

Trường Đại học Khoa học Tự nhiên - Đại học Quốc Gia TP.HCM

Môn học: Cơ sở trí tuệ nhân tạo - 19_21
Project 1: Ứng dụng các giải thuật tìm kiếm

Nhóm có 1 thành viên:

Tên: Bùi Quang Bảo

MSSV: 19120454

Bản đồ không điểm thưởng: Tổng quan

Tóm tắt phần cài đặt 4 thuật toán:

- DFS: Sử dụng frontier là 1 stack
- BFS: Sử dụng frontier là 1 queue
- GBFS: Sử dụng frontier, mà khi chọn ra node kế tiếp sẽ xét xem node nào có giá trị **khoảng cách Manhattan** nhỏ nhất để remove khỏi frontier và xét node đó.
- A*: Giống với GBFS nhưng thay vì chỉ xét khoảng cách Manhattan, A* sẽ xét giá trị: **$h = \text{khoảng cách Manhattan} + \text{chi phí hiện tại}$** của node muốn xét. Lấy node có giá trị **h** nhỏ nhất.

Chi phí hiện tại của 1 vị trí: Số bước để đi đến vị trí đó (trên con đường đang xét).

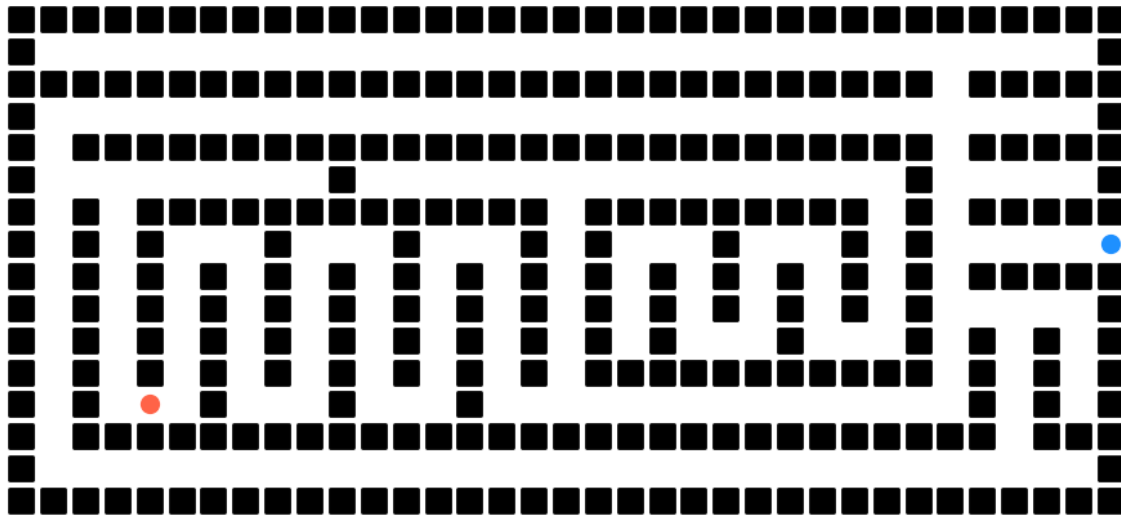
Một số lưu ý:

- Phương thức visualize trong class Maze thực ra là hàm visualize_maze trong file IntroToAI_Demo_DoAn1.ipynb mà thầy trợ giảng đã cung cấp, em mang về và tùy biến một số ký hiệu, màu sắc,... và thêm vào một số tính năng khác (track luôn những node đã đi tới bằng chấm tròn màu xám, in ra những giá trị để xét trong GBFS và A*)
- Em có thay đổi 1 số ký hiệu khi visualize để dễ nhìn hơn:
 - Tường: sử dụng các ô vuông thay vì chữ x
 - Vị trí bắt đầu: Chấm tròn đỏ
 - Vị trí kết thúc: Chấm tròn xanh
 - Đường đi solution: Tam giác xanh dương, chỉ hướng di chuyển
 - Những vị trí đã “khám phá” mà không thuộc solution: Chấm tròn màu xám
- Lí do em chọn khoảng cách Manhattan thay vì khoảng Euclidean trong trường hợp giải mê cung:
 - Ở thuật toán A*, tổng giá trị của khoảng cách Manhantan và chi phí hiện tại sẽ ra kết quả “đẹp” hơn, cụ thể trong các kết quả giải mê cung sẽ rõ, em có in ra giá trị để dễ quan sát.

- Sử dụng khoảng cách Manhattan hợp lý hơn, vì chi phí hiện tại cũng tính theo kiểu Manhattan (chứ agent không thể đi chéo được, chỉ có trên, xuống, trái, phải).
- **Chi phí solution** là chi phí của đường đi tới đích mà thuật toán trả về; solution được thể hiện trong hình ảnh là những tam giác màu xanh dương.
- **Tổng chi phí thực hiện** sẽ là chi phí của **tất cả** các vị trí mà agent đã đi tới (kể cả không nằm trong solution); được thể hiện trong hình ảnh là cả những tam giác màu xanh dương và chấm tròn màu xám.

Mê cung 1 (bất lợi cho DFS, GBFS)

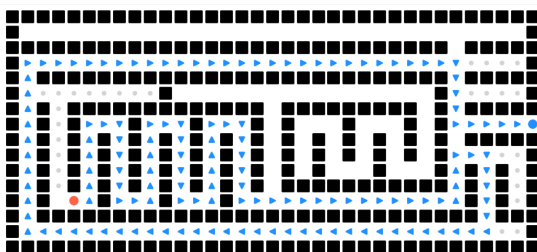
Chiều cao: 16, chiều rộng: 35 (thoả mãn bản đồ lớn)



Maze

Tổng quan giải mê cung bằng 4 thuật toán:

Thuật toán DFS

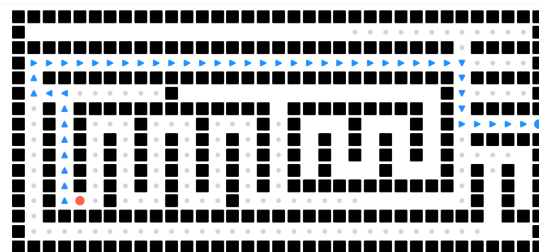


DFS - Depth First Search

Tổng chi phí thực hiện: 172

Solution: 143

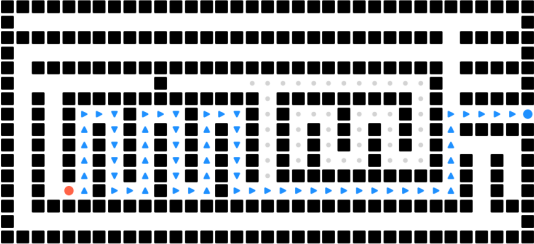
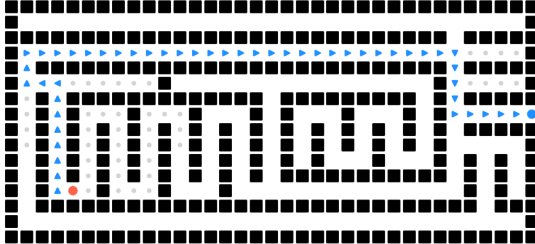
Thuật toán BFS



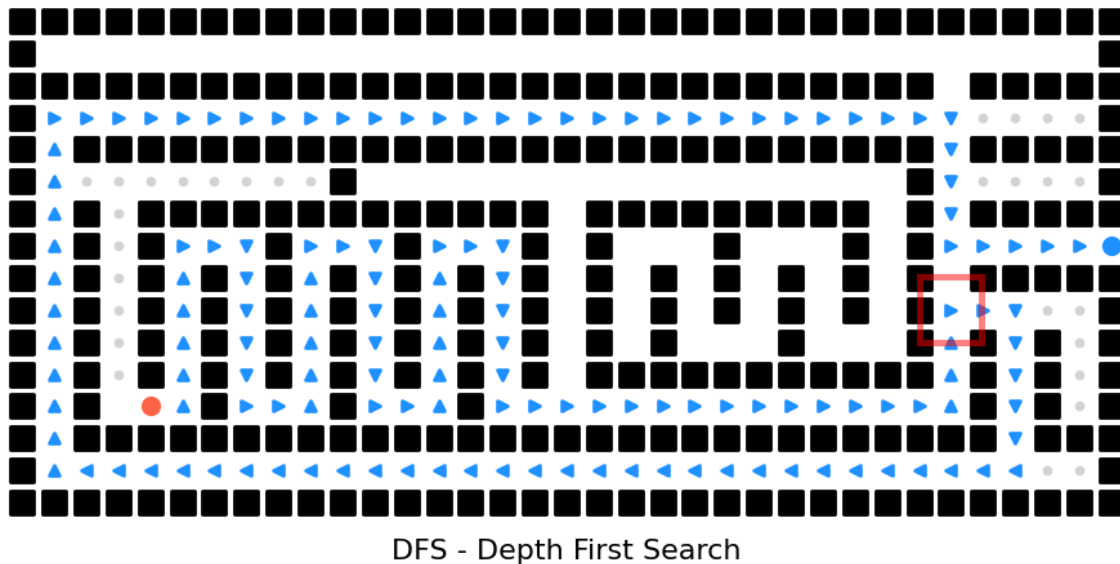
BFS - Breadth First Search

Tổng chi phí thực hiện: 176

Solution: 49 (Optimal solution)

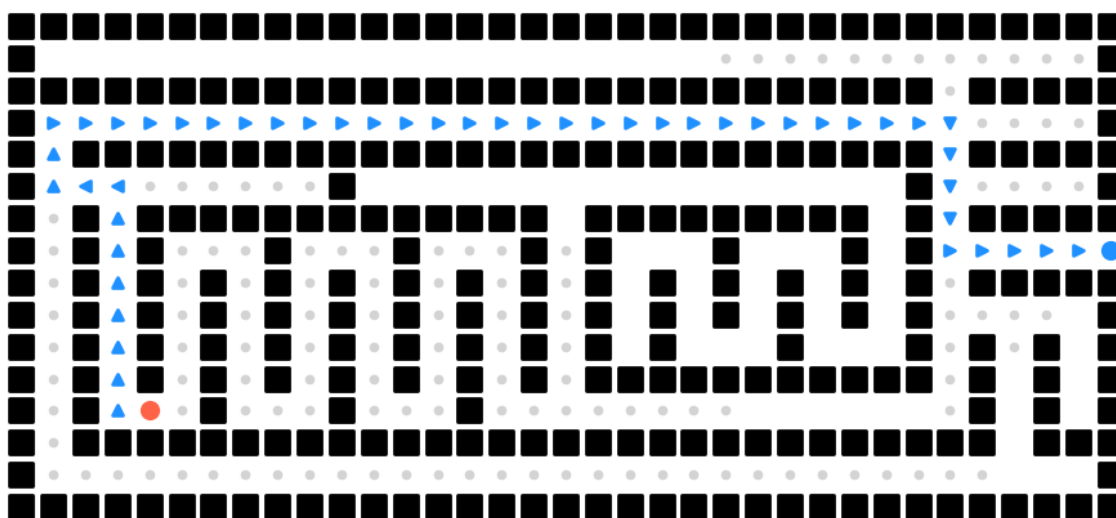
<p>Thuật toán GBFS</p>  <p>GBFS - Greedy Best First Search</p> <p>Tổng chi phí thực hiện: 108</p> <p>Solution: 65</p>	<p>Thuật toán A*</p>  <p>A*</p> <p>Tổng chi phí thực hiện: 91</p> <p>Solution: 49 (Optimal solution)</p>
--	--

Giải mê cung theo thuật toán DFS - Depth First Search:



Ở ma trận này, DSF không chỉ tốn rất nhiều chi phí mà còn đưa ra 1 cách giải rất dài. Ở vị trí được đánh dấu khung đỏ, DFS đã “không may mắn”, thay vì di chuyển lên trên thì lại di chuyển sang phải trước và kết quả là đi 1 vòng mê cung.

