

Đồ án 1: Thuật toán tìm kiếm

-----oOo-----

Họ và tên: Nguyễn Phan Quốc Bảo

Mã số sinh viên: 19120456

Email: 19120456@student.hcmus.edu.vn

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

Mục lục

1.	Định nghĩa bài toán.....	3
1.1.	Trạng thái bắt đầu:.....	3
1.2.	Các hành động.....	3
1.3.	Mô hình di chuyển	3
1.4.	Trạng thái đích	3
2.	Cài đặt các thuật toán:	3
2.1.	Thuật toán tìm kiếm DFS (không có thông tin)	3
2.2.	Thuật toán tìm kiếm BFS (không có thông tin).....	3
2.3.	Thuật toán tìm kiếm tham lam – Greedy Best First Search (có thông tin).....	4
2.4.	Thuật toán tìm kiếm A* (có thông tin).....	4
2.5.	Thuật toán tìm đường đi với chi phí ngắn nhất trên bảng đồ có điểm thưởng.....	4
3.	Lựa chọn hàm heuristic cho các chiến lược tìm kiếm có thông tin.....	5
3.1.	Heuristic là khoảng cách euclid đến đích.....	5
3.2.	Heuristic là khoảng cách manhattan đến đích	5
4.	Các chiến lược tìm kiếm không điểm thưởng trên các loại bản đồ khác nhau.....	6
4.1.	Bản đồ maze_map0.txt - bản đồ có 2 hướng đi để đến đích.	6
4.2.	Bản đồ maze_map1.txt - bản đồ có tường chắn dài ở gần đích.	8
4.3.	Bản đồ maze_map2.txt - bản đồ lớn, có nhiều ngạnh nhô lên trên tường.	10
4.4.	Bản đồ maze_map3.txt - bản đồ có 2 đường đi, 1 ngắn, 1 dài.....	12
4.5.	Bản đồ maze_map4.txt - bản đồ có dạng xoắn ốc.	14
4.6.	Tổng kết	16
5.	Chiến lược tìm kiếm trên bảng đồ có điểm thưởng để có chi phí thấp nhất.....	16
5.1.	Chiến lược.....	16
5.2.	Chạy thuật toán với các bảng đồ có 2, 5, 10 điểm thưởng	17
6.	Trường hợp bản đồ có cổng dịch chuyển.....	18
6.1.	Chiến lược.....	18
6.2.	Thực hiện với bản đồ có điểm dịch chuyển các thuật toán DFS, BFS.....	18
7.	Cách sử dụng mã nguồn.....	18

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

1. Định nghĩa bài toán

1.1. Trạng thái bắt đầu:

Trạng thái bắt đầu tìm kiếm là tọa độ vị trí start của mê cung.

1.2. Các hành động

Với 1 điểm bất kỳ trong mê cung (không phải là tường (có giá trị 'x')), hành động có thể thực hiện là di chuyển lên, xuống, trái, phải 1 đơn vị trong mê cung nếu vị trí đó không phải là tường.

1.3. Mô hình di chuyển

Sau khi chọn 1 trong các hành động có thể thực hiện, vị trí của tác nhân hiện tại là vị trí ô nằm ở hướng di chuyển tương ứng.

Không gian trạng thái là đồ thị với đỉnh là các vị trí của tác nhân trong ma trận và cạnh là các hành động lên, xuống, trái, phải.

Một đường đi trong không gian trạng thái là một chuỗi các vị trí của tác nhân được kết nối bằng chuỗi các hành động di chuyển.

1.4. Trạng thái đích

Bài toán đạt trạng thái đích khi tác nhân đến được vị trí exit.

2. Cài đặt các thuật toán:

2.1. Thuật toán tìm kiếm DFS (không có thông tin)

1. Xây dựng cây tìm kiếm với nút gốc là điểm xuất phát.

2. Sử dụng 1 ngăn xếp để lưu danh sách các nút lá của cây, thêm nút gốc vào hàng đợi.

3. Thực hiện lặp, với mỗi vòng lặp, ta xét node ở đỉnh của ngăn xếp ra và đánh dấu đỉnh này đã thăm. Xét các vị trí trên, dưới, trái, phải của node đang xét, nếu vị trí này không phải tường và chưa được thăm thì thực hiện thêm vào ngăn xếp, lưu giá trị node hiện tại là node cha của vị trí này. Lặp lại bước ba cho đến khi ngăn xếp rỗng hoặc điểm đích đã thăm.

