ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ASSIGNMENT 1

ARTIFICIAL INTELLIGENCE

Đề tài: SOKOBAN

LỚP: CS – Khoa học Máy tính

GVHD: TS. Lương Ngọc Hoàng

SINH VIÊN:

Bùi Nguyễn Anh Trung - 20520332

TP. Hồ Chí Minh - 4/2022

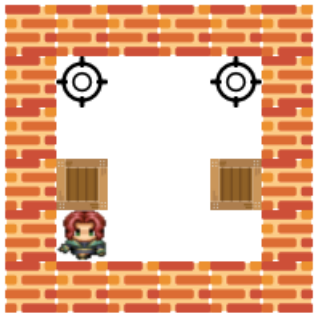
1. GIỚI THIỆU

Trí tuệ nhân tạo đang ngày càng có nhiêu ứng dụng trong thực tế. Và các trò chơi là những môi trường tốt mà chúng ta có thể kiểm thử thuật toán trí tuệ nhân tạo với những ràng buộc đơn giản đơn giản hơn ngoài thực tế

Và trong báo cáo này, ta sẽ xây dựng một tác tử AI để giải quyết một trò chơi kinh điển của Nhật bản là Sokoban dựa trên 3 thuật toán là BFS, DFS và UCS

2. TỔNG QUAN ĐỀ TÀI

1. Mô tả trò chơi và nguyên tắc :



Hình 1: Trò chơi Sokoban

- Sokoban là một trò chơi mà nhân vật trò chơi cố gắng vật chướng (cục đá hoặc thùng gỗ) trong mê cung tới những ô vuông đích.
- **Trạng thái khởi đầu** là không gian trò chơi có nhân vật, vật chướng và ô đích quy định theo màn chơi lúc vừa khởi động.
- **Trạng thái kết thúc** là các vật chướng đã được đặt vào các ô đích thành công
- **Nguyên tắc trò chơi :** Mỗi vật chướng thì có thể đặt ở bất kỳ đích nào, chúng được đẩy bởi nhân vật chơi theo duy nhất 4 hướng lên, xuống, trái, phải được ký hiệu lần lượt là (u,d,r,l) và 4 hành động tương tự nhưng có thêm thao tác đẩy thùng được ký hiệu lần lượt (U,D,R,L)

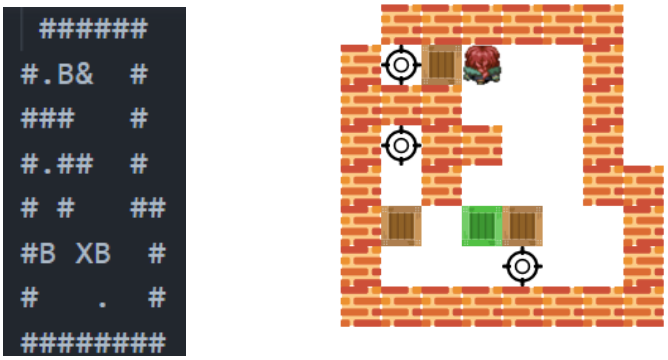
  Các vật chướng không thể xuyên tường, và các vật chướng khác. Khả năng của nhân vật không thể đẩy hai vật chướng cùng một lúc.

  Ô vuông đích cũng được xem như là sàn, nhân vật không bị cản trở khi đi qua ô đích

- Và **mục tiêu** của chúng ta là chuỗi các bước đi để đặt các vật chướng vào các ô đích với số lượng bước đi của nhân vật chơi nhỏ nhất

2. Đặc điểm trò chơi :
   - Trò chơi có ưu điểm các quy tắc rất dễ hiểu
   - Tuy nhiên, trò chơi có nhiều khó khăn :
     + Nếu mô hình chuỗi các bước di chuyển dưới dạng cây đồ thị. Thì số lượng nhánh là rất lớn. Con số này có thể đạt đến 350 trong mức trò chơi tường 20x20
     + Độ dài của lời giải cũng rất lớn. Có trường hợp thì lời giải tối ưu cũng phải cần tới 500 bước
     + Có trường hợp bị deadlock. Khi có vật chướng nằm góc tường (mê cung) nên màn chơi đó không thể giải được.