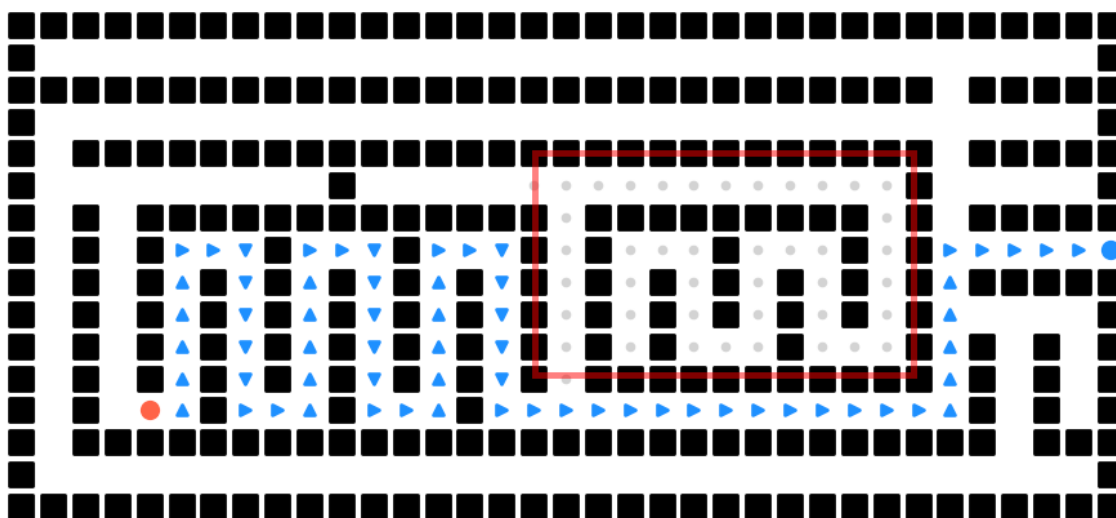
Giải mê cung theo thuật toán BFS - Breadth First Search:



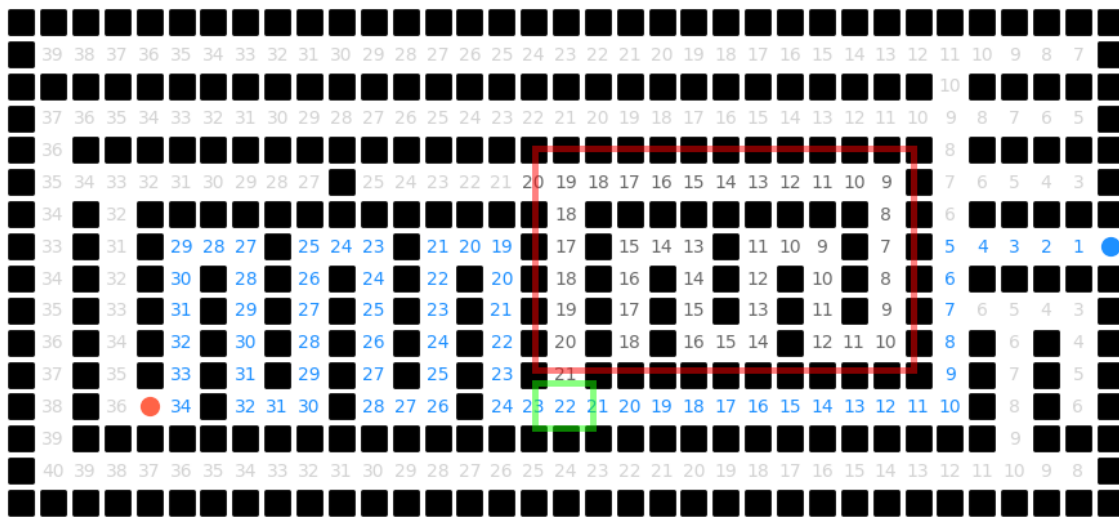
BFS - Breadth First Search

Thuật toán BFS luôn cho lời giải tối ưu, tuy nhiên tổng chi phí vẫn rất lớn vì BFS có xu hướng lan rộng ra khắp mê cung.

Giải mê cung theo thuật toán GBFS - Greedy Best First Search:



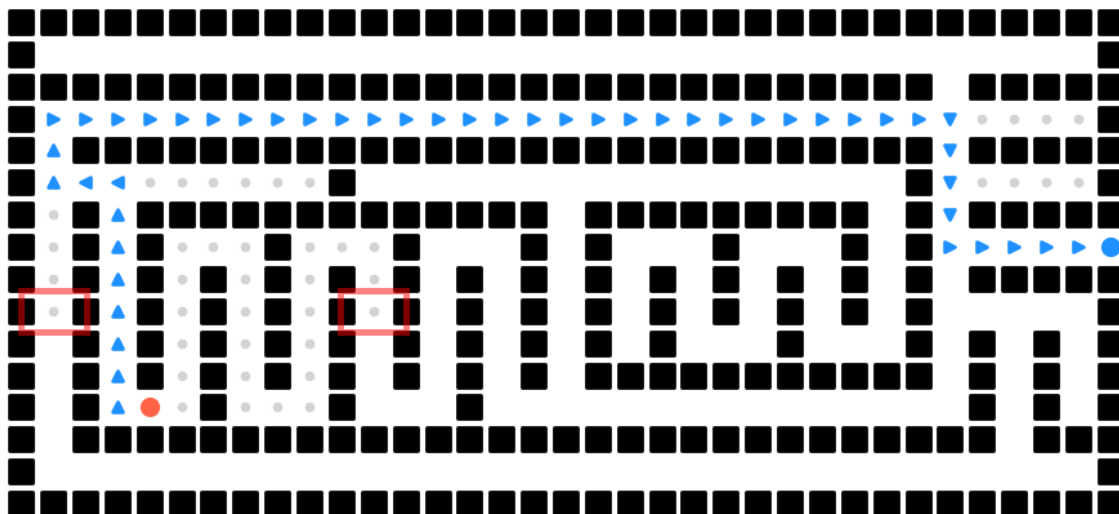
GBFS - Greedy Best First Search



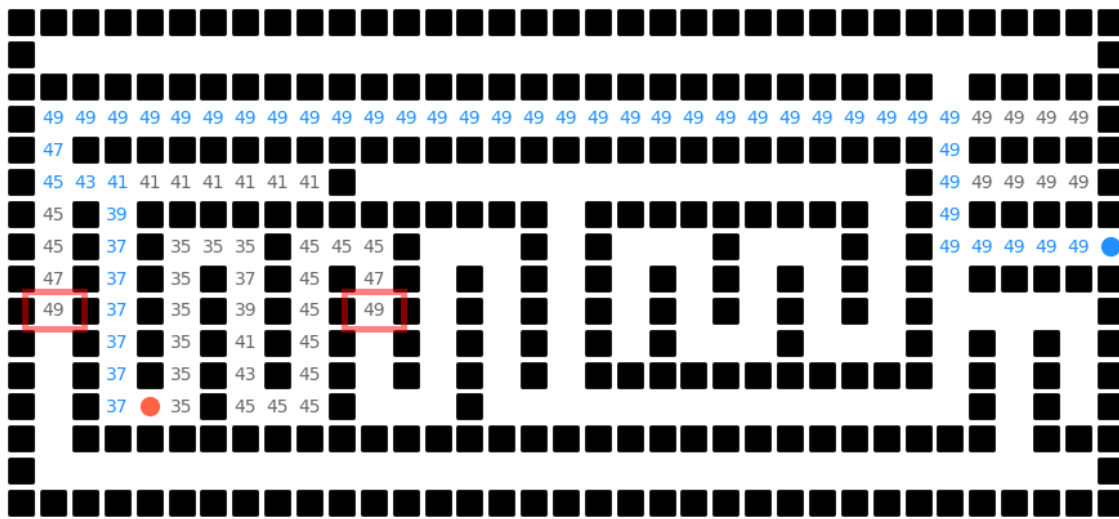
GBFS - Greedy Best First Search (Show Manhattan)

Mê cung này đã cố tình bẫy GBFS ở khung đánh dấu đỏ. Vì các khoảng cách manhattan ở trong khung nhỏ hơn 22 cho nên GBFS đã tốn chi phí để “loay hoay” trong khung, sau đó mới chuyển đường đi xuống phía dưới và tới đích.

Giải mê cung theo thuật toán A*:



A*



A* (Show Manhattan + Cost)

Thực tế thì trong nhiều trường hợp, GBFS sẽ tốn ít chi phí để tìm ra lời giải hơn A* (Mê cung này cố tình bày cho nên GBFS mới tốn chi phí hơn). Tuy nhiên, GBFS không đảm bảo việc ra đường đi ngắn nhất (kết quả tối ưu).

Ở thuật toán A*, tùy theo cài đặt heuristic mà thuật toán có trả về lời giải tối ưu nhất hay không, tuy nhiên sẽ không bị dính “bẫy” như GBFS mà trả về lời giải ngắn hơn.

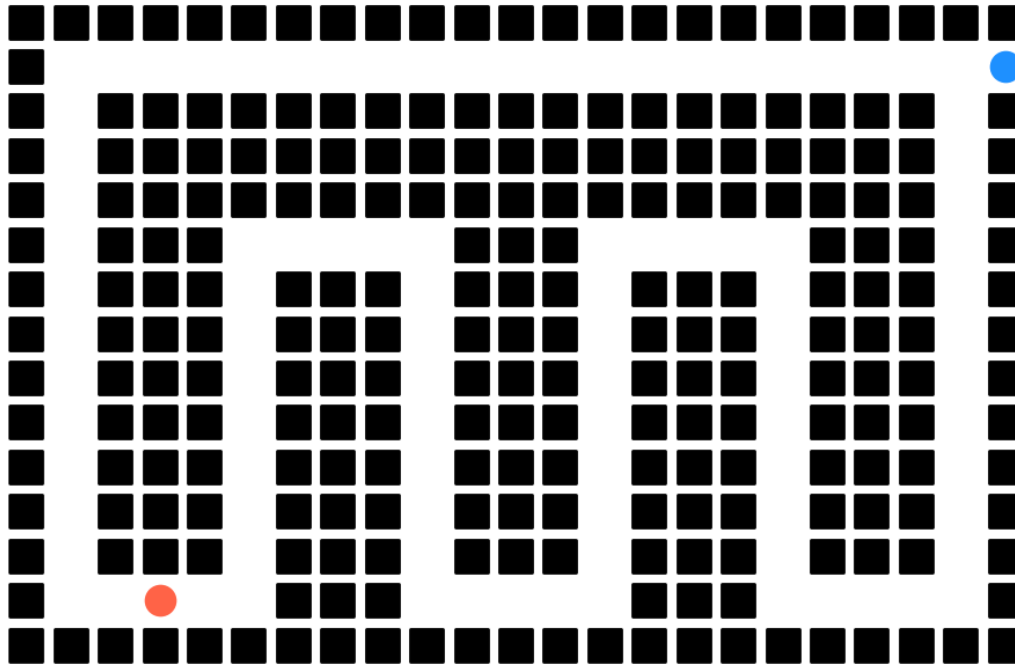
Mê cung này thể hiện rõ điều đó, tại 2 vị trí đánh dấu khung đỏ, ta có thể thấy rằng A* đã “biết dừng đúng lúc” để chuyển hướng sang đường đi ngắn hơn.

2 vị trí đánh dấu khung đỏ, nếu tiến thêm 1 bước (di chuyển xuống dưới), thì tổng giá trị của: khoảng cách manhattan + chi phí hiện tại, đều là 50. Cho nên thay vì tiếp tục thì A* đã chuyển hướng ở những ô có giá trị 49 và tới đích với đường đi ngắn nhất.

So với BFS cũng tìm ra lời giải tối ưu thì A* tốn rất ít chi phí thực hiện.