4. Từ node đích, lần ngược về node gốc theo mảng cha (parent), ta tìm được đường đi theo DFS.

2.2. Thuật toán tìm kiếm BFS (không có thông tin)

1. Xây dựng cây tìm kiếm với nút gốc là điểm xuất phát.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

2. Sử dụng 1 hàng đợi để lưu danh sách các nút lá của cây, thêm nút gốc vào hàng đợi.

(Do mỗi cạnh của cây tìm kiếm đều có trọng số bằng 1, vì vậy, những node nào ở vị trí càng xa node gốc càng có chi phí thực hiện đường đi cao. Để giảm thời gian thực hiện, ta đánh dấu những node được thêm vào hàng đợi là đã duyệt, không cần duyệt lại ở những vị trí xa hơn trên cây)

3. Thực hiện lặp, với mỗi vòng lặp, ta lấy node ở đầu hàng đợi làm node đang xét. Với các vị trí trên, dưới, trái, phải của node đang xét, nếu vị trí này không phải là tường và chưa xét, ta thêm vào hàng đợi, đánh dấu là đã xét và lưu node cha của node. Thực hiện lặp đến khi hàng đợi rỗng hoặc điểm đích đã thăm.

4. Từ node đích, lần ngược về node gốc theo mảng cha (parent), ta tìm được đường đi theo BFS.

2.3. Thuật toán tìm kiếm tham lam – Greedy Best First Search (có thông tin)

1. Tính giá trị heuristic cho tất cả điểm không phải tường trên mê cung.

2. Sử dụng hàng đợi ưu tiên (giá trị heuristic càng nhỏ, độ ưu tiên càng cao) để lưu danh sách các nút lá của cây, thêm nút gốc vào hàng đợi.

3. Thực hiện lặp, Với mỗi vòng lặp, ta lấy node có độ ưu tiên cao nhất đầu tiên của hàng đợi làm node đang xét, đánh dấu node này đã thăm. Với các vị trí trên, dưới, trái, phải của node đang xét, nếu vị trí này không phải là tường và chưa xét, ta thêm vào hàng đợi ưu tiên và lưu node cha của node. Thực hiện lặp đến khi hàng đợi ưu tiên rỗng hoặc điểm đích đã thăm.

4. Từ node đích, lần ngược về node gốc theo mảng cha (parent), ta tìm được đường đi theo Greedy Best First Search.

2.4. Thuật toán tìm kiếm A* (có thông tin)

Thực hiện gần như tương tự với Greedy Best First Search ngoại trừ hàm đánh giá độ ưu tiên là $f(n) = h(n) + g(n)$ với $h(n)$ là heuristic và $g(n)$ là số bước đã đi. Giá trị của $f(n)$ càng nhỏ, độ ưu tiên càng cao.

2.5. Thuật toán tìm đường đi với chi phí ngắn nhất trên bảng đồ có điểm thưởng

Đối với bản đồ có điểm thưởng, ta có thể quy về việc giải nhiều bài toán tìm kiếm đường đi giữa điểm xuất phát đến các điểm thưởng, giữa các điểm thưởng với nhau và giữa các điểm thưởng và đích.

Ta xây dựng đồ thị với đỉnh là điểm xuất phát, đích và các điểm thưởng, cạnh có trọng số là đường đi giữa các điểm. Thực hiện tìm kiếm mọi đường đi trên đồ thị này, với điểm bắt đầu và kết thúc của đường đi là điểm bắt đầu và kết thúc của mê cung. Tính toán chi phí của các đường đi là tổng chi phí của các cạnh và giá trị của điểm thưởng, từ đó tìm ra đường đi có chi phí ngắn nhất.

3. Lựa chọn hàm heuristic cho các chiến lược tìm kiếm có thông tin

3.1. Heuristic là khoảng cách euclid đến đích

Ta có thể hướng tác nhân đến đích bằng cách lấy hàm heuristic là khoảng cách euclid từ điểm đang xét đến đích

Heuristic này không ước lượng quá cao chi phí đến đích và luôn nhỏ hơn chi phí đến đích (đường thẳng là đường đi ngắn nhất giữa 2 điểm). Suy ra đây là Heuristic hợp lý.

