***Tên: Lương Phạm Bảo***

***MSSV: 19521242***

**BÁO CÁO ASSIGNMENT 1**

1. **Astar Search (A\*):**

* Mô hình hóa Sokoban:
  + Trong A\*, Sokoban được mô hình hóa thành một cây gồm các node.
  + Mỗi node bao gồm tập các vị trí player và vị trí box từ node parent đến node hiện tại.(là hai yếu tố có vị trí thay đổi)
  + Mỗi node sẽ mở ra các node con của nó là các bước đi hợp lệ mà node hiện tại có thể sinh ra
  + Một heap được tạo ra để lưu trữ các node đang nằm trong frontier và một heap khác để lưu trữ các hành động (actions) của các node nằm trong frontier. Ta sẽ chọn một node để mở dựa trên môt hàm f(n)=g(n)+h(n) (dựa trên 2 hàm g và h,với g là hàm để tính khoảng cách từ node hiện tại đến beginState và g để ước lượng khoảng cách từ node hiện tại đến goal, heuristic).Dựa trên giá trị hàm f đó ta theo thứ tự tăng dần ta sẽ mở từng node với thêm chúng vào frontier và action
  + Thuật toán sẽ dừng lại khi một node bị lấy ra khỏi frontier và tìm được một node con là trạng trái kết thúc hoặc frontier rỗng (đã duyệt hết tất cả đường đi có thể nhưng không thể tìm được đường đi).
* Trạng thái khởi đầu: vị trí player, vị trí các box lúc bắt đầu game(beginState).
* Trạng thái kết thúc: các box đã ở đúng vị trí target(goalState).
* Không gian trạng thái: được mở rộng theo chiều dọc, liên tục lấy các node cuối cùng trong frontier (node có độ cao lớn nhất) để mở rộng.
* Các hành động hợp lệ: là hành động không lặp lại một trạng thái đã được ghi nhận và vị trí player không trùng vị trí box, vị trí wall hoặc vị trí box không được trùng vị trí box khác, vị trí wall.
* Hàm tiến triển: node có f(n)nhỏ nhất (node ở đầu heap) được ưu tiên mở trước.

1. **Thiết kế heuristic:**

* Dễ thấy ta cần tìm môt hàm h(n) <=Cost( chi phí thật từ vị trí node hiện tại đến mục tiêu) và h(n) không nên quá thấp=> ta cần tìm hmax(n) ( vì dễ thấy h=0 thì bài toán trở thành Dijkstra, có thời gian thực thi lâu,tương tự với các hàm h(n) nhỏ) và ta cũng có thể chọn heuristic xấp xỉ =Cost(chấp nhập việc lớn hơn Cost) sẽ giúp chúng ta tìm ra đường đi nhanh hơn
* **Mahatan**

Với một bản đồ lưới vuông thì việc sử dụng distance Mahatan là một cách thông thường nhất

function heuristic (node):

* dx = abs (node.x - goal.x)
* dy = abs (node.y - goal.y)
* return D \* (dx + dy)

Để thõa tính admissibility ta cần chọn D là khoảng cách giữa cost thấp nhất giữa các ô vuông trên bản đồ(D=1).Với bản đồ ít vật cản,khi di chuyển gần đến ô goal thì sẽ tăng g giảm h(=> g+h ít thay đổi)=>

* Eculid

Với khoảng cách Eculid thì chi phí hàm g sẽ khác nhìu với hàm h. vì khoảng cách Eculid nhỏ hơn Mahatan nên bài toán vẫn sẽ tìm ra đường đi có chi phí tốt nhất tuy nhiên lại tốn khá nhìu thời gian

function heuristic (node):

* dx = abs (node.x - goal.x)
* dy = abs (node.y - goal.y)
* return D \* sqrt(dx\*dx + dy\*dy)