Mê cung 2 (bất lợi cho GBFS)

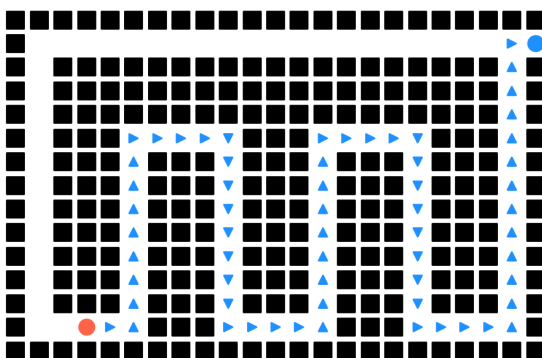
Chiều cao: 15, chiều rộng: 23



Maze

Tổng quan giải mê cung bằng 4 thuật toán:

Thuật toán DFS

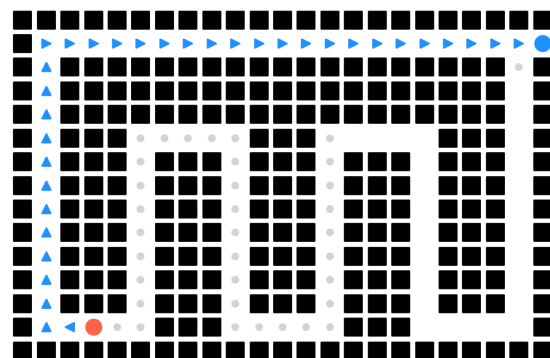


DFS - Depth First Search

Tổng chi phí thực hiện: 63

Solution: 63

Thuật toán BFS

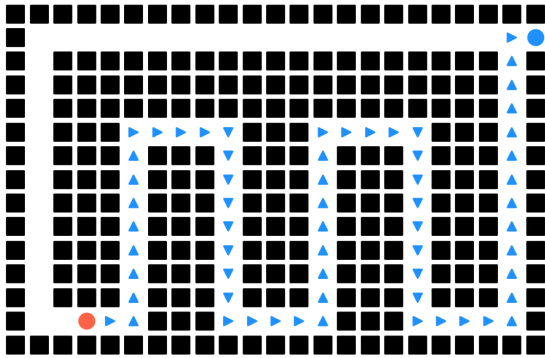


BFS - Breadth First Search

Tổng chi phí thực hiện: 70

Solution: 35 (Optimal solution)

Thuật toán GBFS

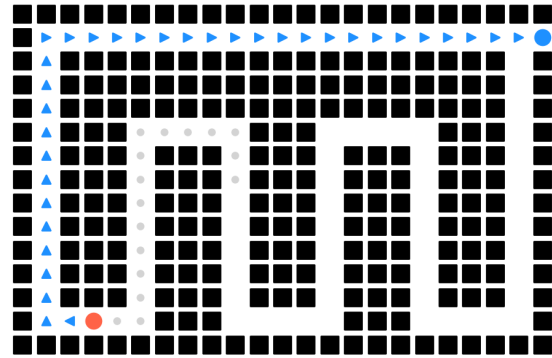


GBFS - Greedy Best First Search

Tổng chi phí thực hiện: 63

Solution: 63

Thuật toán A*

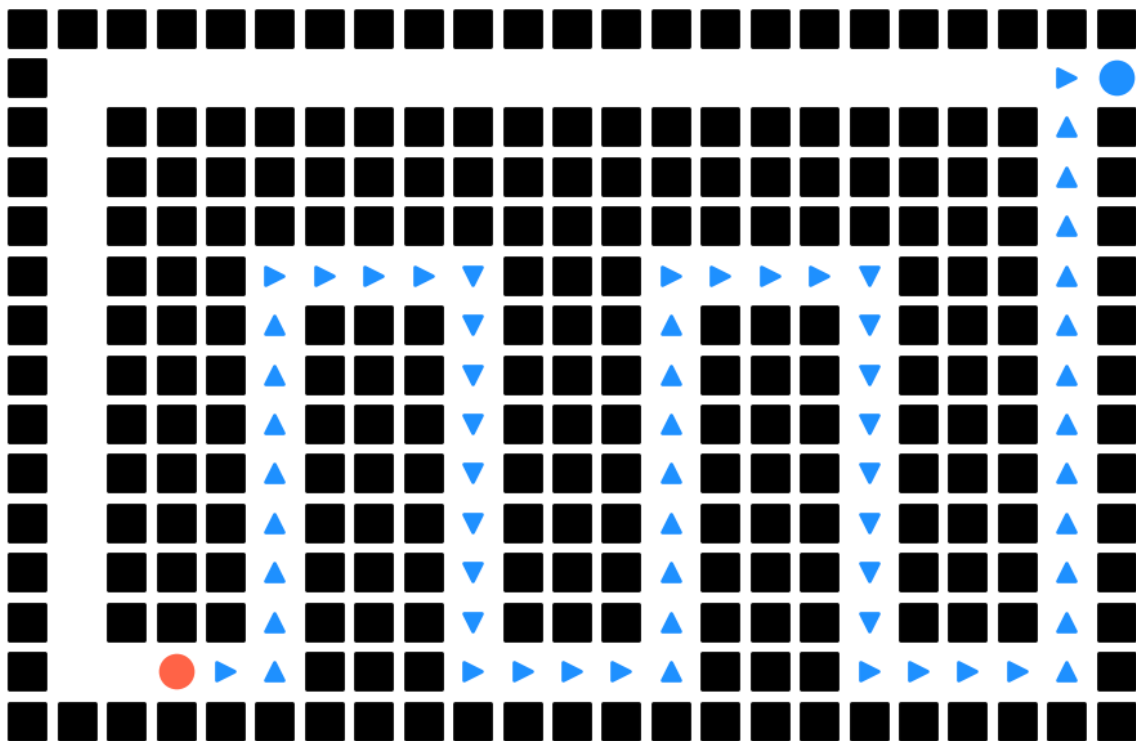


A*

Tổng chi phí thực hiện: 51

Solution: 35 (Optimal solution)

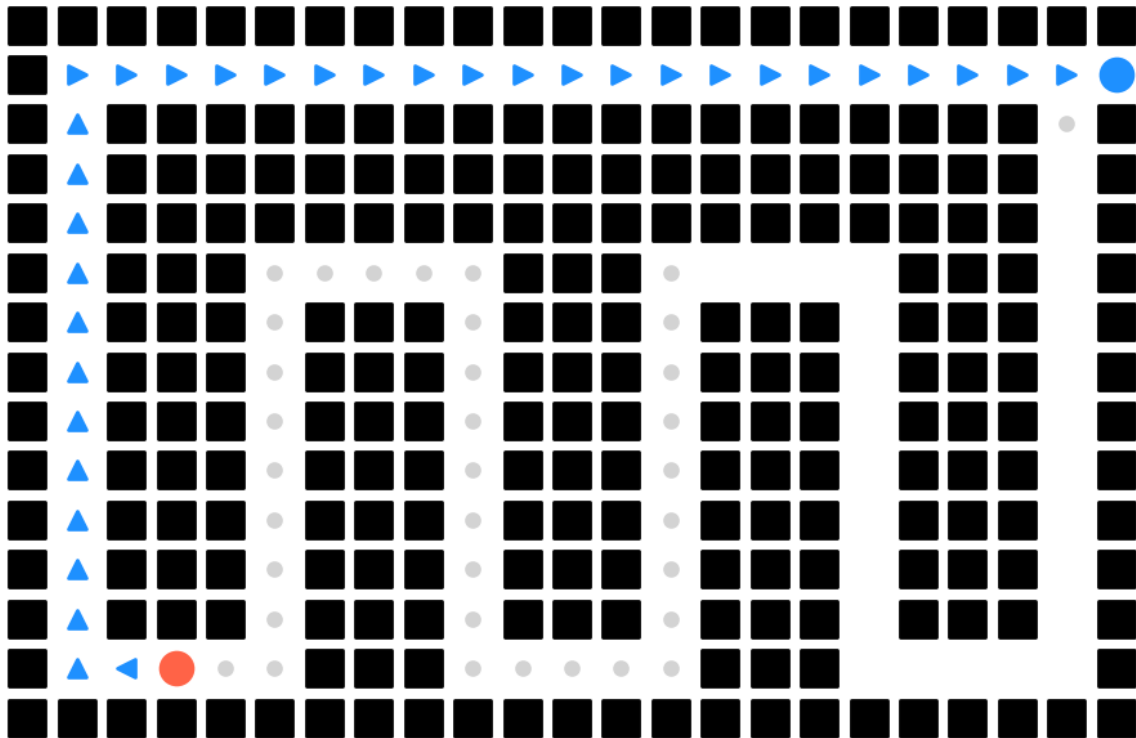
Giải mê cung theo thuật toán DFS - Depth First Search:



DFS - Depth First Search

Ở mê cung này, chỉ có 2 con đường cho DFS. Tùy thuộc vào cài đặt mà DFS sẽ chọn con đường nào. DFS đã “không may mắn” trong mê cung này.

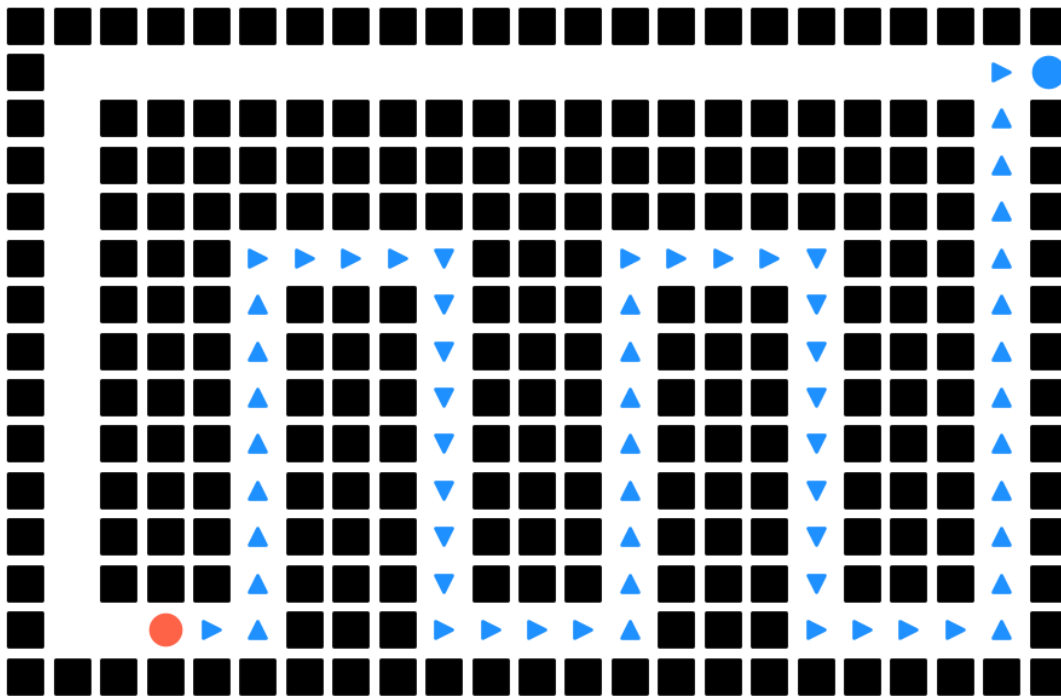
Giải mê cung theo thuật toán BFS - Breadth First Search:



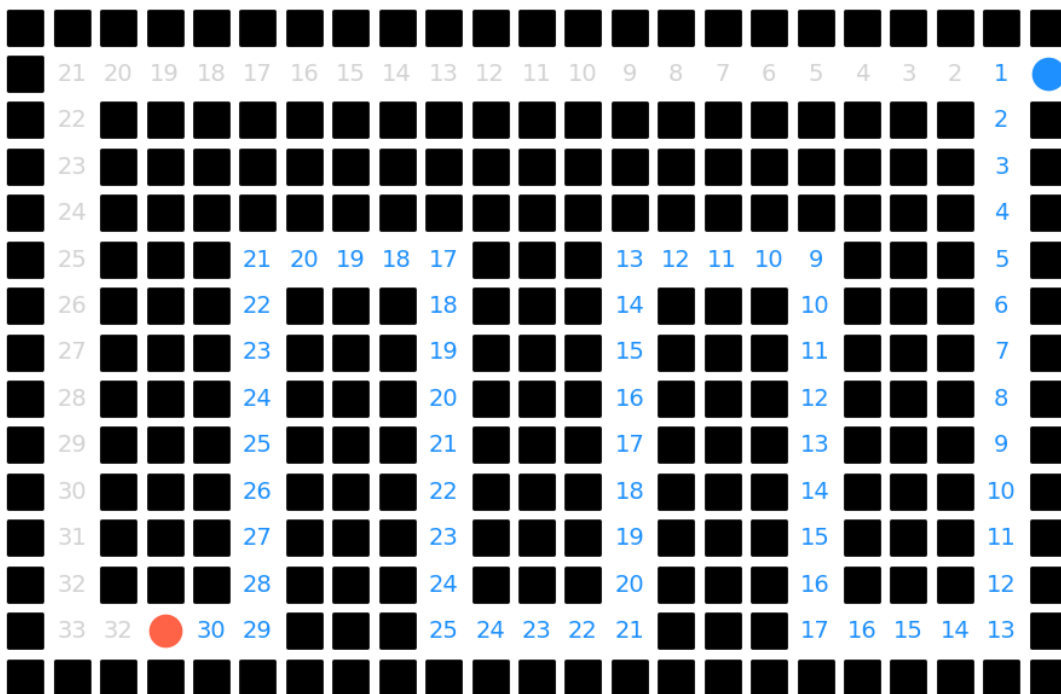
BFS - Breadth First Search

Tuy tốn nhiều chi phí nhưng BFS luôn cho ra đường đi ngắn nhất.

