TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐHQG-HCM
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

ĐỒ ÁN TỔNG HỢP
HƯỚNG TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI

GIẢI RUBIK 2x2
BẰNG CÁC GIẢI THUẬT TÌM KIẾM

SINH VIÊN THỰC HIỆN:

1. Lê Viết Hòa - 1911186

2. Nguyễn Thái Linh - 1913949

3. Trần Hoàng Công Toại - 1912237

4. Nguyễn Văn Vinh Quang - 1911907

5. Trịnh Nguyên Bảo Tuấn - 1912371

NGƯỜI HƯỚNG DẪN:

ThS. TRẦN NGỌC BẢO DUY

TP. HỒ CHÍ MINH - NĂM 2021

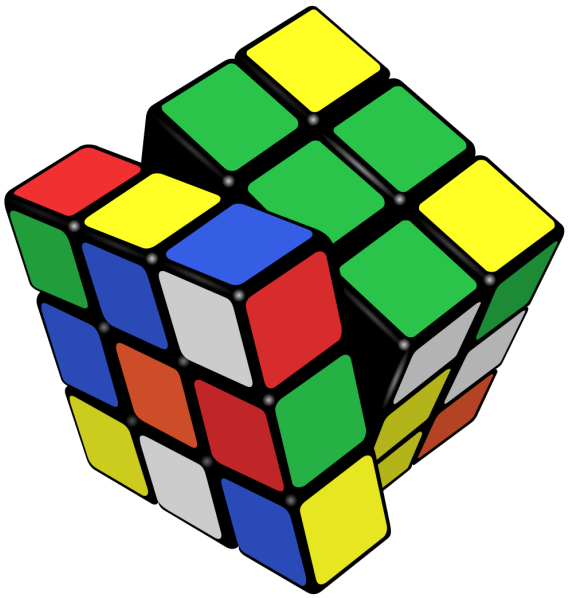
MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG

I. Giới thiệu đề tài

Lập phương Rubik (Khối Rubik hay đơn giản là Rubik) là một trò chơi giải đố cơ học được giáo sư kiến trúc, nhà điêu khắc gia người Hungary, Ernő Rubik phát minh vào năm 1974.

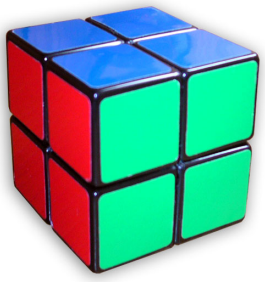


Hình 1: Khối rubik 3x3

Mỗi mặt của phiên bản này có 9 ô vuông và được sơn phủ một trong sáu màu khác nhau, thông thường là trắng, đỏ, vàng, cam, xanh lá cây và xanh dương (một số khối khác thay thế mặt màu trắng bằng màu đen, màu đỏ bằng màu hồng). Bài toán bắt đầu bằng việc xáo trộn tất cả vị trí các ô vuông ở mỗi mặt, tức là các màu sắc xen kẽ nhau. Bài toán chỉ được giải quyết khi mà mỗi mặt của khối là một màu đồng nhất.

Trong bài toán này, nhóm sẽ áp dụng các giải thuật tìm kiếm để giải một biến thể của khối rubik này. Cụ thể là khối **rubik 2x2** hay còn gọi là khối bỏ túi. Khối rubik 2x2 cũng có 6 mặt gồm 6 màu và các bước xoay tương tự như khối rubik 3x3 ở trên. Tuy nhiên ở mỗi mặt chỉ có 4 ô vuông thay vì 9 ô (hình ảnh minh họa ở dưới).

Hình 2: Khối rubik 2x2 chưa giải



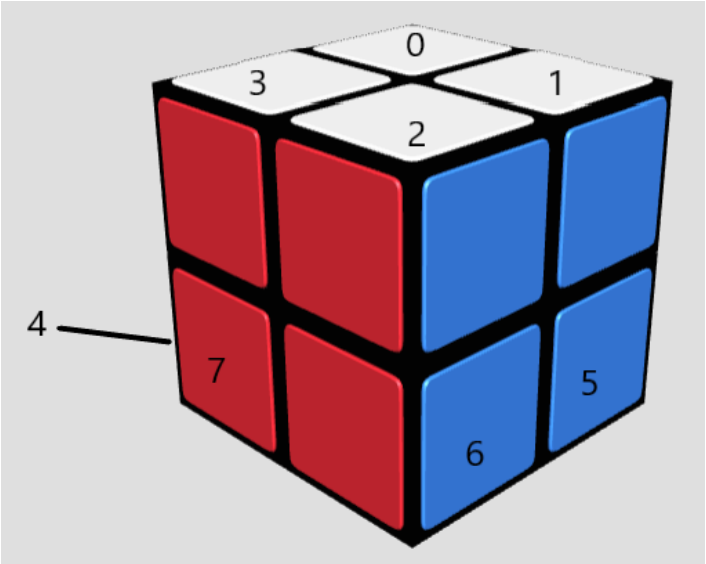
Hình 3: Khối rubik 2x2 đã giải

Số hoán vị: theo Wikipedia bất kỳ hoán vị nào của 8 góc đề khả thi nên có $8!$ hoán vị các vị trí. Ở mỗi vị trí, một góc có thể xoay độc lập theo 3 hướng khác nhau, tuy nhiên phải giữ 1 góc linh hoạt để 7 góc còn lại có thể xoay độc lập tạo nên 3^7 hoán vị góc cho mỗi hoán vị vị trí trên. Các rubik có cạnh chẵn (2x2, 4x4) đều không có tâm nên không xác định được hướng của cả khối lập phương trong không gian nên số hoán vị tổng thể giảm đi 24 lần. Ta có số hoán vị được tính như sau:

$$\frac{8!.3^7}{24} = 3674160 \text{ hoán vị}$$

II. Mô tả bài toán

Định nghĩa trạng thái: Rubik 2x2 được cấu tạo từ 8 khối lập phương (cube) nhỏ. Mỗi cube ta sẽ có một số định danh riêng ứng với vị trí của nó ở goal state. Được miêu tả như hình sau:



Hình 4: Định danh các khối cube theo vị trí ở trạng thái đã giải

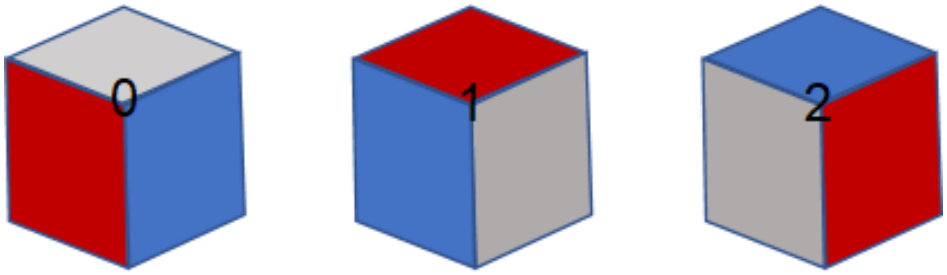
Như vậy, chúng ta sẽ có một danh sách các cube theo vị trí tương ứng dưới dạng mảng 8 phần tử như dưới đây:

$$cube = [0, 1, 2, 3, 4, 5, 6, 7] \text{ (trạng thái đích)}$$

$$cube = [1, 2, 4, 3, 5, 6, 7, 0] \text{ (trạng thái bất kỳ)}$$

(Với trạng thái bất kỳ ở ví dụ trên, cube số 1 ở vị trí 0, cube số 2 ở vị trí 1, cube số 4 ở vị trí 2, cube số 3 ở vị trí 4...)

Ngoài ra, ở mỗi vị trí thì cube có thể có 3 hướng xoay khác nhau, minh họa như hình dưới đây:



Hình 5: Trạng thái của khối cube theo hướng trong không gian

Ta quy ước:

– Hướng mà mặt trắng, vàng hướng lên trên hoặc xuống dưới là 0.

– Hướng mà mặt trắng, vàng từ hướng 0 quay theo chiều kim đồng hồ là 1.

– Hướng mà mặt trắng, vàng từ hướng 0 quay ngược chiều kim đồng hồ là 2.

Ta cũng lưu danh sách các góc của từng cube dưới dạng mảng như dưới đây:

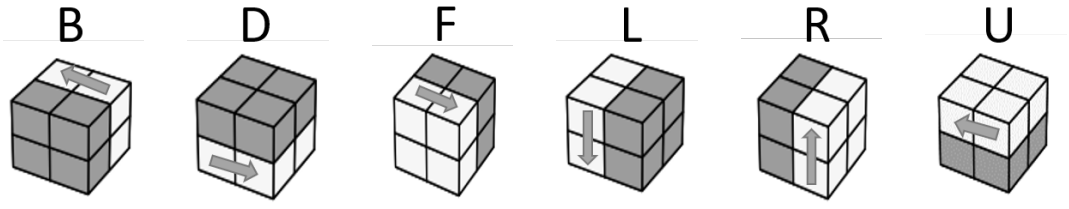
$$orie = [0, 0, 0, 0, 0, 0, 0, 0] \text{ (trạng thái đích)}$$

$$orie = [0, 1, 2, 1, 2, 0, 0, 2] \text{ (trạng thái bất kỳ)}$$

(Với trạng thái bất kỳ như ví dụ trên, cube ở vị trí số 0 có hướng 0, cube ở vị trí 1 có hướng 1, cube ở vị trí 2 có hướng 2, cube ở vị trí 3 có hướng 1...)

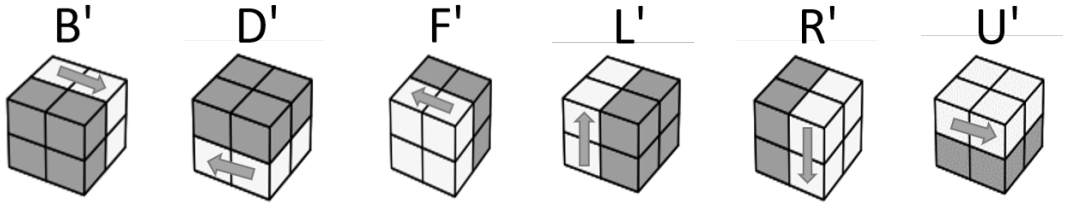
Như vậy, để xác định trạng thái của khối rubik ta cần phải xác định 2 yếu tố là $possition$ và $orie$.

Các bước xoay: các bước xoay của rubik 2x2 theo chiều kim đồng hồ được ký hiệu bằng vị trí các mặt viết tắt bằng tiếng anh B (back) - D (down) - F (front) - L (left) - R (right) - U (up).



Hình 6: Các bước xoay của rubik 2x2 theo chiều kim đồng hồ

Khi các ký tự này được viết cùng với dấu nháy đơn ’ (B’ - D’ - F’ - L’ - R’ –U’) thì chiều xoay sẽ đảo ngược lại, tức là xoay theo chiều ngược chiều kim đồng hồ.



