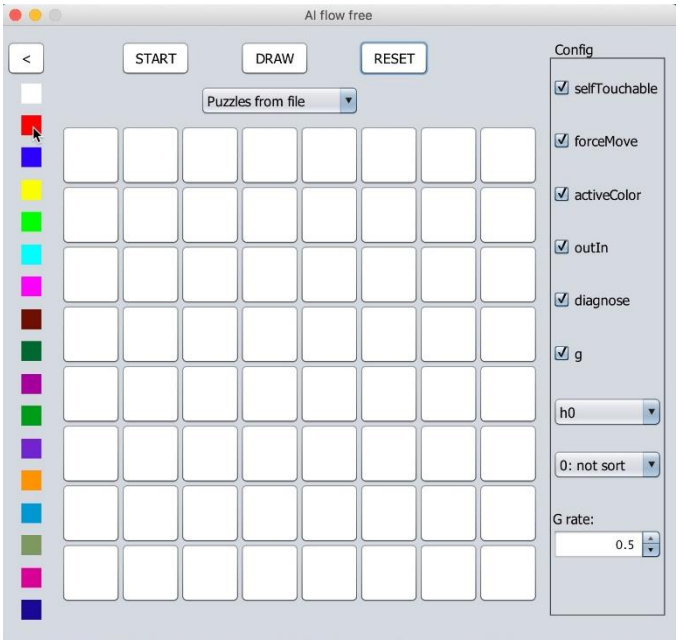


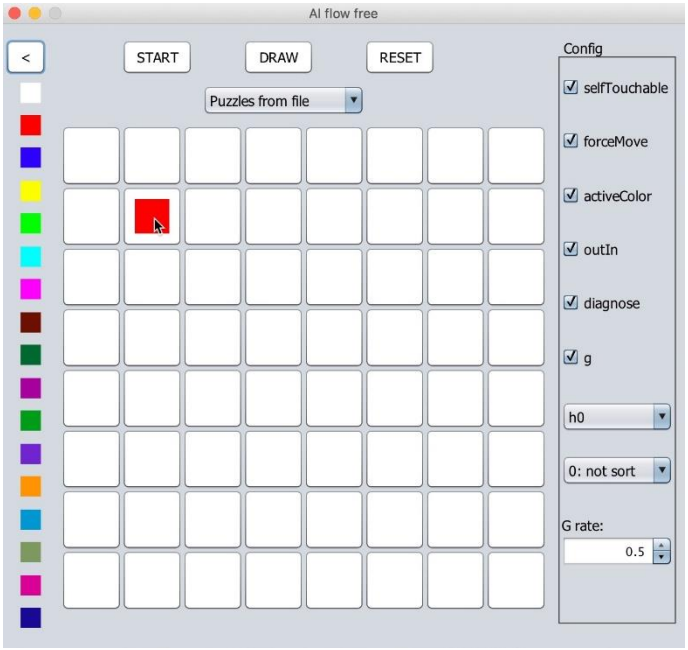
Phần IV: CHỨC NĂNG CHÍNH CỦA HỆ THỐNG

Hệ thống gồm 6 chức năng chính:

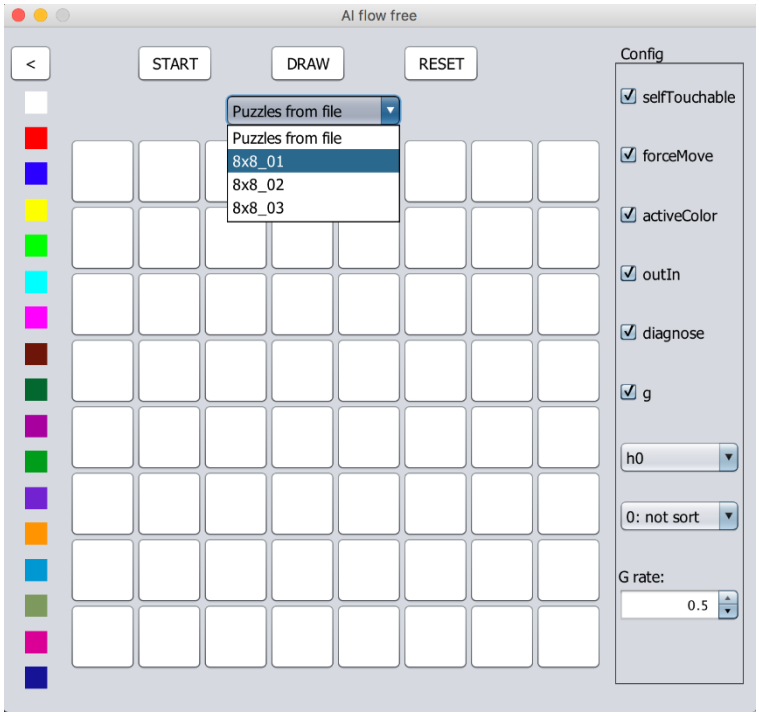
1. Chọn kích thước Puzzle:

