NHẬP MÔN TRÍ TUỆ NHÂN TẠO

GVHD: Vương Bá Thịnh

Đề bài tập lớn I: Trò chơi Bloxorz

Thành viên:

_ Phan Trần Thái Sơn	1512853
_ Huỳnh Minh Thịnh	1513245
_ Lê Công Huy	1511230
_ Nguyễn Hoàng Mẫn Tiến	1513447
_ Nguyễn Phạm Trí Thiện	1513211

Contents

I. PHÂN TÍCH ĐỀ BÀI VÀ ĐỊNH HƯỚNG LÀM VIỆC

1. Phân tích đề bài

Đề bài: Áp dụng trí tuệ nhân tạo giải trò chơi Bloxorz bằng cách hiện thuật 3 giải thuật chính:

- Duyệt theo chiều rộng: Breadth First Search
- Duyệt theo chiều sâu: Depth First Search
- Giải thuật Heuristic

2. Định hướng làm việc

Tổ chức họp nhóm mỗi tuần nhằm thảo luận, phân chia công việc và giải quyết những vấn đề khó khăn còn tồn đọng khi mỗi thành viên hiện thực code.

Thời gian làm việc gồm khoảng 4 tuần, trong đó:

- Tuần thứ nhất họp nhóm, tìm hiểu đề bài, phân tích trò chơi để đưa những giải thuật vào trò chơi, xác định trạng thái và một số luật cơ bản, …
- Tuần 2-4: Hiện thực giải thuật (Code)
- 2 ngày cuối cùng: Tổng hợp và tinh chỉnh lại bài nộp. Đánh giá hiệu quả của từng giải thuật và viết báo cáo.

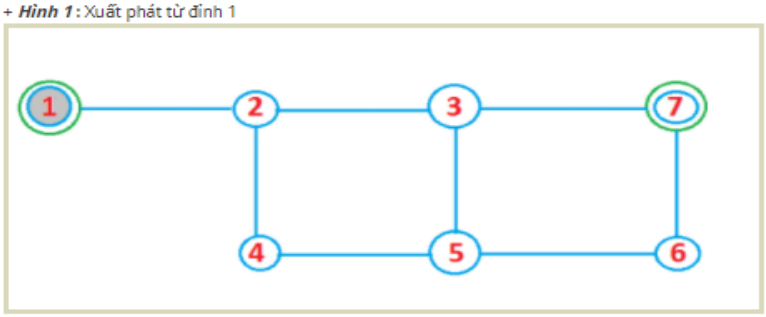
II. LÝ THUYẾT

1. Duyệt theo chiều rộng: Breadth First Search

- Xuất phát từ một đỉnh và đi tới các đỉnh kề nó, tiếp tục cho đến khi không còn đỉnh nào có thể đi.
- Trong quá trình đi đến đỉnh kề, tiến hành lưu lại đỉnh cha của đỉnh kề để đi ngược lại từ đỉnh kết thúc đến đỉnh xuất phát, ta có thể có được đường đi ngắn nhất.
- Sở dĩ thuật toán này tìm được đường đi ngắn nhất là nhờ vào cơ chế tô màu và lưu đỉnh cha. Quá trình tô màu khiến một đỉnh không thể xét 2

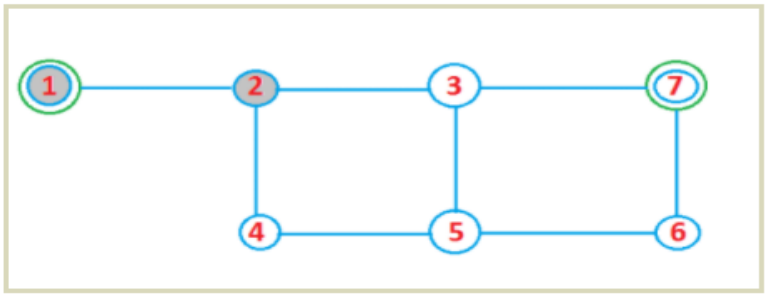
lần trở lên và có thể xem được đường đi từ đỉnh kết thúc đến đỉnh xuất phát dựa vào việc lưu đỉnh cha.