Hình 7: Các bước xoay của rubik 2x2 ngược chiều kim đồng hồ

Khi định nghĩa trạng thái bài toán bằng 2 mảng $possition$ và $orie$ thì các bước xoay sẽ hoán vị vị trí các cube tương ứng trong mảng $possition$ và đồng thời cập nhập các góc tương ứng trong mảng $orie$.

Trạng thái khởi tạo: Rubik được xáo trộn bằng các bước xoay ngẫu nhiên được mô tả ở trên. Mảng $possition$ và mảng $orie$ mang giá trị tương ứng với trạng thái sau các bước xoay ngẫu nhiên.

Trạng thái đích: Rubik được giải khi mảng $possition$ và mảng $orie$ có giá trị:

$$cube = [0, 1, 2, 3, 4, 5, 6, 7]$$

$$orie = [0, 0, 0, 0, 0, 0, 0, 0]$$

III. Cơ sở lý thuyết

1. Thuật toán A*

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

A* xây dựng tăng dần các đường đi từ điểm xuất phát cho tới khi nó tìm thấy một đường đi chạm tới đích. Để biết được tuyến đường nào có khả năng dẫn tới đích thì A* sử dụng một đánh giá heuristic về khoảng cách từ điểm bất kỳ cho trước tới đích. Điểm khác biệt của A* đối với tìm kiếm lựa chọn tốt nhất thông thường là nó còn tính đến khoảng cách đã đi qua. Điều đó làm cho A* có tính "đầy đủ" và "tối ưu", nghĩa là A* sẽ luôn luôn tìm thấy đường đi ngắn nhất nếu đường này tồn tại.

Mô tả

Mỗi nút trong A* sẽ được lưu trong một hàng đợi ưu tiên, thứ tự ưu tiên của một nút x sẽ được quyết định bởi hàm:

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

Trong đó:

- $g(x)$ là chi phí của đường đi cho đến thời điểm hiện tại, nghĩa là tổng trọng số của các cạnh đã đi qua.
- $h(x)$ là hàm đánh giá heuristic về chi phí nhỏ nhất để đến đích từ x.

Ví dụ:
Có 2 điểm trên bản đồ: điểm ban đầu A đến điểm đích là B. Một điểm C sẽ có giá trị heuristic là $f(C) = AC + CB$. Trong đó, $g(C) = AC$ là khoảng cách đã đi qua, và $h(C) = CB$ là một đánh giá heuristic cho khoảng cách còn đi tiếp.

Hàm $f(x)$ có giá trị càng thấp thì độ ưu tiên của x càng cao.

Cần có một tập hợp lưu giữ các nút đã đi qua để tránh việc lặp lại các chu trình.

Tính chất

Nếu hàm heuristic h có tính chất thu nạp được (admissible), nghĩa là nó không bao giờ đánh giá cao hơn chi phí nhỏ nhất thực sự của việc đi tới đích, thì bản thân A* có tính chất thu nạp được (hay tối ưu) nếu sử dụng một tập đóng (tập lưu trữ tất cả các nút cuối cùng hiện tại hay nói cách khác là các nút mà đường đi mới được mở rộng tại đó). Nếu không sử dụng tập đóng thì hàm h phải có tính chất đơn điệu (hay nhất quán) thì A* mới có tính chất tối ưu. Nghĩa là nó không bao giờ đánh giá chi phí đi từ một nút tới một nút kề nó cao hơn chi phí thực.

Phát biểu một cách hình thức, với mọi nút x, y trong đó y là nút tiếp theo của x:

$$h(x) \leq g(y) - g(x) + h(y)$$

Ngoài ra, A* còn có tính chất hiệu quả một cách tối ưu (optimally efficient) với mọi hàm heuristic h, có nghĩa là không có thuật toán nào cũng sử dụng heuristic đó mà chỉ phải mở rộng ít nút hơn A*, trừ khi có một số lời giải chưa đầy đủ mà tại đó h dự đoán chính xác chi phí của đường đi tối ưu.

Độ phức tạp thời gian

Độ phức tạp thời gian của A* phụ thuộc vào đánh giá heuristic. Trong trường hợp xấu nhất thì số nút được mở rộng theo hàm số mũ của độ dài lời giải, nhưng nó sẽ là hàm đa thức khi hàm heuristic h thỏa mãn điều kiện sau:

$$|h(x) - h^*(x) \leq O(\log(h^*(x)))|$$

trong đó h^* là heuristic tối ưu, nghĩa là hàm cho kết quả là chi phí chính xác để đi từ x tới đích.

2. Pattern database

Pattern database là một bảng gồm các giá trị của hàm heuristic tương ứng với từng trường hợp của hàm đó. Cụ thể, pattern databse lưu chi phí cho giải pháp của tất cả trường hợp của một bài toán con của bài toán gốc. Những chi phí này sau đó sẽ được dùng để tính chi phí của bài toán gốc. Pattern database thường được dùng để ước tính cận dưới cho các trò chơi gồm các tổ hợp hay các vấn đề lập kế hoạch...

Khi chi phí tính toán giải pháp của những trường hợp của bài toán con của bài toán gốc cao thì nên sử dụng pattern database. Do đã lưu trữ sẵn chi phí của tất cả các trường hợp trên nên thời gian tính toán chi phí của bài toán gốc sau này sẽ được giảm bớt.

3. Cải tiến thời gian tìm kiếm

Có thể giảm bớt thời gian tìm kiếm bằng cách tạo ra đồ thị có ít trạng thái kế tiếp hơn, có nghĩa là từ một trạng thái có ít trường hợp chuyển sang trạng thái tiếp theo hơn.

IV. Thiết kế và hiện thực

1. Ứng dụng các phương pháp/ giải thuật/ kinh nghiệm vào bài toán

1.1 Heuristic 1

Hiện thực thuật toán A* và sử dụng hàm heuristic $h(x)$ đơn giản là sự sai khác giữa vị trí các cube và các hướng của một trạng thái so với trạng thái đích. Còn $g(x)$ chính là chiều dài đường đi từ trạng thái ban đầu đến trạng thái hiện tại.

$$h(x) = \frac{\text{số vị trí sai} + \text{số hướng sai}}{8}$$

$$g(x) = \text{chiều dài đường đi đến trạng thái hiện tại}$$

Hàm heuristic trong thuật toán A* sẽ là: $f(x) = g(x) + h(x)$.

Ở đây chia 8 bởi vì trong 1 lần chuyển trạng thái (tức là xoay 1 lần) thì có 4 cube thay đổi (mỗi cube có thể thay đổi cả vị trí và hướng), khi đó để đạt tính tối ưu của A* đã nêu ở trên thì ta nên chia 8.

1.2 Heuristic 2

Một hàm heuristic khác của bài toán là

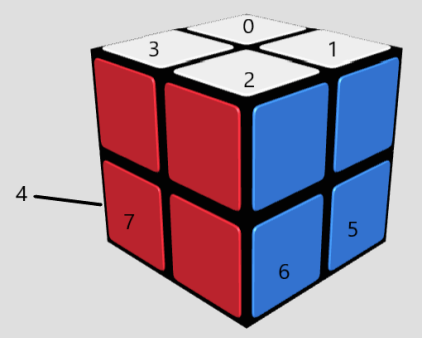
$$f_2(x) = g(x) + h_2(x)$$

Trong đó

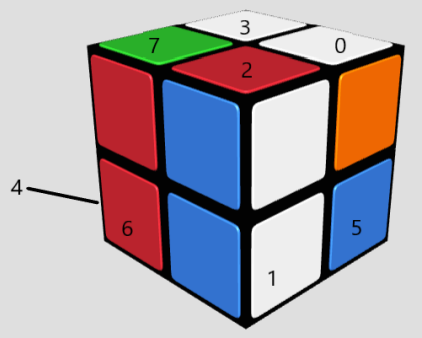
$$h_2(x) = \frac{\text{Tổng chi phí ngắn nhất của từng cube để đi đến vị trí đúng}}{4}$$

Ý tưởng của hàm $h_2(x)$ này là xấp xỉ số bước xoay cả khối rubik về vị trí đích bằng tổng số bước xoay của từng khối con về vị trí đích. Do mỗi một bước xoay của khối rubik sẽ tương ứng với 4 lần xoay 1 bước từng khối con của mặt đang xoay nên cần thực hiện phép chia 4 trong hàm $h_2(x)$ để có kết quả hợp lý.

Ví dụ cụ thể như sau:



Hình 8: Hình mô tả rubik trạng thái đích



Hình 9: Hình mô tả rubik trạng thái ngẫu nhiên

Từ trạng thái đích (Hình 8)

$$cube = [0, 1, 2, 3, 4, 5, 6, 7]$$
$$orie = [0, 0, 0, 0, 0, 0, 0, 0]$$

Xoay 2 bước U F để được trạng thái `state_ex` như hình 9:

$$cube = [3, 0, 2, 7, 4, 5, 1, 6]$$
$$orie = [0, 0, 1, 2, 0, 0, 2, 1]$$

Ta có:

- Cube số 0 đang ở vị trí 1, hướng 0, muốn về vị trí 0, hướng 0 nhanh nhất cần xoay U': 1 bước xoay.
- Cube số 1 đang ở vị trí 6, hướng 2, muốn về vị trí 1, hướng 0 nhanh nhất cần xoay D R': 2 bước xoay.
- Cube số 2 đang ở vị trí 2, hướng 1, muốn về vị trí 2, hướng 0 nhanh nhất cần xoay U' R': 2 bước xoay.

– Cube số 3 đang ở vị trí 0, hướng 0, muốn về vị trí 3, hướng 0 nhanh nhất cần xoay U': 1 bước xoay.

– Cube số 4 và cube số 5 đang ở vị trí đích của chúng.

– Cube số 6 đang ở vị trí 7, hướng 1, muốn về vị trí 6, hướng 0 nhanh nhất cần xoay F': 1 bước xoay.

– Cube số 7 đang ở vị trí 3, hướng 2, muốn về vị trí 7, hướng 0 nhanh nhất cần xoay F': 1 bước xoay.

Vậy

$$h_2(\text{state\_ex}) = \frac{1+2+2+1+1+1}{4} = 2$$

Số trường hợp của bài toán con (tính số bước xoay ngắn nhất để về vị trí đích của một khối con ở một vị trí nào đó) không nhiều nên có thể giải trước các bài toán con này và lưu vào pattern database nhằm giảm thời gian khi tính toán hàm $f_2(x)$. Bài toán con đơn giản nên có thể giải bằng thuật toán tìm kiếm theo chiều rộng (BFS - Breadth-first search).

1.3 Heuristic 3

Hàm heuristic này tương tự với hàm heuristic 2. Hàm $g(x)$ vẫn giữ nguyên và hàm $h_3(x)$ ước lượng số bước xoay của cả khối rubik về vị trí đúng bằng tổng số bước xoay của từng cặp hai khối con về vị trí đích, thay vì từng khối.