2. Tự thiết kế Puzzle: Bấm chọn màu ở bảng màu phía bên trái và bấm vào các ô để set màu





3.Chọn các Puzzle có sẵn từ file test Puzzels



4.Chỉnh các thông số thuật toán

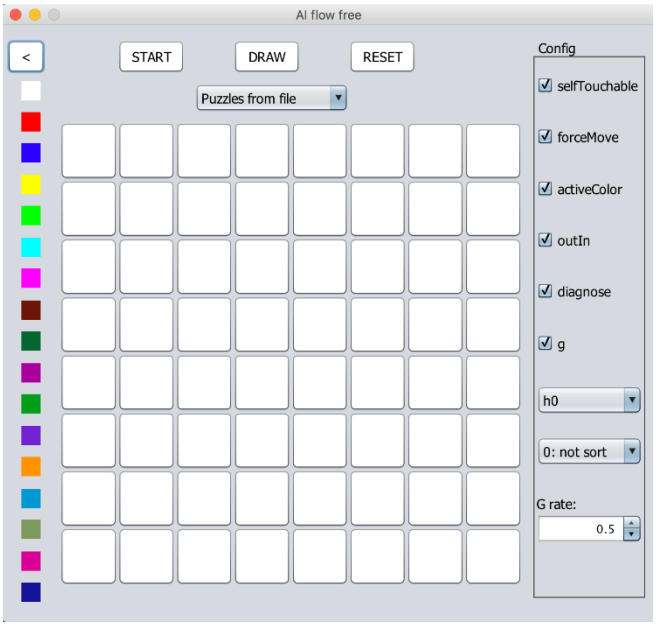
Ta có thể điều chỉnh các hàm phía bên phải của giao diện trò chơi

Có thể điều chỉnh các hàm :

