TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
THUẬT TOÁN TÌM KIẾM

MSSV	HỌ VÀ TÊN
19120686	TRẦN VĂN TÌNH
19120698	MAI DƯƠNG NGUYÊN TRƯỜNG

HỒ CHÍ MINH, ngày 11 tháng 11 năm 2021

I/ Không có điểm thưởng:

Bản đồ 1: 16X42

Depth-First_Search

Breadth First Search

Heuristic: Manhattan distance

greedy_best_first_search

A_Star_Search

Phân tích: Với bản đồ đơn giản trên ta có thể phân tích được các sự khác biệt thuật toán sau:

UnInformed Search: Depth_First_Search, Breadth First Search,

Completeness: Cả hai thuật toán luôn tìm được lối ra (nếu lối ra đó tồn tại) vì thuật toán duyệt hết đồ thị

Time: Do sự khác biệt về thuật toán nên DFS chạy nhanh hơn trong bản đồ này vì chỉ đi theo một hướng, trong khi đó BFS phải mở rộng về mọi hướng tốn thời gian hơn nhiều

Optimality: Đối với bản đồ không điểm thưởng thì BFS có đường đi tối ưu nhất vì tìm được điểm thoát gần nhất trong khi đó DFS tìm hướng đi xa hơn vì chỉ dẫn sai hướng (badly-guide DFS)

Informed Search: GBFS, A*

Completeness: Với bản đồ trên cả hai đều tìm thấy lối ra

Optimality: Đối với GBFS vì ‘”tham lam” tiến lại gần đích đến nên phải đi quãng đường xa hơn (phải lên rồi mới xuống)

Trong khi đó A* Với hàm heuristic: Manhattan hợp lí vẫn đảm bảo tính tối ưu quảng đường đi được là ngắn nhất giống BFS

Kết luận: Xét về quảng đường đi ngắn nhất trong ma trận không có điểm thưởng thì hai thuật toán BFS và A* đều cho ta kết quả tối ưu(đường đi ngắn nhất) tuy nhiên về thời gian thực hiện thì A* thông thường sẽ nhanh hơn (BFS lâu hơn vì phải đi tất cả các hướng).

Bản đồ 2: 31X63

Depth-First_Search

Breadth First Search

Heuristic: Manhattan distance

greedy_best_first_search

A_Star_Search

Phân tích:

UnInformed Search: Depth_First_Search, Breadth First Search

Optimality: Về tính tối ưu rõ ràng đường đi của BFS cho ta số bước đi ngắn hơn so với DFS

Completeness: Cả hai đều tìm được lối ra nếu lối ra đó tồn tại

Informed Search: GBFS, A*

Optimality: GBFS: Đi quãng đường lâu hơn so với A* do tính chất "greedy" của mình nên GBFS không đảm bảo tính tối ưu

Kết luận từ bản đồ trên: Trong bản đồ không có điểm thưởng thì hai thuật toán A* và BFS đảm bảo tính tối ưu (tìm đường đi ngắn nhất) tuy nhiên đa phần A* sẽ chạy nhanh hơn khi có hàm f(n) phù hợp sẽ định hướng hướng di chuyển hợp lí

Thuật toán DFS đảm bảo luôn tìm ra đường đi nếu chúng tồn tại

GBFS cho ta thấy chiến lược "tham lam" của mình đôi lúc cũng tốt hơn so với các hướng tiếp cận khác

Bản đồ 3: 16X42

Depth-First_Search

Breadth First Search

Heuristic: Euleur distance

greedy_best_first_search

A_Star_Search

Phân tích:

<u>UnInformed Search</u>: DFS, BFS

-Optimality: Thuật toán BFS cho ta đường đi tối ưu rõ ràng hơn DFS. Với bản đồ không điểm thưởng BFS cho ta đường đi ngắn nhất.