Có nhiều cách để chia 8 khối con thành danh sách những cặp 2 khối con, tuy nhiên chỉ cần quan tâm đến các cách chia thành những cặp 2 khối con kề nhau ở vị trí đích. Việc giải bài toán xoay 2 khối con về vị trí đích của chúng ở cách xa nhau không mang ý nghĩa thực tế, trái lại, nếu 2 khối con được xoay về vị trí đích kề nhau của chúng thì ta cách trạng thái đích của cả khối rubik không xa.

Các cách chia nêu trên tuy hiệu quả hơn nhưng tổng số vẫn còn khá nhiều. Nhóm đã chọn 3 cách chia trong số đó để đại diện cho các trường hợp còn lại. Với mỗi cách chia, cần chọn một đường thẳng song song với một cạnh của khối rubik và chia khối rubik thành các cặp 2 khối con dọc theo đường thẳng đó.

Chẳng hạn, như hình 8, chọn đường thẳng song song với cạnh đi qua khối con 0 và khối con 1, ta chia khối rubik thành các cặp

$$\{(0,1),(2,3),(4,5),(6,7)\}$$

Ngoài ra còn 2 cách chia theo đường thẳng đi qua khối con 0 và khối con 3, khối con 0 và khối con 4.

Tóm lại,

$$f_3(x) = g(x) + h_3(x)$$

Trong đó

$$h_3(x) = \frac{\text{(Tổng chi phí ngắn nhất của từng cặp cube để đi đến vị trí đúng) theo 3 cách}}{6}$$

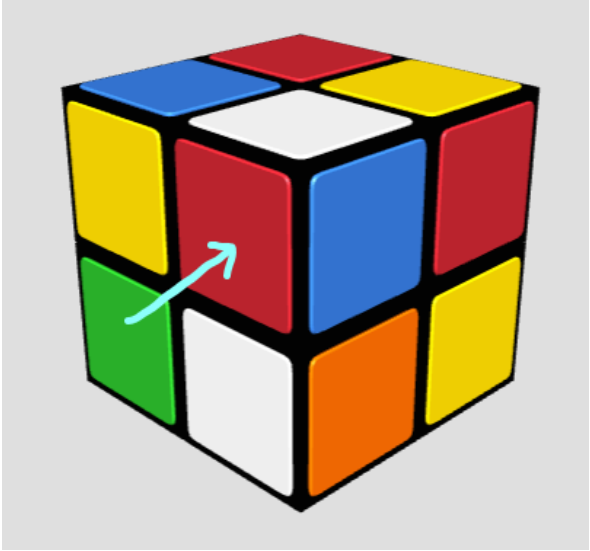
Tương tự với hàm heuristic 2, mỗi bước xoay của khối rubik sẽ tương ứng với 2 lần xoay 1 bước của hai cặp khối con của mặt đang xoay nên cần thực hiện phép chia 2, ngoài ra tính tổng theo 3 cách nên cần chia 3, do đó cần chia $2 \times 3 = 6$ trong hàm $h_3(x)$ để có kết quả hợp lý.

1.4 Cải tiến thời gian tìm kiếm

Các trạng thái sinh ra từ một trạng thái sẽ được xác định bằng 1 lần xoay rubik, vì vậy nếu số khả năng xoay của rubik ít hơn sẽ sinh ra trạng thái ít hơn.

Vì vậy, hướng tiếp cận của nhóm là sẽ chọn 1 cube cố định và chỉ xoay các cube còn lại, khi đó số cách xoay sẽ giảm 1 nửa (từ 12 thành 6).

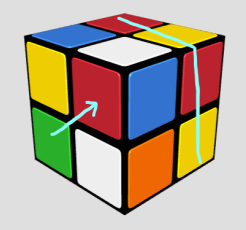
Ví dụ:



Hình 10: Khối rubik 2x2 chọn 1 cube cố định

Các mặt có thể xoay

Hình 11: Xoay bên trái

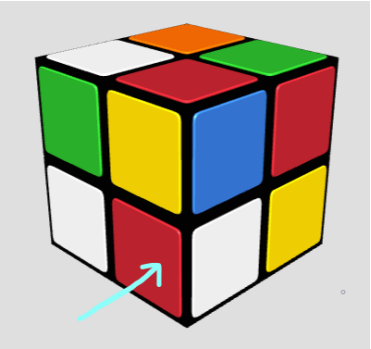


Hình 12: Xoay bên phải

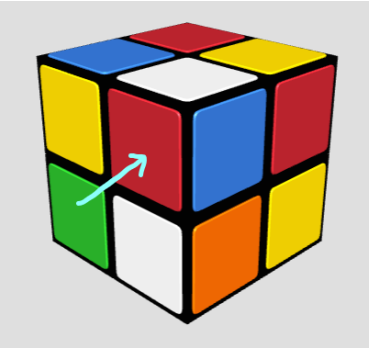
Hình 13: Xoay bên dưới

Ở đây, để có thể chọn 1 cube cố định thì cần đảm bảo cube này phải đúng vị trí như vị trí đích của nó trước, nhóm sẽ sử dụng 1 hàm tìm đường đi để chuyển trạng thái bất kỳ ban đầu sang trạng thái rubik đã đúng ở cube được chọn cố định. Sau đó sẽ giải rubik dựa vào A* như bình thường.

Ở ví dụ trên giả sử trạng thái ban đầu của rubik như sau thì ta cần thực hiện bước **F'** và **B'** (tức là lật rubik theo chiều ngược kim đồng hồ) để chuyển nó về dạng chuẩn (cube được chọn cố định đúng vị trí và đúng hướng)



Hình 14: Khối rubik 2x2 ở trạng thái ban đầu



Hình 15: Khối rubik 2x2 chuyển sang dạng chuẩn

Đánh giá: Nhìn chung, cách cải tiến giảm được rất nhiều trạng thái sinh ra, vì vậy thời gian giảm đáng kể. Thời gian để chuyển 1 trạng thái bất kỳ ban đầu sang trạng thái rubik đã đúng ở 1 cube được chọn không đáng kể so với thời gian được giảm do giảm số trạng thái sinh ra.

2. Thống kê

Trong phần này, chúng ta sẽ sử dụng phép đo thời gian để kiểm chứng độ hiệu quả của các giải thuật.

Chúng ta sẽ chạy các giải thuật với hàm random xoay ngẫu nhiên rubik 5 bước, 10 bước và 20 bước.

Trong phần này sẽ trình bày 2 phép so sánh:

– Giải thuật Breadth-First search với giải thuật A* dùng 3 hàm heuristic $h1$, $h2$, $h3$

– Giải thuật A* với 3 hàm heuristic $h1$, $h2$, $h3$ trước và sau khi cải tiến

Thiết lập timeout cho mỗi lần chạy testcase là 5 phút. Sau đó tính trung bình các thông số để so sánh độ hiệu quả các giải thuật

2.1 So sánh các giải thuật đã được cải tiến và BFS

Random	h1 improved	h2 improved	h3 improved	BFS improved
LdBLU	Time: 0.013964176177978516 Steps: 5 Node visited 994 Node created: 1326	Time: 0.0029909610748291016 Steps: 5 Node visited 161 Node created: 210	Time: 0.0009970664978027344 Steps: 5 Node visited 41 Node created: 48	Time: 0.009972572326660156 Steps: 5 Node visited 1248 Node created: 1680
lrblD	Time: 0.014958620071411133 Steps: 5 Node visited 820 Node created: 1098	Time: 0.0020284652709960938 Steps: 5 Node visited 159 Node created: 203	Time: 0.002019166946411133 Steps: 5 Node visited 71 Node created: 89	Time: 0.009973764419555664 Steps: 5 Node visited 1319 Node created: 1775
lfLdF	Time: 0.00797724723815918 Steps: 5 Node visited 680 Node created: 899	Time: 0.00196075439453125 Steps: 5 Node visited 69 Node created: 89	Time: 0.001995563507080078 Steps: 5 Node visited 25 Node created: 29	Time: 0.015959501266479492 Steps: 5 Node visited 2127 Node created: 2909
bruIU	Time: 0.007978677749633789 Steps: 5 Node visited 749 Node created: 987	Time: 0.0019948482513427734 Steps: 5 Node visited 129 Node created: 162	Time: 0.0009715557098388672 Steps: 5 Node visited 28 Node created: 33	Time: 0.009970903396606445 Steps: 5 Node visited 1183 Node created: 1593
BDRrR	Time: 0.000997781753540039 Steps: 3 Node visited 42 Node created: 52	Time: 0.00097304916381836 Steps: 3 Node visited 15 Node created: 16	Time: 0.0009970664978027344 Steps: 3 Node visited 15 Node created: 16	Time: 0.00097304916381836 Steps: 3 Node visited 113 Node created: 148
LdBfD	Time: 0.003990650177001953 Steps: 3 Node visited 65 Node created: 84	Time: 0.002993345260620117 Steps: 3 Node visited 16 Node created: 18	Time: 0.0030167102813720703 Steps: 3 Node visited 16 Node created: 18	Time: 0.003989219665527344 Steps: 3 Node visited 149 Node created: 198
BuluU	Time: 0.0009970664978027344 Steps: 3 Node visited 43 Node created: 52	Time: 0.0 Steps: 3 Node visited 15 Node created: 16	Time: 0.000972747802734375 Steps: 3 Node visited 15 Node created: 16	Time: 0.000997781753540039 Steps: 3 Node visited 50 Node created: 64

Hình 16: Một phần kết quả của giải thuật A* với $h1$, $h2$, $h3$ và BFS với 5 bước xoay ngẫu nhiên

Vì số bước xoay ngẫu nhiên khá ít nên thời gian chạy của các giải thuật khá nhỏ. Chúng ta vẫn chưa thể so sánh rõ ràng độ hiệu quả của các giải thuật.

