PROJECT 1 – SEARCH

Trường Đại học Khoa học Tự nhiên TP.HCM

Cơ sở Trí tuệ nhân tạo

TA. Hoàng Xuân Trường - hxtruong6@gmail.com

I. Mục tiêu đồ án

Nghiên cứu, cài đặt và trình bày các thuật toán tìm kiếm trên đồ thị.

II. Yêu cầu

Project được thực hiện theo cá nhân.

Thời gian và cách thức nộp, xem trên Moodle.

Nội dung cần nộp:

- Báo cáo trình bày trong file **pdf** chứa:

  - Thông tin sinh viên: họ tên, MSSV…
  - Mức độ hoàn thành của mỗi mức yêu cầu. Tự đánh giá đồ án trên thang điểm 10.
  - Trình bày lý thuyết cơ bản (ý tưởng, độ phức tạp, tính chất,...) của mỗi thuật toán cài đặt.
  - Trình bày điểm khác biệt giữa UCS và A*.
  - **Điểm cộng** nếu có thêm thuật toán Tìm kiếm khác ngoài 4 thuật toán BFS, DFS, UCS(Uniform Cost Search), A*.
  - Chi tiết mỗi thuật toán cài đặt:
    - Đưa ra các trường hợp(testcase) khác nhau của thuật toán, xét trường hợp đặc biệt (nếu có).
    - Có hình minh họa cho mỗi testcase
    - Nhận xét mỗi thuật toán. Đối với thuật toán A*, phải đưa ra heuristic, giải thích và nhận xét tại sao chọn hàm heuristic đó.

- Source code

- Video quay lại màn hình cho mỗi thuật toán của một testcase đã cài đặt được. Ví dụ: testcase 1 có ít nhất 4 video bfs, dfs, ucs, a_star.

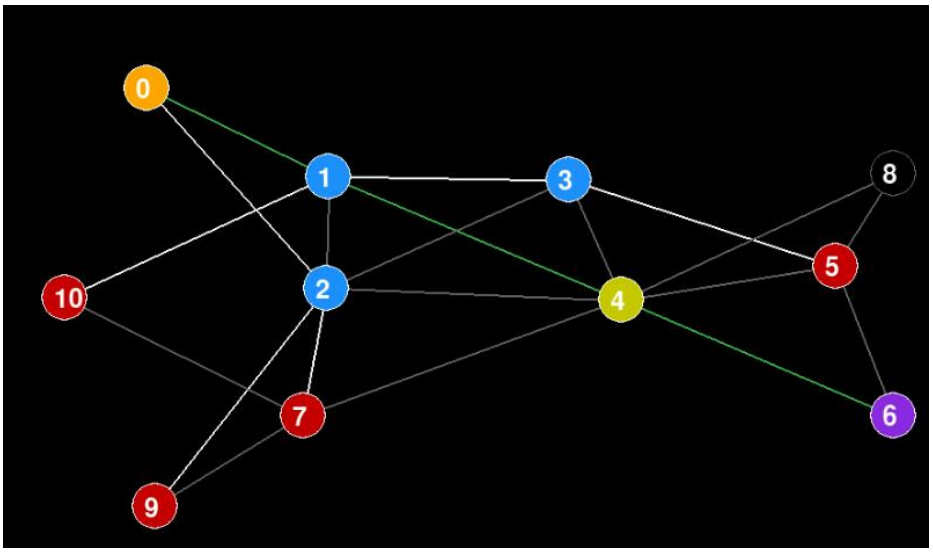
Cây thư mục sẽ như sau:

MSSV.zip:

- MSSV.pdf
- Thư mục source_code
- Thư mục video:
  - testcase1.txt, tc1_bfs.mp4, tc1_dfs.mp4, tc1_a_star.mp4
  - …

Ngôn ngữ lập trình bắt buộc: Python

III. Nội dung



Cài đặt các thuật toán tìm kiếm trên đồ thị. Nhiệm vụ của bạn chỉ cần viết code của các hàm BFS, DFS, UCS, AStar đã được định nghĩa sẵn trong file **search_algorithm.py**. Không được sửa các hàm khác hay các file khác (có thể thêm hàm mới hoặc file mới nếu cần thiết).