-Time: Thời gian thực hiện của DFS sẽ nhanh hơn BFS

<u>Informed Search</u>: GBFS, A*

-Optimality: Thuật toán Greedy tham lam tiến lại gần đích nên quảng đường (theo hình vẽ) đã có một đoạn dư ra khi lại gần đích làm thuật toán không đảm bảo tính tối ưu tìm ra lối đi ngắn nhất

Trong khi đó A* vẫn đảm bảo tính tối ưu và cho ra đường thẳng ngắn nhất giống như BFS.

Kết luận với bản đồ trên:

- Với các bài toán đòi hỏi sự tối ưu có lẽ ta không nên chọn thuật toán DFS

Bản đồ 4: 17X41

Depth-First_Search

Breadth First Search

greedy_best_first_search

A_Star_Search

Heuristic: euclidean distance

Phân tích: Trong trường hợp này:

UnInformed Search: DFS, BFS

- Optimality: DFS không đảm bảo tính tối ưu so với BFS vẫn tìm ra đường đi ngắn nhất
- Time: DFS sẽ nhanh hơn do điểm start và end xa nhau tránh việc làm cho BFS tỏa ra nhiều hướng mất thời gian hơn

Informed Search: GBFS, A*

- Optimality: Rõ ràng A_Star cho ta thấy đường đi ngắn hơn

  Và Greedy vì chọn hàm heuristic là Euclidean distance tham lam theo hàm Heuristic trên làm cho Greedy đi một khoảng xa hơn

  Kiểm tra thử hàm: Manhattan Distance thì cho kết quả khả quan hơn và ngắn hơn rất nhiều

Kết luận từ bản đồ trên:

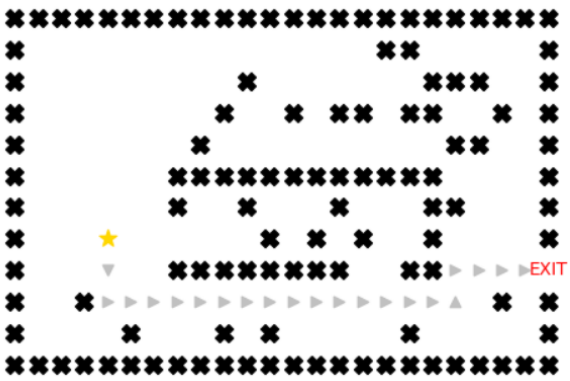
Nên chọn và thử nhiều hàm heuristic khác nhau cho các kết quả khác nhau "rất đáng kể" trong bài toán Greedy

Bản đồ 5: 10X19

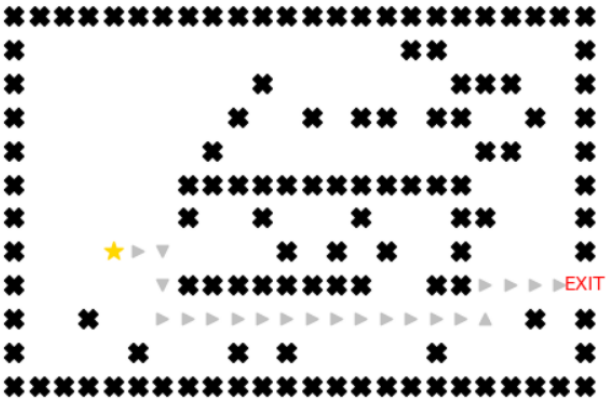
Heuristic: Manhattan distance

Depth-First_Search

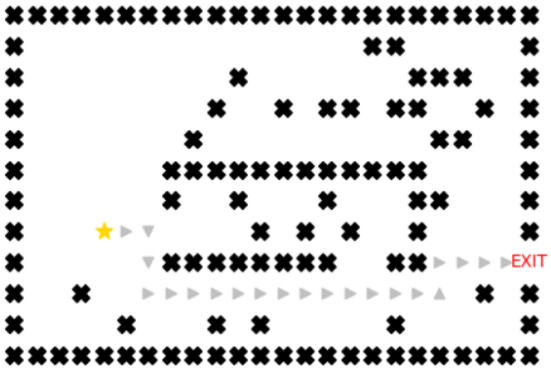
Breadth First Search



greedy_best_first_search



A_Star_Search



Phân tích: Sử dụng bản đồ khác biệt ít các thuật toán

Đây là bản đồ đơn giản: Không thể hiện rõ sự khác biệt thuật toán nhưng từ bản đồ ta cũng có nhận xét rằng:

