Tên: Lý Quốc Hưng

MSSV: 21522117

Lớp: CS106.N21

GVHD: TS. Lương Ngọc Hoàng

Assignment 1 - DFS/BFS/UCS for Sokoban

GIỚI THIỆU

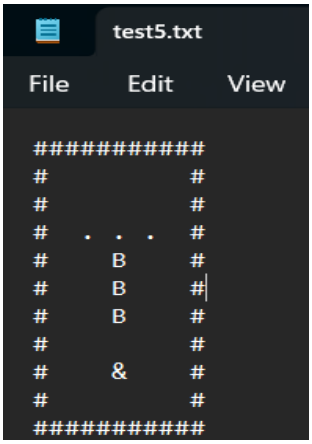
Sokoban là trò chơi dạng câu đố trong đó người chơi phải đẩy một số khối vuông vượt qua chướng ngại vật để đến đích.

Trong báo cáo này, ta sẽ cài đặt thuật toán BFS và UCS để chơi game Sokoban. Qua đó ta có thể rút ra nhận xét về cả 3 thuật toán tìm kiếm không có thông tin là DFS, BFS và UCS.

I. MÔ HÌNH HOÁ SOKOBAN VÀ CÁC TRẠNG THÁI

1. Mô hình hoá Sokoban

- Các level của trò chơi sẽ được lưu trong file .txt với các ký tự:



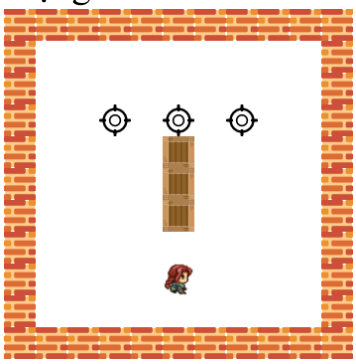
+ Ký tự "#": Tường chắn.
+ Ký tự "&": Nhân vật.
+ Ký tự "B": Thùng.
+ Ký tự ".": Vị trí đích.

Hình 1.1: file txt level 5 của trò chơi sokoban.

- Sử dụng các thuật toán tìm kiếm không có thông tin, Sokoban được mô hình hoá thành một cây gồm các node, mỗi node lưu trữ vị trí của người chơi và các thùng

2. **Trạng thái khởi đầu**

- Trạng thái khởi đầu là vị trí ban đầu của nhân vật và các thùng.



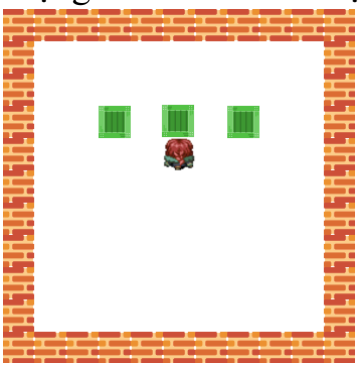
```
startState = (beginPlayer, beginBox)
```

```
((8,5),((6,5),(5,5),(4,5)))
```

Hình 1.2. Trạng thái khởi đầu của level 5.

3. **Trạng thái kết thúc**

- Trạng thái kết thúc là vị trí các thùng được đưa đến đích.



```
node[-1][-1] == ((3, 3), (3, 5), (3, 7))
```

Hình 1.3. Trạng thái kết thúc của level 5.

4. **Không gian trạng thái**

- Là không gian chứa vị trí của nhân vật, các hộp và các vị trí nhân vật có thể di chuyển đến.
- Mỗi level sẽ có không gian trạng thái khác nhau.

5. **Các hành động hợp lệ**

- Các hành động hợp lệ của trò chơi là các hướng đi "u" (lên), "d" (xuống), "l" (trái), "r" (phải) mà người chơi có thể thực hiện. Nếu hành động đó là một hành động "push" (đẩy thùng) thì chữ in hoa tương ứng với hướng đi được sử dụng, còn nếu không thì sử dụng chữ thường.Và sau khi thực hiện 1 hành động người chơi và thùng có thể di chuyển đến vị trí mới sau khi thực hiện hành động đó.