Random10	h1 improved	h2 improved	h3 improved	BFS improved
RDlDUFdBBR	Time: 9.80988621711731 Steps: 10 Node visited 546743 Node created: 934962 lUUBuBBUlb	Time: 1.221175193786621 Steps: 10 Node visited 75484 Node created: 111630 lUUBuBBUlb	Time: 0.13064980506896973 Steps: 10 Node visited 3505 Node created: 4668 lUUBuBBUlb	Time: 11.468313217163086 Steps: 10 Node visited 824716 Node created: 1551120 lUUBuBBUlb
LDdddlrbUB	Time: 0.03694868087768555 Steps: 6 Node visited 3249 Node created: 4444 RdFrfd	Time: 0.004985332489013672 Steps: 6 Node visited 320 Node created: 412 RdFrfd	Time: 0.0009968280792236328 Steps: 6 Node visited 75 Node created: 94 RdFrfd	Time: 0.07782411575317383 Steps: 6 Node visited 10208 Node created: 14950 RdFrfd
UDrDlfuuFB	Time: 2.4094364643096924 Steps: 10 Node visited 148068 Node created: 235856 FFrrFRuFuu	Time: 1.3435771465301514 Steps: 10 Node visited 80062 Node created: 120241 ffrrFRuFUU	Time: 0.16356301307678223 Steps: 10 Node visited 5995 Node created: 8029 FFrrFRuFUU	Time: 14.191128492355347 Steps: 10 Node visited 1073844 Node created: 2104423 FFRRFRuFUU
rFFfrflrfD	Time: 0.70947265625 Steps: 8 Node visited 44411 Node created: 66743 uflLFuuLU	Time: 0.10522723197937012 Steps: 8 Node visited 7677 Node created: 10565 uflLFuuLU	Time: 0.03291130065917969 Steps: 8 Node visited 779 Node created: 1025 uflLFuuLU	Time: 0.8567073345184326 Steps: 8 Node visited 93412 Node created: 152381 FLUlfULU
RFBlbBRbdB	Time: 0.04937481880187988 Steps: 6 Node visited 3185 Node created: 4332 bURuur	Time: 0.009028196334838867 Steps: 6 Node visited 602 Node created: 786 bURUUr	Time: 0.004985809326171875 Steps: 6 Node visited 105 Node created: 132 bURUUr	Time: 0.036900997161865234 Steps: 6 Node visited 5049 Node created: 7164 bURUUr
frBFDubfrd	Time: 0.5645403861999512 Steps: 8 Node visited 42999 Node created: 64323 DRffddRD	Time: 0.0987088680267334 Steps: 8 Node visited 7616 Node created: 10377 DRFFddRD	Time: 0.010971546173095703 Steps: 8 Node visited 525 Node created: 675 DRFFddRD	Time: 0.9778933525085449 Steps: 8 Node visited 103656 Node created: 169905 DRFFDDRD

Hình 17: Một phần kết quả của giải thuật A* với $h1$, $h2$, $h3$ và BFS với 10 bước xoay ngẫu nhiên

Random20	h1 improved	h2 improved	h3 improved	BFS improved
rrulDlBudRblRDUUDRBD	Time: 36.49340462684631 Steps: 12 Node visited 1706827 Node created: 3451399 LFDDLdLfLDDF	Time: 11.051350593566895 Steps: 12 Node visited 706646 Node created: 1196076 ldfDlDlDLDfd	Time: 1.3903350830078125 Steps: 12 Node visited 65917 Node created: 97962 ldfDlDlDLDfd	Time: 58.57999801635742 Steps: 12 Node visited 3404608 Node created: 11588422 FDFDLdfLfdFl
dFLLRRrrdFUbUbddDRuL	Time: 10.088959217071533 Steps: 10 Node visited 479026 Node created: 815722 ffdffRfDrr	Time: 1.171658992767334 Steps: 10 Node visited 81648 Node created: 122433 ffdffRfDRR	Time: 0.1561722755432129 Steps: 10 Node visited 8513 Node created: 11649 ffdffRfDRR	Time: 15.980633020401001 Steps: 10 Node visited 1362218 Node created: 2854023 FFdFFRfDRR
fUUrBLRfuBfblLUFFRrU	Time: 11.762083530426025 Steps: 10 Node visited 590342 Node created: 1016945 rUURuBRbRB	Time: 1.5465009212493896 Steps: 10 Node visited 103976 Node created: 157751 rUURuBRbRB	Time: 0.09372782707214355 Steps: 10 Node visited 5262 Node created: 7199 rUURuBRbRB	Time: 13.075064897537231 Steps: 10 Node visited 1141800 Node created: 2270735 rUURuBRbRB
bdRbrDlfluUuluBdFrDr	Time: 37.23308348655701 Steps: 12 Node visited 1742954 Node created: 3540398 dllfDldFlddl	Time: 16.994866371154785 Steps: 12 Node visited 974624 Node created: 1705787 fllfdffdLflF	Time: 1.2810072898864746 Steps: 12 Node visited 61667 Node created: 90799 fLFdLfDlDlfL	Time: 60.235917806625366 Steps: 12 Node visited 3441304 Node created: 11950455 LFdLdFlFFDlD
UfddFlRDDulrUFFFdLFd	Time: 43.24453544616699 Steps: 12 Node visited 1967103 Node created: 4094438 RurBruuRRbru	Time: 12.094676733016968 Steps: 12 Node visited 698585 Node created: 1174812 ubRRUbRuBURb	Time: 1.374678373336792 Steps: 12 Node visited 63857 Node created: 94096 rbruBrBBURbr	Time: 55.98631286621094 Steps: 12 Node visited 3316636 Node created: 10915112 UrUruBrUUBru
fDllurDrFBuLBllfrRFF	Time: 8.706329345703125 Steps: 10 Node visited 495692 Node created: 839742 BrDrbDDRBd	Time: 1.2184600830078125 Steps: 10 Node visited 84168 Node created: 124464 dBDrDbDRBd	Time: 0.14053630828857422 Steps: 10 Node visited 7806 Node created: 10602 dBDrDbDRBd	Time: 6.264155387878418 Steps: 10 Node visited 686317 Node created: 1268394 BrDrbDDRBd

Hình 18: Một phần kết quả của giải thuật A* với $h1$, $h2$, $h3$ và BFS với 20 bước xoay ngẫu nhiên

Với 10 và 20 bước xoay ngẫu nhiên thì sự chênh lệch thời gian giữa các giải thuật đã đáng kể, trong đó, nhanh nhất là giải thuật A* với hàm lượng giá $h3$, rồi lần lược đến hàm lượng giá $h2$, $h1$ và chậm nhất là BFS.

2.2 So sánh giải thuật A* trước và sau khi cả tiến

Random	h1 improved	h1 not improved	h2 improved	h2 not improved	h3 improved	h3 not improved
FdUUR	Time: 0.0009965896606445312 Steps: 3 Node visited 49 Node created: 64	Time: 0.004988908767700195 Steps: 5 Node visited 525 Node created: 643	Time: 0.0009982585906982422 Steps: 3 Node visited 14 Node created: 16	Time: 0.0059850215911865234 Steps: 5 Node visited 274 Node created: 322	Time: 0.0009982585906982422 Steps: 3 Node visited 14 Node created: 16	Time: 0.0009963512420654297 Steps: 5 Node visited 54 Node created: 58
ubBbb	Time: 0.0009984970092773438 Steps: 3 Node visited 21 Node created: 25	Time: 0.0009636878967285156 Steps: 3 Node visited 62 Node created: 73	Time: 0.0009968280792236328 Steps: 3 Node visited 16 Node created: 19	Time: 0.0009965896606445312 Steps: 3 Node visited 34 Node created: 37	Time: 0.0010211467742919922 Steps: 3 Node visited 12 Node created: 13	Time: 0.0 Steps: 3 Node visited 24 Node created: 25
LffBr	Time: 0.001994609832763672 Steps: 3 Node visited 34 Node created: 42	Time: 0.08277750015258789 Steps: 5 Node visited 5735 Node created: 7896	Time: 0.0009968280792236328 Steps: 3 Node visited 16 Node created: 18	Time: 0.0029993057250976562 Steps: 5 Node visited 223 Node created: 274	Time: 0.000997781753540039 Steps: 3 Node visited 16 Node created: 18	Time: 0.0009982585906982422 Steps: 5 Node visited 54 Node created: 58
brurB	Time: 0.011976718902587891 Steps: 5 Node visited 749 Node created: 987	Time: 0.044879913330078125 Steps: 5 Node visited 4528 Node created: 6045	Time: 0.0039899349212646484 Steps: 5 Node visited 129 Node created: 162	Time: 0.004976749420166016 Steps: 5 Node visited 419 Node created: 498	Time: 0.0 Steps: 5 Node visited 28 Node created: 33	Time: 0.00099535986328125 Steps: 5 Node visited 50 Node created: 54
LfbDf	Time: 0.011967182159423828 Steps: 5 Node visited 758 Node created: 1008	Time: 0.04886913299560547 Steps: 5 Node visited 4690 Node created: 6378	Time: 0.0019922256469726562 Steps: 5 Node visited 148 Node created: 192	Time: 0.003025054931640625 Steps: 5 Node visited 269 Node created: 312	Time: 0.0019712448120117188 Steps: 5 Node visited 39 Node created: 48	Time: 0.0009984970092773438 Steps: 5 Node visited 55 Node created: 60
UBLlB	Time: 0.0009970664978027344 Steps: 3 Node visited 22 Node created: 26	Time: 0.0 Steps: 3 Node visited 63 Node created: 74	Time: 0.0020055770874023438 Steps: 3 Node visited 17 Node created: 20	Time: 0.0 Steps: 3 Node visited 35 Node created: 38	Time: 0.0009963512420654297 Steps: 3 Node visited 13 Node created: 14	Time: 0.0 Steps: 3 Node visited 25 Node created: 26
DDDlR	Time: 0.0 Steps: 1 Node visited 1 Node created: 1	Time: 0.0 Steps: 3 Node visited 29 Node created: 31	Time: 0.0 Steps: 1 Node visited 1 Node created: 1	Time: 0.0009653568267822266 Steps: 3 Node visited 50 Node created: 55	Time: 0.0 Steps: 1 Node visited 1 Node created: 1	Time: 0.0009975433349609375 Steps: 3 Node visited 33 Node created: 35

Hình 19: Một phần kết quả của giải thuật A* trước và sau khi cải tiến với 5 bước xoay ngẫu nhiên

Vì số bước xoay ngẫu nhiên khá ít nên thời gian chạy của các giải thuật khá nhỏ. Chúng ta vẫn chưa thể so sánh rõ ràng độ hiệu quả của các giải thuật.