Giải mê cung theo thuật toán GBFS - Greedy Best First Search:



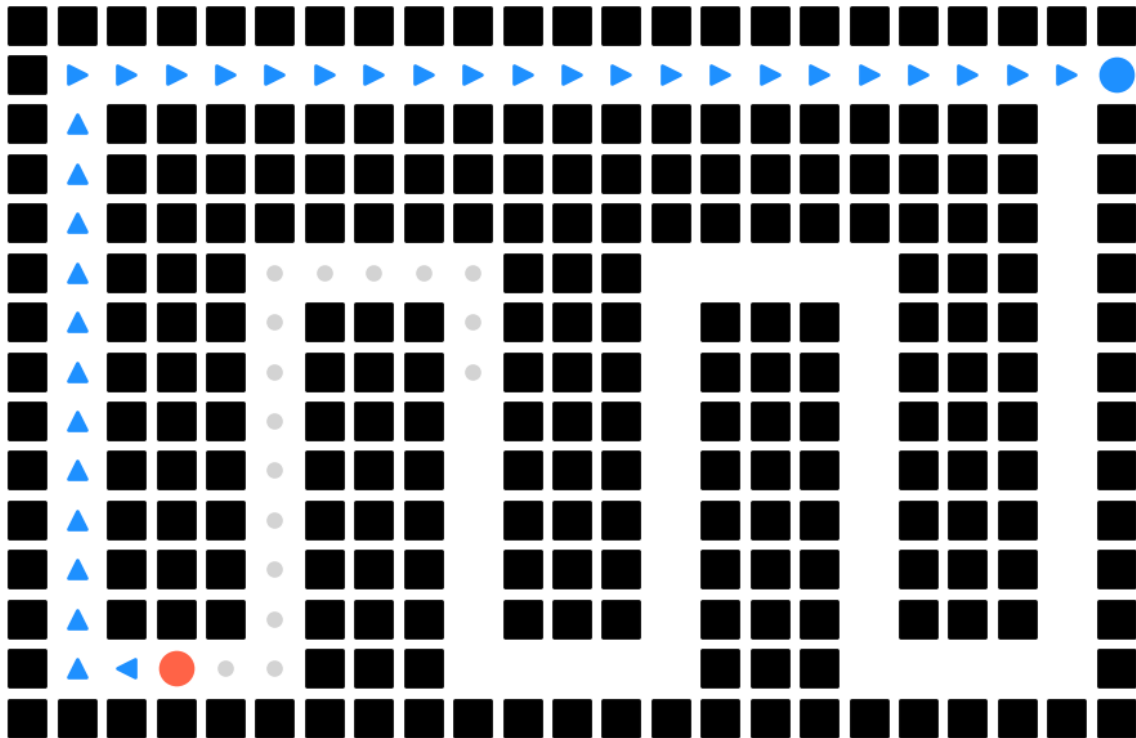
GBFS - Greedy Best First Search



GBFS - Greedy Best First Search (Show Manhattan)

Với 2 giá trị “32” và “30”, không khó để hiểu vì sao GBFS chọn đường đi này, vì tất cả các giá trị đều không vượt được 32. Tuy nhiên, GBFS đã chọn đường đi không tối ưu.

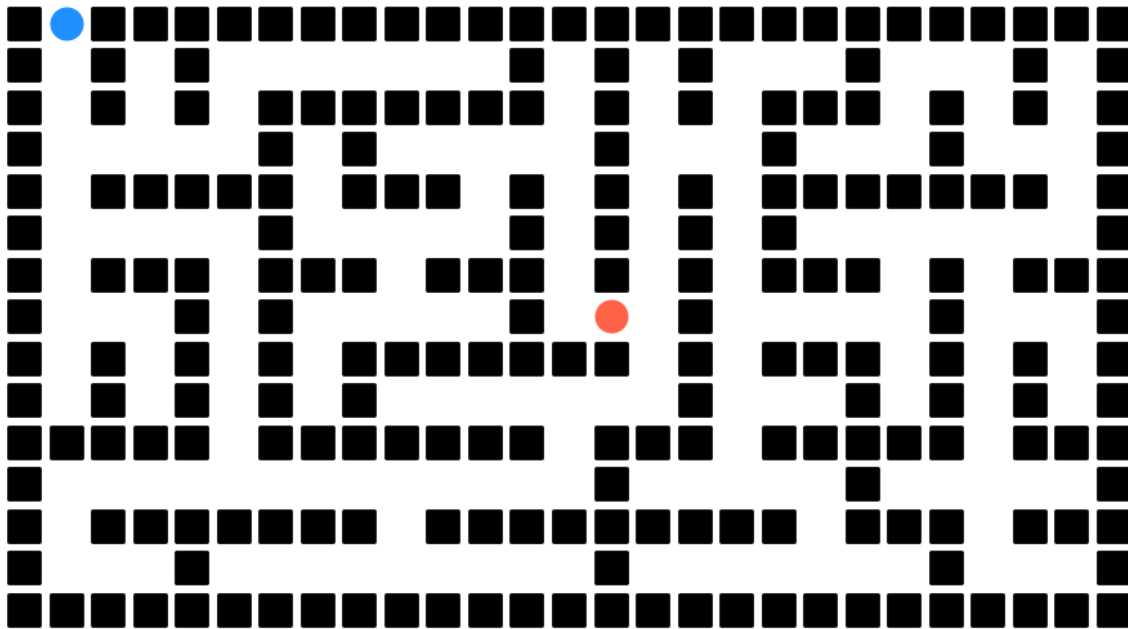
Giải mê cung theo thuật toán A*:



A*

Mê cung 3 (bất lợi cho BFS)

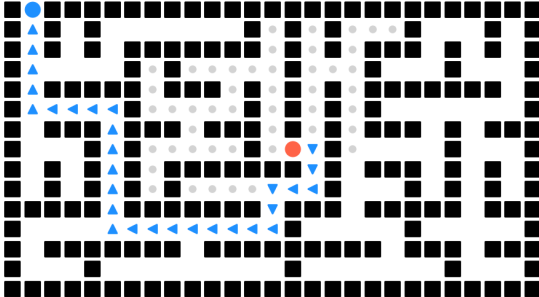
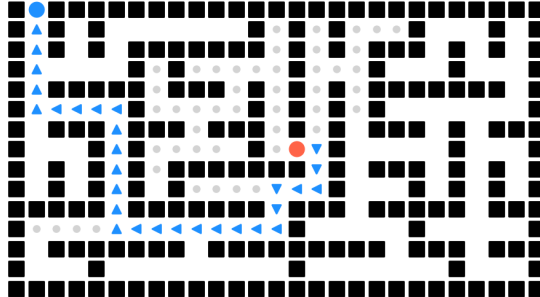
Chiều cao: 15, chiều rộng: 27



Maze

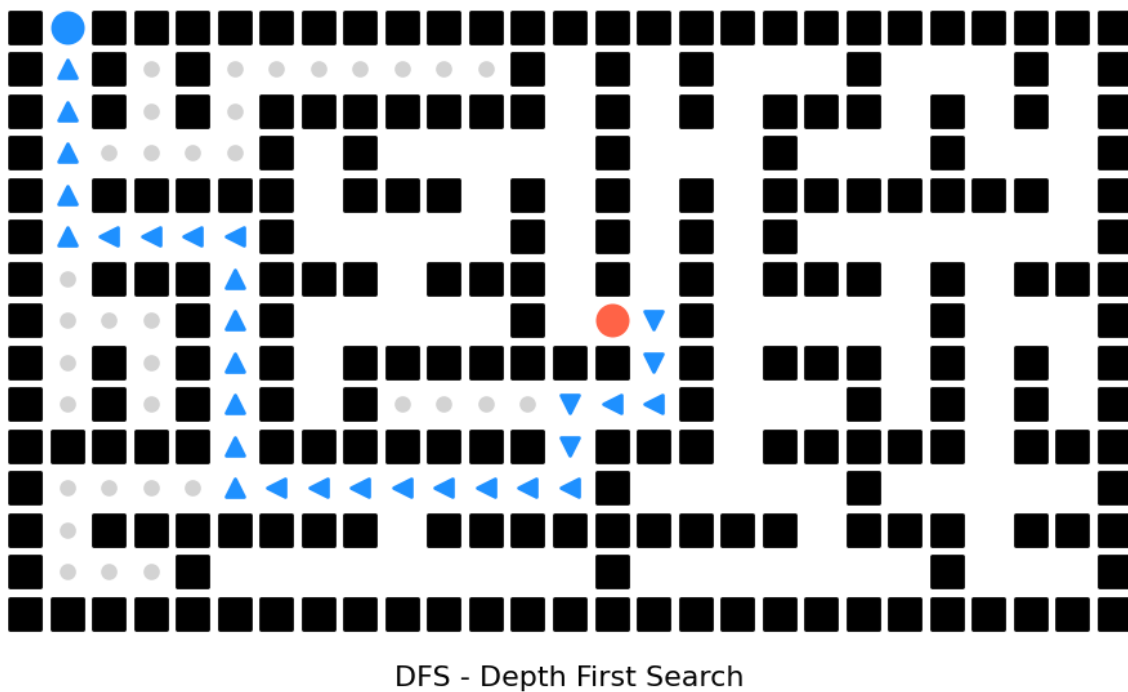
Tổng quan giải mê cung bằng 4 thuật toán:

Thuật toán DFS	Thuật toán BFS
DFS - Depth First Search	BFS - Breadth First Search
Tổng chi phí thực hiện: 64 Solution: 30 (Optimal solution)	Tổng chi phí thực hiện: 164 Solution: 30 (Optimal solution)

<p>Thuật toán GBFS</p>  <p>GBFS - Greedy Best First Search</p> <p>Tổng chi phí thực hiện: 77</p> <p>Solution: 30 (Optimal solution)</p>	<p>Thuật toán A*</p>  <p>A*</p> <p>Tổng chi phí thực hiện: 77</p> <p>Solution: 30 (Optimal solution)</p>
--	--

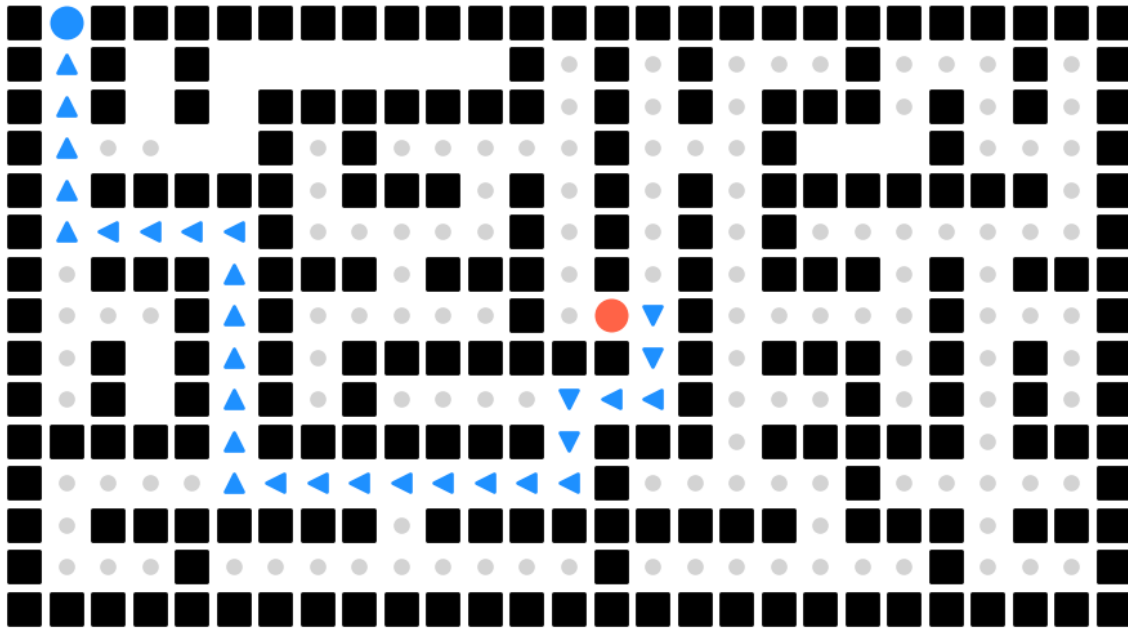
Nhận xét chung: Ở mê cung này, cả 4 thuật toán đều tìm được đường đi ngắn nhất. Tất nhiên đối với DFS và GBFS thì đó là do “may mắn”. Tuy nhiên ta vẫn thấy rõ sự khác nhau trong chi phí tìm kiếm, đặc biệt là đối với BFS.

Giải mê cung theo thuật toán DFS - Depth First Search:



DFS đã “may mắn” ở mê cung này khi tìm ra đường đi ngắn nhất, với chi phí không quá lớn.