Gọi n là 1 điểm trên mê cung, n' là 1 successor của n .

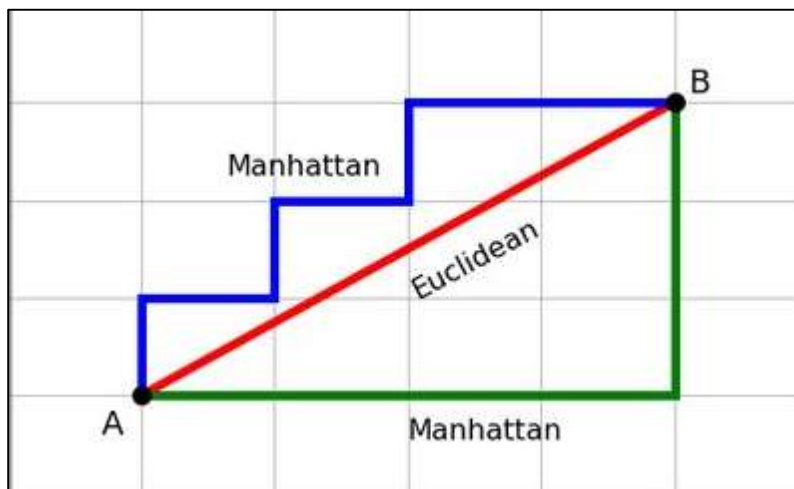
Gọi $d(a,b)$ là khoảng cách Euclid từ a đến b .

Ta có: $d(n, exit) \leq d(n, n') + d(n', exit) \Rightarrow h(n) \leq c(n, n') + h(n')$

Vậy đây là 1 heuristic nhất quán.

3.2. Heuristic là khoảng cách manhattan đến đích

Thực tế trong mê cung, ta hoàn toàn không có cách nào đi theo đường thẳng đến đích vì không gian mê cung của chúng ta là rời rạc. Vì vậy, ta có thể ước lượng chi phí đến đích chính xác hơn thông qua chi phí ít nhất cần có để đến đích. Đó là khoảng cách manhattan.



Heuristic này không ước lượng quá cao chi phí đến đích và luôn nhỏ hơn hoặc bằng chi phí đến đích (đường đi theo manhattan luôn đi qua số điểm là ít nhất do chỉ đi về 1 hướng theo trục Ox, và Oy). Suy ra đây là Heuristic hợp lý.

Gọi n là 1 điểm trên mê cung, n' là 1 successor của n .

Gọi $d(a,b)$ là khoảng cách Manhattan từ a đến b .




THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

Ta có: $d(n, exit) \leq d(n, n') + d(n', exit) \Rightarrow h(n) \leq c(n, n') + h(n')$




Vậy đây là 1 heuristic nhất quán.

4. Các chiến lược tìm kiếm không điểm thưởng trên các loại bản đồ khác nhau

4.1. Bản đồ maze_map0.txt - bản đồ có 2 hướng đi để đến đích.




Chiến lược	Chi Phí	Đường đi	Ghi chú
DFS	124		Đường đi của DFS phụ thuộc rất lớn vào thứ tự duyệt vị trí lân cận của 1 điểm. Trong trường hợp điểm thoát mê cung ở vị trí bất lợi so với thứ tự duyệt (ví dụ duyệt trên, trái trước mà đích ở hướng dưới phải) đường đi sẽ có chi phí cao.
BFS	48		Phép duyệt BFS sẽ luôn cho ta được đường đi đi qua ít đỉnh nhất của đồ thị, trong trường hợp trọng số các cạnh là 1, nó cũng cho ra đường đi ngắn nhất, tuy nhiên, thời gian thực hiện của BFS lâu hơn các thuật toán khác.
Greedy euclidean	88		Với tham lam, ta không quan tâm đến số bước đã đi, vì vậy, thuật toán sẽ cố gắng tìm đường đi đến đích theo heuristic. Trong trường hợp này, khoảng cách Euclid đã hướng đường đi theo hướng khác đường đi tối ưu.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO




