|  |  |  |
| --- | --- | --- |
| Họ và tên | MSSV | Công việc |
| Phạm Thành Đạt | 19120473 | Code BFS và DFS ở bản đồ không có điểm thưởng  Code thuật toán ở bản đồ có cổng dịch chuyển  Viết báo cáo ở các phần tương ứng |
| Đoàn Việt Nam | 19120297 | Code Greedy Best First Search và A\* ở bản đồ không có điểm thưởng  Code thuật toán ở bản đồ có điểm thưởng  Viết báo cáo ở các phần tương ứng |

**BÁO CÁO ĐỒ ÁN AI - TÌM KIẾM**

***Bảng phân công***

**A. Bản đồ không có điểm thưởng**

**1. Tìm kiếm không có thông tin**

***So sánh ý tưởng 2 thuật toán BFS và DFS:***

* BFS là thuật toán tìm kiếm duyệt theo chiều rộng trong bản đồ. Nó sử dụng cấu trúc hàng đợi (*queue)* để lưu lại các đỉnh đã được truy cập tới. Mỗi lần duyệt nó sẽ lưu vào queue các node có trạng thái chưa được duyệt qua. Các đỉnh được lưu vào sẽ được duyệt một cách tuần tự từng nút và tiếp đến duyệt các nút liền kề nó.
* Còn DFS là thuật toán tìm kiếm theo chiều sâu. DFS sử dụng ngăn xếp (*stack*) để lưu trữu các đỉnh đã được duyệt. Nó hoạt động theo cơ chế đệ quy khi khám phá dọc theo một đường đi (1 cạnh). DFS sẽ duyệt đến các node sâu nhất chưa được khám phá.

***Map 1:***

A picture containing shape

Description automatically generated

Đường đi khi duyệt theo BFS:

A picture containing calendar

Description automatically generated

Chi phí và số trạng thái đường đi khi duyệt theo BFS:



Đường đi khi duyệt theo DFS:

A picture containing calendar

Description automatically generated

Chi phí và số các ô đã được duyệt qua khi thực hiện DFS:



***Nhận xét:*** Ta nhận thấy 2 thuật toán đều tìm được đường đi có chi phí giống nhau. Tuy vậy thời gian thực hiên (số trạng thái các ô đã duyệt qua) của thu ật toán BFS là nhiều hơn so với DFS.

***Map 2:***

**Table

Description automatically generated with medium confidence**

Đường đi khi duyệt theo BFS

A picture containing shape

Description automatically generated

Chi phí đường đi và số ô đã duyệt qua



Đường đi khi duyệt qua DFS

A picture containing shape

Description automatically generated

Chi phí đường đi và tổng các ô đã được duyệt qua



***Nhận xét:*** Đường đi khi thực hiện 2 thuật toán đều là đường đi ngắn nhất. Thời gian thực hiện của DFS có lâu hơn so với BFS tuy nhiên không nhiều.

***Map 3:***

**A picture containing table

Description automatically generated**

Đường đi khi duyệt theo BFS:

A picture containing shape

Description automatically generated

Chi phí đường đi và số các ô đã duyệt qua



Đường đi khi duyệt theo DFS

Table

Description automatically generated with medium confidence

Chi phí đường đi và số các ô đã được duyệt qua

Text

Description automatically generated

***Nhận xét:*** 2 thuật toán đều tìm được đường đi ngắn nhất. Thời gian thực hiện DFS lâu hơn so với BFS tuy nhiên không đáng kể.

***Map 4:***

**A picture containing table

Description automatically generated**

Đường đi khi duyệt theo BFS

Scatter chart

Description automatically generated with medium confidence

Chi phí đường đi và tổng các ô đã duyệt qua



Đường đi khi duyệt theo DFS

A picture containing scatter chart

Description automatically generated

Chi phí đường đi và tổng các ô đã được duyệt qua



***Nhận xét:*** 2 thuật toán đều tìm được đường đi có chi phí nhỏ nhất, nhưng ở thuật toán BFS thì thời gian duyệt lâu hơn so với DFS đáng kể (gần gấp đôi số trạng thái được duyệt của DFS)

***Map 5 (map lớn)***

**Scatter chart

Description automatically generated**

Đường đi khi duyệt theo BFS:

Chart, scatter chart

Description automatically generated

Chi phí đường đi và số các ô đã duyệt qua



Đường đi khi duyệt theo DFS

A picture containing scatter chart

Description automatically generated

Chi phí đường đi và số các ô đã được duyệt qua

Graphical user interface, text

Description automatically generated

***Nhận xét:*** Ta nhận thấy ở bản đồ này chi phí đường đi khi duyệt theo BFS nhỏ hơn đáng kể so với chi phí đường đi khi duyệt theo DFS. Tuy nhiên số các trạng thái (các ô đã duyệt qua) ở thuật toán BFS lớn hơn so với khi duyệt DFS chứng tỏ thời gian duyệt của BFS là lâu hơn.