Random10	h1 improved	h1 not improved	h2 improved	h2 not improved	h3 improved	h3 not improved
BfuubrbUrB	Time: 0.5815086364746094 Steps: 8 Node visited 44370 Node created: 66708 bLbLBLbb	Time: 14.009512186050415 Steps: 8 Node visited 749752 Node created: 1415610 fUlFRFrr	Time: 0.08774042129516602 Steps: 8 Node visited 5275 Node created: 7109 LBLbbllB	Time: 0.8357610702514648 Steps: 8 Node visited 64824 Node created: 97122 RUbUUFBr	Time: 0.015956640243530273 Steps: 8 Node visited 557 Node created: 725 LBLbbLLB	Time: 0.05085897445678711 Steps: 8 Node visited 2872 Node created: 3827 fDfLDFLL
bbbLFlfrlF	Time: 0.012993812561035156 Steps: 6 Node visited 1115 Node created: 1483 fLdlDL	Time: 0.0897934436798095 7 Steps: 6 Node visited 8074 Node created: 11383 bDfdLD	Time: 0.007978200912475586 Steps: 6 Node visited 488 Node created: 631 fLdlDL	Time: 0.028905630111694336 Steps: 6 Node visited 2545 Node created: 3319 bDfdLD	Time: 0.0019969940185546875 Steps: 6 Node visited 45 Node created: 55 fLdlDL	Time: 0.002991914749145508 Steps: 6 Node visited 139 Node created: 163 fLubRD
DlUUrLUfUD	Time: 0.29052281379699707 Steps: 8 Node visited 23565 Node created: 34208 uuLbUULb	Time: 6.548522233963013 Steps: 8 Node visited 396754 Node created: 722558 DDLullFu	Time: 0.11967873573303223 Steps: 8 Node visited 7275 Node created: 9926 uuLbUULb	Time: 0.9035971164703369 Steps: 8 Node visited 67965 Node created: 102062 DDLuLLFu	Time: 0.013960361480712 89 Steps: 8 Node visited 606 Node created: 782 uuLbUULb	Time: 0.007979154586791992 Steps: 8 Node visited 473 Node created: 554 udFdLLFu
LLubDBFbBF	Time: 0.03286337852478027 Steps: 6 Node visited 2882 Node created: 3918 BuBUbb	Time: 0.5505273342132568 Steps: 6 Node visited 33920 Node created: 50099 FrFULL	Time: 0.00902557373046875 Steps: 6 Node visited 720 Node created: 935 BuBUBB	Time: 0.022920846939086914 Steps: 6 Node visited 2069 Node created: 2664 FlUBrl	Time: 0.00299167633056640 62 Steps: 6 Node visited 158 Node created: 203 BuBUBB	Time: 0.006949186325073242 Steps: 6 Node visited 440 Node created: 527 FrFULL
fbURfuLrrU	Time: 0.346315860748291 Steps: 8 Node visited 20344 Node created: 29365 dBRRbrBD	Time: 4.2898101806640625 Steps: 8 Node visited 242539 Node created: 425131 dFduldRD	Time: 0.0658257007598877 Steps: 8 Node visited 5029 Node created: 6801 BrrDrbrB	Time: 0.8557112216949463 Steps: 8 Node visited 64944 Node created: 96070 dBUBdrbb	Time: 0.01695561408996582 Steps: 8 Node visited 726 Node created: 939 BRRDrbrB	Time: 0.05388355255126953 Steps: 8 Node visited 3086 Node created: 4033 dBlrdlFU
LuDrlBLDrD	Time: 0.5668318271636963 Steps: 8 Node visited 43507 Node created: 65322 dRdrdffr	Time: 15.1315855979919 43 Steps: 8 Node visited 818515 Node created: 1563018 uBurbddr	Time: 0.09968137741088867 Steps: 8 Node visited 7731 Node created: 10596 dRdrdFFr	Time: 1.5478265285491943 Steps: 8 Node visited 95677 Node created: 147564 dRdrdFFl	Time: 0.02596378326416015 6 Steps: 8 Node visited 1019 Node created: 1344 dRdrdFFr	Time: 0.16456961631774902 Steps: 8 Node visited 6861 Node created: 9270 dRdlfUUr

Hình 20: Một phần kết quả của giải thuật A* trước và sau khi cải tiến với 10 bước xoay ngẫu nhiên

Random20	h1 improved	h1 not improved	h2 improved	h2 not improved	h3 improved	h3 not improved
rrulDlBudRblRDUUDRBD	Time: 36.49340462684631 Steps: 12 Node visited 1706827 Node created: 3451399 LFDDLdLfLDDF	Timeout	Time: 11.051350593566895 Steps: 12 Node visited 706646 Node created: 1196076 ldfDlDlDLDfd	Timeout	Time: 1.3903350830078125 Steps: 12 Node visited 65917 Node created: 97962 ldfDlDlDLDfd	Time: 26.842226506484985 Steps: 12 Node visited 1109978 Node created: 2224376 dFlrdRLFuLfd
dFLLRRrrdFUbUbddDRuL	Time: 10.088959217071533 Steps: 10 Node visited 479026 Node created: 815722 ffdffRfDrr	Timeout	Time: 1.171658992767334 Steps: 10 Node visited 81648 Node created: 122433 ffdfRfDRR	Time: 18.432169198989868 Steps: 10 Node visited 1233631 Node created: 2385502 FFuLLBuLfb	Time: 0.1561722755432129 Steps: 10 Node visited 8513 Node created: 11649 ffdfRfDRR	Time: 1.5153248310089111 Steps: 10 Node visited 77330 Node created: 121521 bbdlrDbUBB
fUUrBLRfuBfblLUFFRrU	Time: 11.762083530426025 Steps: 10 Node visited 590342 Node created: 1016945 rUURuBRbRB	Timeout	Time: 1.5465009212493896 Steps: 10 Node visited 103976 Node created: 157751 rUURuBRbRB	Time: 26.94685411453247 Steps: 10 Node visited 1734141 Node created: 3464729 lbbRfLFrFR	Time: 0.09372782707214355 Steps: 10 Node visited 5262 Node created: 7199 rUURuBRbRB	Time: 1.093442440032959 Steps: 10 Node visited 56032 Node created: 86825 lBFUlDFdBR
bdRbrDlfluUuluBdFrDr	Time: 37.23308348655701 Steps: 12 Node visited 1742954 Node created: 3540398 dllfDldFldd l	Timeout	Time: 16.994866371154785 Steps: 12 Node visited 974624 Node created: 1705787 fllfdffdLflF	Timeout	Time: 1.2810072898864746 Steps: 12 Node visited 61667 Node created: 90799 fLFdLfDlDlfL	Time: 23.385149717330933 Steps: 12 Node visited 977931 Node created: 1904527 bDRbRuLdBrbD
UfddFlRDDulrUFFFdLFd	Time: 43.24453544616699 Steps: 12 Node visited 1967103 Node created: 4094438 RurBruuRRbru	Timeout	Time: 12.094676733016968 Steps: 12 Node visited 698585 Node created: 1174812 ubRRUbRuBURb	Timeout	Time: 1.3746783733367 92 Steps: 12 Node visited 63857 Node created: 94096 rbruBrBBURbr	Time: 24.603622913360596 Steps: 12 Node visited 1031872 Node created: 2044221 LDfrDlbDFdLB
fDllurDrFBuLBllfrRFF	Time: 8.706329345703125 Steps: 10 Node visited 495692 Node created: 839742 BrDrbDDRBd	Time: 198.38760328292847 Steps: 10 Node visited 10244575 Node created: 25616880 FuFubddLBl	Time: 1.2184600830078125 Steps: 10 Node visited 84168 Node created: 124464 dBDrDbDRBd	Time: 18.667664527893066 Steps: 10 Node visited 1250895 Node created: 2409162 FuFubddRDr	Time: 0.14053630828857422 Steps: 10 Node visited 7806 Node created: 10602 dBDrDbDRBd	Time: 0.9685268402099609 Steps: 10 Node visited 50944 Node created: 77346 FuFubduBUr

Hình 21: Một phần kết quả của giải thuật A* trước và sau khi cải tiến với 20 bước xoay ngẫu nhiê

Từ hai hình trên, ta có thể thấy được các giải thuật đã cải tiến có thời gian thực hiện ngắn hơn nhiều so với giải thuật chưa được cải tiến.

2.3 Tính toán trung bình các thông số

Giải thuật	Runtime (s)	Steps	Node visited	Node created
BFS + improved	0.0050366	3.5	564.85	763.95
A* + $h1$	0.0260818	4	2390.4	3210.4
A* + $h1$ + improved	0.0049371	3.5	302.9	400.15
A* + $h2$	0.0030916	4	207.45	245.05
A* + $h2$ + improved	0.0013967	3.5	52.9	66.5
A* + $h3$	0.0007979	4	45.25	48.75
A* + $h3$ + improved	0.0010472	3.5	20.75	24.15

Bảng 1: Trung bình các thông số sau khi xoay ngẫu nhiên rubik 5 lần

Giải thuật	Runtime (s)	Steps	Node visited	Node created
BFS	1.8093448	7	149621.2	272453.8
A*+$h1$	22.648288	7.2	981101.8	2217337.6
A*+$h1$+improved	0.8580980	7	52587.9	85020.15
A*+$h2$	3.2692878	7.2	179172.6	333556.95
A*+$h2$+improved	0.1737502	7	11047.5	16029.15
A*+$h3$	0.1290545	7.2	6007.75	8721.7
A*+$h3$+improved	0.0245343	7	843.15	1110.9

Bảng 2: Trung bình các thông số sau khi xoay ngẫu nhiên rubik 10 lần

Giải thuật	Runtime (s)	Steps	Node visited	Node created	Timeout cases
BFS	8.103868757	9.3	725193.1	1418293.2	0
A* + $h1$	106.0843174	9.3	5690110.4	13999891.9	6
A* + $h1$ +improved	5.511020865	9.3	302043.5	507860.7	0
A* + $h2$	15.78089237	9.3	951336.1	1849544.2	4
A* + $h2$ +improved	0.965179324	9.3	65509	97686.7	0
A* + $h3$	0.866992218	9.3	41084.3	63234.5	0
A* + $h3$ +improved	0.104494248	9.3	5400.9	7338.4	0

Bảng 3: Trung bình các thông số sau khi xoay ngẫu nhiên rubik 20 lần

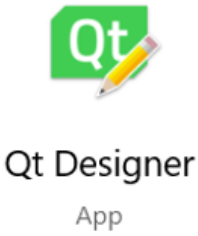
Từ các bảng thống kê trung bình ta có thể rút ra được kết luận đó là hàm lượng giá $h3$ đã được cải tiến có hiệu quả nhất, tiếp theo đó lần lượt là các hàm lượng giá $h2$ đã được cải tiến và $h1$ đã được cải tiến. Tất cả các giải thuật đã cải tiến có hiệu quả hơn giải thuật chưa được cải tiến và BFS.

Link bảng thống kê chi tiết:
docs.google.com/spreadsheets/d/1BEO9w8cIVbdnQDulpm6StBSYIO4n-fPTrV67prAoXzg

3. Hiện thực GUI

3.1 Ngôn Ngữ Và Phần Mềm Hỗ Trợ

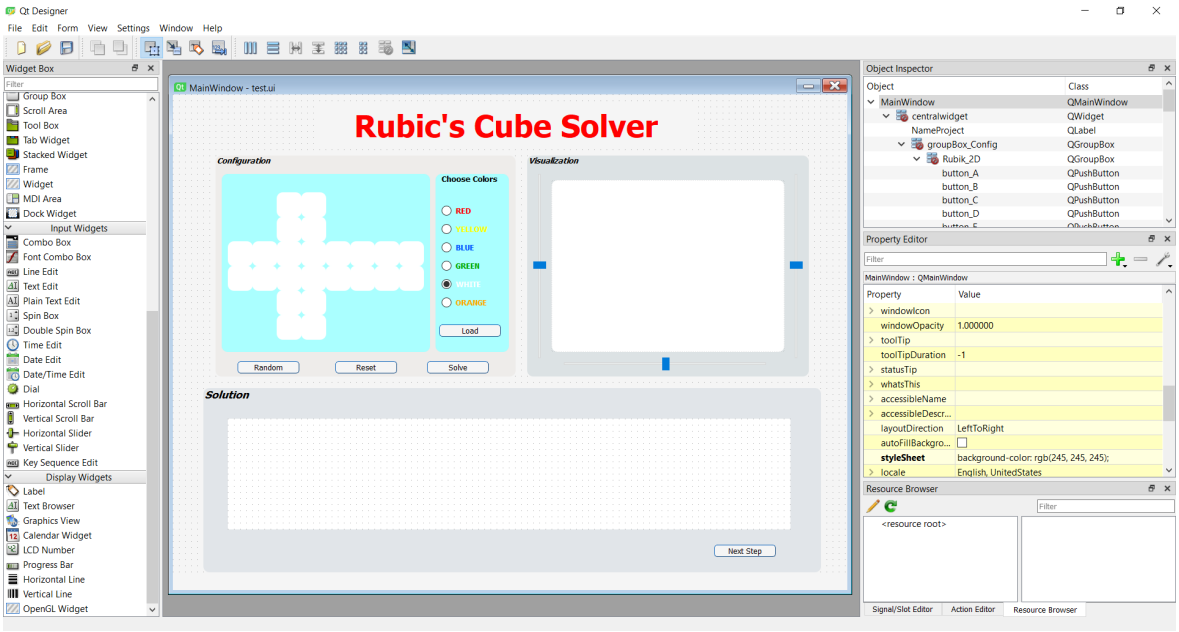
- Ngôn ngữ: Python
- Package: PyQt5 và matplotlib
- Phần mềm hỗ trợ: Qt Designer