A* euclidean	48		Cũng với heuristic là khoảng cách euclid, khi quan tâm thêm số bước đã đi của đường đi, thuật toán A* đã hướng đường đi về lại đúng hướng của đường đi ngắn nhất.
Greedy Manhattan	48		Sử dụng Heuristic tốt hơn khiến cho thuật toán tham lam không cần quan tâm đến số bước đi vẫn tìm ra được đường đi ngắn nhất.
A* Manhattan	48		A* với heuristic là khoảng cách manhattan cũng cho ra đường đi tốt nhất.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

4.2. Bản đồ maze_map1.txt - bản đồ có tường chắn dài ở gần đích.

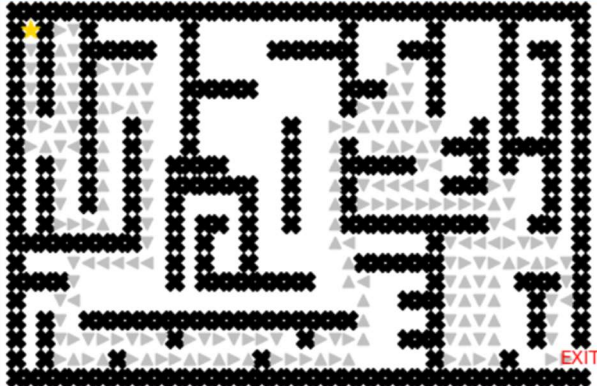
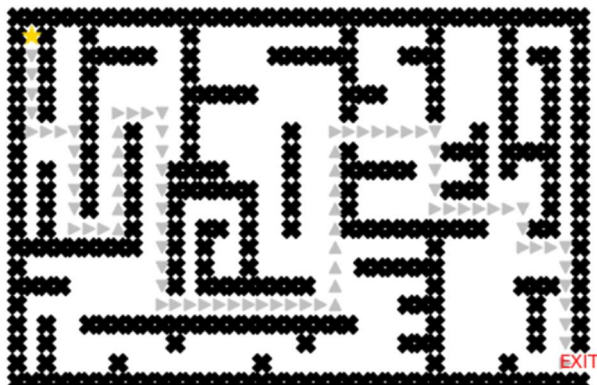
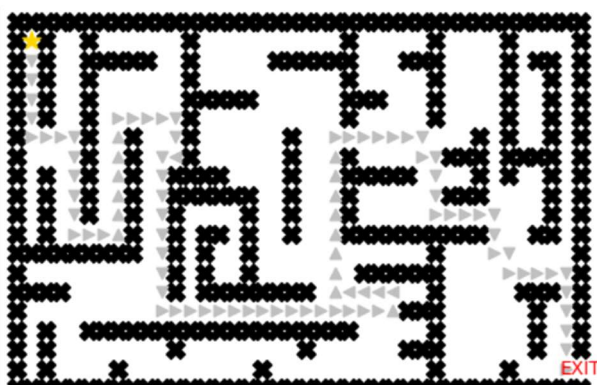
Chiến lược	Chi Phí	Đường đi	Ghi chú
DFS	138		DFS vẫn cho ra đường đi rất xấu và chi phí cao so với chi phí thấp nhất do sự tương quan giữa thứ tự duyệt và vị trí đích.
BFS	42		BFS trong trường hợp này sớm kết thúc và cho ta đường đi ngắn nhất.
Greedy euclidean	56		Với tham lam và khoảng cách euclid, đường đi sẽ thường có những đoạn zigzag và cố gắng lao về đích, khi bị tường cản thì mới tìm hướng khác để đi tiếp. Vì vậy, cây tìm kiếm trong trường hợp này sẽ rất sâu làm thuật toán thực hiện lâu.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

