**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**NGUYỄN KHẮC THÁI – 20521888**

# TRÍ TUỆ NHÂN TẠO-CS106.M21

## Báo cáo: Assignment 1

## BFS/DFS/UCS for Sokoban

**GIẢNG VIÊN HƯỚNG DẪN**

**LƯƠNG NGỌC HOÀNG**

**TP. HỒ CHÍ MINH, 2022**

MỤC LỤC

# I. GIỚI THIỆU CHUNG

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.

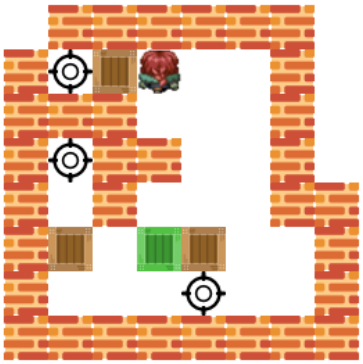
Trò chơi có dạng bảng ô vuông. Có một số khối vuông được đẩy đến đích (số ô đích đúng bằng số khối vuông). Chỉ có thể đẩy từng khối vuông một, và không thể kéo, cũng như không thể đẩy một dãy hai hay nhiều khối.

Khối vuông bị *dính tường* nếu như nó bị đẩy sát vào tường mà hai bên tường đều là góc. Vì không thể kéo khối lại được nên coi như khối này bị mất, nó không thể được đưa đến đích trừ phi đích nằm đúng trên cạnh tường đó. Dính tường là một trường hợp cần tránh khi chơi

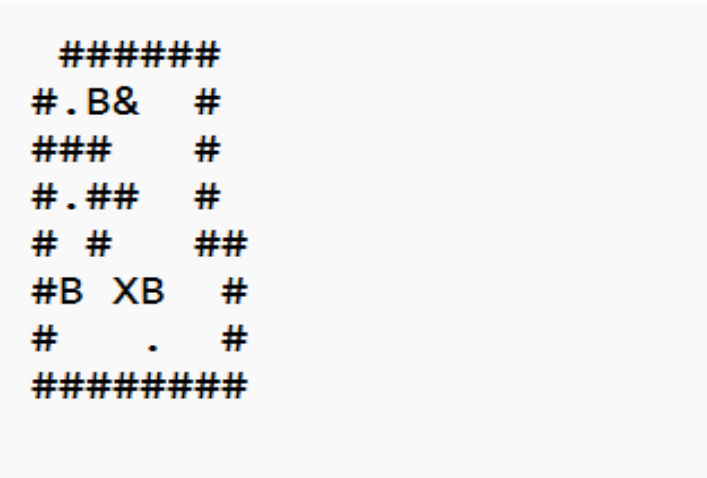
# II. MÔ HÌNH HÓA GAME SOKOBAN

Để mô hình hóa trò chơi, game đã có một kí hiệu tiêu chuẩn để phân biệt các đối tượng trong trò chơi một cách độc lập và rõ ràng. Các giá trị đầu vào cho mô hình hóa của trò chơi cụ thể gồm các kí hiệu sau: “ ”, “#”, “&”, “B”, “.”, “X”. Mỗi đối tượng có một kí hiệu đặc biệt của trò chơi gán cho nó, cụ thể như sạu:

- “ ” là không gian trống, người điều khiên sokoban có thể đi vào những vị trí này
- “#” là bức tường, người điều kiển sokoban không thể đi qua vị trí này
- “&” là sokoban, người chơi sẽ điều khiển sokoban này để tìm ra lời giải cho từng màng chơi
- “B” là  một chiếc hộp, người chơi sẽ điều khiển sokoban để đưa những chiếc họp này tới vị trí đích(mục tiêu)
- “.” là vị trí đích(mục tiêu)
- “X” khi ô vuông năm trên mục tiêu