Hình 22: Logo app Qt Designer

3.2 Mô Tả Hiện Thực

Phần 1: Sử dụng Qt Designer để tiến hành thiết kế những phần chính của GUI dựa trên những tính năng mà app hỗ trợ



Hình 23: App Qt Designer

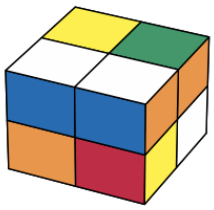
Sau đó chuyển file .ui sang file .py và tiếp tục thực hiện.

Phần 2: Kết nối GUI với project

Phần chính của GUI được thực hiện trong file layout.py và run file app.py để hiển thị GUI.

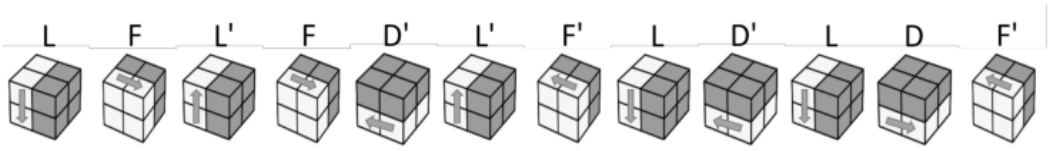
File layout.py gồm có 4 Class:

- Class ***RubikCube***: Thực hiện một số công việc liên quan đến Rubik: reset Rubik, random Rubik, solve solution
- Class ***my3Dcanvas***



Hình 24: Ảnh 3D

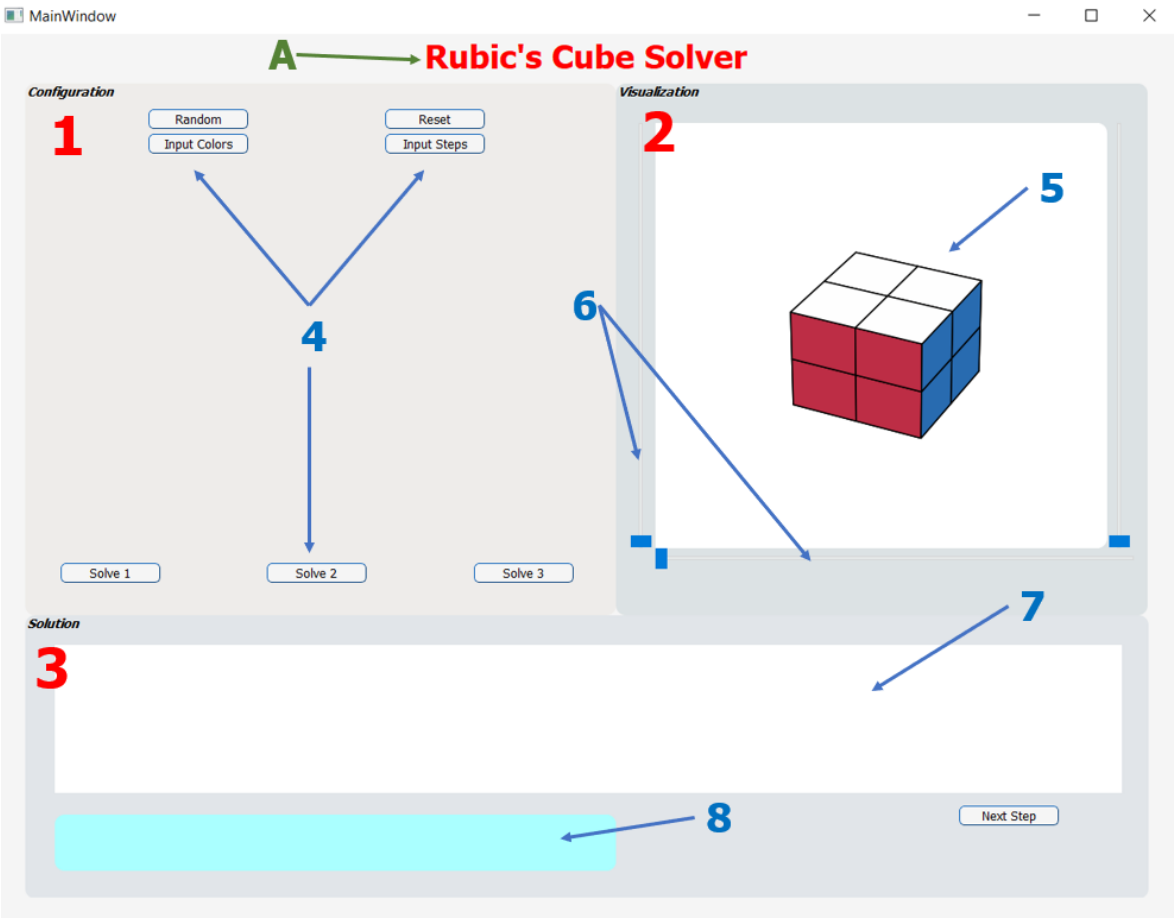
– Class ***my2Dcanvas***



Hình 25: Ảnh 2D

– Class ***Ui_MainWindow***: Setup thành phần GUI (title, scroll bar, button, ...)

3.3 Mô Tả Giao Diện



Hình 26: GUI

Mô tả:

A. Title "Rubic's Cube Solve"

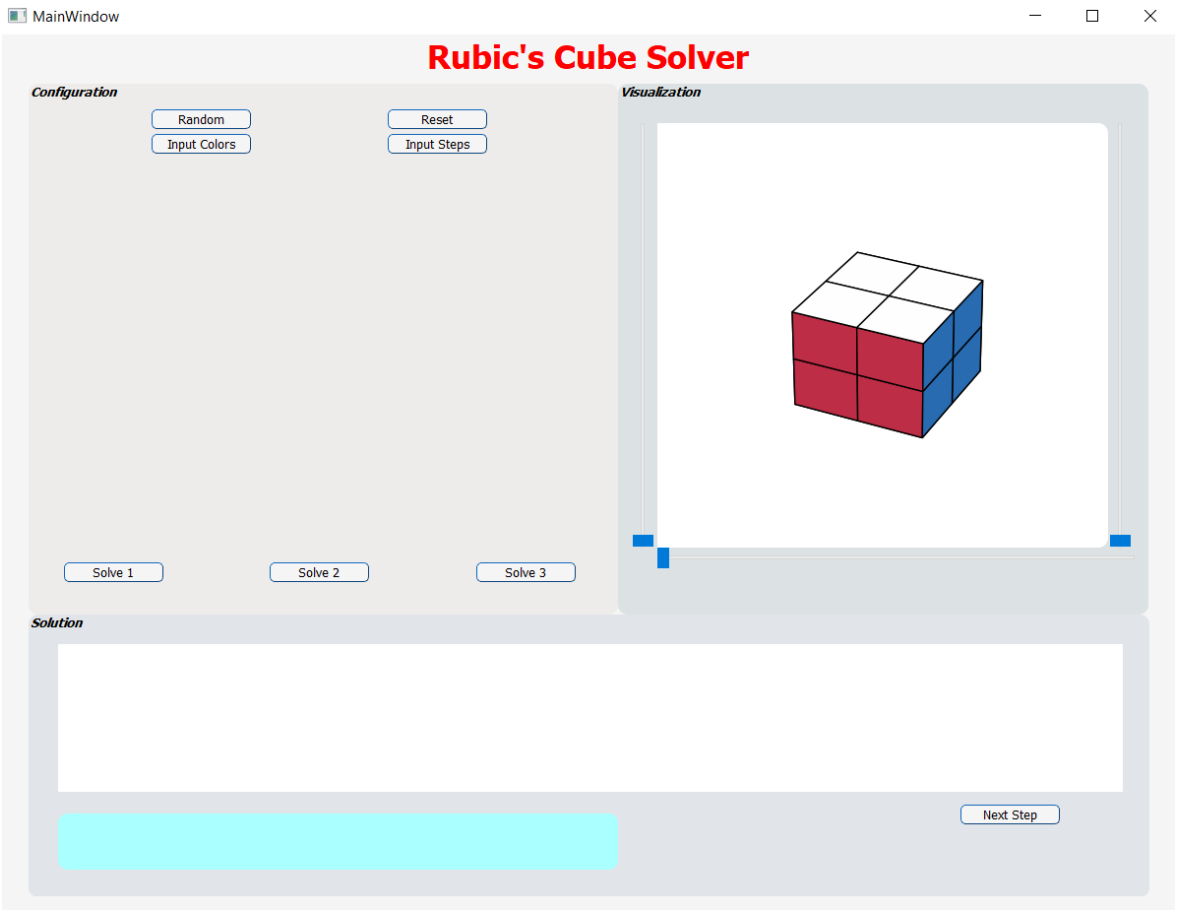
1. GroupBox Configuration: vùng nhập input

2. GroupBox Visualization: vùng hiển thị hình ảnh Rubik 3D

3. GroupBox Solution: vùng hiển thị solution
4. Các button
5. Hiển thị Rubik
6. Các thanh Scroll Bar
7. Hiển thị Solution
8. Hiển thị thông tin về số lượng generated states, visited states và time run