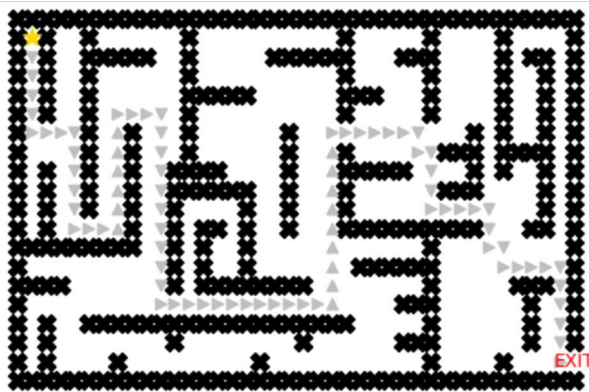
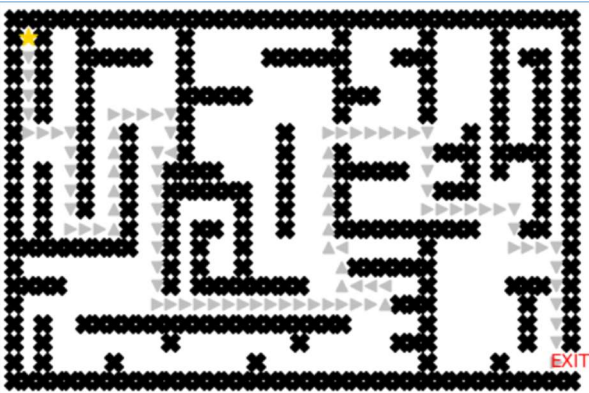
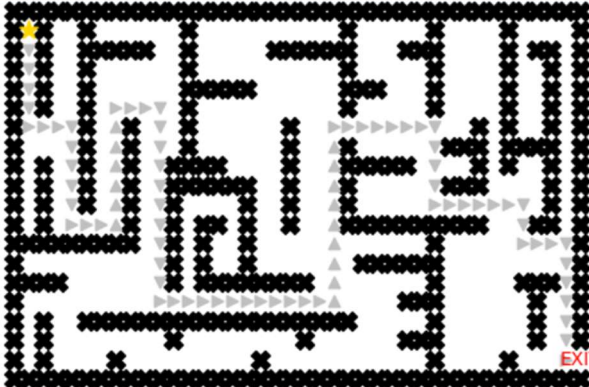
A* euclidean	42		Với A*, đoạn zigzag về đích sẽ tốn rất nhiều bước đi, vì vậy A* sẽ ưu tiên đi những đường khác thay vì tiếp tục nhánh tìm kiếm đó. Tuy nhiên, do tính chất của khoảng cách Euclid, đường đi sẽ có 1 đoạn zigzag ở gần đích
Greedy Manhattan	42		Sử dụng Heuristic tốt hơn, cụ thể đây là trường hợp mà chi phí đường đi bằng đúng khoảng cách Manhattan, nhờ vào thứ tự duyệt tốt sẽ làm cho đường đi ngắn nhất và tìm được nhanh chóng. Không tốn bộ nhớ để lưu như A*
A* Manhattan	42		A* trường hợp này cũng cho đường đi tốt nhất, không còn đoạn zigzag ở gần đích. Tuy nhiên, tốn bộ nhớ hơn và thời gian thực hiện lâu hơn so với tham lam.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

4.3. Bản đồ maze_map2.txt - bản đồ lớn, có nhiều ngạnh nhô lên trên tường.


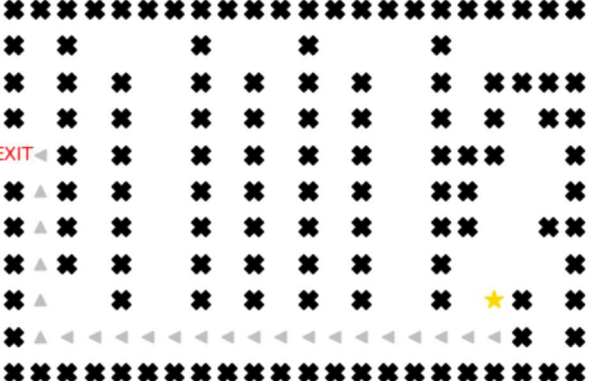
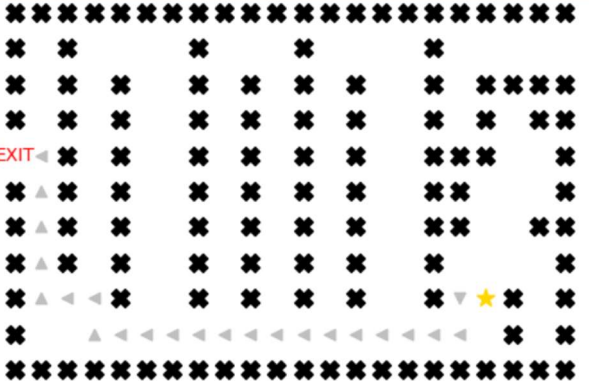
Chiến lược	Chi Phí	Đường đi	Ghi chú
DFS	202		DFS vẫn rất cao và gần như lấp đầy tất cả đường đi hẹp.
BFS	86		BFS cho đường đi ngắn nhất.
Greedy euclidean	96		Đường đi cố gắng đi những nơi gần đích, do đó có nhiều khúc phải quay lại. Thuật toán đi theo heuristic là khoảng cách Euclid, do đó có nhiều khúc zigzag.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