DFS: Có vẽ vẫn cho ta đi đường đi tốn kém hơn

Còn thuật toán Greedy_BFS: Thấy rằng thỉnh thoảng "tham lam" vẫn cho ta kết quả đáng mong đợi

Trong trường hợp này cả 3 thuật toán BFS, Greedy, A* cho ta đường đi tối ưu

Time: Rõ ràng thời gian chạy của Greedy sẽ nhanh hơn trong trường hợp này

Kết luận từ bản đồ : Trong một số trường hợp thì "tham lam" vẫn cho ta một lời giải "khá" tốt và chấp nhận được.

II/ Có điểm thưởng:

Đề xuất giải thuật: **Dùng thuật toán đệ qui**

Gọi hàm tìm đường đi có chi phí thấp nhất lấy tham số start LeastCost(start)

Từ điểm start sẽ có hai trường hợp:

- Đi thẳng tới đích (end): Ta sẽ tính trường hợp này và lưu lại các biến (cost,path)
- Đi qua điểm thưởng đầu tiên: Chi phí thấp nhất trong trường hợp này là A*(start->u) + LeastCost(u->end)

Với ý tưởng này ta có thể xây dựng thuật toán đệ qui để giải quyết bài toán một cách tối ưu (vì xét hết tất cả các trường hợp)

Hạn chế: Thuật toán có thể tìm lời giải tối ưu tuy nhiên thời gian chạy là hạn chế khi số điểm thưởng tăng lên. Cụ thể khi (n =10) thuật toán chạy chưa cho ra kết quả.

Đề xuất Cải tiến: cụ thể TH (n=10)

Ta có thể đánh đổi giữa "bộ nhớ" và tốc độ:

Cụ thể lưu các quảng đường từ điểm thưởng u-> điểm thưởng v

(Sẽ cần lưu 10C2 = 45 đường thẳng từ các điểm thưởng khác nhau)

Sau đó tránh việc gọi lại hàm A*(start->u) nhiều lần

Với trường hợp n lớn:

Đề xuất chiến lượt:

Bước 1: Xem xét các điểm thưởng phía sau lưng:

(là nữa mặt phẳng còn lại của ma trận không chứa điểm bắt đầu đi và lối thoát)