3.4 Hướng dẫn sử dụng phần mềm

Bước 1: Run file app.py

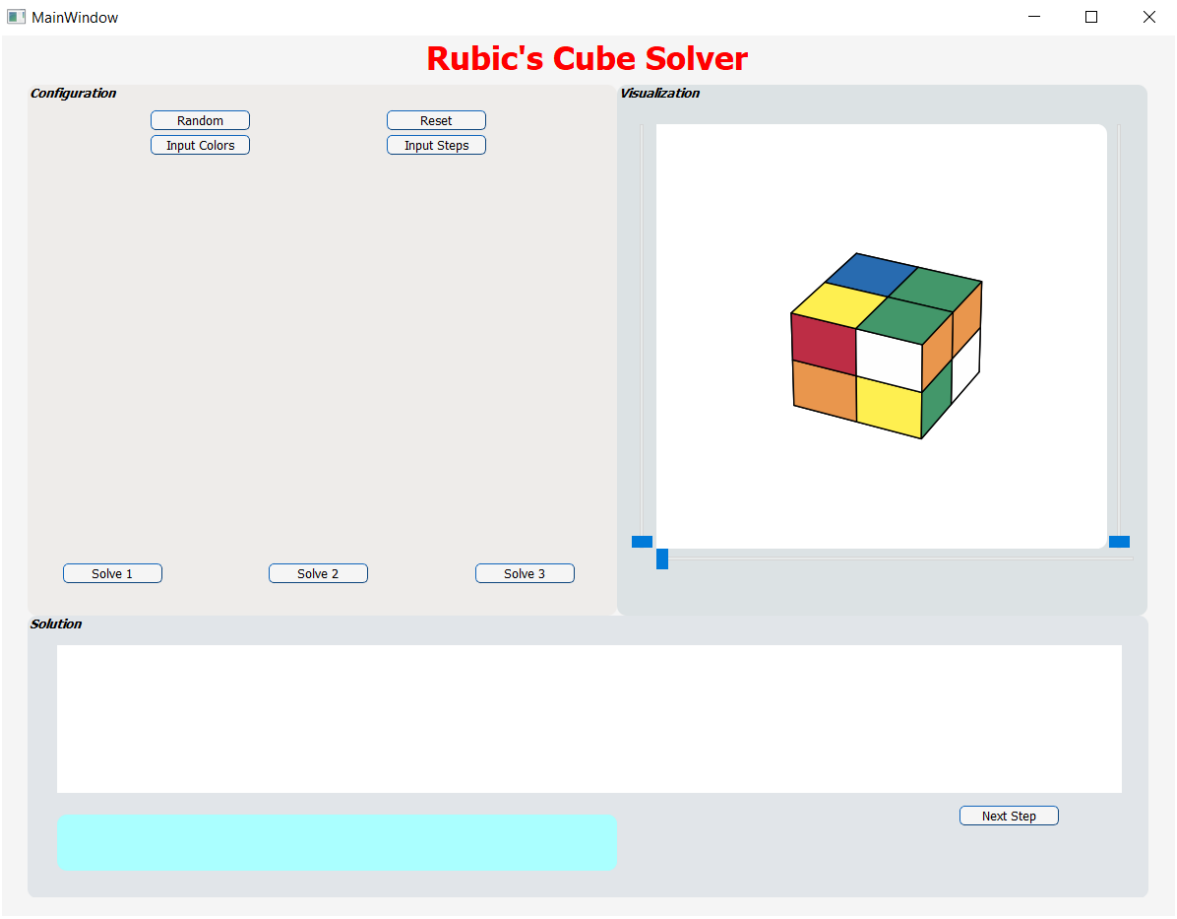


Hình 27: Hướng dẫn sử dụng bước 1

Bước 2: Chọn cách nhập Input

a) Input Random

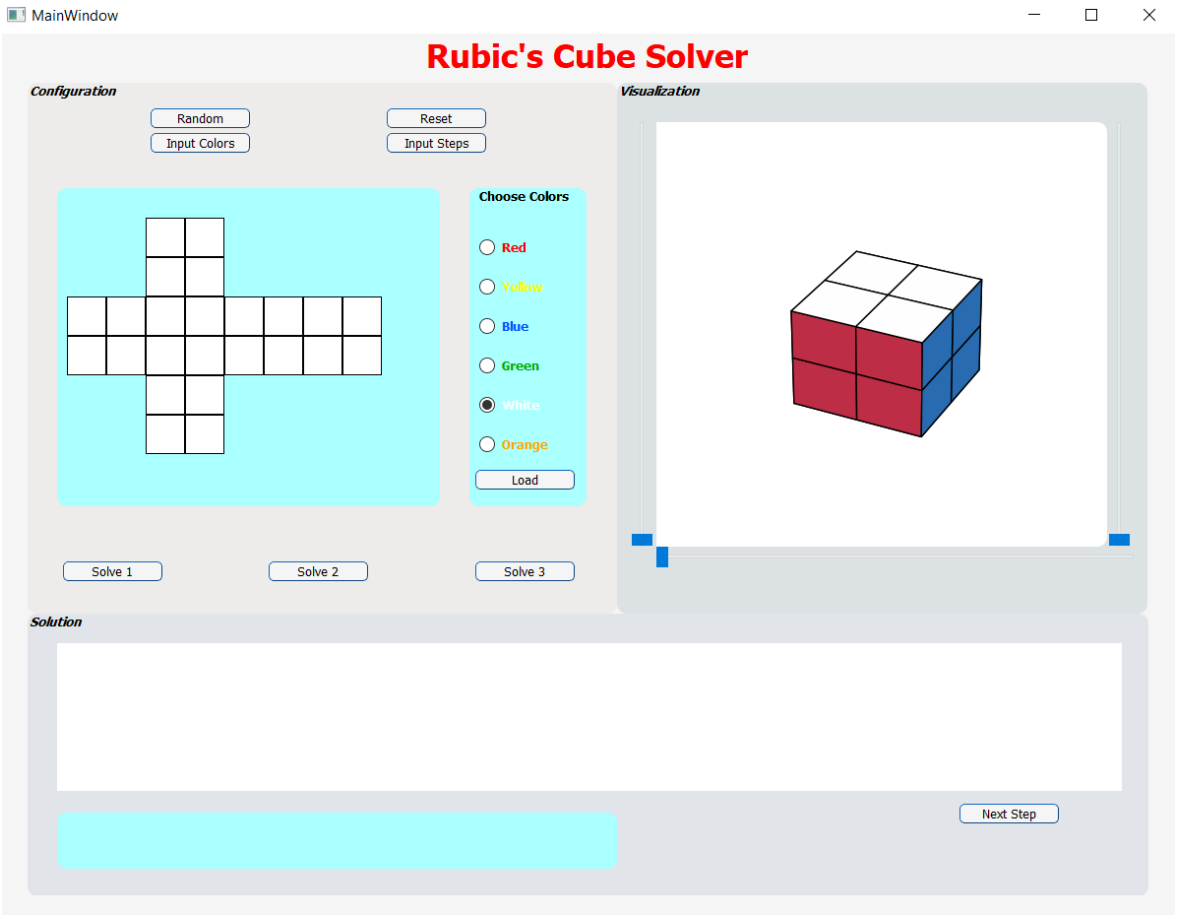
Chọn "Random", Rubik sẽ tự động xáo trộn và hiển thị bên Visualization