A* euclidean	86		Với A*, những đoạn quay lại đã bị hạn chế do những đoạn này tốn nhiều chi phí thực hiện.
Greedy Manhattan	96		Sử dụng Heuristic tốt hơn, ít đoạn zigzag, tuy nhiên, trong trường hợp này, chi phí thực hiện vẫn không giảm do vẫn còn nhiều đoạn quay lại.
A* Manhattan	86		Cho đường đi có chi phí thấp nhất và ít đoạn zigzag.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

4.4. Bản đồ maze_map3.txt - bản đồ có 2 đường đi, 1 ngắn, 1 dài.

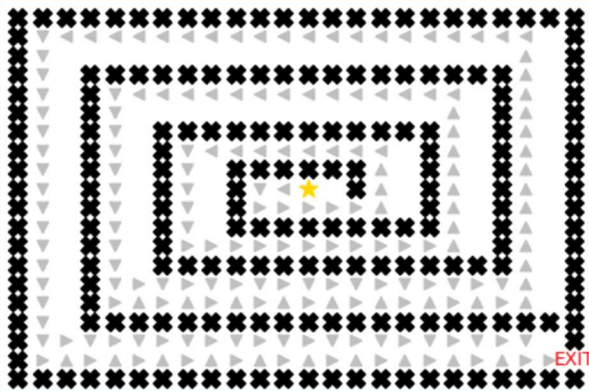


Chiến lược	Chi Phí	Đường đi	Ghi chú
DFS	73		Mặc dù lúc này, thứ tự duyệt DFS đã phù hợp hơn, ưu tiên đi lên trên và sang trái để về đích. Tuy nhiên do ngã rẽ này quá sớm, làm cho DFS đi vào đường đi rất xa.
BFS	25		BFS cho đường đi ngắn nhất.
Greedy euclidean	25		Trong trường hợp này, thuật toán tham lam vừa tìm được đường đi có chi phí thấp nhất, vừa chạy nhanh nhất nhờ thứ tự duyệt hợp lý.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

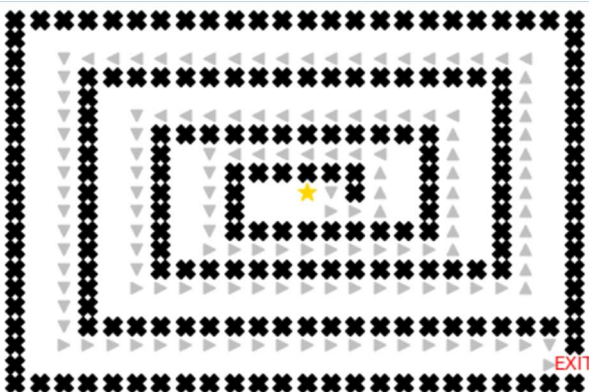
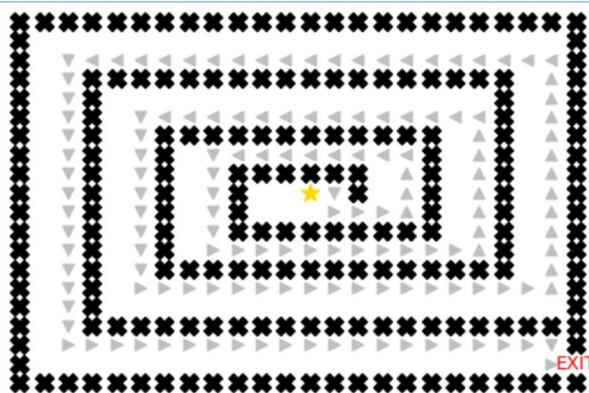
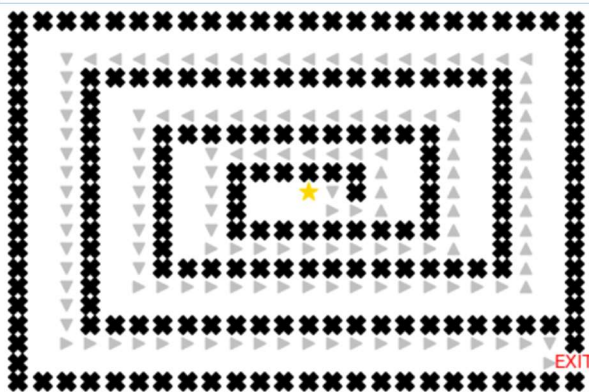
A* euclidean	25		Chi phí tốt nhất nhưng thời gian thực hiện lâu hơn tham lam.
Greedy Manhattan	25		Khử zigzag.
A* Manhattan	25		Khử zigzag