1. **GreedySearch:**

* Mô hình hóa Sokoban:
  + Trong Greedy Search, Sokoban được mô hình hóa thành một cây gồm các node.
  + Mỗi node bao gồm tập các vị trí player và vị trí box từ node parent đến node hiện tại.
  + Mỗi node sẽ có các node con của nó là các bước đi hợp lệ mà node hiện tại có thể sinh ra.
  + Một heap(là môt hàng đợi ưu tiên) được tạo ra để lưu trữ các node đang nằm trong frontier và một heap khác để lưu trữ các hành động (actions) của các node nằm trong frontier. Các node được đưa vào heap dựa trên giá trị heuristc( là một hàm để ước lượng khoảng cách nhỏ nhất từ vị trí node cuối trong hàng đợi đến vị trí mục tiêu, một node có cost càng thấp thì sẽ được mở ra càng sớm). Lần lượt lấy phần tử đầu heap ra để tìm các node con hợp lệ của nó và các hành động để đến các node con sau đó push vào frontier và actions.
  + Thuật toán sẽ dừng lại khi một node bị lấy ra khỏi frontier và tìm được một node con là trạng trái kết thúc hoặc frontier rỗng (đã duyệt hết các đường đi có thể nhưng không thể tìm được đường đi).
* Trạng thái khởi đầu: vị trí player, vị trí các box lúc bắt đầu game.
* Trạng thái kết thúc: các box đã ở đúng vị trí target.
* Không gian trạng thái: Được mở rộng dần dựa theo node có cost thấp nhất (node nằm ở đầu frontier).
* Các hành động hợp lệ: là hành động không lặp lại một trạng thái đã được ghi nhận và vị trí player không trùng vị trí box, vị trí wall hoặc vị trí box không được trùng vị trí box khác, vị trí wall.
* Hàm tiến triển: node có cost thấp hơn (node ở đầu heap) sẽ được ưu tiên mở rộng trước.
* Nhận xét:
  + Với thuật toán Greedy nếu không gian tìm kiếm mở( việc mở một node có chi phí thấp) sẽ cho thuật toán hoạt động cực kì hiệu quả( vì bài toán sẽ tìm dựa trên Cost từ node hiện tại đến goal mà bản đồ mở thì thuật toán sẽ tìm ra đường đi ngắn nhất và thời gian rất nhanh (hên như DFS tìm thấy đường ngay lần đầu)
  + Thuật toán A\* là thuật toán tìm đường mang tính hiệu quả và ổn định nhất( chỉ cần ta tìm được heuristic càng tốt thì kết quả sẽ càng nhanh và tốt nhất). Thuật toán A\* sẽ tìm ra đường tốt nhất nếu thõa tính  admissibility/consistency. Thuật toán sẽ tốn thời gian đầu để ước lượng và sẽ chạy khi tìm ra đường đi (tùy vào việc heuristic có gần với cost hay không và có nhỏ hơn cost hay không)
* So với 2 thuật toán thì đường đi tìm được bởi UCS(nến tồn tại đường đi) sẽ là đường đi tốt nhất về chi phí ,tuy nhiên thuật toán trên sẽ không quan tâm về số độ sâu của cây tìm kiếm mà chỉ quan tâm chi phí đi qua điểm vì vậy nó thật sự không quá hiệu quả với bài toán trên (việc tìm đường đi tối ưu bài toán trên là tương đối khó(vì ta không biết chi phí lẫn độ sâu tối thiểu cần đi để tìm ra được đường đi vì không gian để tìm kiếm quá rộng cũng như bài toán sẽ duyệt hết các đường đi(trong khi BFS có thể dừng lại khi tìm được đường đi
* Với các thuật toán Uniform Search ở trên thì đường đều có ưu và khuyết điểm riêng để giải quyết hiệu quả bài toán trên rằng việc tìm dường đi ở các bản đồ phức tạp hay rộng lớn đều không hiệu quả( tốn nhìu thời gian và không gian tính toán vì đa phần đường đi đều không dẫn tới đích cũng như có nhìu hộp( việc chọn ưu tiên hộp để di chuyển là rất khó khăn và nếu các điểm đặt hộp quá gần thì sẽ dẫn đến việc tìm ra đường đi dễ bị lâm vào bế tắc. Vì vậy thuật toán Astar là thuật toán cân bằng giữa tốc độ và việc tìm ra đường đi tốt nhất ,còn UCS chỉ hiệu quả khi chi phí mở một node là lớn và đường đi cần tìm cũng là chi phí nhỏ nhất còn greedy thì lại hiệu quả với bản đồ mở(việc mở node chi phí thấp)