*Hình 1 Game sokoban*




*Hình 2 Mô hình hóa sokoban*

## III. TRẠNG THÁI CỦA GAME

- Trạng thái bắt đầu là trạng thái do nhà phát triển trò chơi gốc đưa ra,bao gồm một người chơi tại một vị trí x, y nhất định trên lưới và một số vị trí nhất định được đánh dấu là thùng (hoặc hộp) và cửa hàng mục tiêu.
- Người chơi có thể đẩy nhiều nhất một ô vào một ô trống không phải là tường hoặc ô khác. Người chơi không được phép kéo tường hoặc nhiều hơn 1 thùng hàng cùng một lúc. Game có số lượng thùng và kho mục tiêu bằng nhau và người chơi sẽ thành công khi tất cả các thùng đều ở vị trí lưu trữ mục tiêu. Người chơi sẽ thất bại nếu một thùng bị nhốt trong một góc hoặc với một thùng khác với các vị trí lưu trữ (hoặc vị trí) không có sử dụng.
- Trạng thái kết thúc là trạng thái khi tất cả các thùng được chuyển đến vị trí lưu trữ thích hợp, cụ thể là các thùng hàng đều nằm ở vị trí mục tiêu.
- Hàm tiến triển (Successor Function): Mô tả lại những hành động có thể xảy ra giữa các trạng thái. Trong bài toán này, hàm tiến triển sẽ thực hiện kiểm tra và cập nhật lại vị trí của người chơi và cái box của từng trạng thái sau các hành động.

## IV. THUẬT TOÁN

Game sokoban có nước đi là không thể thay đổi, vì vậy việc di chuyển không tốt có thể khiến bạn thua trận. Với cơ chế trò chơi, một số yếu tố được tính đến để quyết định tối ưu thuật toán:

- Cần phải trả lại một loạt các nước đi một cách nhanh chóng vì trò chơi cũng được chơi theo thời gian thực. Do đó, một thuật toán có thể tạo ra các bước di chuyển trong một khoảng thời gian ngắn và trả về một bước đi dẫn đến chiến thắng là điều mong muốn.
- Ràng buộc được thêm vào trò chơi. Ràng buộc trong trò chơi này đề cập đến các bức tường hiện diện bên trong ranh giới bên ngoài và hạn chế đường đi của người chơi.
- Độ sâu mà tìm kiếm được xây dựng. Vì vấn đề tìm kiếm này có thể dẫn đến vô số trạng thái và đôi khi có thể dẫn đến độ sâu tìm kiếm vô hạn, một mối quan tâm là phải biết thuật toán có thể khai thác bao nhiêu trong không gian tìm kiếm.

Với những vấn đề trên, chúng ta quyết định đánh giá các thuật toán và so sánh chúng dựa trên độ phức tạp về không gian/thời gian tìm ra đường đi của chúng. Điểm mạnh của các thuật toán trong không gian này nằm ở khả năng nhanh chóng xác định chuỗi di chuyển mang lại kết quả tương đối. Chúng ta sẽ triển khai ba thuật toán cụ thể là:

- Tìm kiếm theo chiều sâu (DFS)
- Tìm kiếm theo chiều rộng (BFS)
- Tìm kiếm chi phí thấp nhất (UCS)

### 1) Tìm kiếm theo chiều sâu(DFS)

Thuật toán tìm kiếm theo chiều sâu là một trong các thuật toán tìm kiếm mù(không có thông tin). Thuật toán này sẽ bắt đầu từ nút gốc và tiến hành đến nút xa nhất rồi quay lại. Thuật toán sẽ đảm bảo tìm ra lời giải nếu như có lời giải và là số tần của cây(hay độ sâu của cây)là hữu hạn. Nếu như thuật toán tìm ra lời giải thì lời giải đó không phải là lời giải tối ưu nhất. Thuật toán sẽ dừng nếu như tìm ra lời giải hoặc là đã duyệt hết cây nhưng không tìm ra lời giải.Độ phức tạp về không gian trong trường hợp sấu nhất là O(bm), độ phức tạp về thời gian trong trường hợp sấu nhất là O($b^m$)(với m là số tầng của cây, b là số cây con tại mỗi node).