+ ***Hình 1*** : Xuất phát từ đỉnh 1



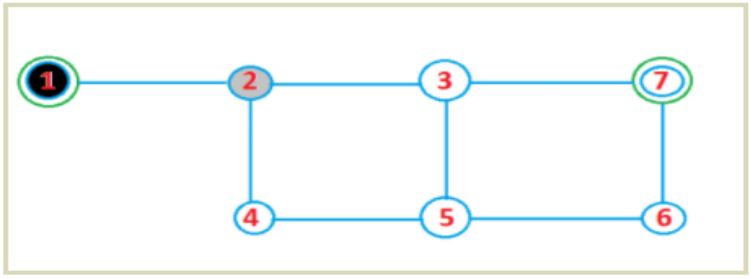
Hình 1

+ ***Hình 2*** : Đi đến đỉnh 2, như vậy nút 1 là nút cha của nút 2



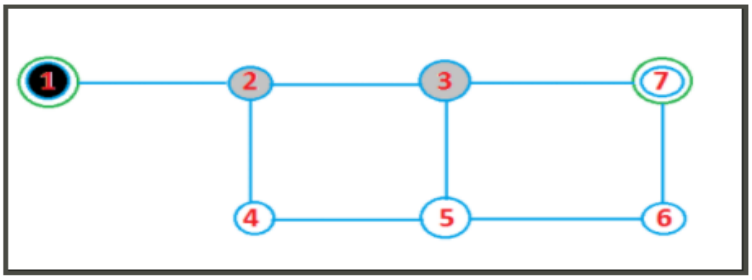
Hình 2

+ ***Hình 3*** : Đã đi hết tất cả các đỉnh kề của đỉnh 1, tiến hành bôi đen đỉnh 1



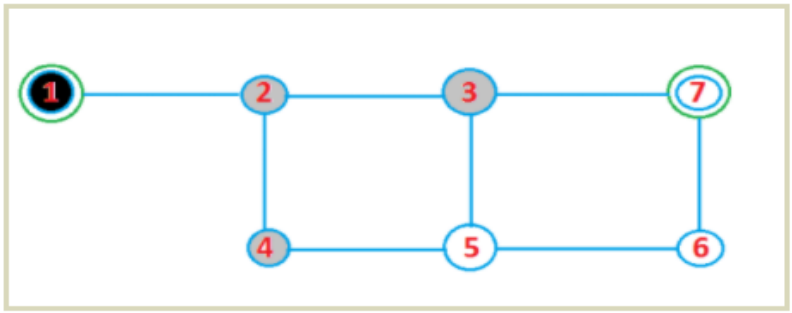
Hình 3

+ ***Hình 4*** : Xuất phát từ đỉnh 2, chọn đỉnh 3, nút cha của đỉnh 3 là đỉnh 2



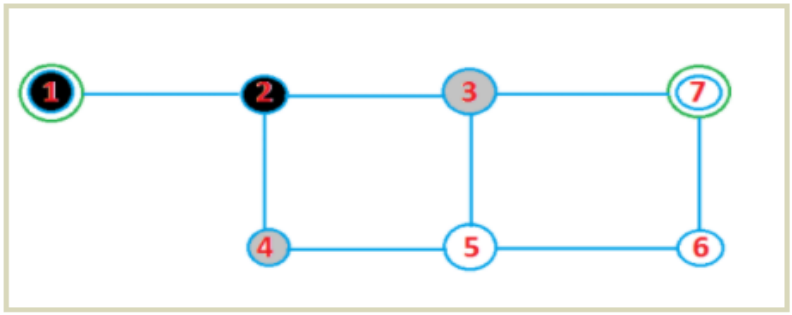
Hình 4

+ ***Hình 5*** : Xuất phát từ đỉnh 2, bôi đen đỉnh 4, nút cha của đỉnh 4 là đỉnh 2



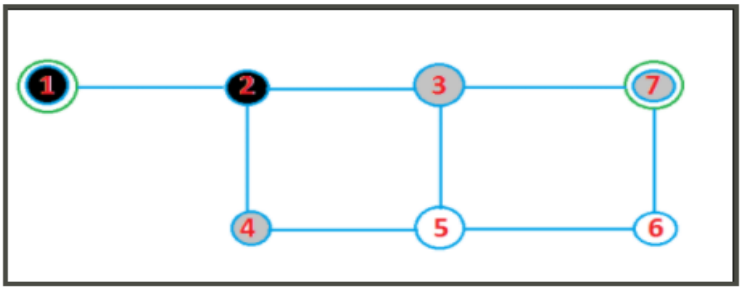
Hình 5

+ ***Hình 6*** : Đã đi hết tất cả các đỉnh kề của đỉnh 2, tiến hành bôi đen đỉnh 2