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

4.5. Bản đồ maze_map4.txt - bản đồ có dạng xoắn ốc.

Chiến lược	Chi Phí	Đường đi	Ghi chú
DFS	195		Do thứ tự duyệt, các đoạn đường đi theo chiều ngang của xoắn ốc ở bên trên sẽ không tốn nhiều chi phí. Ngược lại, các đoạn chiều ngang của xoắn ốc ở dưới sẽ bị lấp đầy bởi đường đi.
BFS	143		BFS cho đường đi ngắn nhất.
Greedy euclidean	151		Thuật giải tham lam cố gắng đi đến đích, vì vậy tại mọi vị trí trong xoắn ốc, nó đều cố gắng men theo mép tường ở bên phải. thời gian thực hiện là rất lớn vì đường đi đúng khác xa rất nhiều so với đường đi heuristic dự đoán.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

A* euclidean	143		Cho chi phí tốt nhất và giống với đường đi BFS
Greedy Manhattan	149		Khử zigzag và vẫn thực hiện rất lâu
A* Manhattan	143		Giống với A* theo euclid nhưng thực hiện nhanh hơn.

4.6. Tổng kết

Tất cả thuật toán tìm kiếm đường đi đều sẽ cho ra đường đi tới đích nếu tồn tại ít nhất 1 đường đi. Vì trong trường hợp xấu nhất, cây tìm kiếm của tất cả thuật toán sẽ duyệt qua toàn bộ vị trí có thể đi đến trong mê cung.

Thuật toán DFS cho ta lời giải không tốt trong nhiều trường hợp, có thể tăng hiệu suất bằng cách thực hiện sắp xếp thứ tự duyệt hợp lý với vị trí điểm ra. Tuy nhiên, trong các trường hợp đường đi đầu tiên đến vị trí điểm ra có chi phí lớn DFS vẫn không thể cải thiện nhiều.

Thuật toán BFS luôn cho ta đường đi có chi phí thấp nhất do trọng số các cạnh của các node trên cây tìm kiếm là 1, tuy nhiên, thời gian duyệt BFS có thể rất lâu.

Thuật toán tham lam tỏ ra khá hữu ích trong các trường hợp đường đi tới đích ít cạnh chẵn, những trường hợp này thời gian thực hiện thuật toán khá nhanh và tốn ít chi phí bộ nhớ hơn so với A*. Tuy nhiên trong thực tế, xác suất xuất hiện những mê cung này là rất thấp.

Thuật toán A* có thể xem là tối ưu nhất trong 1 trường hợp ngẫu nhiên, A* vừa đảm bảo đi ít bước, vừa đảm bảo đi đường đi có khả năng về đích nhất.

Heuristic là khoảng cách Manhattan tỏ ra vượt trội hơn hẳn khoảng cách Euclid trong mọi bản đồ kiểm thử.

5. Chiến lược tìm kiếm trên bảng đồ có điểm thưởng để có chi phí thấp nhất

5.1. Chiến lược

1. Thực hiện nhiều lần thuật toán A* với điểm đầu là điểm xuất phát hoặc các điểm thưởng, điểm đích là các điểm thưởng hoặc lối ra, Vậy có tổng cộng $C_{2+\text{số điểm thưởng}}^2 - 2$ lần chạy A*. Tương ứng với số điểm 2, 5, 10 là 4, 19, 64 lần chạy A*.