3. MÔ HÌNH HÓA



Hình 2: Ví dụ mô phỏng trò chơi level 3

1. Mô phỏng trò chơi :

Ban dầu sơ dồ bài toán sẽ được lưu trữ trong một file .txt với các ký tự

+ Ký tự ‘#’ : Tường chắn

+ Ký tự ‘&’ : Mô tả vị trí nhân vật chơi

+ Ký tự ‘B’ : Các vật chướng chưa được chuyển tới đích

+ Ký tự ‘.’ : Vị trí cần đưa box đến

+ Ký tự ‘X’ : Vị trí đích đã có vật chướng

2. **Không gian trò chơi :**
   - Là toàn bộ mê cung
   - Khi được mô hình hóa thành đồ thị, mỗi đỉnh vị trí là một ô vuông trống (không bị chiếm chỗ bởi vật chướng)
   - Hai đỉnh vị trí được nối với nhau nếu giữa chúng có thể di chuyển qua lại
   - Vì người chơi có thể hoàn tác bước đi gần nhất nên đồ thị không có hướng
   - Không gian trò chơi sẽ thay đổi nếu vật chướng bị thay đổi vị trí

3. **Không gian trạng thái :**
   - Trạng thái của mỗi màn là vị trí của nhân vật và vật chướng
   - Chơi trò chơi chính là sự chuyển đổi từ trạng thái này đến trạng thái khác
   - Mỗi mức độ trò chơi sẽ có một sơ đồ trạng thái riêng. Với mỗi đỉnh là là một trạng thái có thể xảy ra của nhân vật và vật chướng
   - Hai đỉnh được nối với nhau nếu có sự thay đổi hợp lí từ trạng thái này đến trạng thái khác
   - Sơ đồ trạng thái này có hướng do không thể đưa trạng thái (gồm nhân vật và vật chướng) quay trở lại như cũ (này phụ thuộc vào cài đặt trò chơi)
   -