* **Nhận xét chung: Qua các bản đồ trên ta có thể thấy thuật toán BFS luôn tìm được đường đi tối ưu (ngắn nhất), còn ở DFS thì chưa chắc tìm được đường đi ngắn nhất. Tuy nhiên thời gian duyệt của BFS sẽ lâu hơn đáng kể so với khi duyệt theo DFS. Điều này chứng tỏ đúng tính chất của mỗi thuật toán trên.**

**2. Tìm kiếm có thông tin**

***So sánh 2 thuật toán Greedy Best First Search với A\*:***

Greedy Best First Search tìm kiếm theo hàm heuristic (Khoảng cách từ node này đến đích). Ở mỗi lần di chuyển ta sẽ thêm vào priority queue các node con liền kề mà có trạng thái chưa được đi qua. Priority queue được xếp theo thứ tự ưu tiên cao nhất là những node có heuristic nhỏ nhất( tức là những node chúng ta ước tính gần đích nhất). Rồi cứ tiếp tục lấy ra các node rồi lại thêm node con liền kề như vậy cho đến khi chúng ta lấy được node đích( Exit).

Còn A\* chỉ khác GBFS ở chỗ không chỉ lấy hàm heuristic mà chúng ta còn cộng cả chi phí từ điểm xuất phát đến node đang xét.

***Map 1:***

A picture containing scatter chart

Description automatically generated

Khi thực hiện thuật toán GBFS, ta được đường đi:

A picture containing scatter chart

Description automatically generated

Chi phí đường đi và tổng số trạng thái phải xét của thuật toán GBFS:



Khi thực hiện thuật toán A\*, ta được đường đi:

Scatter chart

Description automatically generated with medium confidence

Chi phí đường đi và tổng số trạng thái phải xét của thuật toán A\*:



***Nhận xét:*** Ở bản đồ này ta thấy chi phí đường đi giống nhau và số trạng thái phải xét cũng xấp xỉ nhau nên 2 thuật toán ở trường hợp này không có nhiều sự khác biệt.

***Map 2:***

Ảnh có chứa bàn

Mô tả được tạo tự động

Khi ta chạy thuật toán Greedy Best First Search, ta được đường đi như sau:

Ảnh có chứa bàn

Mô tả được tạo tự động

Chi phí đường đi và tổng số trạng thái phải xét:



Khi chạy A\*, ta được đường đi:

Ảnh có chứa bàn

Mô tả được tạo tự động

Chi phí đường đi và tổng số trạng thái phải xét:



***Nhận xét:*** Nhìn chung ở map này cả 2 thuật toán đều tìm được đường đi ngắn nhất giống nhau. Còn thời gian thực hiện thì A\* có vẻ nhiều hơn 1 chút nhưng không đáng kể.

***Map 3:***

Ảnh có chứa bàn

Mô tả được tạo tự động

Khi thực hiện GBFS, ta được đường đi:

Ảnh có chứa bàn

Mô tả được tạo tự động

Chi phí đường đi và tổng số trạng thái phải xét:



Khi thực hiện A\*, ta được đường đi:

Ảnh có chứa bàn

Mô tả được tạo tự động

Chi phí đường đi và tổng số trạng thái phải xét:



Ở bản đồ này ta thấy thuật toán A\* tối ưu hơn 1 cách rõ rệt khi chi phí đường đi giảm gần 1 nửa so với GBFS và chi phí thời gian cũng nhỏ hơn.