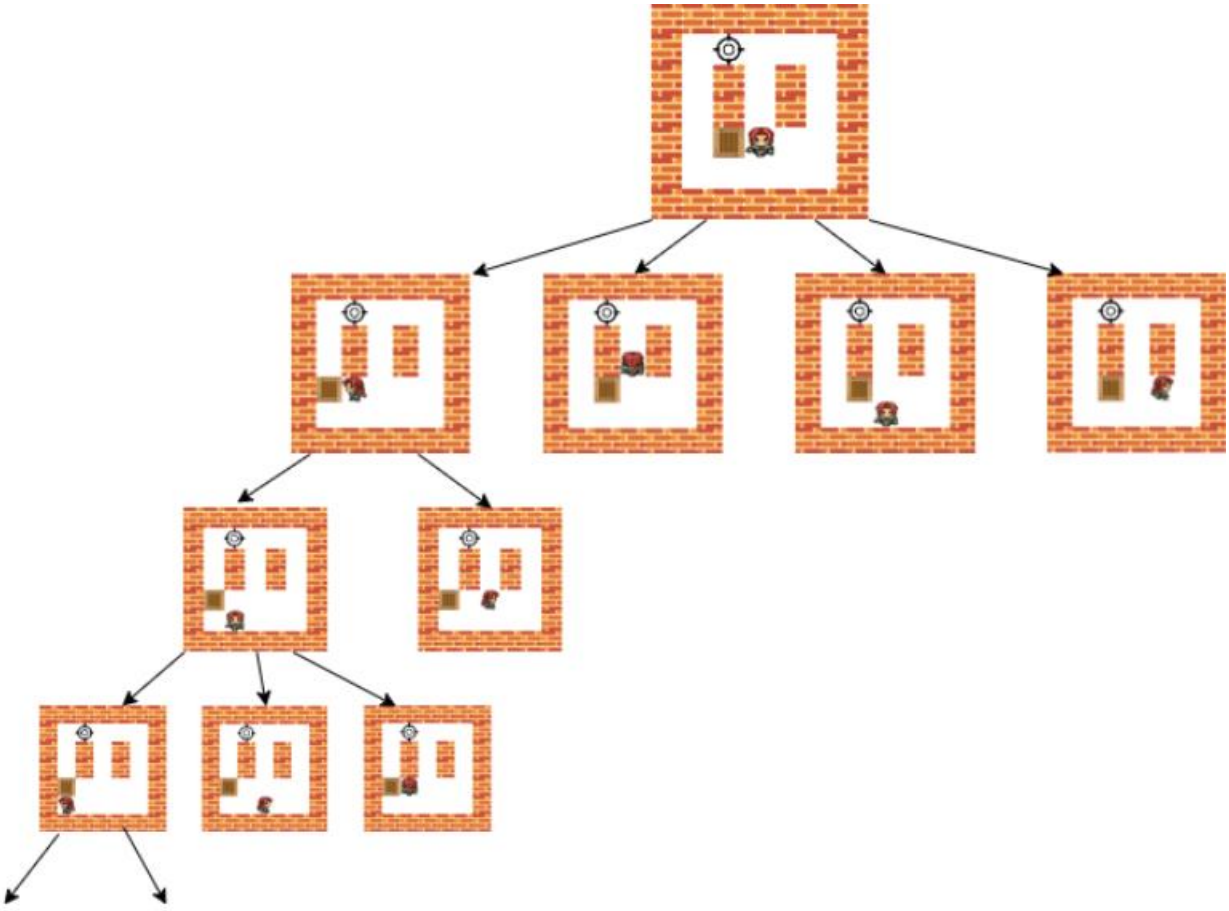
6. **Hàm tiến triển**

- Là một hàm nhận đầu vào là trạng thái hiện tại của trò chơi và một hành động hợp lệ, tính toán các hành động tiếp theo và trả về trạng thái mới của trò chơi sau khi thực hiện hành động đó.

II. CÁC HÀM TIẾN TRIỂN

1. Depth – First Search(Tìm kiếm theo chiều sâu)

- Là một giải thuật vét cạn, DFS là thuật toán tìm kiếm không có thông tin, sử dụng cấu trúc dữ liệu stack với nguyên tắc LIFO với trạng thái khởi đầu là trạng thái bắt đầu trò chơi. Sau đó kiểm tra các hành động hợp lệ tiếp theo.
- Thuật toán ưu tiên kiểm tra các node con cho đến node ở tầng sâu nhất. Khiến cho thuật toán tốn rất nhiều bước để thực hiện. Khi ta chạy Sokoban bằng thuật toán DFS, nó sẽ chạy rất nhiều bước thừa.