2. Xây dựng đồ thị không trọng số, với đỉnh là điểm đầu, điểm cuối, các điểm thưởng.

3. Tìm tất cả đường đi từ đỉnh xuất phát tới đích trên đồ thị này.

4. Thực hiện tính toán chi phí là tổng các chi phí A* giữa các đường đi và tổng điểm thưởng nếu có đi qua các điểm thưởng.


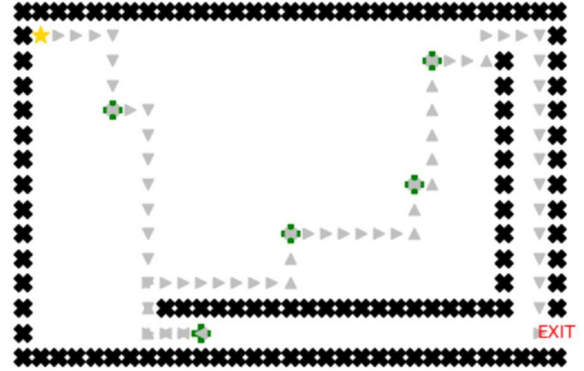

6. Trường hợp cần tìm là trường hợp có chi phí trên là thấp nhất.

Thuật toán sẽ có điểm dừng, tuy nhiên, với trường hợp có 10 điểm, thuật toán sẽ chạy với thời gian khá lâu.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

Tính chính xác của chi phí phụ thuộc vào tính chính xác của thuật toán A*, có thể thay thế A* bằng BFS để đảm bảo độ chính xác, tuy nhiên phải đánh đổi thời gian thực hiện.

5.2. Chạy thuật toán với các bản đồ có 2, 5, 10 điểm thưởng

Bản đồ	Chi Phí	Đường đi	Ghi chú
2 điểm thưởng 1 19 -25 9 6 -4	42		Đường đi ăn điểm có trọng số là -25 thay vì đi đường đi ngắn nhất về số bước đi (với chi phí là 48).
5 điểm thưởng	-2		Đường đi ăn tất cả các điểm thưởng do chi phí đi đến điểm thưởng thấp hơn chi phí được trừ khi ăn điểm thưởng.
10 điểm thưởng Các điểm thưởng có trọng số là -10	78		Đường đi chỉ ăn những điểm thưởng phù hợp để đạt được chi phí đường đi thấp nhất.

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO



6. Trường hợp bản đồ có cổng dịch chuyển

6.1. Chiến lược

Nếu tác nhân đến điểm dịch chuyển, thay vì chỉ có các bước di chuyển là trên, dưới, trái, phải, ta thêm 1 trường hợp di chuyển nữa là cổng dịch chuyển ở đầu bên kia.

Thuật toán phải thêm 1 hàm kiểm tra điểm hiện tại có phải điểm dịch chuyển không, làm giảm hiệu suất thực hiện của các thuật toán gốc.

6.2. Thực hiện với bản đồ có điểm dịch chuyển các thuật toán DFS, BFS

Chiến lược	Đường đi	Ghi chú
DFS		Mặc dù đã có sự hỗ trợ của điểm dịch chuyển, tuy nhiên do thứ tự duyệt, tác nhân lại đi đường vòng để về đích
BFS		BFS cho ra đường đi có chi phí thấp nhất thông qua điểm dịch chuyển.

7. Cách sử dụng mã nguồn

Mở file notebook đính kèm với mã nguồn.

```
from main import find_path
```

```
find_path(<TênFileBảnĐồ.txt>,<TênChiếnLượcTìmKiếm>,<TênHeuristicSửDụngNếuCó>).
```

THỰC HÀNH CƠ SỞ TRÍ TUỆ NHÂN TẠO

Trong đó:

<TênFileBảnĐồ.txt>: đường dẫn đến file bản đồ có sẵn.

<TênChiếnLượcTìmKiếm>: Các chiến lược tìm kiếm ("DFS", "BFS", "GBFS", "A*").

<TênHeuristicSửDụngNếuCó>: 2 phương pháp tính Heuristic ("euclidean_distance", "manhattan_distances")

Ví dụ:

```
: from main import find_path  
  
find_path("maze_map7.txt", "eat_points")
```

Cost: 78



-HẾT-