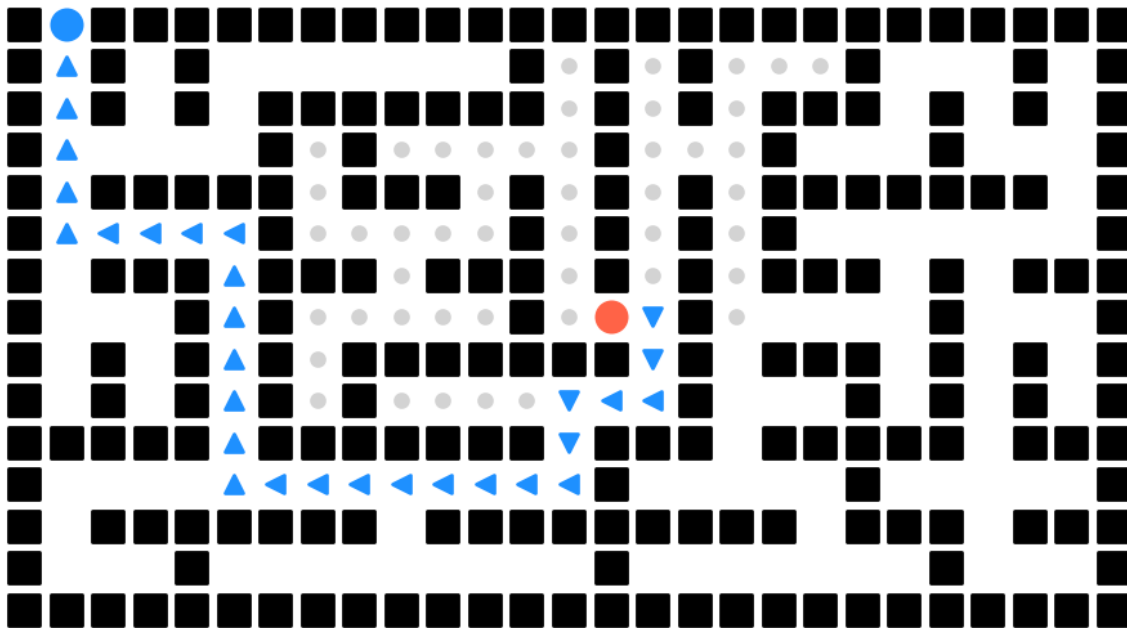
Giải mê cung theo thuật toán BFS - Breadth First Search:



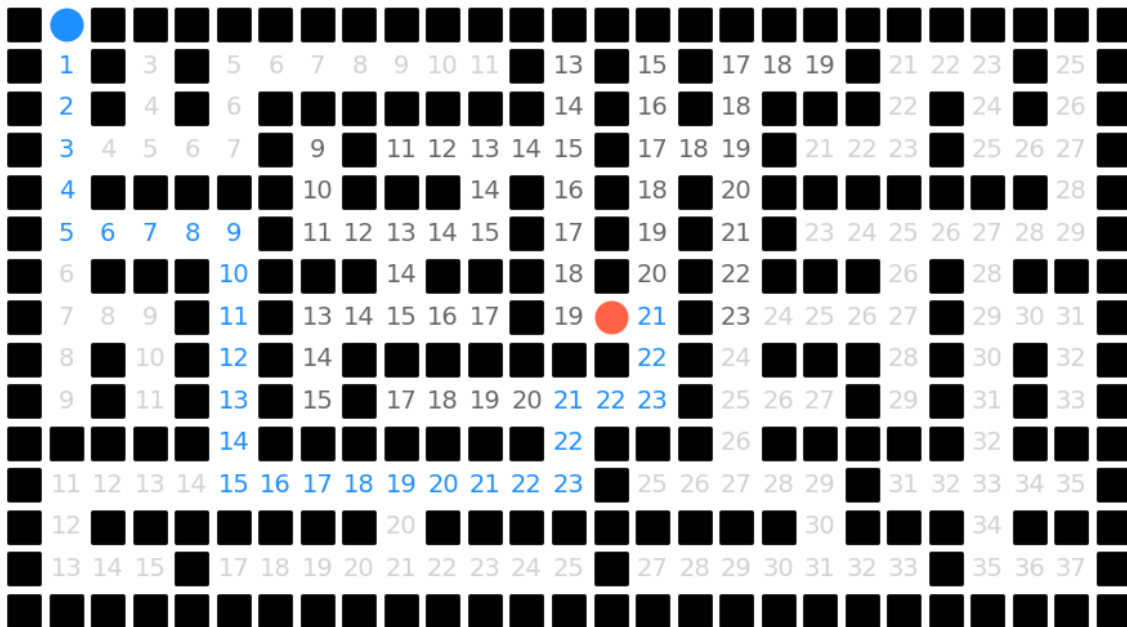
BFS - Breadth First Search

Vì cố tình đặt điểm xuất phát của agent ngay giữa mê cung, cho nên BFS đã có tổng chi phí thực hiện cao hơn rất nhiều so với các thuật toán khác. “Chúc mừng” BFS.

Giải mê cung theo thuật toán GBFS - Greedy Best First Search:

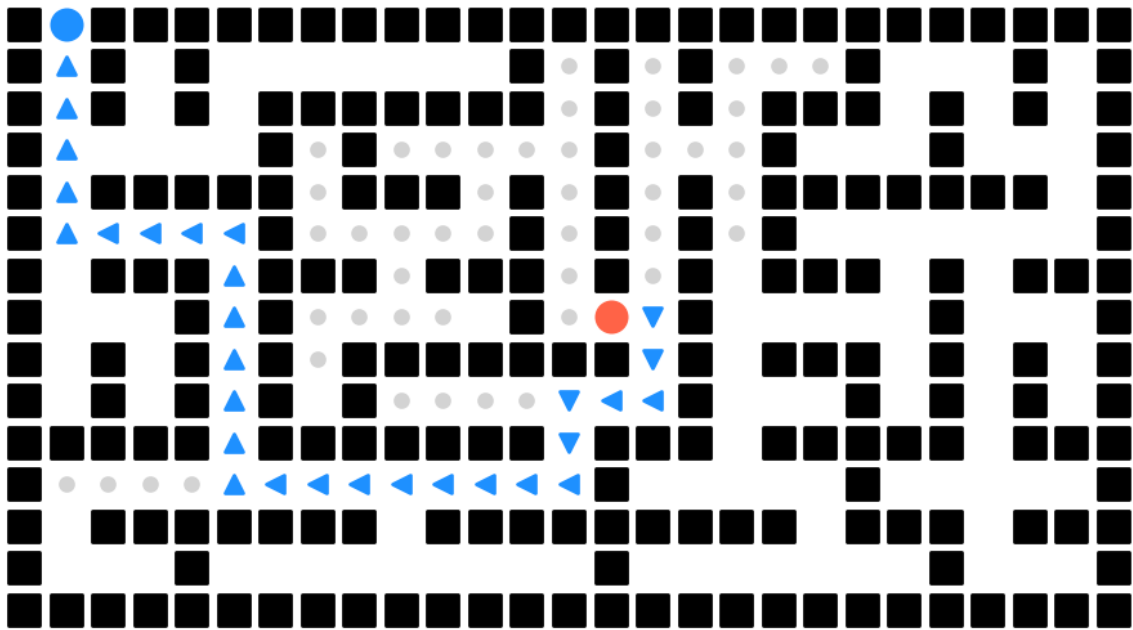


GBFS - Greedy Best First Search

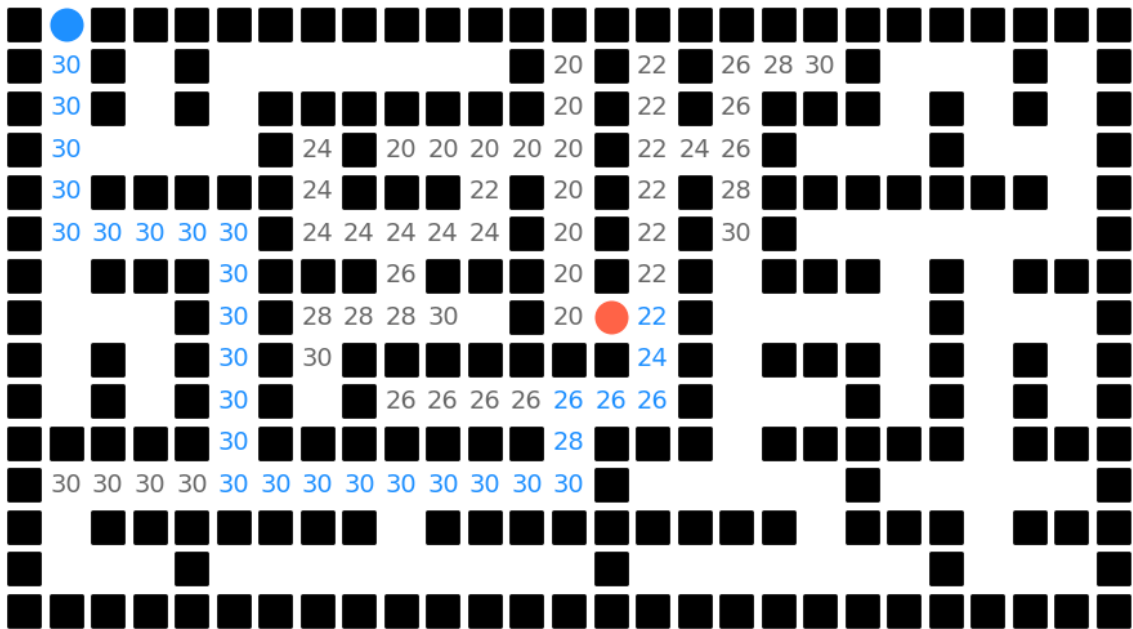


GBFS - Greedy Best First Search (Show Manhattan)

Giải mê cung theo thuật toán A*:



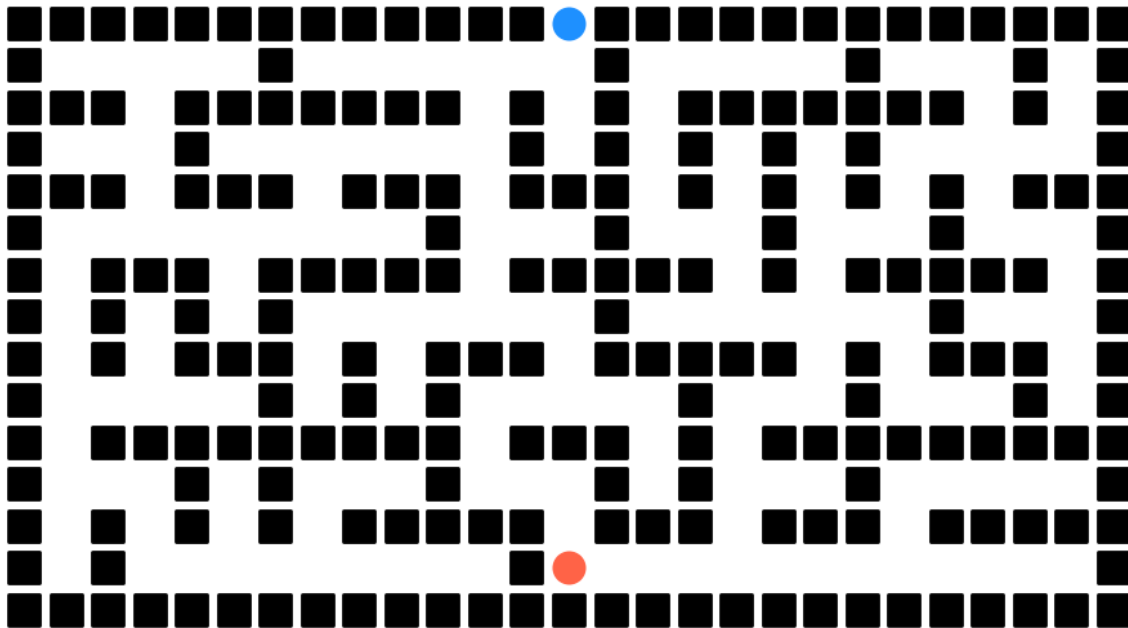
A*



A* (Show Manhattan + Cost)