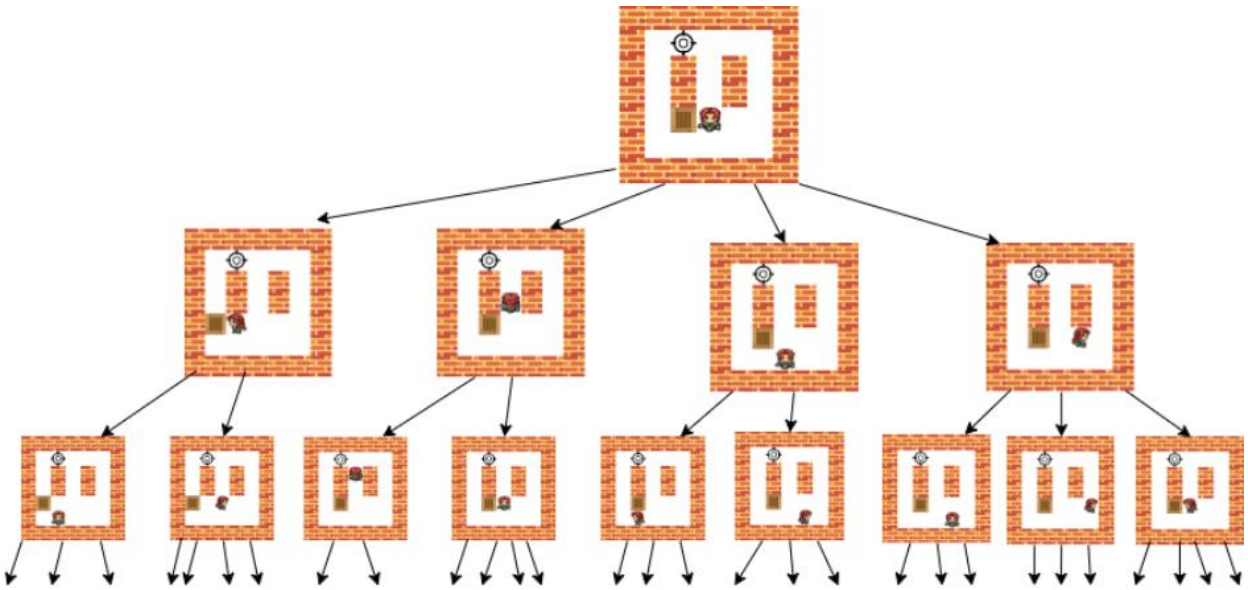


Hình 2.1. Mô tả đồ thị trò chơi level 6 sử dụng thuật toán DFS

2. Breadth – First Search(Tìm kiếm theo chiều rộng)

- Là một giải thuật vét cạn, BFS là thuật toán tìm kiếm không có thông tin, sử dụng cấu trúc dữ liệu queue với nguyên tắc FIFO với trạng thái khởi đầu là trạng thái bắt đầu trò chơi. Sau đó kiểm tra các hành động hợp lệ tiếp theo.
- Thuật toán ưu tiên kiểm tra các node nông nhất, các node được mở rộng theo chiều rộng, các node con được thêm vào cuối hàng đợi. Khiến cho thuật toán

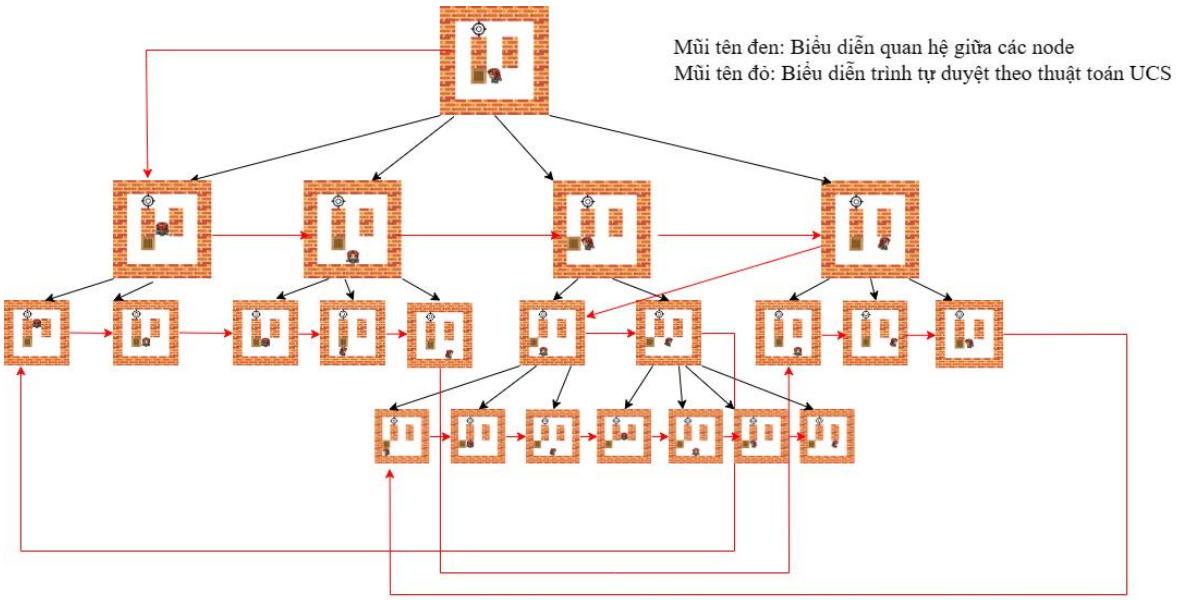
sẽ phải tiêu tốn nhiều bộ nhớ để lưu trữ các node con, tuy nhiên lại cho ra số bước đi khá tối ưu.