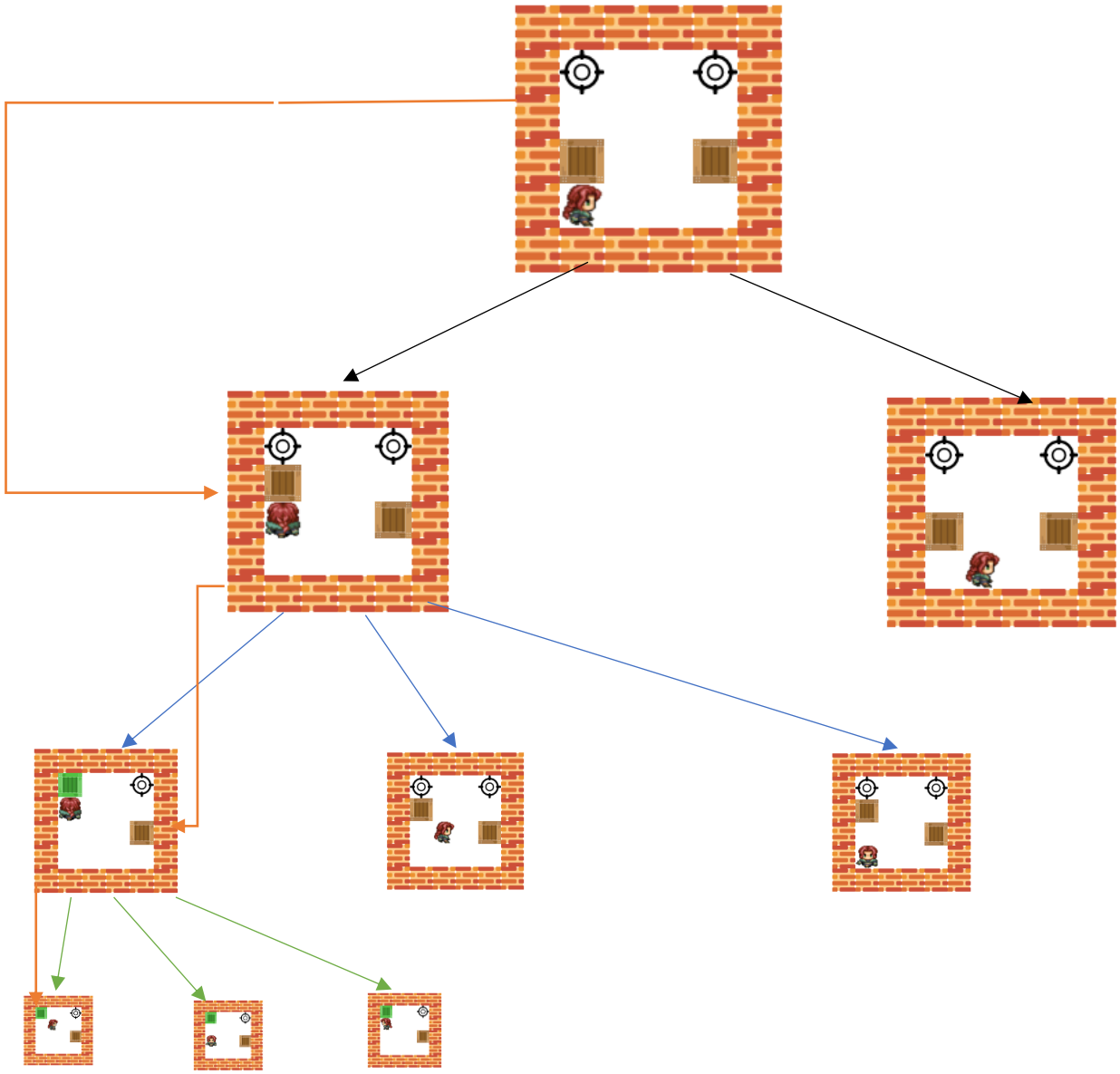
4. HÀM TIẾN TRIỂN

Ba thuật toán được giới thiệu là thuật toán tìm kiếm mù (Uniformed Search)

a. Depth First Search (Tìm kiếm theo chiều rộng)

- Là một giải thuật vét cạn, với trạng thái khởi đầu là trạng thái bắt đầu trò chơi. Sau đó kiểm tra các các trạng thái mới bằng các bước di chuyển hợp lệ (lên, xuống, trái, phải). Ta sẽ lưu lại trạng thái hợp lí đó và trạng thái cha của nó. Nếu node đang xét chính là trạng thái cần tìm, ta sẽ dừng lại và xuất ra chuỗi hành động để được trạng thái này dựa trên cấu trúc lưu cặp trạng thái cha – con trước đó. Nếu không là trạng thái cần tìm, ta sẽ xét xem các node con của nó đã được duyệt qua chưa, nếu chưa thì ta sẽ thêm chúng vào trong ngăn xếp có cấu trúc LIFO (Last in, First out), rồi tiếp tục kiểm tra với nút tiếp theo trong ngăn xếp
- Thuật toán sẽ ưu tiên xét các nút con cho tới nút ở tầng sâu nhất (không còn nút con nào khác) thay vì xét hết toàn bộ đỉnh ở cùng một độ sâu.
- Điều này giúp tránh tình trạng nghẽn cổ chai (dữ liệu chờ được duyệt quá nhiều)