### 2) Tìm kiếm theo chiều rộng(BFS)

Thuật toán tìm kiếm theo chiều rông là một trong các thuật toán tìm kiếm mù(không có thông tin). Thuật toán này sẽ bắt đầu từ nút gốc và tiến hành đến nút ở tầng nông nhất rồi tiếp tục tìm kiếm ở tầng tiếp theo. Thuật toán sẽ đảm bảo tìm ra lời giải nếu như có lời giải và lời giải được tìm ra sẽ nằm ở tầng nông nhất. Nếu như thuật toán tìm ra lời giải thì lời giải đó không phải là lời giải tối ưu nhất,nó sẽ là thuật toán có số bước đi ít nhất, nhưng nếu như mỗi bước đi sẽ có một chi phí như nhau thì thuật toán sẽ tìm ra lời giải tối ưu nhất. Thuật toán sẽ dừng nếu như tìm ra lời giải hoặc là đã duyệt hết cây nhưng không tìm ra lời giải. Độ phức tạp

về không gian là O($b^s$), độ phức tạp về thời gian là O($b^s$)(với s là tầng mà tại đó chứa lời giản, b là số cây con tại mỗi node).

### 3) Tìm kiếm với chi phí thấp nhất (UCS)

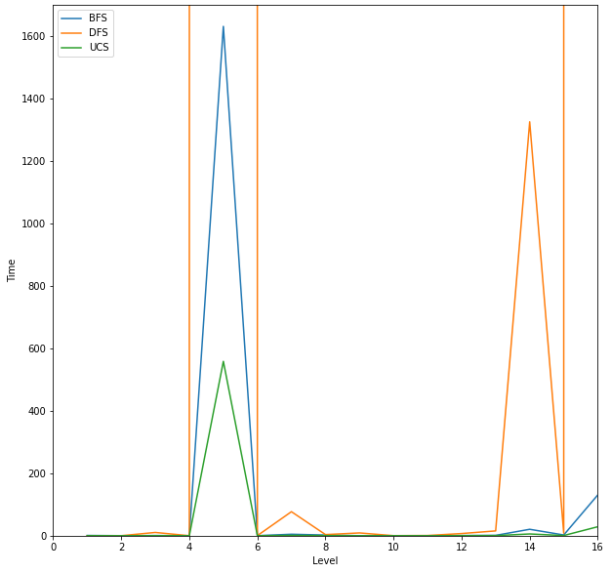
Thuật toán tìm kiếm với chi phí thấp nhất(UCS) là một trong các thuật toán tìm kiếm mù(không có thông tin). Thuật toán sẽ tìm kiếm khám phá các nút nhánh mà có chi phi thấp nhất hoặc nhiều nhất. Điều này dẫn đến việc chọn một phần tử từ hằng đợi, mà ưu tiên cái phần tử từ vị trí gốc đến nút đó sao cho có chi phí nhỏ nhất(hoặc lớn nhất). Thuật toán sẽ đảm bảo tìm ra lời giải, và lời giải đó là lời tối ưu nhất của bài toán. Thuật toán sẽ dừng nếu như tìm ra lời giải hoặc là đã duyệt hết cây nhưng không tìm ra lời giải. Độ phức tạp về không gian là O($b^{C\backslash\varepsilon}$), độ phức tạp về thời gian là O($b^{C\backslash\varepsilon),}$)(với C là chi phí của lời giải, ε là chi phí thấp nhất của đường đi , b là số cây con tại mỗi node).

