ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

KHOA Toán - Tin Học

~~~~~~*~~~~~~

BÁO CÁO THỰC HÀNH NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Sinh viên	**:**	**VÒNG VĨNH PHÚ**
MSSV	**:**	**19110413**
Môn Học	**:**	**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**
Trường	**:**	**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 12 NĂM 2021
~~~~~~

Nội dung

1. Ý TƯỞNG BÀI TOÁN

Đề bài : The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without

ever leaving a group of missionaries in one place outnumbered by the cannibals in

that place. This problem is famous in AI because it was the subject of the first paper

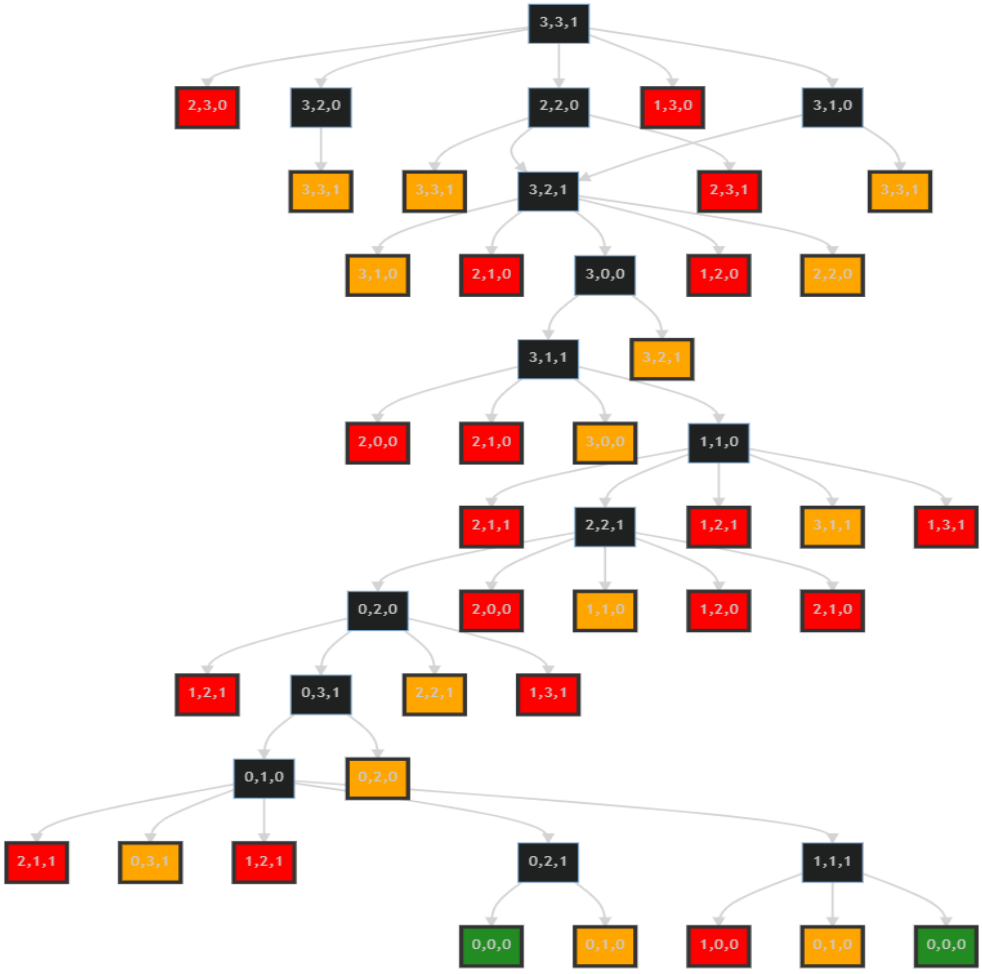
that approached problem formulation from ananalytical viewpoint (Amarel, 1968).
a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

b. Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

1.1 Mục tiêu bài toán

- Tìm kiếm giải pháp tốt nhất cho bài toán The missionaries and Cannibals problem
- Start : [3,3,1] Với Start[0] : là số $0 \leq$ missionaries ≤ 3; Start[1] : là số $0 \leq$ Cannibals ≤ 3; Start[2] : là 1 nếu thuyền ở bở trái và 0 thuyền là bờ phải
- Goal : [0,0,0] : Tất cả missionaries và Cannibals và thuyền ở bờ phải

1.2 Các đường đi được



Hình 1 Đồ thị các khả năng tìm ra lời giải cho The missionaries and Cannibals problem

- Theo đồ thị ta có các Node đen là các giá trị đường đi dẫn tới giải pháp, Node đỏ là node missionaries < Cannibals ở 1 trong 2 bên bờ hay nói cách khác là không thể đi tiếp (Thua), Node vàng là node trở lại node đã từng đi (lặp lại) về mô hình chung nó cũng không dẫn đến giải pháp cuối cùng khi nó sẽ tạo ra các vòng lặp nếu ta sử dụng thuật toán DFS

