**TÌM ĐƯỜNG ĐI NGẮN NHÂT GIỮA HAI ĐIỂM TRONG MÊ CUNG**

***GVHD: Huỳnh Xuân Phụng***

**Nguyễn Văn Hà**

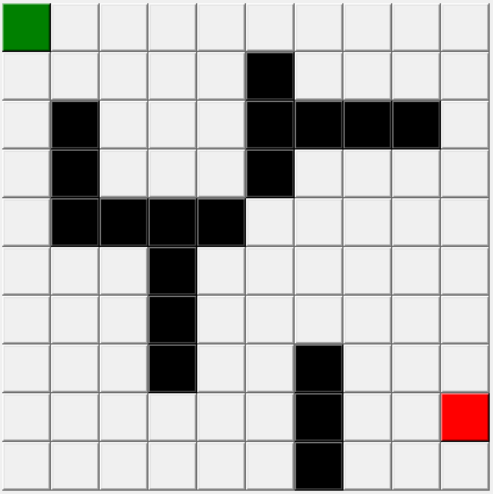
*1 Trường đại học Sư phạm Kỹ thuật TP.HCM*

**Tóm tắt:**

Thông thường để giúp xe dò được đường trong mê cung, ta có nhiều thuật toán khác khau. **Phổ biến** và **đơn giản nhất** có lẽ là thuật toán tìm đường ngẫu nhiên và **thuật toán bám tường**. Với thuật toán tìm đường ngẫu nhiên, xe bạn sẽ mò mẫm trong mê cung theo kiểu "ngẫu nhiên", may mắn thì sẽ tìm ra nhanh, còn không thì có thể không tìm ra được.

Các phương pháp kể trên có một nhược điểm, đó là quãng đường thu được sau khi đến đích thường rất dài do trước đó gặp phải khá nhiều ngõ cụt. Thế nên sau khi có được đường đi, ta cần phải tối ưu nó để ra được cách di chuyển ngắn nhất.

1. **Mô tả**



Mê cung có tất cả 10 hàng và 10 cột, chi phí di chuyển từ 1 ô sang một ô kế bên là 1. Những ô màu đen là những vật cản, ô màu xanh là điểm bắt đầu di chuyển, ô màu đỏ là đích mà mình cần phải đi đến. Có thể đi sang trái, sang phải, đi lên trên, đi xuống dưới, đi chéo lên trên bên phải, đi chéo lên trên bên trái, đi chéo xuống dưới bên phải, đi chéo xuống dưới bên trái và không thể đi ra ngoài mê cung, không thể đi lên các vật cản mà phải tìm một con đường khác. Như vậy ở một vị trí tối ưu thì từ một ô có thể di chuyển đến 8 ô xung quanh. Mê cung sẽ được mô phỏng lại bằng một mảng 2 chiều với 10 dòng và 10 cột. Phần tử có giá trị 0 thể hiện là vật cản, phần tử có giá trị 1 thể hiện là ô trống có thể di chuyển.

1. **Thuật toán**

Thuật toán A\* (đọc là A sao) là một [thuật toán tìm kiếm trong đồ thị](https://vi.wikipedia.org/wiki/Duy%E1%BB%87t_c%C3%A2y). Thuật toán này tìm một đường đi từ một [nút](https://vi.wikipedia.org/w/index.php?title=N%C3%BAt_(l%C3%BD_thuy%E1%BA%BFt_%C4%91%E1%BB%93_th%E1%BB%8B)&action=edit&redlink=1) khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này. Do đó, thuật toán A\* là một ví dụ của [tìm kiếm theo lựa chọn tốt nhất](https://vi.wikipedia.org/wiki/T%C3%ACm_ki%E1%BA%BFm_theo_l%E1%BB%B1a_ch%E1%BB%8Dn_t%E1%BB%91t_nh%E1%BA%A5t) (best-first search).