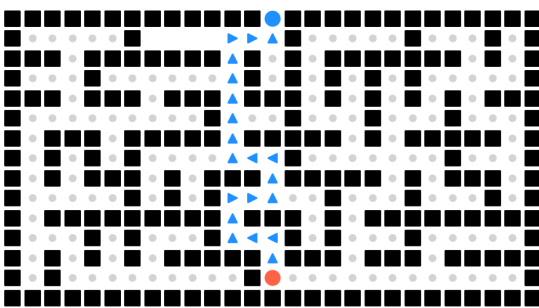
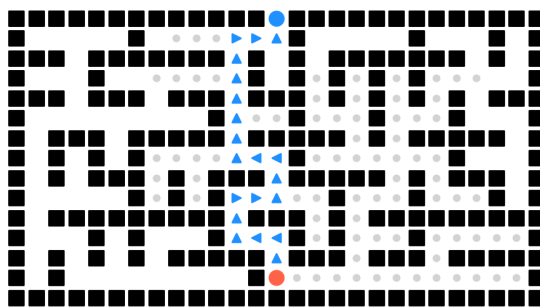
Mê cung 4

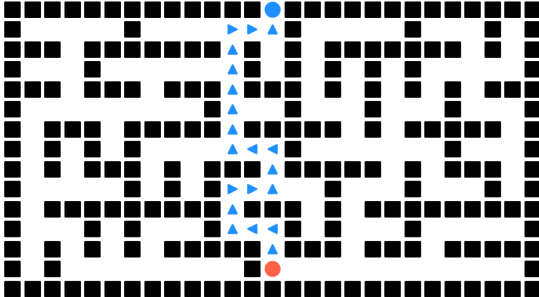
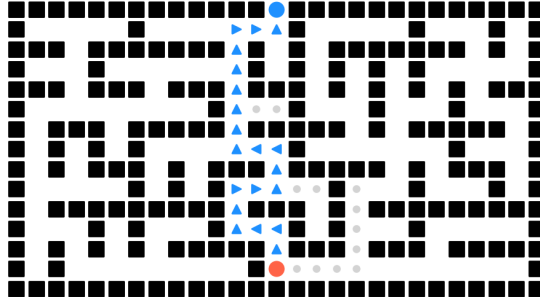
Chiều cao: 15, chiều rộng: 27



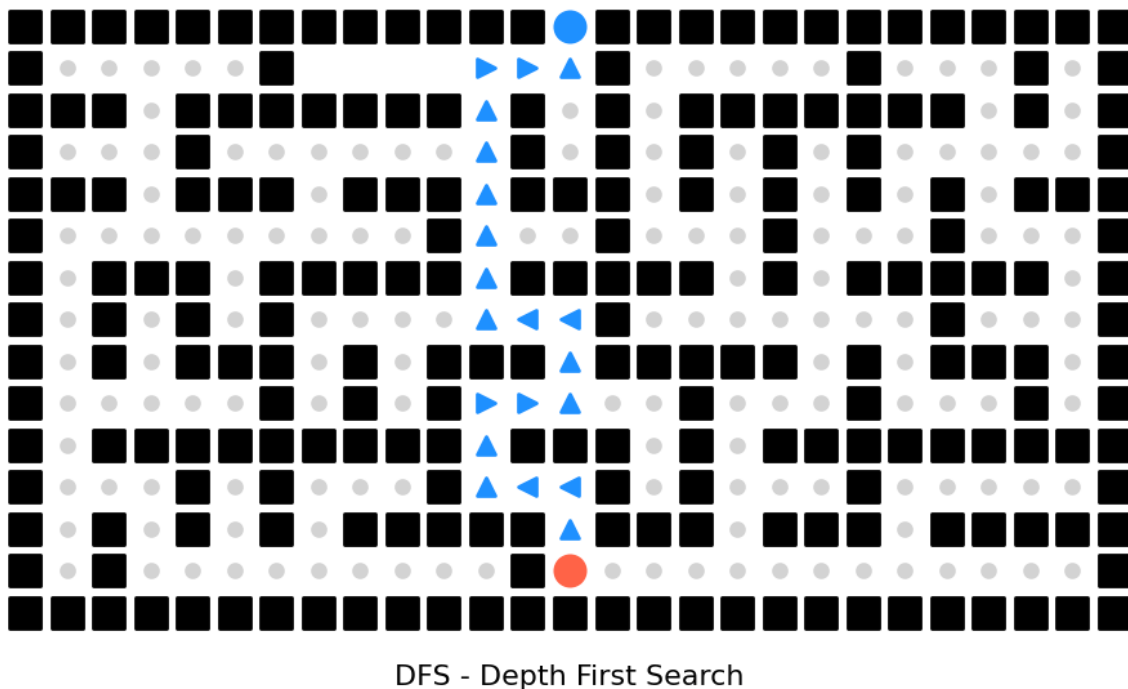
Maze

Tổng quan giải mê cung bằng 4 thuật toán:

Thuật toán DFS	Thuật toán BFS
 DFS - Depth First Search	 BFS - Breadth First Search
Tổng chi phí thực hiện: 177 Solution: 21	Tổng chi phí thực hiện: 98 Solution: 21 (Optimal solution)

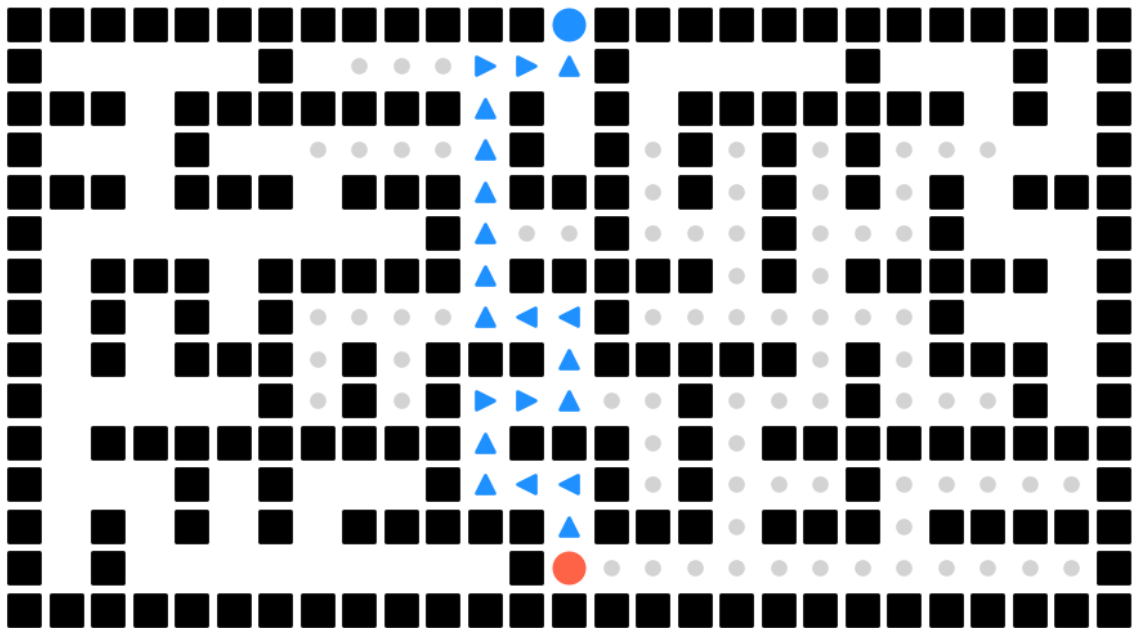
<p>Thuật toán GBFS</p>  <p>GBFS - Greedy Best First Search</p> <p>Tổng chi phí thực hiện: 21</p> <p>Solution: 21</p>	<p>Thuật toán A*</p>  <p>A*</p> <p>Tổng chi phí thực hiện: 33</p> <p>Solution: 21 (Optimal solution)</p>
---	--

Giải mê cung theo thuật toán DFS - Depth First Search:



Mặc dù tìm được đường đi ngắn nhất (như mọi khi, vẫn là do “may mắn”), tuy nhiên, ở mê cung này DFS đã tốn chi phí thực hiện rất lớn, thậm chí còn lớn hơn cả BFS!

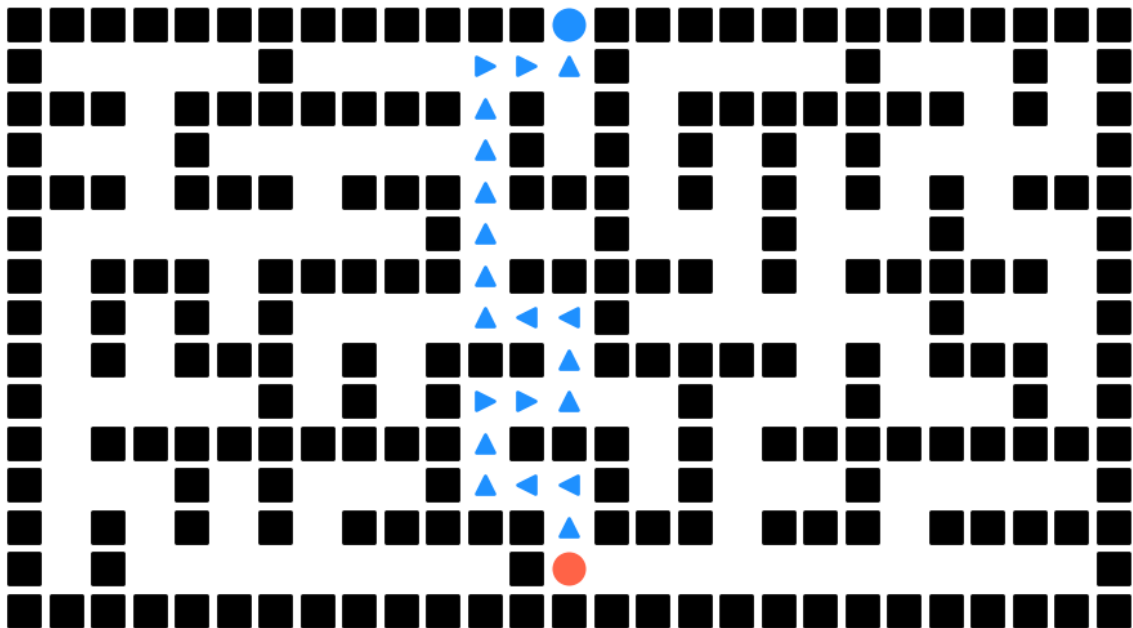
Giải mê cung theo thuật toán BFS - Breadth First Search:



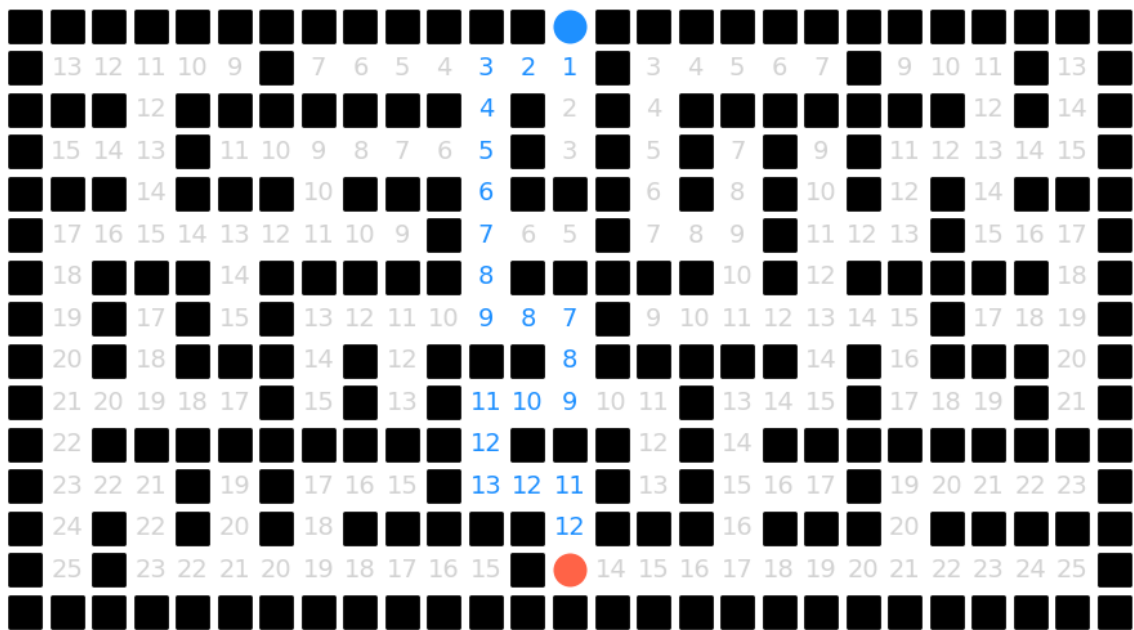
BFS - Breadth First Search

BFS vẫn tốn nhiều chi phí và tìm được đường đi ngắn nhất.