1.3 Giải pháp

- Với mô hình không có chi phí ta sẽ sử dụng BFS (DFS) vì bài toán có vòng lặp và độ sâu là 12 nên ta sẽ cần phải tối ưu Graph để sử dụng BFS/DFS hiệu quả

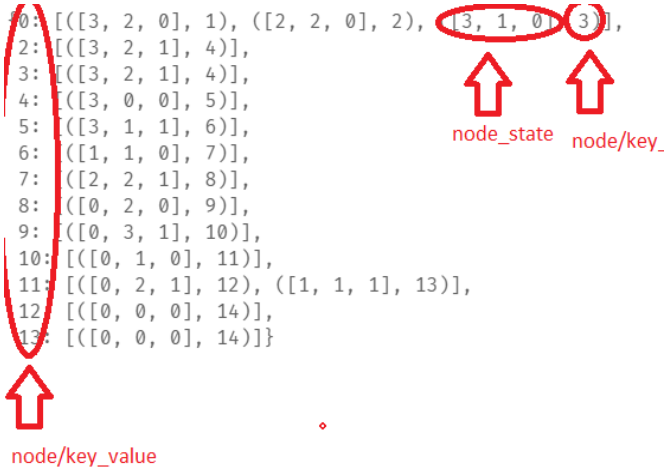
2. Xây Dựng node và graph

2.1 Ý tưởng

- Ta sẽ xây dựng các node(Graph) như đồ thị trên bằng cách loại bỏ các ô không thể đi được (Vàng và Đỏ) và chỉ lưu các đường dẫn tới solution bằng cách này ta sẽ cho thuật toán DFS không tạo ra vòng lặp tìm kiếm vô hạn (có thể xây dựng một list cho biết các node DFS đã duyệt qua để tránh việc này) và BFS không cần phải duyệt tất cả các node nếu gặp node vàng và độ sâu node là 12 (giảm thiểu được độ phức tạp không gian thời gian). Nhưng để chắc chắn tìm ra kết quả đáp án thì trong bài ta sẽ sử dụng BFS

2.2 Xây dựng node và graph

- Để bài toán trở nên dễ hình dung ta sẽ xây dựng một biến Graph lữu trữ các thông tin đường đi cụ thể ta sẽ sử dụng Class Dictionary của python để lưu trữ thông tin các node có thể tìm thấy lời giải bài toán bao gồm node_state: [3,3,1], [2,3,1], [3,2,0], ….; node: [1,2,3,4,5, …] trong đó node chính là key_value để dẫn tới value node_state



3. Breath First Search (BFS)

3.1 Ý tưởng thuật toán

Breath First Search là thuật toán sử dụng cấu trúc dữ liệu hàng đợi (Queue) để lưu trữ thông tin trung gian thu được trong quá trình tiếp kiếm.

- Từ 1 gốc ban đầu xác định và lần lượt duyệt các đỉnh liền kề xung quanh đỉnh gốc vừa xét
    - Tiếp tục quá trình duyệt qua các đỉnh kề vừa xét cho đến khi đạt được kết quả cần tìm hoặc duyệt qua các đỉnh

3.2 Input/Output

- Input: (Graph, Start, Goal)
    - Graph: Dictionary đã được tạo từ đồ thị Hình 1
    - Start: Nơi bắt đầu [3,3,1]
    - Goal: Nơi kết thúc [0,0,0]
- Output: Giải pháp đi từ Start đến Goal với đường đi ngắn nhất

3.3 Xây dựng Hàm và thuật toán

- Khởi tạo 1 hàm để tìm đường đi BFS(graph,start,goal)
    - Xây dựng 1 list queue cho phép lưu trữ thông tin vào
    - Ta sẽ truyền thông tin [Start],[0] vào trong queue để khởi tạo node đầu tiên với Start chính là node_state đầu tiên, và 0 là node đầu tiên
    - Nếu queue chưa rỗng thì ta sẽ cho thực hiện 1 vòng lặp để tìm kiếm, ngược lại ta sẽ xuất ra thông báo không có đường đi dẫn tới kết quả
        - Ta lấy phần tử đầu tiên trong queue cho list(Path),list(Path_state) để Path có node đầu tiên

- Tạo 2 biến node và node_state là phần tử cuối của path và path_state và thực hiện kiểm trả node_state để kiểm tra đã ở đích đến chưa nếu chưa thì tiếp tục tìm kiếm
- Tạo 1 vòng lặp để lấy thông tin các đỉnh liền kề cho biến adj (for adj in graph.get(node, []))
- Tạo 1 mảng explored để lấy thông tin mà path đã đi qua và thêm phần tử thông tin tìm kiếm liền kề (adj)
- Truyền explored này vào queue để tạo 1 danh sách các đường đi đã explored được từ các node liền kề
- Khi đã hết explored hết tất cả các đỉnh liền kề queue sẽ trả về cho path danh sách các đường đi và thực hiện như các bước trên đến khi node_state == goal (nếu danh sách path_state nào được đến được goal đầu tiên sẽ trả về cho hàm path_state đó)

- Kết thúc thuật toán