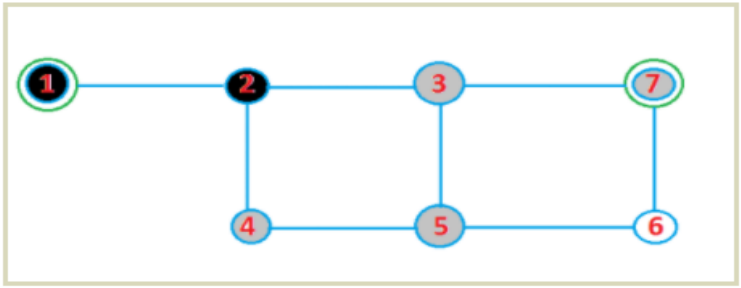
Hình 6

+ ***Hình 7*** : Xuất phát từ đỉnh 3, đi đến đỉnh 7, như vậy đỉnh 3 là đỉnh cha của đỉnh 7



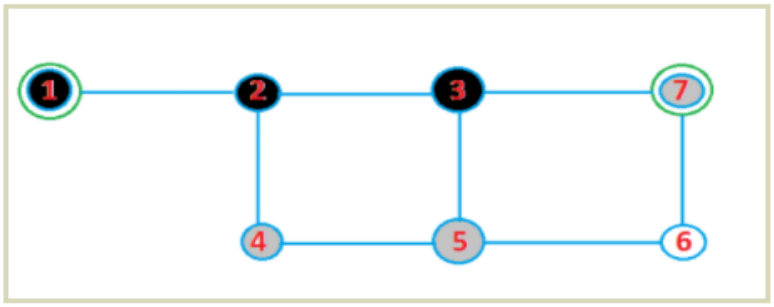
Hình 7

+ ***Hình 8*** : Xuất phát từ đỉnh 3, đi đến đỉnh 5, như vậy đỉnh 3 là đỉnh cha của đỉnh 5



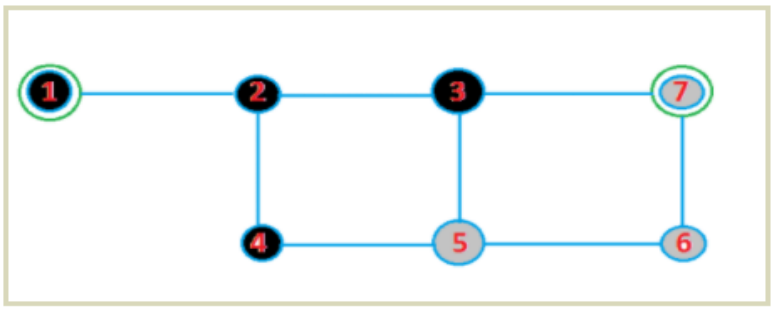
Hình 8

+ ***Hình 9*** : Đã đi hết tất cả các đỉnh kề của đỉnh 3, tiến hành bôi đen đỉnh 3



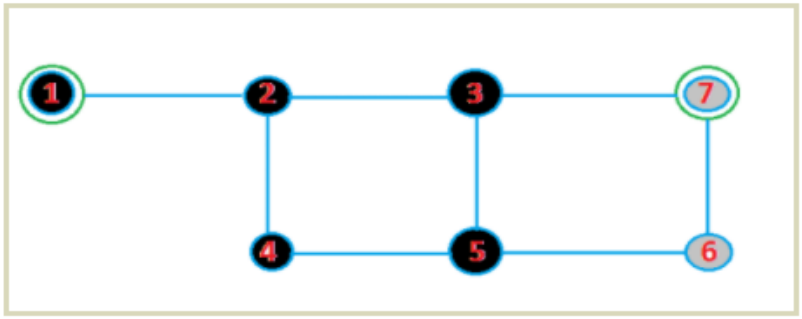
Hình 9

+ ***Hình 10*** : Xuất phát từ đỉnh 5, đi đến đỉnh 6, như vậy đỉnh 5 là đỉnh cha của đỉnh 6



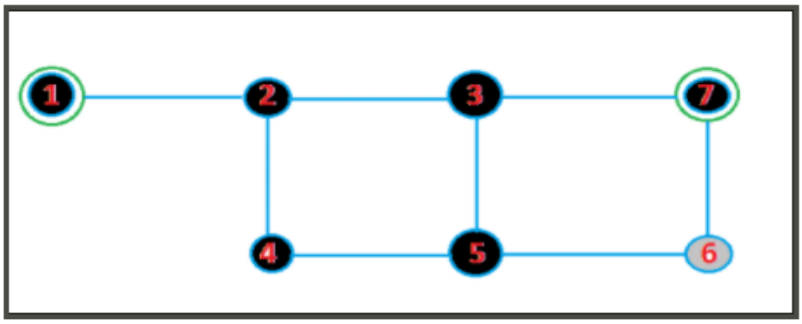
Hình 10

+ ***Hình 11***: Đã đi hết tất cả các đỉnh kề của đỉnh 5, tiến hành bôi đen đỉnh 5