Với mỗi thuật toán tìm kiếm đã được truyền tham số đầu vào gồm:

graph: thông tin dữ liệu của đồ thị. Bao gồm một danh sách cách object chưa thông tin của một định.

Ví dụ:

```
graph = [
        [(139, 140),      # vị trí của đỉnh khi vẽ ra màn hình
          [1, 2],        # danh sách các đỉnh kề với đỉnh 0
          (100, 100, 100),     # màu viền của đỉnh - xám
          (0, 0, 0)            # màu của đỉnh - đen
        ],
```

```
            [(312, 224), [0, 2, 3, 4], (100, 100, 100), (0, 0, 0)],

              ...

]
```

Ở đây, ta có `graph[0]` là tại đỉnh 0, `graph[1]` là tại đỉnh 1. Như ví dụ, ta có đỉnh 0 nối tới đỉnh 1 và đỉnh 2 (truy cập bằng cách `graph[0][1]`)

`edges`: danh sách cách cạnh của đồ thị. Ví dụ: `edges[edge_id(0,1)] = [(0,1),` (0,0,0)]. Nghĩa là gán cạnh từ đỉnh 0 tới đỉnh 1 có màu (0,0,0) là màu đen.

`edge_id:` mỗi cạnh đều có mội id riêng thể hiện một cạnh giữa hai đỉnh.

`start`: đỉnh xuất phát

`goal`: đỉnh cần tìm kiếm

Input: input.txt

```
0
3
0 1
0 2
2 3
```

Dòng 1 đỉnh xuất phát `start`

Dòng 2 đỉnh tìm kiếm `goal`

Các dòng tiếp theo là các cạnh của đồ thị thể hiện liên kết giữa hai đỉnh.

Muốn thêm testcase khác, chỉnh sửa hoặc thêm file mới (testcase.txt) có định dạng tương tự như đã trình bày.

Chạy chương trình : `python main.py input_file search_algorithm`

Đã trình bày trong file README.md để trong thư mục source_code.

Gợi ý: đọc code hàm `example_func()` trong file **search_algorithm.py** để hiểu cách thức tô màu đỉnh, cạnh của đồ thị.

Muốn cập nhập trạng thái mới trong lúc tìm kiếm gọi hàm:
`graphUI.updateUI()`

Yêu cầu

Viết code của mỗi thuật toán vào các hàm tương ứng đã được định nghĩa. Không được chỉnh sửa tham số đầu vào của các hàm (BFS, DFS, UCS, AStar)

Các màu thể hiện cho các trạng thái của đồ thị được ghi tại file **node_color.py** như sau:

- Màu xanh dương: đỉnh đã duyệt qua
- Màu đỏ: đỉnh vừa duyệt tới
- Màu đen: đỉnh chưa được duyệt
- Màu vàng: đỉnh hiện tại
- Màu tím: đỉnh đích
- Màu cam: đỉnh bắt đầu
- Màu xanh lá: cạnh của đường đi tìm kiếm được
- Màu xám: đỉnh hoặc cạnh chưa duyệt qua
- Màu trắng: cạnh đã duyệt qua hoặc viền của đỉnh

Sau khi thực hiện tìm kiếm xong cần phải thể hiện được đường đi đã tìm kiếm được từ đỉnh `Start` tới đỉnh `Goal`.

Xem video demo để hiểu rõ hơn.

Điểm cộng: Thêm thuật toán tìm kiếm mới

Tại dòng 65 của file **graphUI.py,** chỉnh sửa tên thuật toán bạn viết trùng với tham số truyền vào. Và thêm một hàm tương ứng ở file **search_algorithm.py.**

IV. Đánh giá

- Theo mức độ hoàn thành đồ án
- Mỗi thuật toán chạy thử ít nhất 3 testcase khác nhau.
- Testcase nào không có báo cáo sẽ không được chấm
- Báo cáo/mã nguồn có tham khảo cần phải ghi nguồn rõ ràng ở cuối báo cáo.
- **Bài giống nhau sẽ 0 điểm môn học**

V. Liên hệ

Mọi thắc mắc trong quá trình thực hiện vui lòng gửi mail về hxtruong6@gmail.com