## V. KẾT QUẢ

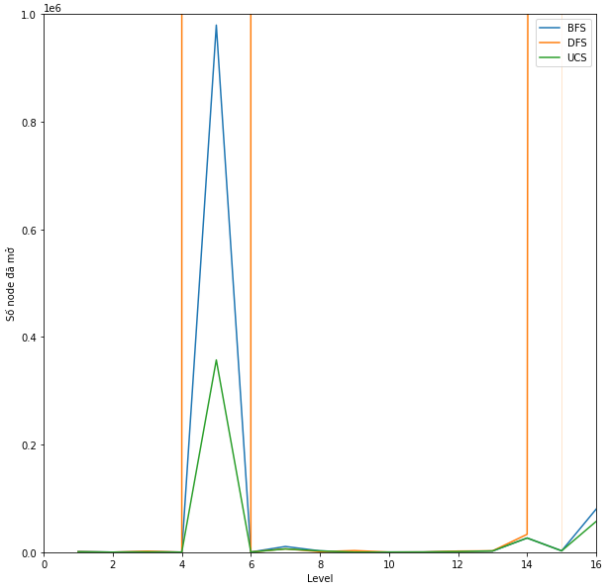
Hiệu suất của tất cả các thuật toán được tính toán dựa trên các số liệu như sau: thời gian tìm ra đường đi, tổng số nút con đã mở và độ dài của các giải pháp được thể hiện bằng đồ họa trong Hình 3, Hình 4 và Hình 5. Các thuật toán đã được thực hiện để dừng lại khi nó vượt quá thời gian tối đa(1h), Time-out. Điều này là cần thiết để ngăn chặn các thuật toán tìm kiếm trong không gian tìm kiếm vô hạn. Sau khi đã giải được các ta được bảng kết quả như sau:

| Thuật toán / Level | DFS | | | BFS | | | UCS | | |
|---|---|---|---|---|---|---|---|---|---|
| | Số nút đã mở | Thời gian | Độ dài đường đi | Số nút đã mở | Thời gian | Độ dài đường đi | Số nút đã mở | Thời gian | Độ dài đường đi |
| 1 | 818 | 0.02 | 24 | 1197 | 0.79 | 12 | 720 | 0.13 | 12 |
| 2 | 30 | 0.03 | 24 | 82 | 0.03 | 9 | 64 | 0.01 | 9 |
| 3 | 1874 | 10.42 | 403 | 1109 | 0.54 | 15 | 509 | 0.19 | 15 |
| 4 | 27 | 0.02 | 27 | 108 | 0.03 | 7 | 55 | 0.01 | 7 |
| 5 | Time out | Time out | Time out | 979435 | 1631.55 | 20 | 357203 | 558.5 | 20 |
| 6 | 236 | 0.32 | 55 | 263 | 0.08 | 19 | 250 | 0.03 | 19 |
| 7 | 6181 | 76.93 | 706 | 10568 | 4.67 | 21 | 6046 | 1.08 | 21 |
| 8 | 884 | 3.6 | 323 | 2498 | 2.05 | 97 | 2383 | 0.45 | 97 |
| 9 | 3006 | 8.87 | 74 | 80 | 0.04 | 8 | 74 | 0.02 | 8 |
| 10 | 172 | 0.17 | 37 | 220 | 0.09 | 33 | 218 | 0.04 | 33 |
| 11 | 297 | 0.63 | 36 | 305 | 0.14 | 34 | 296 | 0.05 | 34 |
| 12 | 2029 | 6.93 | 109 | 1265 | 0.53 | 23 | 1225 | 0.22 | 23 |
| 13 | 2485 | 15.41 | 185 | 2032 | 1.07 | 31 | 2342 | 0.4 | 31 |

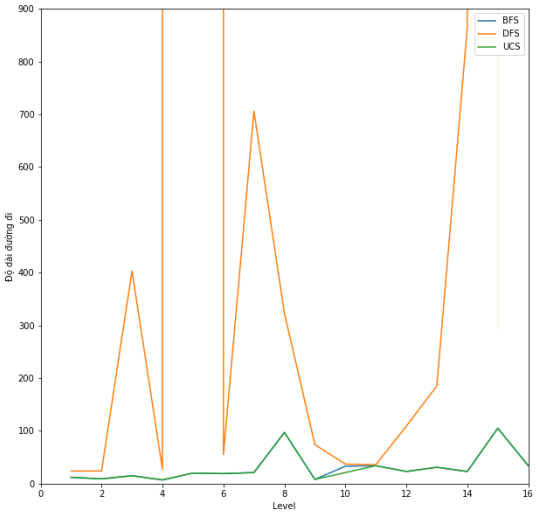
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 14 | 33209 | 1326 | 865 | 26386 | 20.6 | 23 | 26352 | 5.56 | 23 |
| 15 | 1630 | 8.11 | 291 | 2526 | 2.32 | 105 | 2505 | 0.54 | 105 |
| 16 | Time out | Time out | Time out | 80141 | 130.57 | 34 | 57275 | 28.75 | 34 |