Giải mê cung theo thuật toán GBFS - Greedy Best First Search:



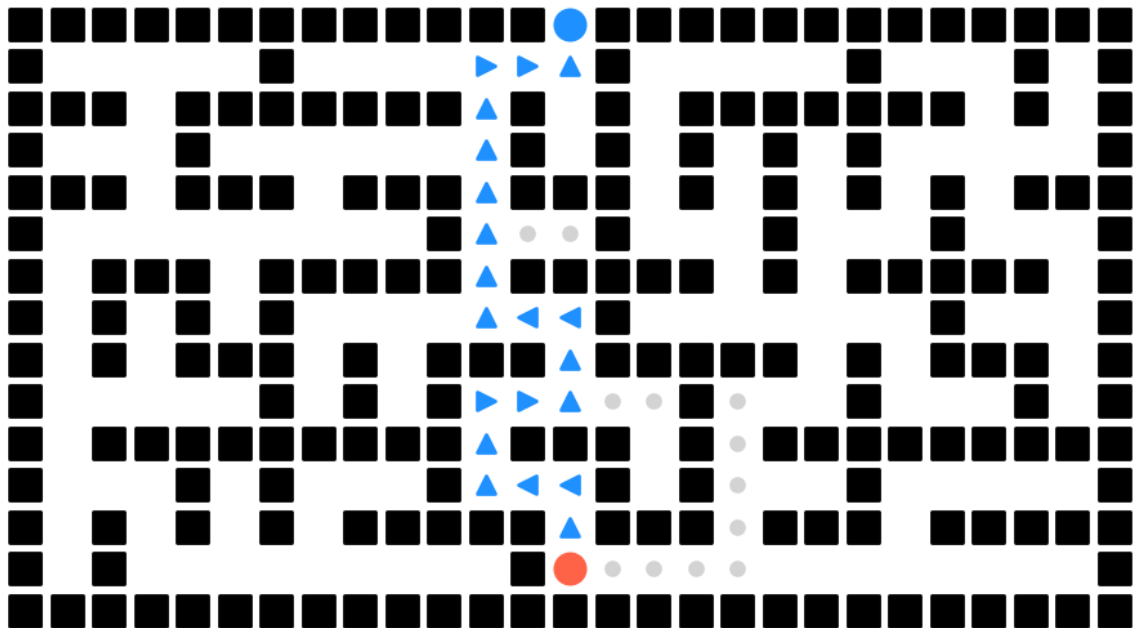
GBFS - Greedy Best First Search



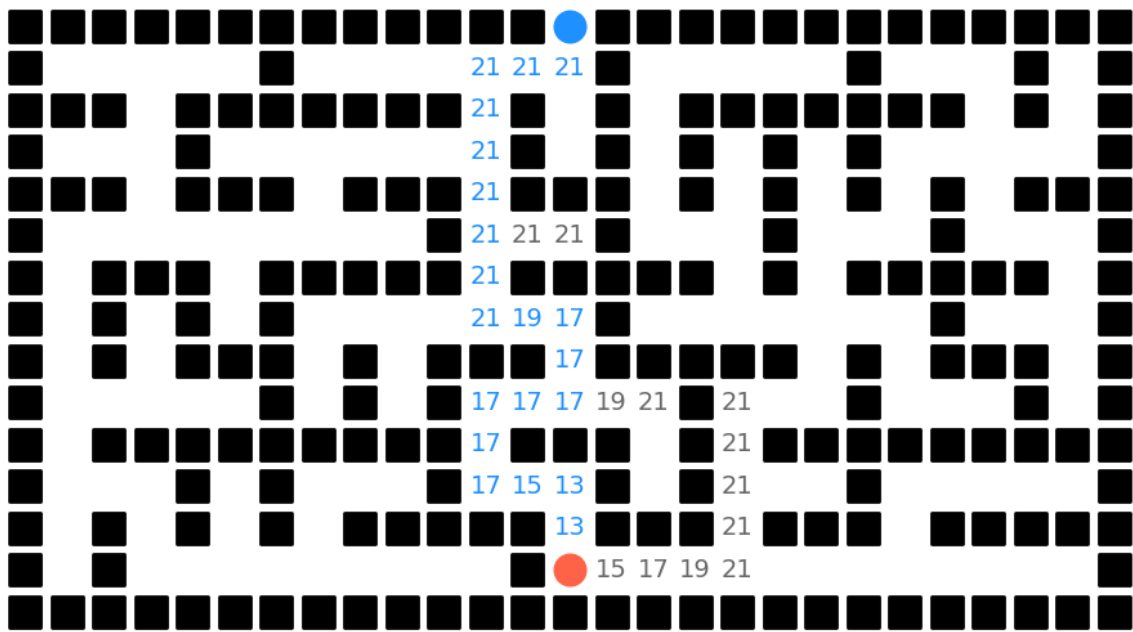
GBFS - Greedy Best First Search (Show Manhattan)

GBFS đi thẳng 1 đường tới đích trong mê cung này.

Giải mê cung theo thuật toán A*:



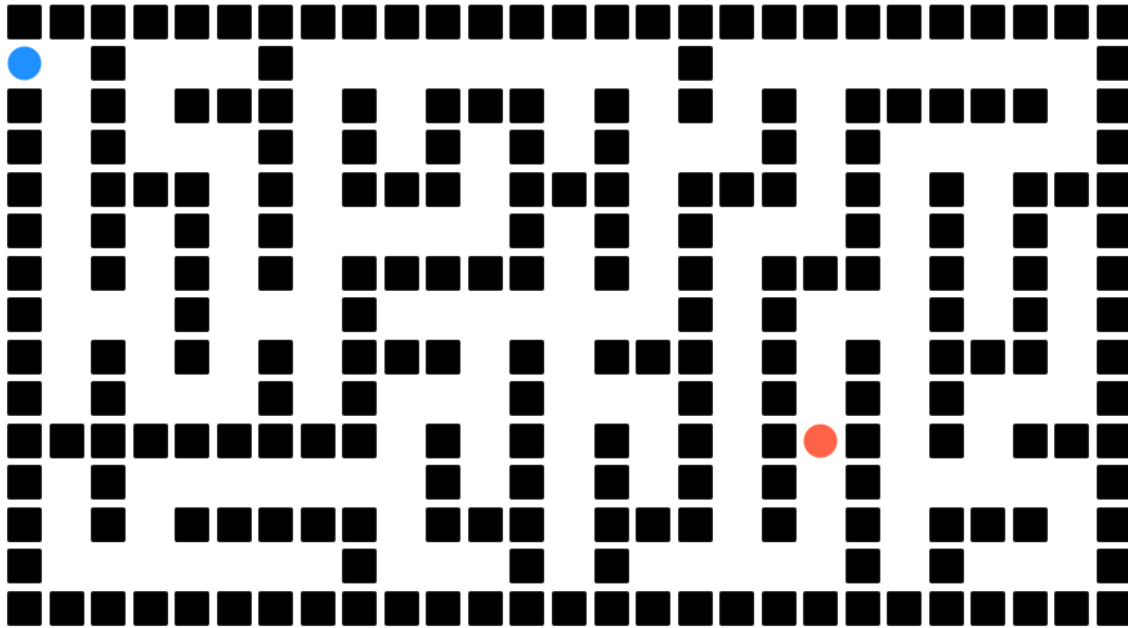
A*



A* (Show Manhattan + Cost)

Mê cung 5

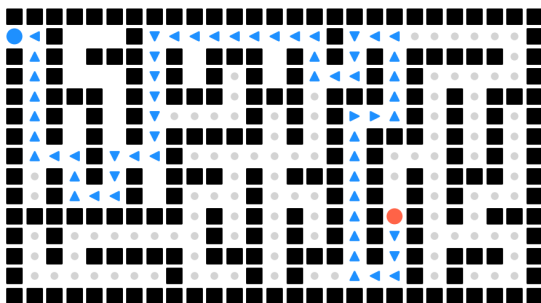
Chiều cao: 15, chiều rộng: 27



Maze

Tổng quan giải mê cung bằng 4 thuật toán:

Thuật toán DFS

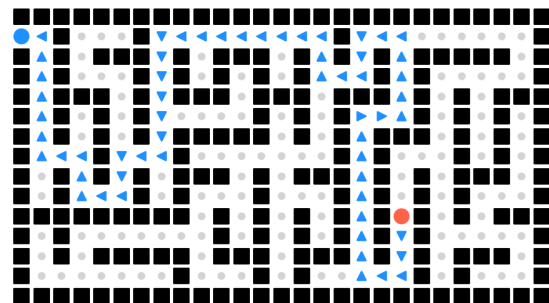


DFS - Depth First Search

Tổng chi phí thực hiện: 163

Solution: 58

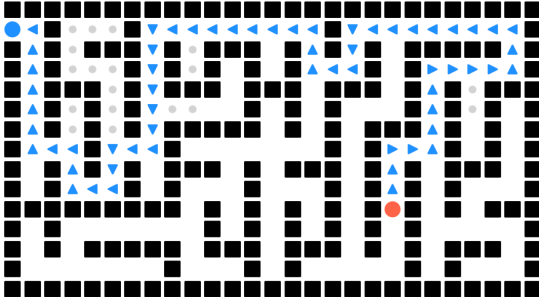
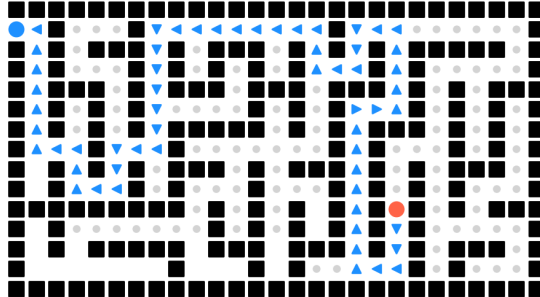
Thuật toán BFS



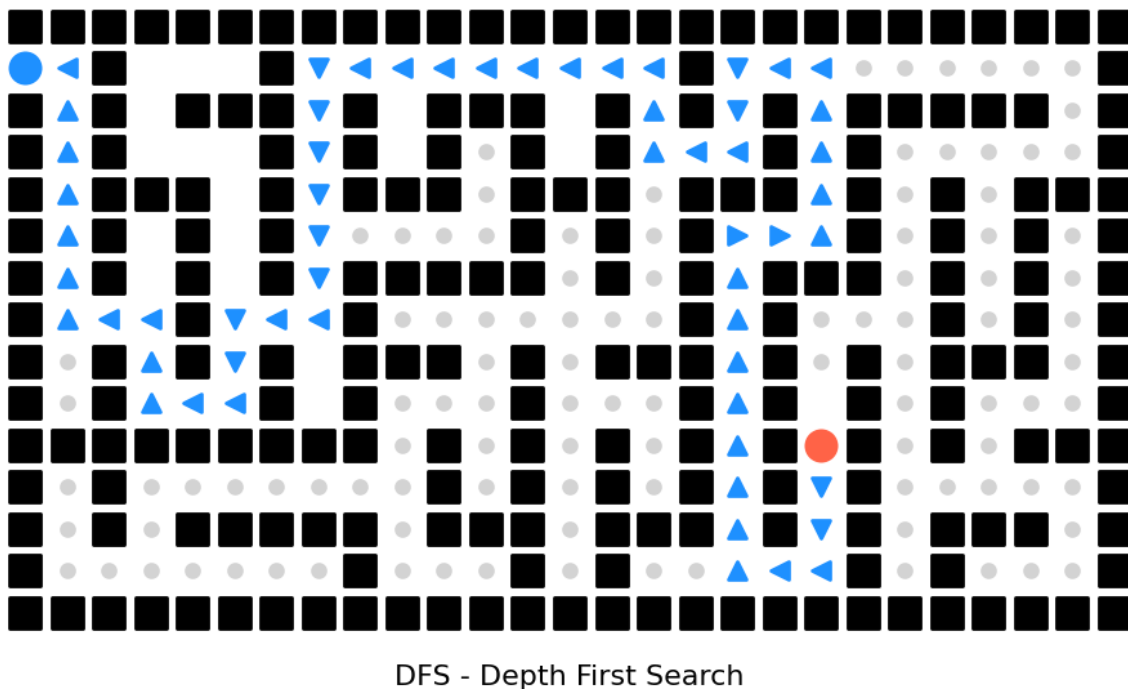
BFS - Breadth First Search

Tổng chi phí thực hiện: 182

Solution: 58 (Optimal solution)