Tham lam ăn các điểm thưởng có giá trị thưởng cao hơn chi phí cả đi lẫn về

Điều kiện: 2*distance(start->bonus) + value(bonus) <0

Sau đó quay lại điểm xuất phát

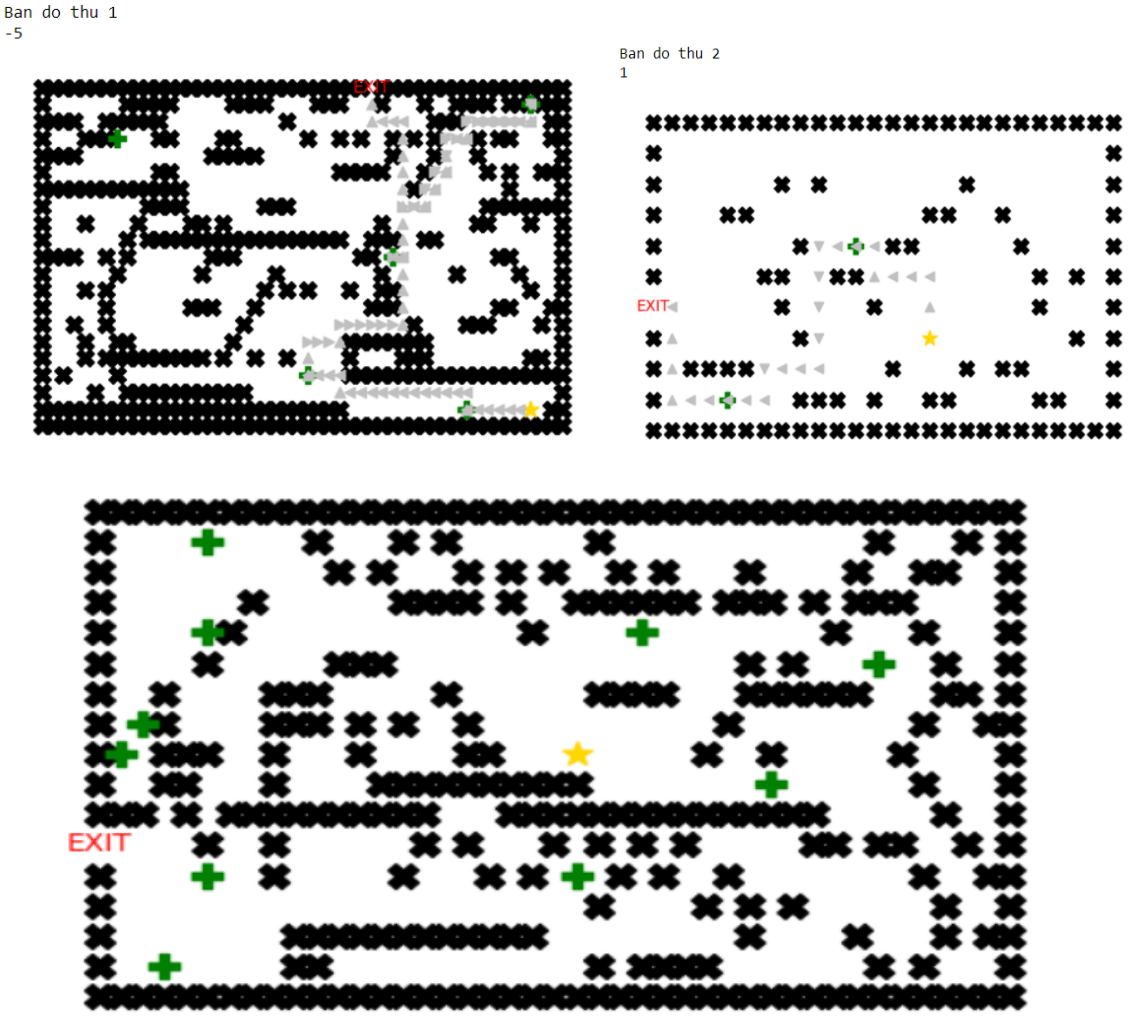
Bước 2: Xem xét các điểm thưởng phía trước mặt

Sẽ ăn các điểm thưởng có chi phí thấp, và ở gần với lối ra mê cung

Ta sẽ ăn các điểm thỏa mãn hàm F(n) sau nhỏ nhất

F(n) = A*(start->điểm thưởng)+A*(điểm thưởng->Exit)+Value(điểm thưởng)

-Nếu có điểm thưởng nào bị bỏ lại phía sau lưng quay lại bước 1



Phân tích 2 bản đồ: Gồm lần lượt 5, 2 điểm thưởng thuật toán cho ta các đường đi tối ưu nhất theo thuật toán đệ quy đề xuất phía trên.

Bản đồ 10 điểm thưởng: Chưa thể tìm ra đường thẳng tối ưu vì rất tốn thời gian để chạy. Sử dụng chiến lược tham lam như đề xuất ở trên

Reference:

1/https://www.geeksforgeeks.org/best-first-search-informed-search/
2/https://en.wikipedia.org/wiki/Best-first_search
3/https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_chi%E1%BB%81u_s%C3%A2u