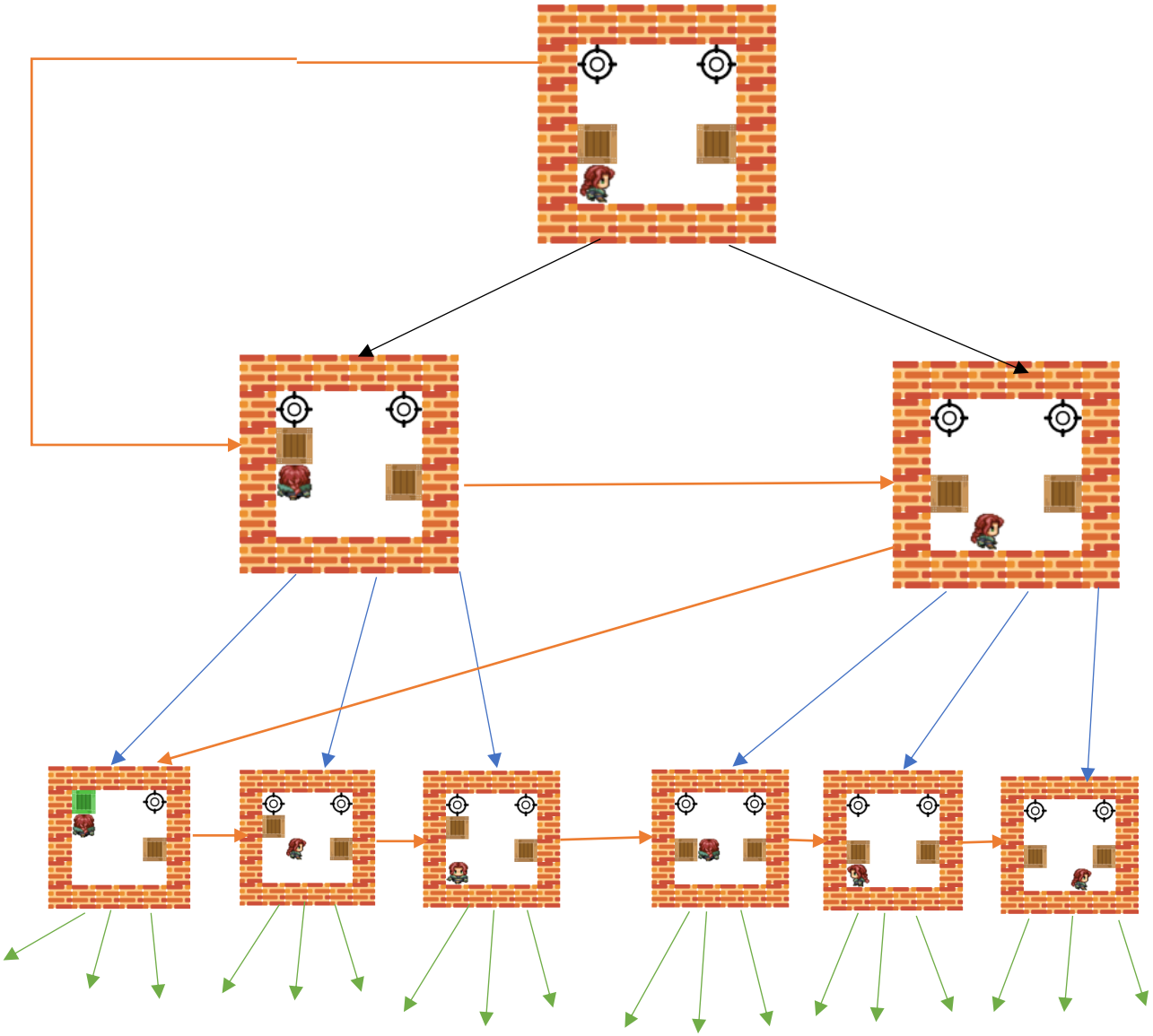
- Khi mà quá trình tìm kiếm không thể xuống sâu thêm nữa, nó sẽ thực hiện quay lui để thực hiện với đỉnh sâu nhất chưa được xét
- Tại mỗi thời điểm thì DFS chỉ lưu trữ những đỉnh nó xét dọc theo một đường nhất định. Điều này giúp DFS chỉ sử dụng bộ nhớ là $O(bd)$
- Điểm hạn chế lớn của DFS là không thể giới hạn được độ dài lời giải trong không gian tìm kiếm vô hạn, kể cả khi lời giải thực sự có độ dài rất ngắn



Hình 3: Mô tả đồ thị trạng thái trò chơi level 2. Nút gốc là trạng thái bắt đầu. Theo mũi tên đen, xanh lần lượt biểu diễn quan hệ các nút trạng thái. Mũi tên cam là mẫu trình tự duyệt theo thuật toán DFS

b. Breadth First Search (Tìm kiếm theo chiều rộng)

- Cũng là thuật toán vét cạn như DFS, chỉ khác cấu trúc lưu node con là hàng đợi với nguyên tắc FIFO (First in, First Out) nghĩa là ưu tiên các node vào trước
- Tìm kiếm trên cây, bắt đầu từ nút gốc, thêm các nút con của chúng vào cuối hàng đợi.
- Sau đó tiếp tục xét tới nút đầu tiên trong hàng đợi
- Thuật toán này sẽ để có đỉnh mà sắp được xét phải được giữ trong tập đỉnh hàng đợi. Và tập đỉnh này còn được gọi là search frontier
- Giả sử một cây có số lượng nhánh cố định ở mỗi nút là b và ở độ sâu d, thì độ lớn của search frontier là $O(b^d)$.
- Trong không gian tìm kiếm lớn, mỗi đỉnh sẽ có nhiều hơn 1 đỉnh con nên sẽ gây ra vấn đề bộ nhớ



Hình 4: Mô tả đồ thị trạng thái trò chơi level 2. Nút gốc là trạng thái bắt đầu. Theo mũi tên đen, xanh lần lượt biểu diễn quan hệ các nút trạng thái. Mũi tên cam là mẫu trình tự duyệt theo thuật toán BFS