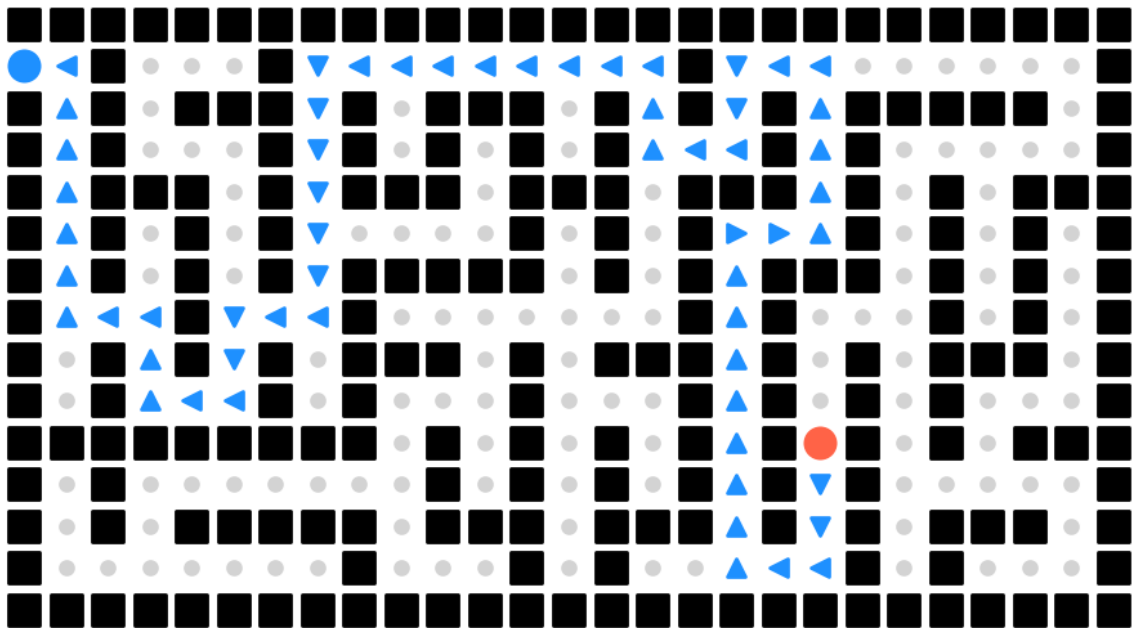
<p>Thuật toán GBFS</p>  <p>GBFS - Greedy Best First Search</p> <p>Tổng chi phí thực hiện: 78</p> <p>Solution: 60</p>	<p>Thuật toán A*</p>  <p>A*</p> <p>Tổng chi phí thực hiện: 162</p> <p>Solution: 58 (Optimal solution)</p>
---	---

Giải mê cung theo thuật toán DFS - Depth First Search:



Tuy không bằng BFS nhưng ở mê cung này, DFS vẫn tốn rất nhiều chi phí.

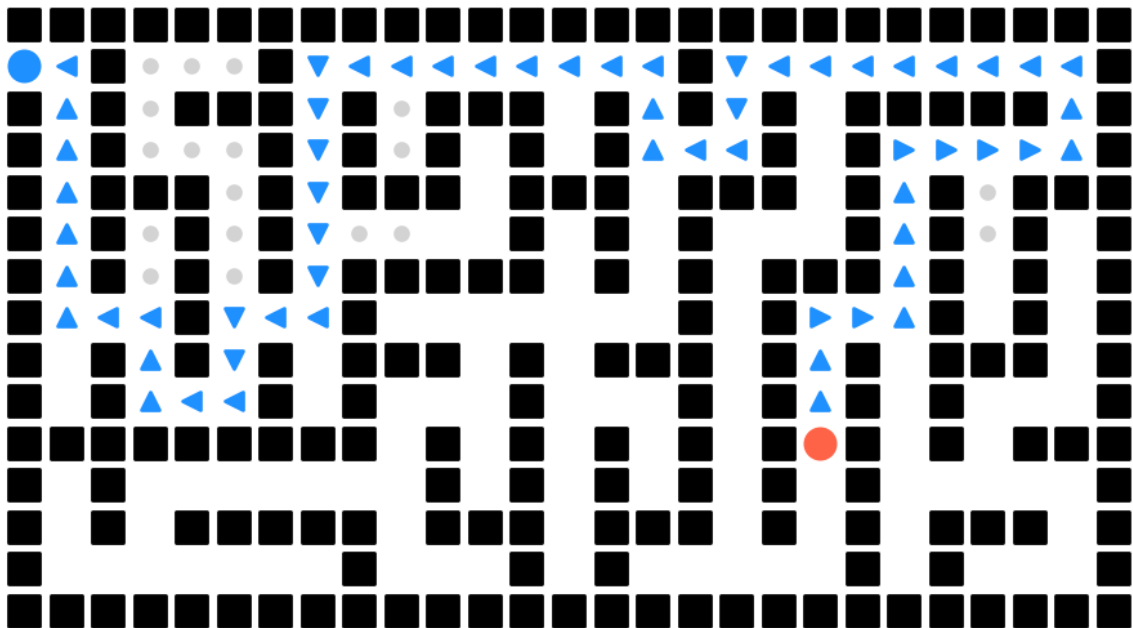
Giải mê cung theo thuật toán BFS - Breadth First Search:



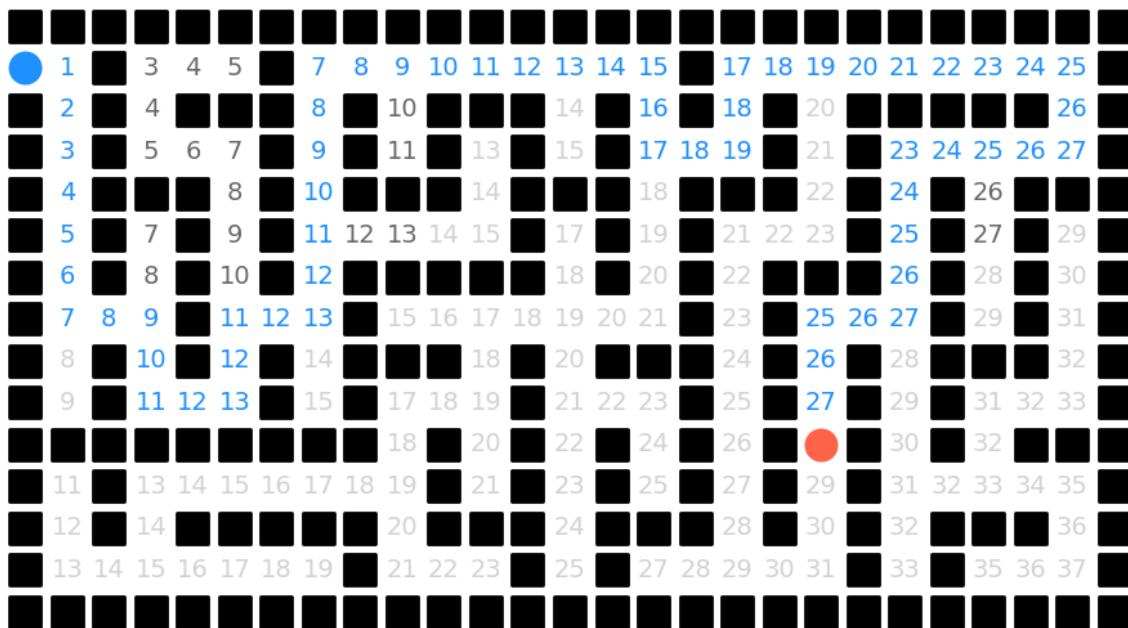
BFS - Breadth First Search

BFS đã “check in” tại mọi điểm có thể đi được của mê cung.

Giải mê cung theo thuật toán GBFS - Greedy Best First Search:



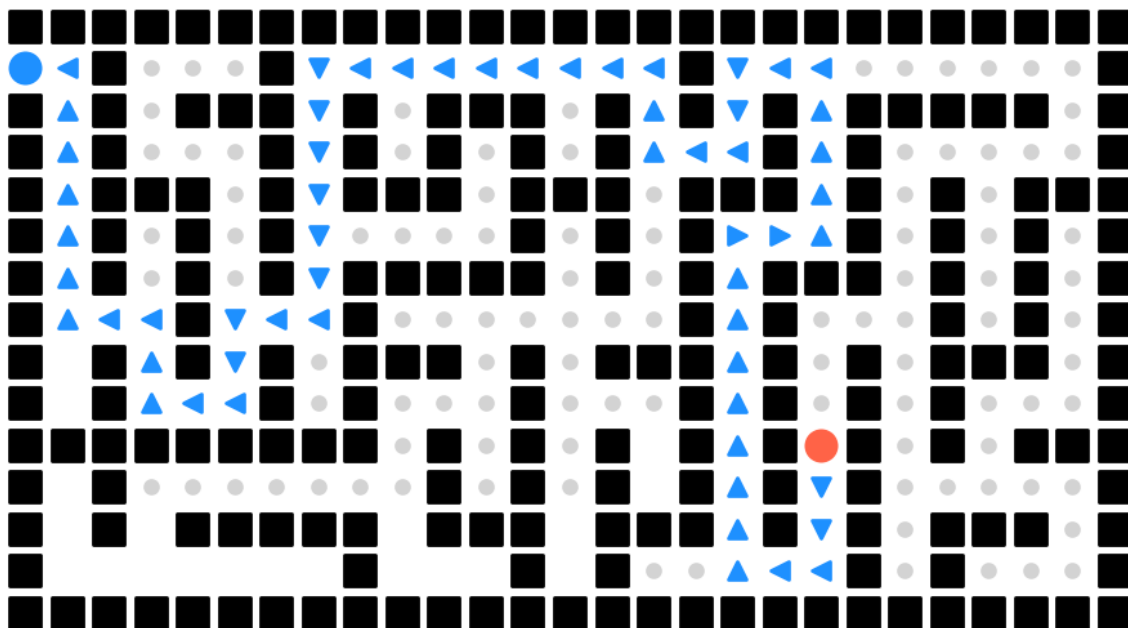
GBFS - Greedy Best First Search



GBFS - Greedy Best First Search (Show Manhattan)

Ở mê cung này, GBFS tốn khá ít chi phí, tuy nhiên, đường đi không ngắn nhất.

Giải mê cung theo thuật toán A*:



A*

Một số kết luận

Kết luận được áp dụng cho cách cài đặt các thuật toán ở bài làm này, có thể sai khác ở một vài điểm với cách cài đặt hoặc hàm heuristic khác.

- BFS và A* luôn cho đường đi ngắn nhất, tuy nhiên A* sẽ tốn ít chi phí hơn.
- DFS và GBFS không đảm bảo cho kết quả là đường đi ngắn nhất, GBFS thường sẽ tốn ít chi phí hơn DFS.
- A* vs GBFS: A* đảm bảo đường đi ngắn hơn (nhất), nhưng tốn nhiều chi phí hơn; GBFS lại thường có chi phí ít hơn (nếu không cố tình bẫy) nhưng không đảm bảo đường đi ngắn nhất.

Bản đồ có điểm thưởng:

Ý tưởng:

- **Ý tưởng 1:** Tưởng tượng điểm thưởng như những “ổ gà”, điểm thưởng càng cao thì “ổ gà” càng rộng. Agent chọn sử dụng thuật toán A*, tìm đường đi đến đích như bình thường, tuy nhiên sẽ có thay đổi nhỏ: Nếu agent đi vào ô thuộc “ổ gà” thì sẽ ưu tiên đi ăn điểm thưởng trước, sau đó mới tiếp tục tìm đường tới đích.
- **Ý tưởng 2:** Agent sử dụng thuật toán A*, trên đường tới đích nếu có 1 điểm thưởng “gần đó” thì sẽ “tiện đường” tới ăn điểm thưởng. “Gần” bao nhiêu và điểm thưởng lớn tới mức nào thì agent sẽ ăn do người cài đặt xác định. (Ví dụ: Khoảng cách Manhattan từ vị trí đang xét tới điểm thưởng < 5 và giá trị điểm thưởng > 12 thì agent sẽ đi ăn điểm thưởng đó chẳng hạn.)
- **Ý tưởng 3** (Đảm bảo tối ưu): Vét cạn. Cho agent lần lượt thử hết tất cả các trường hợp: không ăn điểm thưởng, ăn 1 điểm thưởng A, ăn 1 điểm thưởng B, ăn 2 điểm thưởng A \rightarrow B, ăn n điểm thưởng,... rồi chọn ra trường hợp có kết quả tối ưu nhất. Mặc dù về lý thuyết thì đảm bảo tối ưu, tuy nhiên sẽ không khả thi nếu có quá nhiều điểm thưởng.

Hết