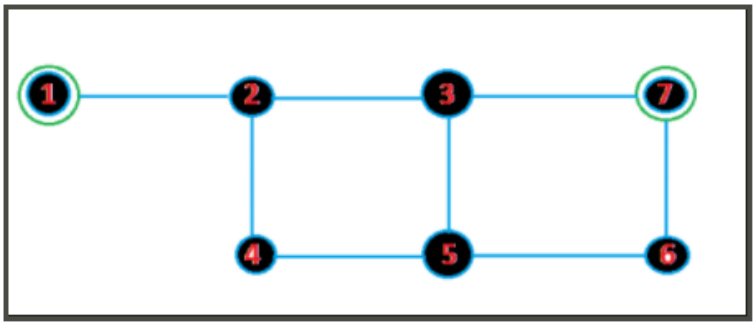
Hình 11

+ ***Hình 12***: Đã đi hết tất cả các đỉnh kề của đỉnh 7, tiến hành bôi đen đỉnh 7



Hình 12

+ ***Hình 13***: Đã đi hết tất cả các đỉnh kề của đỉnh6, tiến hành bôi đen đỉnh 6



Hình 13

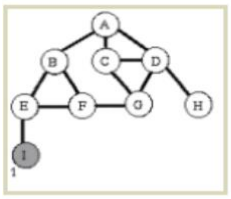
2. Duyệt theo chiều sâu: Depth First Search

- Xuất phát từ một đỉnh và đi mãi cho đến khi không thể đi tiếp được nữa, sau đó quay lại đỉnh đầu.
- Trong quá trình quay lại:

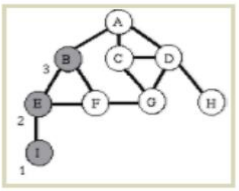
+ Nếu gặp đường đi khác thì đi cho đến khi không đi tiếp được nữa.

+ Nếu không tìm được đường đi nào khác thì ngừng việc tìm kiếm.

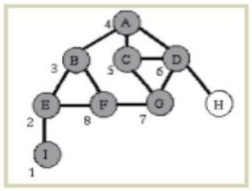
- Trong quá trình đi đến đỉnh khác, thuật toán sẽ lưu lại đỉnh cha vừa đi qua để khi đi ngược lại từ đỉnh kết thúc đến đỉnh xuất phát.
- Sở dĩ thuật toán này tìm được đường đi là nhờ vào cơ chế tô màu và lưu đỉnh cha. Quá trình tô màu khiến một đỉnh không thể xét hai lần trờ lên và có thể xem được đường đi từ đỉnh kết thúc đến đỉnh bắt đầu.



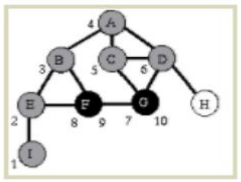
Hình 1



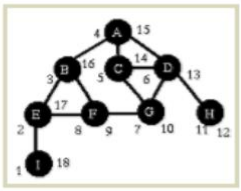
Hình 3



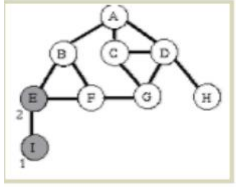
Hình 5



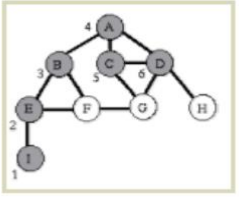
Hình 7



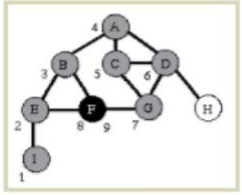
Hình 9



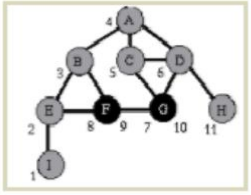
Hình 2



Hình 4



Hình 6



Hình 8

III. QUÁ TRÌNH LÀM VIỆC

- Như đã trình bày trong phần I nhóm đã bắt đầu thảo luận và làm việc kể từ tuần đầu tiên được giao đề assignment. Trong đó phân chia công việc như sau:
- Thảo luận trình bày tìm hiểu thuật toán cũng như trò chơi: Tất cả thành viên nhóm
- Hiện thực giải thuật Tìm kiếm theo chiều rộng: 3 người
- Hiện thực giải thuật Tìm kiếm theo chiều sâu: 2 người
- Hiện thực giải thuật Heuristic: 2 người
- Hiện thực map: 1 người
- Hiện thực phần hiển thị trực quan map và các nước đi: 1 người
- Thảo luận đánh giá hiệu năng giải thuật và viết báo cáo: 2 người

IV. PHÂN TÍCH CODE

Map là 1 mảng 2 chiều chứa các số nguyên được qui ước như sau:

- 0: vực, nếu khối chạm vào ô này sẽ thua.
- 1: ô bình thường.