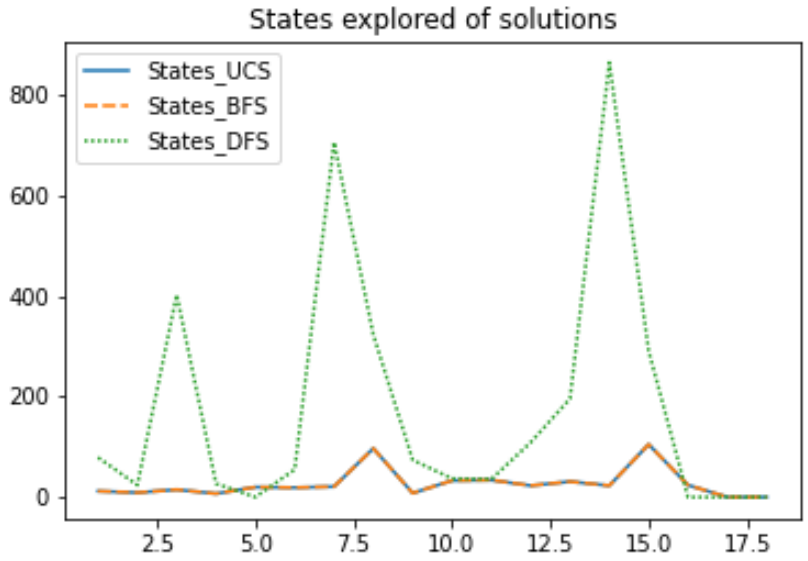
c. Uniform Cost Search (Tìm kiếm với chi phí đồng nhất)

- Thuật toán UCS ta bắt đầu định nghĩa hàm chi phí (cost) quy định lại một hành động di chuyển hợp lí sẽ có chi phí là 1, và hành động đẩy thùng sẽ không được điểm. Do đó, chi phí của hành động đẩy thùng sẽ được lược bỏ
- Ta sẽ sử dụng cấu trúc hàng đợi ưu tiên, Với mục đích là ưu tiên node có chi phí thấp nhất thực hiện trước
- Thuật toán UCS trong trường hợp này có vai trò khá tương đồng với BFS là đều khai thác theo chiều rộng do các chi phí đều đồng nhất bằng 1 nên việc khai thác càng sâu thì chi phí càng lớn (sẽ không được ưu tiên thực hiện). Dù tương đồng với BFS là vậy, UCS vẫn có ưu thế về mặt chi phí và thời gian do có cải tiến lượt bỏ chi phí hành động đẩy thùng