Hình 2.2. Mô tả đồ thị trò chơi Sokoban level 6 sử dụng thuật toán BFS.

3. Uniformed – Cost Search(Tìm kiếm tối ưu chi phí)

- UCS là thuật toán tìm kiếm không có thông tin, hoạt động theo cấu trúc dữ liệu hàng đợi ưu tiên(Priority queue).
- Các node sẽ được sắp xếp theo độ ưu tiên. UCS sẽ chọn node có chi phí thấp nhất để mở rộng. Giống với BFS, UCS cho ra số bước đi khá tối ưu.



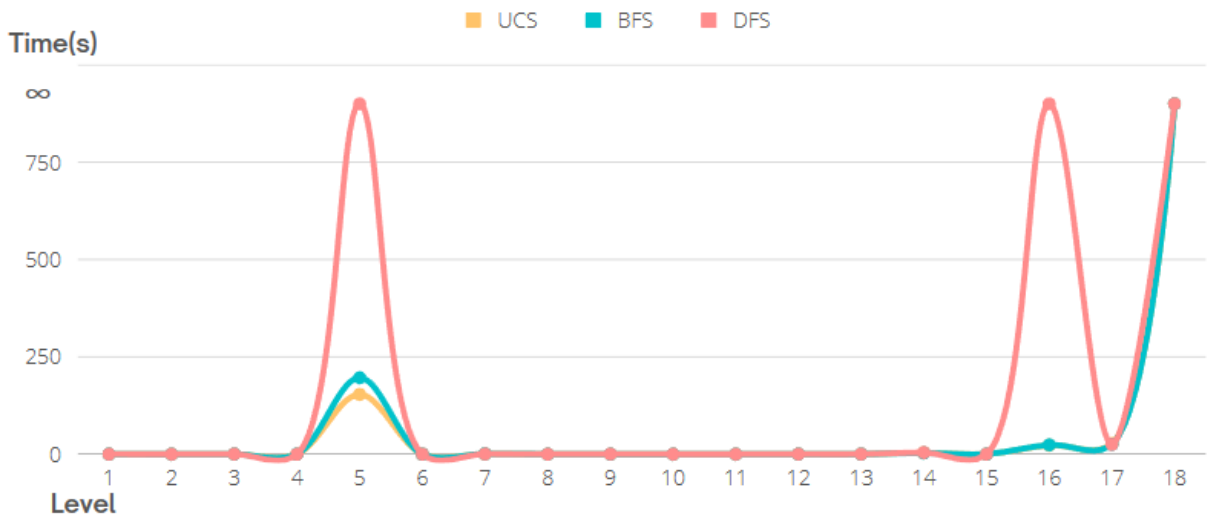
Hình 2.3. Mô tả đồ thị trò chơi Sokoban level 6 sử dụng thuật toán UCS.