- -2: Vật thể O có tác dụng đóng/mở một số ô khác khi khối chạm vào nó.
- 2: Vật thể X có tác dụng đóng/mở một số ô khác khi và chỉ khi khối ở trạng thái ĐỨNG trên nó.
- 4: đích đến.
- 5: ô yếu, nếu khối ở trạng thái ĐỨNG trên nó thì sẽ thua.
- 6: Vật thể dịch chuyển có tác dụng tách khối ra làm 2 và dịch chuyển khối đến vị trí khác khi khối ở trạng thái ĐỨNG trên nó.

Các hàm/class cần chú ý là:

- Node
  Class Node bao gồm toạ độ của Block, địa chỉ Node trước đó trong cây, nước di chuyển, map và trạng thái(state) của XO object.
- is_stand
  Hàm is_stand trả về giá trị Boolean, cho biết trạng thái của Block (true nếu khối đang đứng, false nếu khối đang nằm).
- State
  Class State bao gồm vị trí của Block, map, XOObject, SplitObject, vị trí bắt đầu, state của Block và các nước đã đi.
- next_position
  Hàm next_position kiểm tra trạng thái của Block và gọi hàm add_move
- add_move
  Hàm add_move kiểm tra trạng thái các Object và thêm các nước đi có thể vào node.
- notContain
  Hàm notContain kiểm tra xem Object có lặp lại nước trước không.

- is_valid
  Hàm is_valid kiểm tra tính khả thi của nước đi.
- add_state
  Hàm add_state thêm node vào state
- add_valid_state
  Hàm add_valid_state chạy hàm next_position để sinh ra các state tiếp theo từ state hiện tại, sau đó kiểm tra state hợp lệ bằng hàm is_valid, nếu hợp lệ thì xài hàm add_state để add vào.
- is_goal
  Hàm is_goal kiểm tra xem vị trí hiện tại của Block có phải đích không.
- check_goal
  Hàm check_goal kiểm tra nếu Block đã đủ điều kiện qua màn hay chưa.
- set_player_posistion
  Hàm set_player_position thay đổi map hoặc vị trí của Block khi Block tiếp xúc với XOObject.
- XOObject
  Class XOObject chứa loại Object, vị trí và vị trí chịu tác động của các Object đặc biệt trong màn.
- ManagedPosition
  Class ManagedPosition chứa vị trí và tác dụng của Object (Enable/Disable/Both).
- bfs
  Duyệt cây được tạo ra theo chiều rộng.
- dfs
  Duyệt cây được tạo ra theo chiều sâu.
- Level
  Class Level lưu state ban đầu của màn.

- draw_map
  Hàm draw_map hiển thị map và các bước di chuyển trực quan để tiện cho việc quan sát.
- map_copy
  Hàm map_copy sao chép màn sang 1 array khác để tránh việc thay đổi map ban đầu khi chạy giải thuật.
- init_level
  Hàm init_level lưu state bắt đầu và các XOObject của các màn vào level_array để sử dụng khi chạy giải thuật.
- test
  Hàm test chạy giải thuật bfs hay dfs trên array_level của init_level, tính thời gian thực thi của giải thuật trên từng màn.
- SplitObject
  Class SplitObject chứa vị trí của Object (.) (Object teleport) và vị trí của Block khi dịch chuyển.

V. ĐÁNH GIÁ HIỆU QUẢ GIẢI THUẬT

1. Đánh giá bộ nhớ tiêu tốn từng giải thuật

Bảng so sánh dưới đây tính kết quả bộ nhớ từng giải thuật trong 33 màn. Đơn vị là bytes.

	DFS	BFS
Màn 1	19070976	18980864
Màn 2	19795968	19537920
Màn 3	19886080	19771392
Màn 4	20185088	19968000
Màn 5	21303296	21016576
Màn 6	21565440	21344256
Màn 7	21893120	21803008
Màn 8	22265856	23576576
Màn 9	23547904	25030656