1. **Mã giả**

Initialize the open list

Initialize the closed list put the starting node on the open list (you can leave its f at zero).

While the open list í not empty:

Find the node with the least f on the open list, call it “q”

Pop q off the open list

Generate q’s 8 successors and set their parents to q

For each succesor:

If successor is the goal, stop search successor.g = q.g + distance between successor and q

Successor.h = distance from goal to successor (this can be done ussing many ways, we will discuss three heuristic – Manhattan, Diagonal and Euclidean Heuristic).

Successor.f = successor.g + successor.h;

If a node with the same position as successor is in the open list which has a lower f than successor, skip this successor.

If a node with the same position as successor is in the closed list which has a lower f than successor, skip this successor otherwise, add the node to the open list

End (for loop).

Push q on the closed list

End (while loop).

1. **Độ phức tạp: O(b^m)**
2. **Ví dụ**

|  |  |
| --- | --- |
| **Đầu vào** | **Đầu ra** |
| int A[R][C] =  {  { 1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },  { 1, 1, 1, 0, 1, 1, 1, 0, 1, 1 },  { 1, 1, 1, 0, 1, 1, 0, 1, 0, 1 },  { 0, 0, 1, 0, 1, 0, 0, 0, 0, 1 },  { 1, 1, 1, 0, 1, 1, 1, 0, 1, 0 },  { 1, 0, 1, 1, 1, 1, 0, 1, 0, 0 },  { 1, 0, 0, 0, 0, 1, 0, 0, 0, 1 },  { 1, 0, 1, 1, 1, 1, 0, 1, 1, 1 },  { 1, 1, 1, 0, 0, 0, 1, 0, 0, 1 },  { 1, 1, 1, 0, 0, 0, 1, 0, 0, 1 }  }; | Duong di: -> (0, 0)-> (1, 1)-> (2, 2)-> (3, 2)-> (4, 2)-> (5, 3)-> (5, 4)-> (6, 5)-> (7, 5)-> (8, 6)-> (7, 7)-> (7, 8)-> (8, 9)-> (9, 9) |
| Nhap hang cua nguon: 0 |
| Nhap cot cua nguon: 0 |
| Nhap hang cua dich: 9 |
| Nhap cot cua dich: 9 |

Ma trận hai chiều với 10 hàng và 10 cột mô phỏng lại mê cung của chúng ta. Chương trình cho phép chúng ta nhập vị trí bắt đầu và vị trí đích cần tìm (ở đây vị trí bắt đầu là ô đầu tiên bên góc trái trên và vị trí đích là ô dưới cùng bên phải). Sau đó thuật toán sẽ tìm đường đi ngắn nhất và nếu tìm được đích sẽ hiển thị thông báo đã tìm thấy đích và in ra đường đi từ vị trí bắt đầu đến đích.

1. **Kết luận**

Với thuật toán A\* có sử dụng hàm heuristic ta đã có thể tìm được đường đi ngắn nhất mà không va chạm phải vật cản, nó có thể được áp dụng rộng rãi trong lĩnh vực giao thông, khai thác,… rất có ích cho cuộc sống con người. Tuy nhiên cũng có những trường hợp mà thuật toán chưa thể tìm ra được, đó còn là hạn chế mà thuật toán chưa thể giải quyết được.

**Tài liệu tham khảo**

1. <http://arduino.vn/bai-viet/5553-gioi-thieu-thuat-toan-tim-duong-di-ngan-nhat-trong-me-cung>
2. <https://simplecodecjava.blogspot.com/2015/08/bai-toan-tim-uong-i-trong-ma-cung.html>
3. [https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i\_thu%E1%BA%ADt\_t%C3%ACm\_ki%E1%BA%BFm\_A\*](https://vi.wikipedia.org/wiki/Gi%E1%BA%A3i_thu%E1%BA%ADt_t%C3%ACm_ki%E1%BA%BFm_A*)
4. <https://www.geeksforgeeks.org/a-search-algorithm/>