KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN MÔN HỌC
CƠ SỞ TRÍ TUỆ NHÂN TẠO

ĐỒ ÁN 1: CÁC THUẬT TOÁN TÌM KIẾM

Trợ giảng phụ trách đồ án:

Nguyễn Duy Khánh (duykhanhnguyen360@gmail.com)
Nguyễn Ngọc Băng Tâm (bangtamnguyenn@gmail.com)

1 Mục tiêu đồ án

Nghiên cứu, cài đặt và so sánh các thuật toán tìm kiếm đường đi.

2 Quy định đồ án

- Các bạn làm đồ án theo nhóm với tối đa 3 thành viên.
- Thời gian hoàn thành đồ án là 2 tuần (xem chi tiết trên Moodle). Chỉ chấp nhận các trường hợp nộp trễ khi đã gửi mail xin phép trước deadline đồ án ít nhất 1 ngày.
- Không được copy code của nhau hoặc bài giải của các sinh viên khóa trước. Sinh viên chỉ được tham khảo ý tưởng, phần cài đặt phải tự mình thực hiện. Phải ghi rõ các nguồn tham thảo khi nộp bài. Nếu vi phạm các quy định trên sẽ bị 0 điểm đồ án.
- Sau khi hoàn thành đồ án, không khuyến khích các bạn đăng công khai code của mình lên Github, chỉ nên public kết quả visualization của mình (ảnh, video demo, ...).

3 Nội dung đồ án

3.1 Mô tả bài toán

Cho một bản đồ mê cung như hình 1, trong đó:

- Ký hiệu $\star$ đại diện cho vị trí xuất phát của tác nhân,
- Ký hiệu $\times$ thể hiện các bức tường và các chướng ngại vật, tác nhân sẽ không thể di chuyển lên các vị trí này,
- Ký hiệu + thể hiện các ô điểm thưởng mà nếu tác nhân di chuyển vào các ô này sẽ được giảm chi phí thực hiện đường đi.
- Ký hiệu **EXIT** đại diện cho vị trí đích mà tác nhân cần di chuyển

Hình 1: Bản đồ trò chơi.

đến, là vị trí **duy nhất** trên biên của các bức tường mà tác nhân có thể thoát khỏi mê cung.

Mục tiêu của các bạn sinh viên là cài đặt các thuật toán tìm kiếm đường đi từ vị trí xuất phát đến vị trí đích (vị trí thoát khỏi mê cung) cho tác nhân. Trong đó tác nhân chỉ có thể di chuyển lên, xuống, trái, phải với chi phí bằng nhau. Các bạn cần cài đặt các thuật toán sau:

- Thuật toán tìm kiếm không có thông tin:
  - Thuật toán tìm kiếm DFS (Depth First Search).
  - Thuật toán tìm kiếm BFS (Breadth First Search).
  - Thuật toán tìm kiếm UCS (Uniform-Cost Search).
- Thuật toán tìm kiếm có thông tin:
  - Thuật toán tìm kiếm tham lam (Greedy Best First Search).
  - Thuật toán tìm kiếm A*.

Đối với các thuật toán tìm kiếm có thông tin, các bạn tự định nghĩa hàm heuristic phù hợp cho bài toán này. Các bạn có thể thử nhiều

hàm heuristic khác nhau (tối thiểu 2 hàm) và báo cáo kết quả.

Về bản đồ trò chơi, sẽ có hai loại: bản đồ không có điểm thưởng và bản đồ có điểm thưởng. Nhiệm vụ của các bạn là cài đặt các thuật toán tìm kiếm và tự thiết kế các bản đồ cho hai loại trên.

- Với trường hợp bản đồ không có điểm thưởng, các bạn sẽ cài đặt 5 thuật toán trên để giải quyết. Các bạn sẽ tự thiết kế và báo cáo ít nhất 5 bản đồ tiêu biểu (các bạn nên chọn những bản đồ mà các thuật toán có sự khác biệt với nhau nhiều) và so sánh sự khác nhau giữa cách tìm đường đi trong các loại bản đồ này với từng chiến lược tìm kiếm khác nhau. Cụ thể với một bản đồ được chọn để báo cáo, các bạn sẽ nhận xét:
  - Sự khác nhau giữa các chiến lược tìm kiếm đối với bản đồ này được thể hiện như thế nào? (cụ thể các bạn cần nhận xét về tính đầy đủ, tính tối ưu, độ mở các nút, thời gian chạy, ... của các thuật toán). Các bạn cần vận dụng lý thuyết để giải thích và nên có hình vẽ hoặc video thể hiện quá trình tìm kiếm đường đi để minh họa.
  - Với các chiến lược tìm kiếm có thông tin, các bạn cần chọn nhiều hàm heuristic khác nhau và báo cáo sự khác nhau giữa các chiến lược đối với từng hàm heuristic.
- Với trường hợp bản đồ có điểm thưởng, các bạn sẽ suy nghĩ cách giải quyết và đề xuất chiến lược để tác nhân di chuyển sao cho chi phí đường đi từ điểm bắt đầu đến điểm thoát khỏi mê cung là nhỏ nhất (lưu ý rằng tác nhân không nhất thiết phải đi qua hết các điểm thưởng). Các bạn cần thiết kế bản đồ với tối thiểu 3 trường hợp: có 2, 5, 10 điểm thưởng (với các giá trị điểm thưởng khác nhau) trên bản đồ. Nếu các bạn không thể tìm được lời giải tối ưu (bất kể trong trường hợp số lượng điểm thưởng là nhiều hay ít), hãy đề xuất chiến lược heuristic để giải quyết, chẳng hạn tham lam ăn tất cả điểm thưởng theo độ lớn giá trị của chúng rồi mới tìm đường thoát khỏi mê cung.