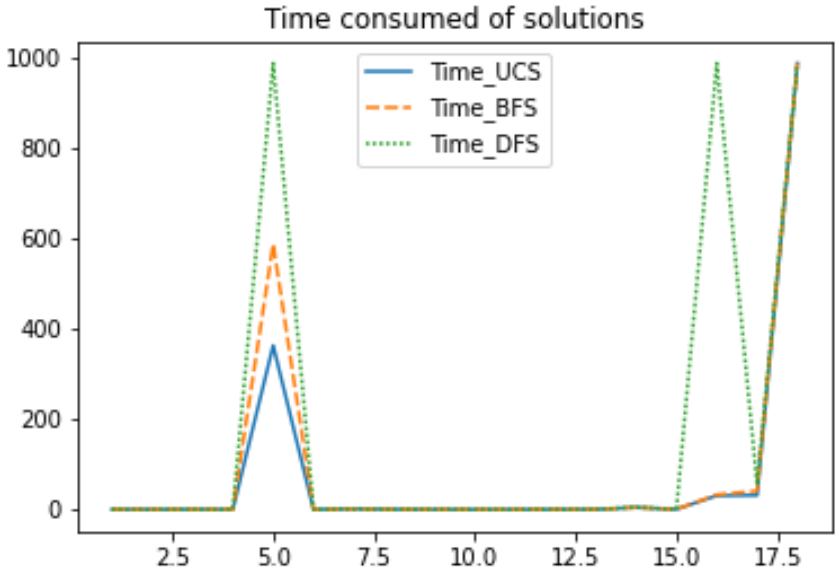
5. THỐNG KÊ KẾT QUẢ THỰC NGHIỆM

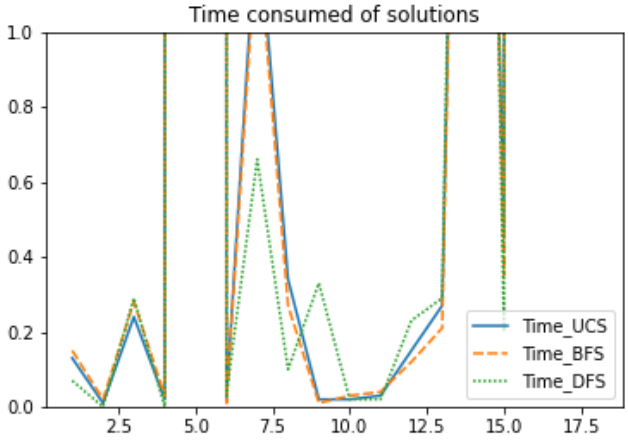
Level	DFS		BFS		UCS	
	Time (s)	States	Time (s)	States	Time (s)	States
1	0.07	79	0.15	12	0.13	12
2	0.00	24	0.02	9	0.01	9
3	0.29	403	0.28	15	0.24	15
4	0.00	27	0.03	7	0.02	7
5	990	0	589.25	20	363.74	20
6	0.03	55	0.01	19	0.02	19
7	0.66	707	1.28	21	1.34	21
8	0.10	323	0.27	97	0.34	97
9	0.33	74	0.01	8	0.02	8
10	0.02	37	0.03	33	0.02	33
11	0.02	36	0.04	34	0.03	34
12	0.23	109	0.12	23	0.15	23
13	0.29	196	0.21	31	0.27D	31
14	4.82	865	4.17	23	4.25	23
15	0.20	291	0.35	105	0.45	105
16	990	0	32.65	34	30.66	34
17	55.54	0	41.15	0	31.54	0
18	990	0	990	0	990	0

Trực quan hóa thực nghiệm



Hình 5.a : Số lượng bước di chuyển của mỗi thuật toán trong các màn





Hình 5.b và 5.c : Thời gian tìm ra lời giải của các thuật toán qua các màn với các khoảng giá trị trục thời gian khác nhau

Dựa vào bảng kết quả trực nghiệm và đồ thị mô hình hóa kết quả, ta có những nhận xét cơ bản sau :

- Thuật toán DFS cho ra kết quả nhanh chóng trong những màn chơi mức độ dễ (theo kết quả hình c) , tuy nhiên độ hiệu quả chưa cao, cần phải thực hiện quá nhiều bước đi để đạt tới trạng thái cuối (hình a). Như trong chương trình, level 7 DFS tìm ra kết quả tận 707 bước đi trong khi chỉ cần 21 bước ở BFS. Điều này là do DFS tìm ra kết quả theo chiều sâu nên các kết quả được tìm ra nằm ở tầng đáy của cây đồ thị trạng thái, khi này càng sâu thì phải càng thực hiện nhiều bước
- Ngược lại, thuật toán BFS và UCS lại cho ra các kết quả gần như giống nhau về số bước đi, nhưng UCS nhỉnh hơn về mặt thời gian. Còn so về mặt thời gian, hai thuật toán này có tốc độ thực hiện lâu DFS ở những màn dễ (nhưng không thực sự đáng kể), bù lại, ở những màn khó, chúng thực sự vượt trội hơn DFS khi có thể tìm ra kết quả và do đó tốc độ cho ra kết quả là tốt hơn. Và giữa BFS với UCS, việc bỏ qua chi phí cho việc đẩy thùng khiến UCS trở nên vượt trội hơn hơn BFS
- Chung quy lại, thuật toán **UCS là tốt nhất**, còn DFS là tệ nhất (theo thời gian thực hiện và độ tốt của lời giải)
- Do thời gian thực hiện ở những màn khó là rất lớn, nên ta cài đặt ngưỡng thời gian tối đa để dừng và kết luận khả năng là 990 giây
- Bênh cạnh so sánh các thuật toán, độ khó các màn chơi còn quyết định độ hiệu quả. Rõ ràng, màn chơi 5,16 thật sự gây trở ngại cho DFS (thời gian chạy quá lớn mà không cho ra kết quả). Level 17 thì cả 3 giải thuật đều trả về kết quả không tìm thấy đường đi. Còn level 18 thì cả 3 giải thuật không thể giải được nên chắc chắn đây là **màn khó nhất**