*Hình 3 Thời gian tìm ra đường đi theo các màng*




*Hình 4 Số node đã mở theo các màng*




*Hình 5 Độ dài đường đi theo các màng*

Khi giải các màng có thể quan sát thấy rằng DFS vượt quá thời gian tối đa (ở màng 5 và 16) nguyên nhân là do vì hai màng này có không gian dịch chuyển rộng nên chiều sâu của cây mà nó xây dựng quá lớn, còn UCS và BFS có thể vượt qua 2 màng này mà không bị Time-out. Trong các thuật toán thì dựa vào hình vẽ trên thì ta thấy rằng thuật toán UCS là cho kết quả tốt nhất, DFS là cho kết quả không tốt nhất trong 3 thuật toán này.

Về phần số nút đã mở theo các màng thì ta có thể thấy rằng DFS là có số nút mở lớn nhất ở một số màng(màng 5,7,14,16), điều này có thể giải thích rằng là chiều sâu của cây xây dựng quá lớn, và việc đâm vào nào những nút không có lời giải nên dẫn đến việc số node đã mở của DFS là nhiều. UCS và BFS thì ở hai màng 5 và 16, UCS số nút mở ít hơn, trong khi UCS phải nhìn về mọi phía để biết đi đường nào có chi phí nhỏ nhất. Điều này có thể giải thích rằng là do chiều rộng và chiều sâu của hai màng này khá lớn, nên dẫn đến viêc duyệt theo chiều sâu và rộng ở hai màn này không tốt bằng việc theo đường đi có chi phí nhỏ nhất. Ngoài ra ở các màng khác thì giữa BFS, DFS và UCS thì số node đã mở ở các màng không chênh lệch nhau nhiều.

Đối với phần thời gian tìm ra đường đi thì ta có thể thấy rằng DFS là cho kết quả không tốt ở một số màng(màng 5,7,14,16), nhất điều này có thể giải thích là do số nút đã mở của DFS nhiều. UCS và BFS thì UCS là vượt trội hơn BFS ở các màng 5 và 16, trong khi UCS thì nó phải cần chi phí tính toán cũng như nó cần phải nhìn về mọi hướng. Điều này có thể giải thích rằng là do số nút để mở của UCS ít hơn của BFS. Ngoài ra thời gian tìm ra đường đi ở các màng khác thì giữa các thuật toán BFS, DFS và UCS là không chênh lêch nhiều.

Về phần độ dài đường đi theo các màng thì ta có thể thấy DFS là cho kết quả tệ nhất ở gần như tất cả các màng, điều này có thể giải thích rằng là do DFS duyệt theo chiều sâu nên viêc đụng tới nút mà tại đó là nút mục tiêu là sâu nên đường đi dẫn đến nút mục tiêu sẽ dài. Còn UCS và BFS thì lại có độ dài đường đi tương đối như nhau ở tất cả các màng trừ màng(10), giải thích cho điều này là vì chi phi đường đi ở các nút gần như là bằng nhau.

## VI. KẾT LUẬN

Chúng ta đã chạy thử nghiệm một số thuật toán tìm kiếm và quan sát thấy rằng hầu hết các thuật toán đều có những động tác tương tự trong việc chơi trò chơi nhưng hiệu suất của chúng khác nhau về thời gian và độ dài đường đi và các trạng thái được khám phá. Với những màng có không gian di chuyển nhỏ thì các thuật toán đáp ứng tốt và có độ chênh lêch với nhau về :thời gian, số trạng thái đã mở, độ dài đường đi là không lớn. Nhưng với những màng có kích thước không gian di chuyển lớn hơn(màng 5,7,14,16), tất cả các thuật toán bị hạn chế bởi thời gian cũng như tăng số trạng thái mở thì thuật toán UCS là có hiệu quả tốt nhất.