selfTouchable,

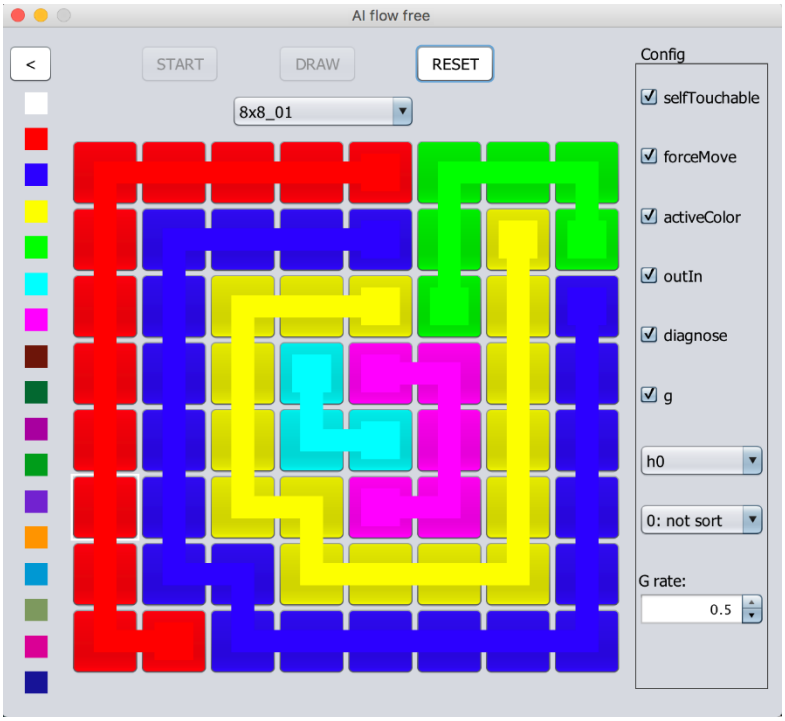
activeColor,

forceMove.…



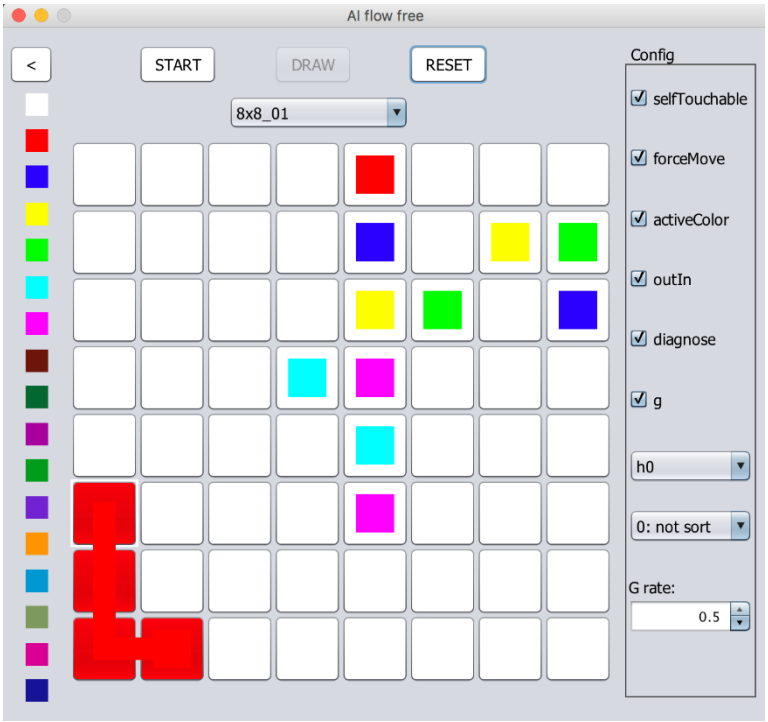
5.Các nút ấn

- **START**: tự động giải trò chơi



- **DRAW**: người chơi giải trò chơi từ một nước đi nào đó

Ví dụ bên dưới là đi từ nút màu đỏ:



- **RESET**: để reset bảng lại trạng thái ban đầu

Phần V: MÔI TRƯỜNG, GÓI PHẦN MỀM SỬ DỤNG

- Môi trường Eclipse IDE
- Framework Java Swing

Phần VI: KẾT LUẬN

1.Giải quyết khó khăn

- Rắc rối do lựa chọn hướng tiếp cận
  - Bản thân Flow Free có nhiều hướng tiếp cận để giải quyết, ví dụ hướng thỏa mãn ràng buộc khá phù hợp. Trong khi đó, A* là lựa chọn khá khó khăn.
  - Giải pháp: Bên cạnh kết hợp các Heuristics cơ bản, bổ sung một số phương pháp tang tốc độ tìm kiếm của A*.
- Khó khăn trong tiến hành thực nghiệm
  - Sai sót do lập trình, hạn chế về bộ nhớ và thời gian,…
  - Giải pháp: Thực nghiệm nhiều lần, hiệu chỉnh các thông số.

2.Kết quả

- Đưa ra được giải thuật và các phương pháp giải quyết bài toán lên đến kích thước puzzle 14x14
- Đánh giá được hiệu quả các phương pháp.
- Giao diện dễ sử dụng, hiệu chỉnh linh động các thông số của bài toán và giải thuật.

3.Hướng phát triển

- Đề xuất và thử nghiệm kết hợp các hàm heuristics hiệu quả hơn.
- Mở rộng giải thuật cho các dạng Flow Free mới: Bridge (cầu), Hexes (lục giác), Warps (cổng dịch chuyển),…
- Cải tiến theo hướng kết hợp thỏa mãn ràng buộc.

Phần VII: TÀI LIỆU THAM KHẢO

- Bài viết của Mzucker: https://mzucker.github.io/2016/08/28/flow-solver.html
- Bài giảng của Dr.Griffin: http://cs.mwsu.edu/~terry/?route=/courses/3013/content/assignments/page/Program1.md
- Wikipedia: https://www.wikipedia.org/