Màn 10	27226112	35635200
Màn 11	27684864	36163584
Màn 12	28303360	37154816
Màn 13	28733440	37613568
Màn 14	29519872	38993920
Màn 15	31989760	48218112
Màn 16	32161792	48484352
Màn 17	33996800	51601408
Màn 18	35418112	53444608
Màn 19	36036608	54099968
Màn 20	47525888	70438912
Màn 21	48111616	71172096
Màn 22	49090560	72470528
Màn 23	49549312	82300928
Màn 24	50429952	83873792
Màn 25	51122176	85446656
Màn 26	60977152	100290560
Màn 27	61743104	101265408
Màn 28	69607424	113283072
Màn 29	72179712	119906304
Màn 30	73228288	121634816
Màn 31	75190272	124809216
Màn 32	75923456	126730240
Màn 33	77496320	127401984

Nhận xét:

_ Càng đến các màn sau, ta càng thấy rõ giải thuât DFS sử dụng bộ nhớ ít hơn so với BFS, phù hợp với lý thuyết

_ Trường hợp càng phức tạp thì chênh lệch bộ nhớ được sử dụng giữa 2 giải thuật càng lớn.

2. Đánh giá thời gian thực thi từng giải thuật

Bảng so sánh dưới đây tính kết quả thời gian thực thi từng giải thuật trong 33 màn. Đơn vị là giây.

	DFS	BFS
Màn 1	0.0	0.0
Màn 2	0.0468	0.0156
Màn 3	0.0	0.0156
Màn 4	0.0156	0.0
Màn 5	0.0156	0.0356
Màn 6	0.0156	0.008
Màn 7	0.008	0.0
Màn 8	0.0	0.1503
Màn 9	0.0937	0.2222
Màn 10	0.2409	3.5914
Màn 11	0.008	0.024
Màn 12	0.012	0.056
Màn 13	0.0156	0.016
Màn 14	0.0162	0.6992
Màn 15	0.1416	3.1111
Màn 16	0.0	0.004
Màn 17	0.0959	0.2226
Màn 18	0.0569	0.925
Màn 19	0.012	0.0408
Màn 20	5.5219	8.3084
Màn 21	0.0126	0.0156
Màn 22	0.0156	0.0563
Màn 23	0.0	2.6795
Màn 24	0.0313	0.06
Màn 25	0.0156	0.7502
Màn 26	3.0899	10.0796
Màn 27	0.0156	0.0314
Màn 28	1.8122	5.3665
Màn 29	0.1299	0.5801
Màn 30	0.032	0.0679

Màn 31	0.0872	0.168
Màn 32	0.0156	0.08
Màn 33	0.0625	0.0081

Nhận xét:

_ Trong 33 màn nhìn chung phần lớn giải thuật DFS đều thực thi nhanh hơn giải thuật BFS.

_ Tuy nhiên trong một vài trường hợp do nhánh DFS phải duyệt quá nhiều khiến cho thời gian của BFS có nhanh hơn đôi chút tuy nhiên với những trường hợp này thời gian chênh lệch 2 bên không nhiều.

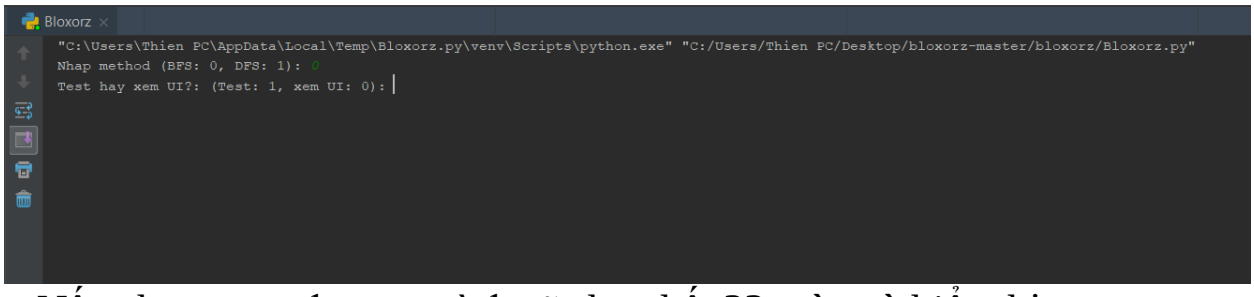
VI. CÁCH SỬ DỤNG:

_ Trước khi chạy chương trình, người dùng cần cài đặt pygame. https://www.pygame.org/wiki/GettingStarted