Hình 28: Hướng dẫn sử dụng bước 2a

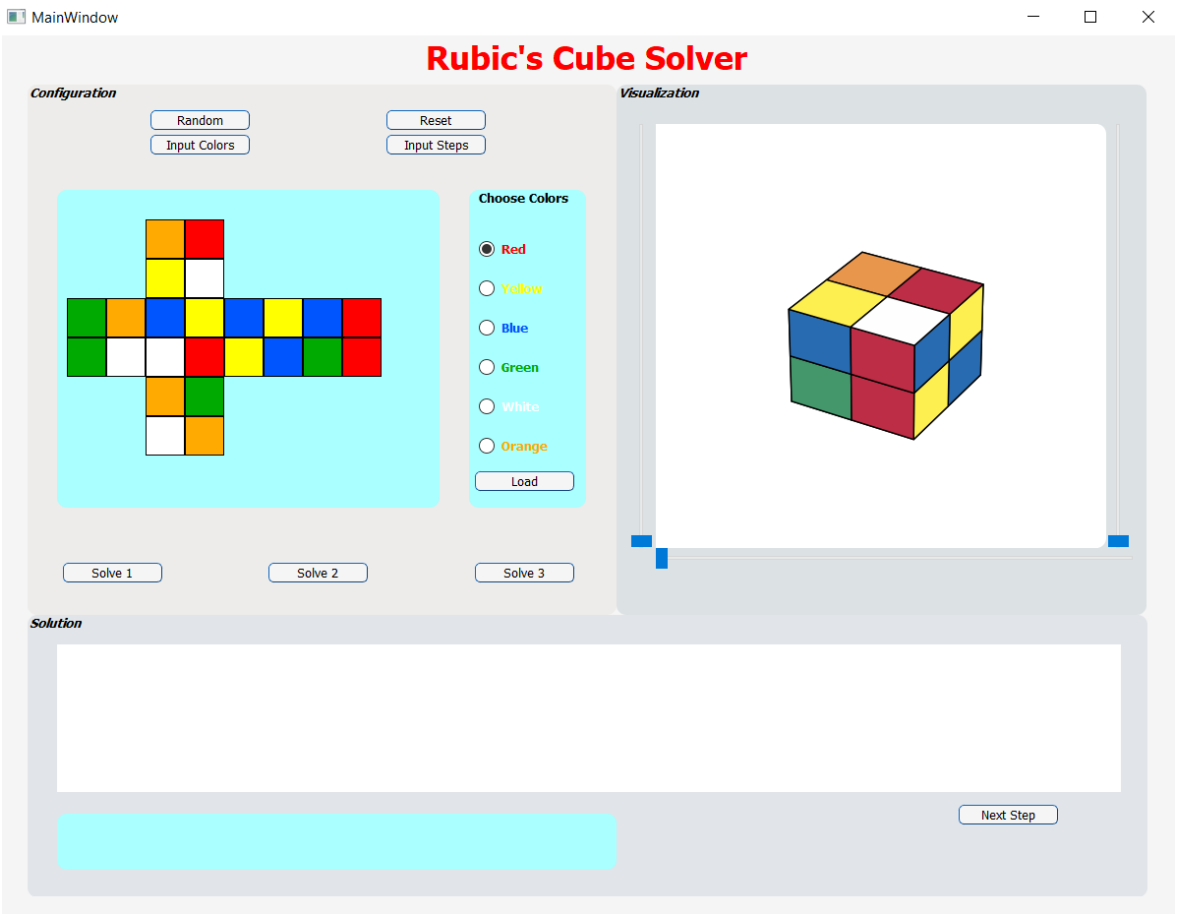
b) Input Colors

Chọn “Input Colors”, mặc định sẽ là White



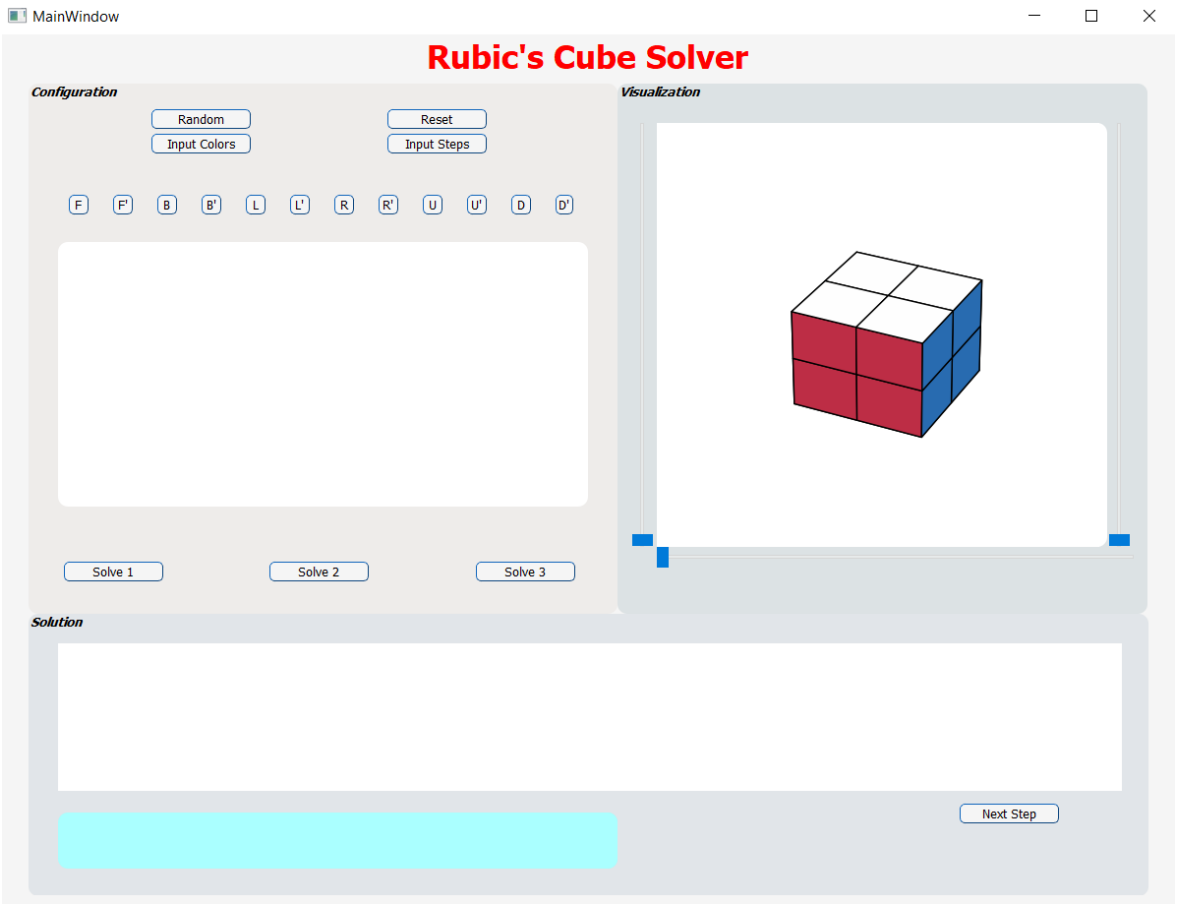
Hình 29: Hướng dẫn sử dụng bước 2b1

Sau đó, chọn màu ở bên group “Choose Colors” rồi “tô” vào các ô vuông Rubik ở dạng 2D. Sau khi xong, chọn “Load” , phần input sẽ được hiển thị sang bên Visualization



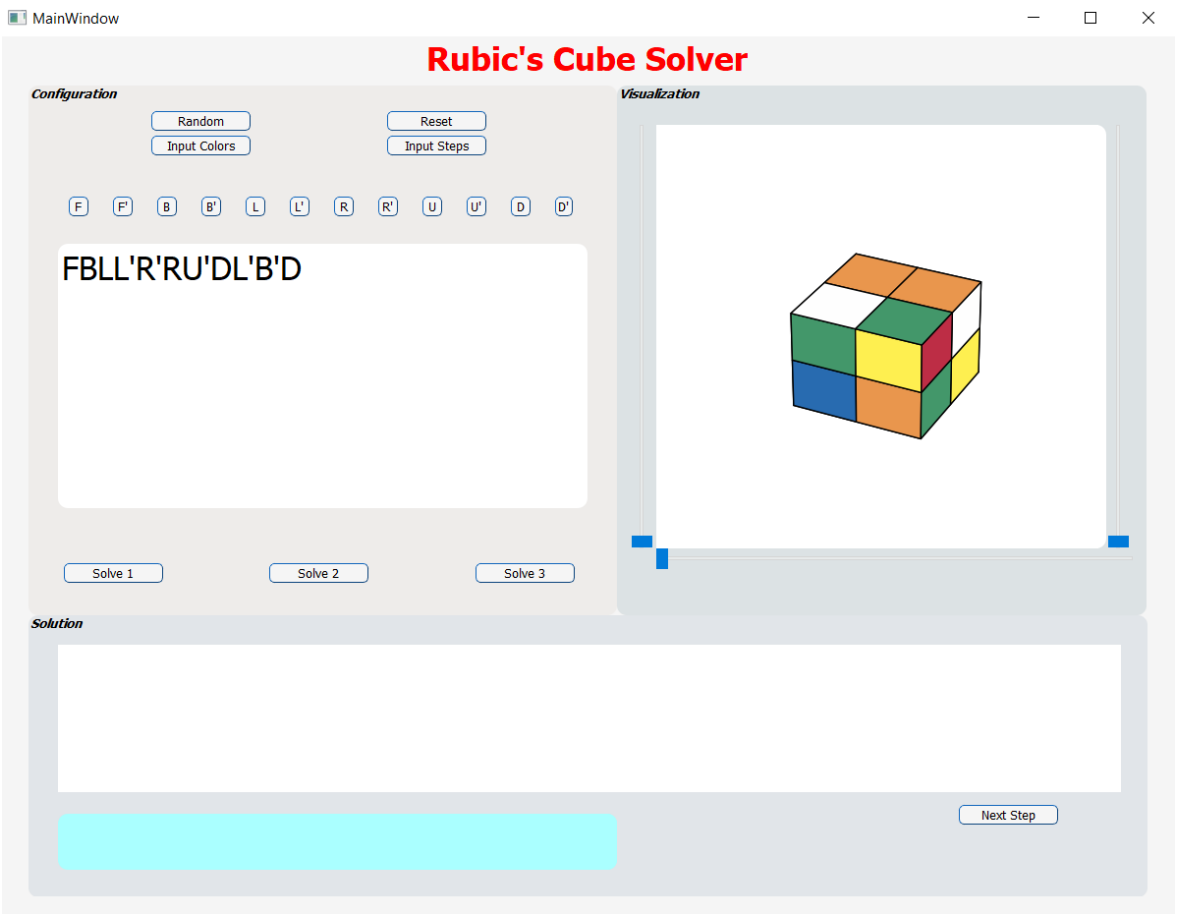
Hình 30: Hướng dẫn sử dụng bước 2b2

c) Input Steps Chọn "Input Steps"



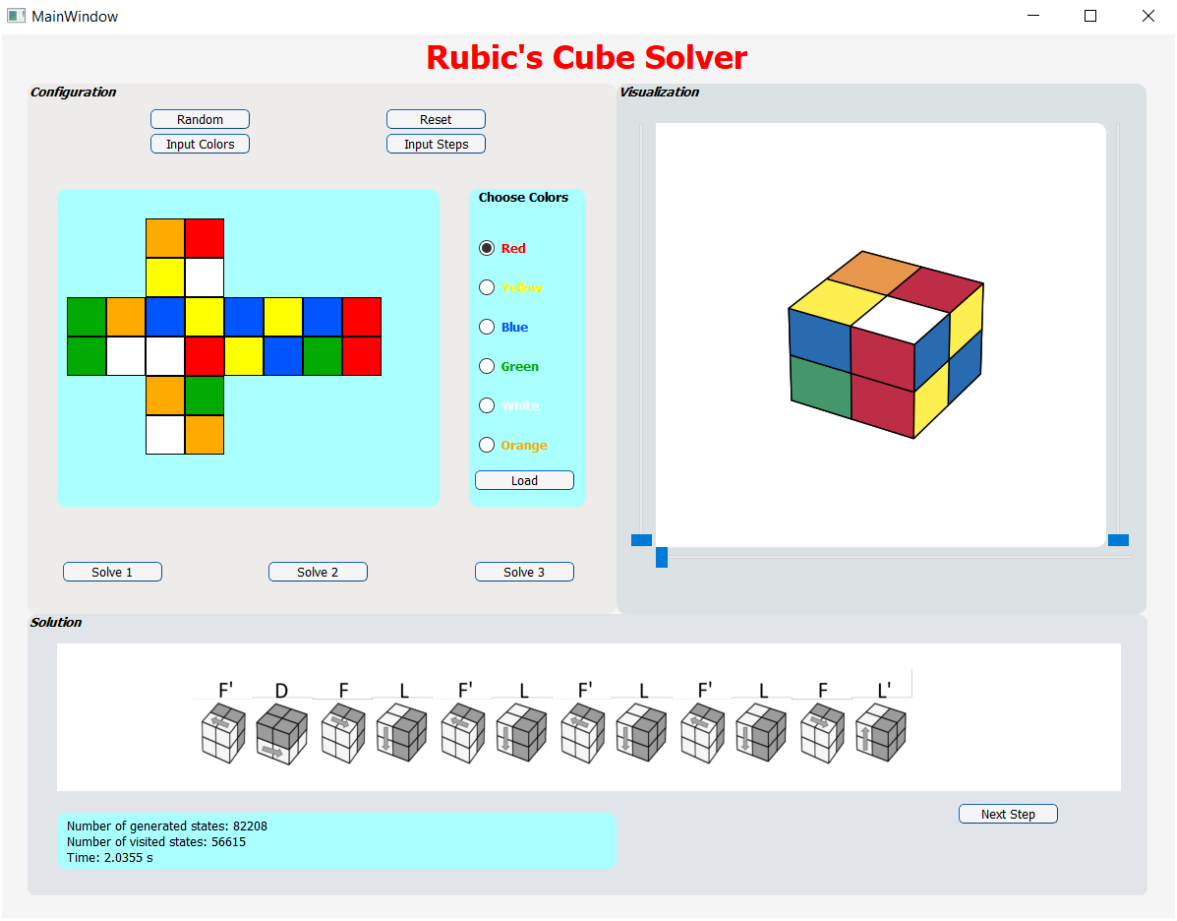
Hình 31: Hướng dẫn sử dụng bước 2c1

Sau đó click chọn các nút: F, F’, B, B’,⋯để thực hiện việc xáo trộn Rubik.



Hình 32: Hướng dẫn sử dụng bước 2c2

Bước 3: Chọn một trong ba nút “Solve 1”, “Solve 2”, “Solve 3” để tiến hành tìm và in solution



Hình 33: Hướng dẫn sử dụng bước 3

TÀI LIỆU THAM KHẢO

[1] Wikipedia. Rubik's Cube, Pocket Cube, Giải thuật tìm kiếm A*. Last access: 10/2021.

[2] Richard E. Korf. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. Computer Science Department University of California, Los Angeles. Last access: 10/2021.

[3] TutorialsPoint. PyQt5 tutorial. Last access: 10/2021