III. THỐNG KÊ KẾT QUẢ

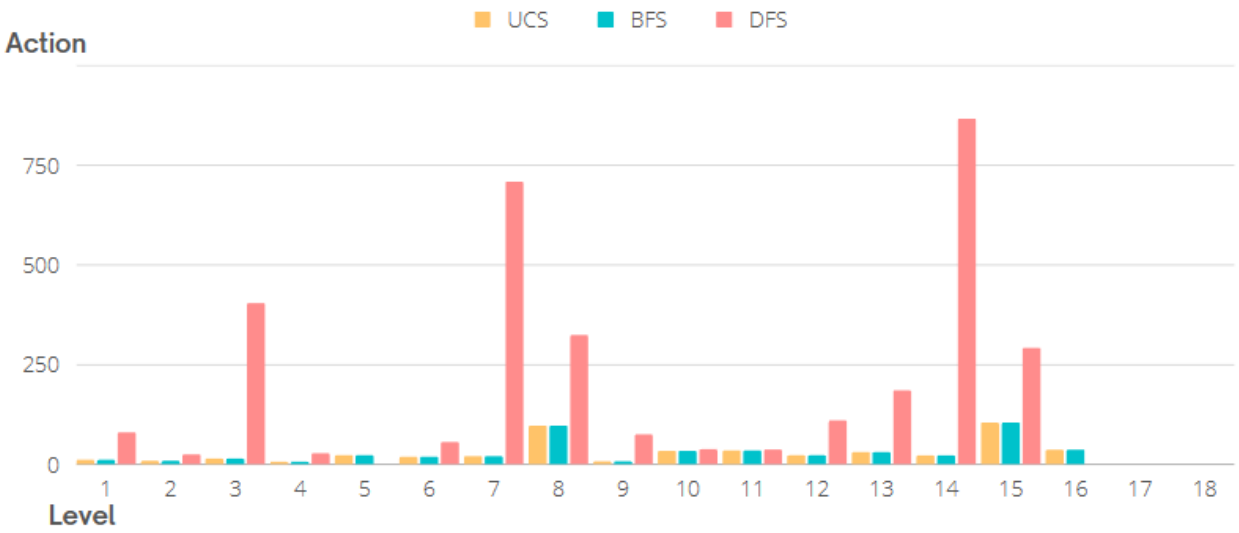
- Sau đây là bảng thống kê thời gian và số bước thực hiện của 3 thuật toán UCS, BFS và DFS. Những trường hợp not found là do thuật toán tìm không ra đường đi.

Level	Time(s)			Actions		
	UCS	BFS	DFS	UCS	BFS	DFS
1	0.07	0.09	0.06	12	12	79
2	0.01	0.01	0	9	9	24
3	0.13	0.2	0.25	15	15	403
4	0.01	0.01	0	7	7	27
5	153.6	196.32	not found	20	20	0
6	0.02	0.02	0.01	19	19	55
7	0.71	0.91	0.56	21	21	707
8	0.23	0.21	0.08	97	97	323
9	0.02	0.01	0.28	8	8	74
10	0.02	0.02	0.02	33	33	37
11	0.02	0.02	0.02	34	34	36
12	0.11	0.1	0.15	23	23	109
13	0.2	0.16	0.21	31	31	185
14	3.6	3.04	4.22	23	23	865
15	0.31	0.29	0.18	105	105	291
16	22.89	24.01	not found	34	34	0
17	27.21	25.18	24.99	0	0	0
18	not found	not found	not found	0	0	0

Hình 3.1. Bảng thống kê thời gian và số bước thực hiện của 3 thuật toán.



Hình 3.2. Biểu đồ thời gian tìm ra lời giải của từng thuật toán.



Hình 3.3. Biểu đồ số bước đi của tường thuật toán.

- Dựa vào bảng kết quả và đồ thị của cả 3 thuật toán, ta có thể thấy:
  + Thời gian chạy của UCS là tối ưu nhất. BFS có thời gian chênh lệnh không đáng kể so với UCS. Đối với những màn dễ, DFS lại cho thời gian nhanh hơn 2 thuật toán còn lại.
  + BFS và UCS có số bước đi tương đồng nhau nhưng UCS có tốc độ nhanh hơn. DFS có số bước nhiều hơn hẳn BFS và UCS do duyệt theo chiều sâu nên đã thực hiện rất nhiều bước đi thừa.
  + Ở level 5 và 16, DFS đã không thể tìm được đường đi do có quá nhiều legalAction(level 5 do chỉ có 4 bức tường xung quanh nên mỗi bước đi sẽ sinh ra rất nhiều legalAction, level 16 lại có khá nhiều thùng).
  + Level 17 do không thể giải được nên cả 3 giải thuật đều trả về kết quả không tìm được đường đi.
  + Level 18 là level khó nhất khi đây là level có thể tìm ra được đường đi nhưng cả 3 thuật toán đều không thể giải ra. Có quá nhiều thùng và các tường chắn.
- Kết luận:

  + UCS là thuật toán **tối ưu nhất** trong 3 thuật toán.

  + Level 18 là level **khó nhất.**