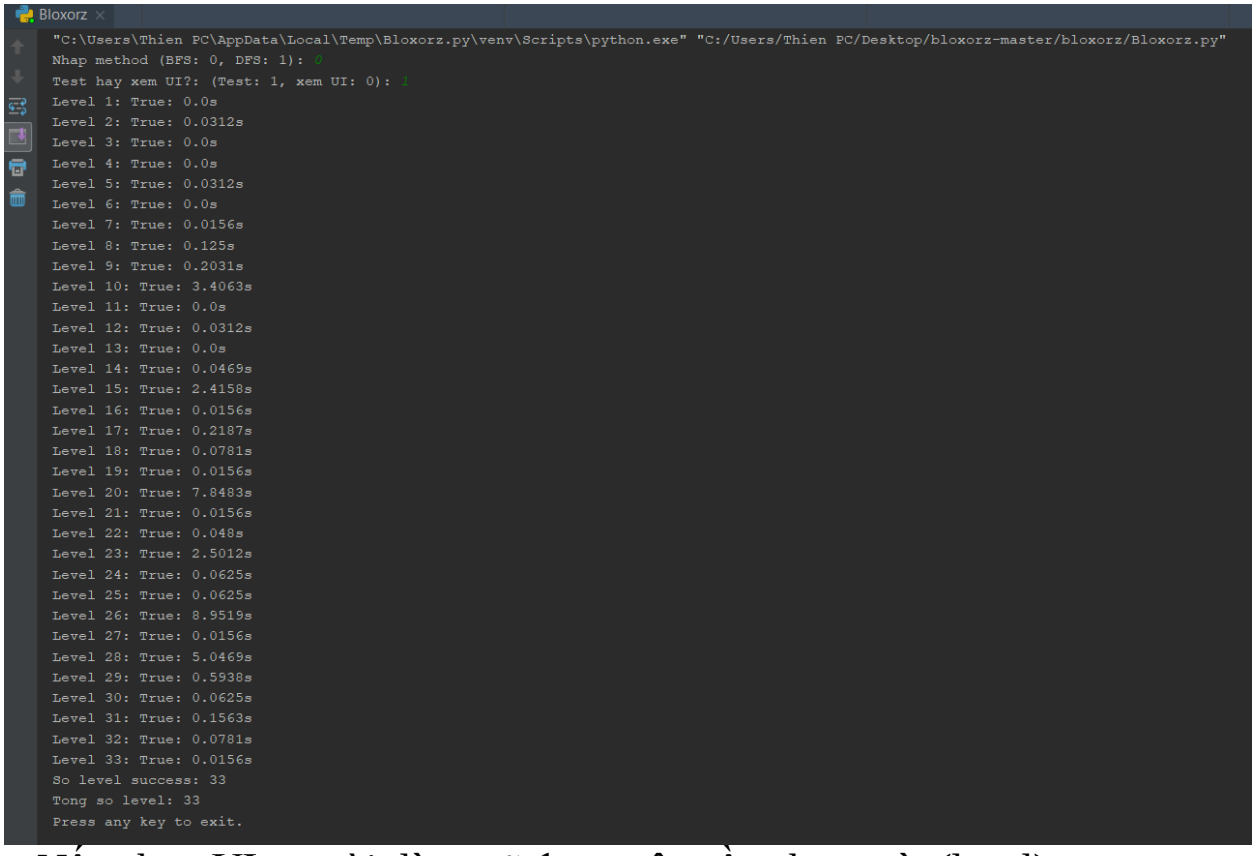
_ Để chạy chương trình, người dùng chạy file Bloxorz.bat.

_ Người dùng chọn method để giải bloxorz (0 nếu là BFS và 1 nếu là DFS).

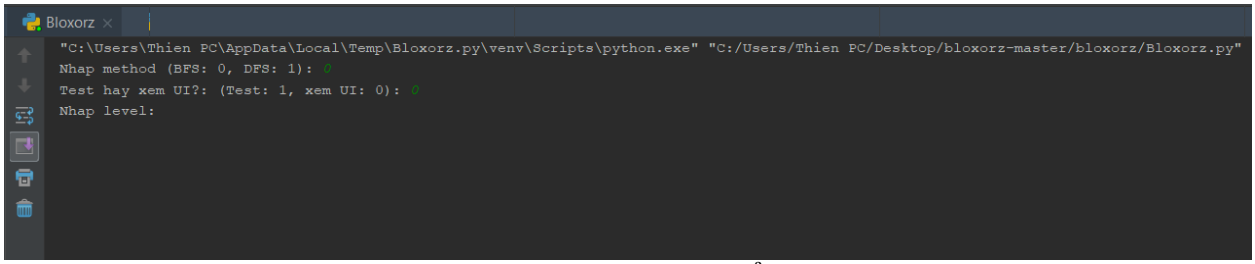
_ Chọn chế độ test hoặc xem hiển thị trực quan cách giải.