***Map 4:***

Ảnh có chứa bàn

Mô tả được tạo tự động

Khi chạy GBFS, ta được đường đi:

A picture containing scatter chart

Description automatically generated

Chi phí và tổng số trạng thái phải xét:



Khi chạy A\*, ta được đường đi:

Ảnh có chứa bàn

Mô tả được tạo tự động

Chi phí và tổng số trạng thái phải xét:



***Nhận xét:*** Đối với bản đồ này, thuật toán A\* vẫn tìm được đường đi tối ưu hơn so với GBFS.

***Map 5 (Map lớn):***

Ảnh có chứa bàn

Mô tả được tạo tự động

Khi thực hiện GBFS, ta được đường đi:

Ảnh có chứa văn bản, người, ảnh chụp màn hình

Mô tả được tạo tự động

Chi phí đường đi và tổng trạng thái phải xét của GBFS là:



Khi thực hiện A\*, ta được đường đi:

Ảnh có chứa văn bản, người, ảnh chụp màn hình

Mô tả được tạo tự động

Chi phí đường đi và tổng số trạng thái phải xét là:



***Nhận xét:*** Ở bản đồ này ta thấy thuật toán A\* vẫn tìm được đường đi ngắn hơn nhưng mà số các trạng thái phải xét lại nhiều hơn đáng kể (gấp đôi so với GBFS).

**🡪*Nhận xét chung:***

Ta thấy thuật toán A\* là 1 thuật toán tối ưu về tìm kiếm đường đi còn thuật toán GBFS lại quá phụ thuộc vào hàm heuristics. Mà khi hàm heuristics không tốt (ước tính không chuẩn) sẽ dẫn đến những đường đi không tối ưu.

**B. BẢN ĐỒ CÓ ĐIỂM THƯỞNG**

Mô tả thuật toán: Ở đây em sử dụng thuật toán UCS, tức là ở mỗi lần di chuyển em sẽ thêm vào các node con lân cận mà có trạng thái chưa được đi qua bỏ vào priority queue. Các node con được sắp xếp trong priority queue theo trọng số được tính bởi hàm g(n) (chi phí đến node đó) cộng cho điểm thưởng nếu node đó có điểm thưởng với độ ưu tiên càng cao cho những node có trọng số càng nhỏ. Cứ lần lượt lấy ra các node rồi lại thêm vào các node liền kề như vậy cho đến khi lấy được node đích.

***Map 1:***

Scatter chart

Description automatically generated



***Map 2:***

A picture containing scatter chart

Description automatically generated



***Nhận xét:*** ở bản đồ này nếu chúng ta không ăn điểm thưởng thì chúng ta sẽ có đường đi ngắn nhất theo thuật toán A\* ở phần trước có chi phí là 20. Trong khi ở thuật toán này ta đã cài đặt thành công thuật toán để ăn được điểm thưởng giúp chi phí thấp hơn ( chỉ còn 14).

***Map 3( Map lớn):***

Ảnh có chứa bàn

Mô tả được tạo tự động



***Nhận xét:*** Ở bản đồ này ta cũng thấy chi phí đi là thấp hơn so với chi phí đi không qua điểm thưởng của thuật toán A\* ở phần trước (ở phần trước chi phí là 46)

***Nhận xét chung:*** Nhìn chung thuật toán đã biết tìm đường đi ngắn nhất bằng cách ăn điểm thưởng. Nhưng sẽ có những trường hợp vì thuật toán này được cài đặt không đi lại những node có trạng thái đã đi rồi nên có những trường hợp nó sẽ không lấy điểm thưởng khi chỉ có 1 đường vào điểm thưởng nhưng không có đường ra mặc dù nếu lấy điểm thưởng thì chúng ta sẽ lời chi phí đường đi.

**C. BẢN ĐỒ CÓ CỔNG DỊCH CHUYỂN**

*Mô tả bản đồ:* Trong bản đồ sẽ có các thêm các ô đóng vai trò là cổng dịch chuyển, mỗi cổng dịch chuyển sẽ có ô vào (entrance) và ô đi ra (exit) tương ứng của cổng đó.

*Ý tưởng:* Ở đây em sẽ duyệt tìm đường đi theo BFS vì thuật toán này luôn tối ưu được đường đi tỏng bản đồ. Khi gặp mỗi cổng dịch chuyển thì em sẽ thay đổi tọa độ điểm tại đó thành tọa độ của lối ra cổng dịch chuyển đó và tiếp tục BFS như cũ cho đến khi tìm đường đường ra khỏi bản đồ.

***Map 1:***

Scatter chart

Description automatically generated Graphical user interface, text

Description automatically generated

***Map 2:***

A picture containing calendar

Description automatically generated

Graphical user interface, text, application

Description automatically generated

***Map 3 (map lớn):***

Background pattern

Description automatically generated

Text

Description automatically generated

* ***Nhận xét chung:*** Ta có thể thấy thuật toán sẽ được tối ưu hơn về chi phí đường đi khi tìm được các cổng dịch chuyển thích hợp.