- Kích thước của các bản đồ được chọn để báo cáo phải đảm bảo có ít nhất một bản đồ có chiều dài lớn hơn 35 và một bản đồ có chiều rộng lớn hơn 15.
- Số lượng bản đồ tối thiểu cần báo cáo và phân tích: 5 bản đồ đối với bản đồ không có điểm thưởng và 3 bản đồ (ứng với 3 trường có 2, 5, 10 điểm thưởng trên bản đồ) đối với bản đồ có điểm thưởng.

Các bạn có thể tham khảo hàm vẽ bản đồ ở notebook này, hoặc tham khảo thư viện Pygame để tạo video minh họa.

3.2 Thiết kế kịch bản kiểm thử và bản đồ

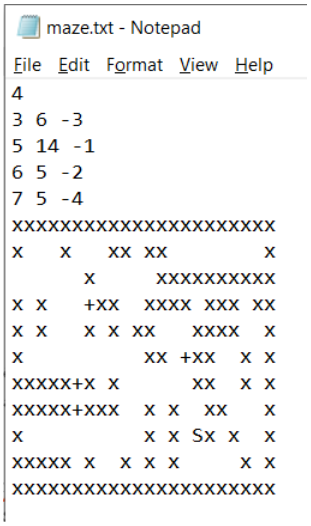
Mỗi kịch bản kiểm thử là một file **.txt** được thiết kế như sau:

- Dòng đầu là số lượng điểm thưởng $\mathbf{n}$ ($\mathbf{n} = 0$ với bản đồ không có điểm thưởng).
- $\mathbf{n}$ dòng tiếp theo, mỗi dòng sẽ bao gồm 3 số nguyên $\mathbf{x}, \mathbf{y}, \mathbf{z}$ với $\mathbf{x}, \mathbf{y}$ là tọa độ của điểm thưởng trong ma trận; $\mathbf{z}$ là giá trị của điểm thưởng, sẽ là các số nguyên âm.
- Các dòng tiếp theo mô tả bản đồ của trò chơi. Các bạn lưu ý điểm kết thúc của hành trình sẽ là điểm thoát khỏi mê cung (ví dụ trong hình 2 thì điểm kết thúc sẽ là điểm ở dòng 2 và cột 0); điểm bắt đầu của tác nhân được ký hiệu bằng ký tự **S**; các ký tự **x** sẽ là các bức tường; các ký tự + sẽ là các điểm thưởng.

Ví dụ về file input **'maze.txt'** như trong hình 2.

Lưu ý về điểm thoát khỏi mê cung: Các bạn lưu ý bản đồ mê cung phải được thiết kế sao cho tồn tại **duy nhất** một vị trí nằm trên các cạnh của bản đồ để tác nhân có thể thoát ra ngoài được.

Đầu ra cho mỗi bản đồ: kết quả trả về cho mỗi bản đồ sẽ là các file đồ họa (**.jpg** hoặc **.mp4**) chứa đồ họa biểu diễn **đường đi trả về** và các file text (**.txt**) có duy nhất một dòng chứa số nguyên thể hiện



Hình 2: Minh họa về tập tin đầu vào của mỗi bản đồ.

chi phí thực hiện đường đi (hoặc chứa từ NO nếu không tìm được đường đi). Lưu ý ứng với một bản đồ, mỗi thuật toán sẽ output ra một file đồ họa và một file text (tên các file này trùng với tên thuật toán, cụ thể là **dfs**, **bfs**, **ucs**, **gbfs**, **astar** hoặc **algo1**, **algo2** nếu là thuật toán do các bạn đề ra). Ví dụ về file ảnh minh họa đường đi kết quả được biểu diễn ở hình 3.