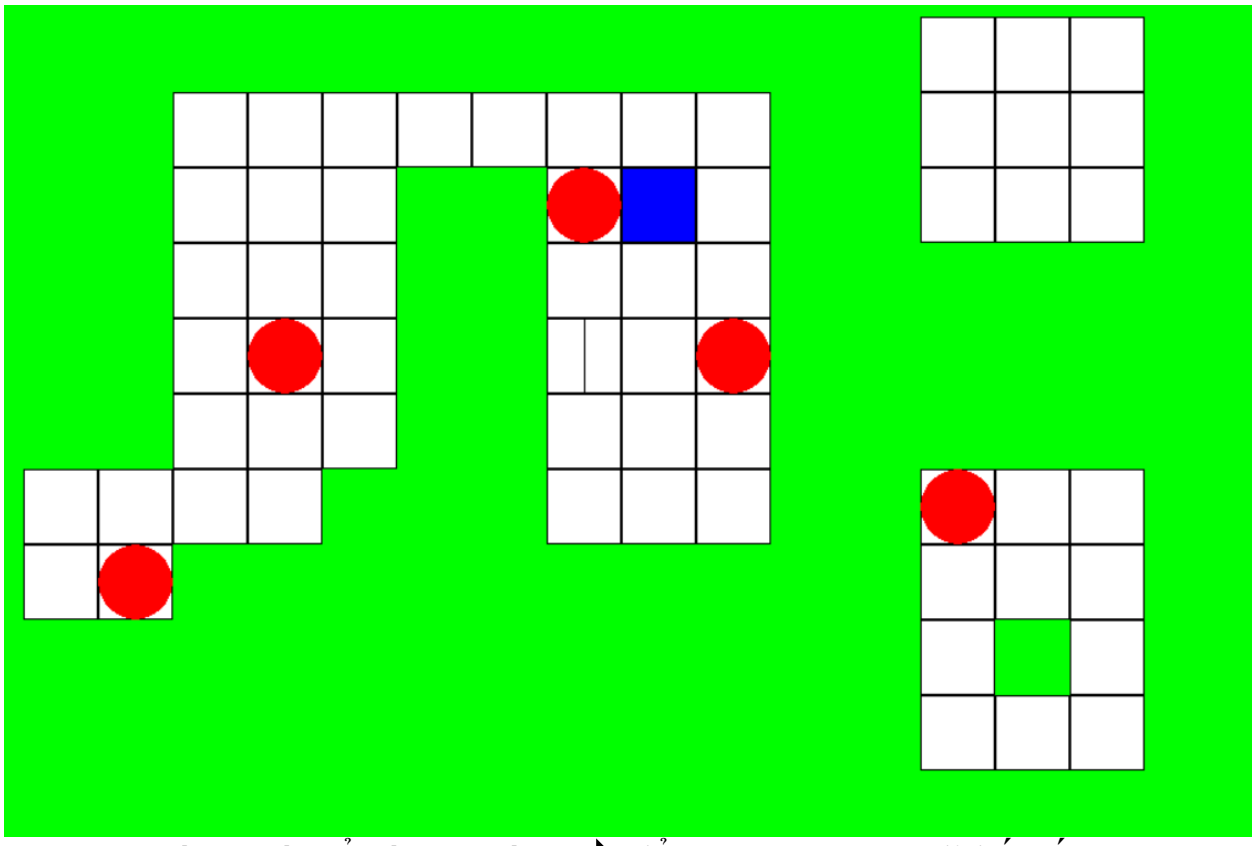
_ Nếu chọn test, chương trình sẽ chạy hết 33 màn và hiển thị true(nếu giải thành công) hoặc false(nếu không giải được) cùng với thời gian thực thi.



_ Nếu chọn UI, người dùng sẽ được yêu cầu chọn màn(level).



_ Sau khi chọn màn, chương trình sẽ cho hiển thị hình ảnh như sau.



_ Người dùng có thể dùng phím ⇨ để xem các nước đi kế tiếp, phím ⇦ để xem các nước trước đó.

VII. ĐÁNH GIÁ DỰ ÁN

1. Ưu điểm:

_ Có demo trực quan bằng hình ảnh sau khi giải quyết xong bài toán

_ Có sự chung sức của cả nhóm

_ Hoàn thành đúng deadline Bài tập lớn.

2. Nhược điểm

_ Dư án của nhóm đã hoàn thành khoảng 80%. Hiện thực thành công hai giải thuật của đề bài là DFS và BFS tuy nhiên chưa thể hiện thực được giải thuật heuristic.

_ Code còn khá dài.

_ 1 số màn tiêu tốn nhiều thời gian thực thi.

TÀI LIỆU THAM KHẢO

_ Stackoverflow

_ Tài liệu học tập của môn học

_ http://www.coolmath-games.com/0-bloxorz