**Báo cáo Assignment 02 - Greedy and A\* for Sokoban**

•••

Họ và tên: Lý Hoàng Thuận

MSSV: 19522315

* **Phần 1 : Thuật toán Heuristic:**
  + **Xây dựng hàm Heuristic cho game sokoban:**

-Trong game sokoban ta đã biết trước cần phải đưa các hộp (box) về vị trí cho trước. Việc xây dựng hàm heuristic được áp dụng trong game này chính bằng cách tính toán nhằm ước lượng khoảng cách để đưa các box về các vị trí đích.

-Việc tính toán ước lượng khoảng cách có thể thực hiện được bằng các thuật toán tình khoảng cách có thể kể đến như Manhattan, Euclidean, etc.

-Ở mỗi hành động thực hiện ta sẽ tính toán khoảng cách từ trạng thái đó đến trạng thái kết thúc.

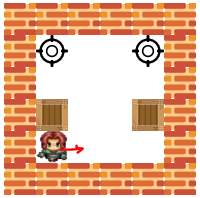
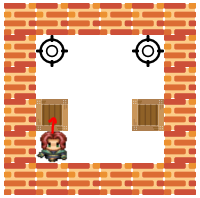
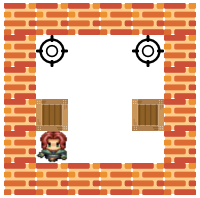
* **Phần 2: Thuật toán tìm kiếm greedy(tham lam):**
  + **Tổng quan:**

-Đúng như tên gọi, thuật toán tìm kiếm tham lam tìm đường đi dẫn đến kết quả mà nó cho rằng ngắn nhất (dù có thể không tối ưu)

* + **Áp dụng vào bài toán sokoban:**

-Với mỗi trạng thái của player hàm heuristic tính toán khoảng cách từ trạng thái đó đến goal và đưa vào 1 hàng đợi có tính ưu tiên(priority queue), tức là hàng đợi sẽ ưu tiên các node được heuristic tính toán nhỏ hơn xếp vào đầu và ngược lại.

Ví dụ: vị trí của player ban đầu ở (4,1) các goal lần lượt là (2,1) và (2,4) hàm heuristic tính toán theo khoảng cách Euclidean thì sẽ nhận được kết quả lần lượt là 2 và sqrt(12) đối với trạng thái kế tiếp(U) kết quả nhận được là 1 và sqrt(10). Với state(R) là sqrt(5) và 2



U

R

-Các trạng thái tiếp theo được mở rộng dựa trên kết quả mà hàm Heuristic tính toán cho ra kết quả nhỏ hơn kết quả của trạng thái trước đó dù không rõ ràng nước đi cụ thể cho game.

* **Phần 3 : Thuật toán tìm kiếm A\*:**
  + **Tổng quan thuật toán:**

-Nếu Uniform-cost có chi phí đồng nhất theo chi phí đường đi hoặc chi phí lùi lại g(n), thì Greedy sẽ tính toán theo khoảng cách mục tiêu, hoặc chi phí chuyển tiếp h(n)

-Thuật toán A\* sắp xếp bởi trật tự từ tổng của g(n) và h(n).

* + **Áp dụng vào bài toán Sokoban:**

-Gần như giống với Greedy, nhưng A\* là tổng của g(n) và h(n) nên chi phí mỗi hành động sẽ được thêm vào.

-Nếu như Greedy luôn tìm kiếm theo shortest(ngắn nhất) nhưng không đảm bảo tính tối ưu thì A\* khắc phục được nhược điểm này do có thêm chi phí cho hành động để đến được mục tiêu.

* **Phần 4: So sánh 2 thuật toán với các thuật toán trước đó:**

-So sánh ở level 3 các thuật toán thu được kết quả như sau:

|  |  |  |
| --- | --- | --- |
| **Algorithms** | **Step** | **Runtime** |
| **DFS** | 4582. | 0.39 second. |
| **BFS** | 2137. | 0.33 second. |
| **UCS** | 792. | 0.16 second. |
| **Greedy** | 21870. | 2.92 second. |
| **A\*** | 6398. | 0.97 second. |