4 Đánh giá

4.1 Thang điểm đồ án

- Điểm đồ án được tính theo các mức như sau:
  1. **Bản đồ không có điểm thưởng**:
     (a) **Mức 1a (5 điểm)**: thiết kế đủ 5 bản đồ và hoàn thành 3 thuật toán tìm kiếm không có thông tin (bao gồm cài đặt thuật toán, báo cáo, hình vẽ minh họa).
     (b) **Mức 1b (8 điểm)**: hoàn thành tiếp 2 thuật toán tìm kiếm có thông tin.

Hình 3: Ví dụ về output đường đi thoát khỏi mê cung.

2. **Bản đồ có điểm thưởng**:
   (a) **Mức 2a (9 điểm)**: đề xuất thuật toán và các hàm heuristic phù hợp để giải quyết, thiết kế đủ 3 bản đồ và mô phỏng cách thuật toán chạy trên các bản đồ.
   (b) **Mức 2b (10 điểm)**: cài đặt thành công thuật toán (đảm bảo kết quả giống với mô phỏng ở mức 2a).
3. **Điểm cộng**: các bạn có thể thực hiện thêm các yêu cầu sau để có thêm điểm cộng cho đồ án:
   – Kịch bản nâng cấp (1 điểm cộng): Bản đồ có các cánh cửa để dịch chuyển bất kỳ từ điểm (i, j) sang điểm (i', j') (giống teleportation - dịch chuyển tức thời). Các bạn tìm lời giải tối ưu nhất có thể cho kịch bản đó, sau đó tự thiết kế bản đồ (tối thiểu 3 bản đồ) và nhận xét.
   – Video minh họa (0.5 điểm cộng): thay vì chỉ output ra

đường đi thoát khỏi mê cung như hình 3, các bạn có thể output ra video thể hiện quá trình tìm kiếm (bao gồm các nút được mở, đường đi cuối cùng, ...) tương tự như video minh họa này. Với các kịch bản nâng cấp, các bạn nên có video để thể hiện thuật toán trực quan hơn.

- Điểm tối đa (S) có thể đạt được là 11.5, điểm đồ án cuối cùng (G) của các bạn sẽ là $G = \min(10, S)$.

4.2 Yêu cầu bài nộp

4.2.1 Báo cáo

- Báo cáo cần trình bày rõ ràng, cấu trúc hợp lý. Báo cáo phải có thông tin các thành viên trong nhóm, phải có phần tự đánh giá mức độ hoàn thành đồ án theo các mức ở mục 4.1.
- Cần giải thích và mô tả ngắn gọn cách cài đặt từng thuật toán mà bạn sử dụng (lưu ý không chèn code để giải thích).
- Cần có chi phí tìm kiếm và hình minh họa (hoặc link video) ứng với output của từng thuật toán cho tất cả các bản đồ.
- Trong báo cáo phải có mục tham khảo, liệt kê các tài liệu (bao gồm cả các link chứa code tham khảo) mà nhóm sử dụng trong quá trình làm đồ án.
- Nên có phần mô tả các câu lệnh trong file **run.sh** (sẽ được mô tả trong mục 4.2.2).

4.2.2 Cấu trúc bài nộp

- Thư mục bài làm của sinh viên phải đặt tên là mã số sinh viên của các thành viên trong nhóm nối lại với nhau bằng ký tự '_', ví dụ nhóm có 3 sinh viên thì đặt tên là MSSV1_MSSV2_MSSV3.

- Thư mục bài nộp (ví dụ MSSV1_MSSV2_MSSV3) bao gồm các file và thư mục được đặt tên như sau:
  - File **run.sh**: file shell script chứa các câu lệnh để chạy **toàn bộ** các thuật toán cho tất cả các bản đồ. Yêu cầu cụ thể các bạn xem ở mục 4.2.3.
  - File **report.pdf**: báo cáo trình bày các vấn đề được yêu cầu trong đề bài. Trang đầu tiên ghi thông tin nhóm (tên thành viên, mã số sinh viên).
  - Folder **source**, trong đó chứa ít nhất một file đặt tên là **main.py** (file này sẽ là file được gọi khi chạy mã nguồn). Các bạn có thể thêm các file mã nguồn khác chứa các hàm phụ trợ khi chạy chương trình. Lưu ý ngôn ngữ sử dụng làm đồ án là Python 3. Ở đầu các file mã nguồn, cần ghi rõ nguồn tham khảo (nếu có).
  - Folder **input**, trong đó có các folder **level_1**, **level_2**, **advance** chứa các file định dạng **.txt** mà các bạn tự thiết kế ứng với các bản đồ không có điểm thưởng, có điểm thưởng, và bản đồ cho kịch bản nâng cấp. Lưu ý trong từng folder trên, các file bản đồ được đặt tên theo thứ tự **input1.txt**, **input2.txt**, ... tăng dần (nếu không có kịch bản nâng cấp thì folder **advance** các bạn để trống).
- Sinh viên phải nén thư mục bài làm thành định dạng **.zip** (các bài nộp với định dạng khác sẽ bị 0 điểm đồ án) rồi nộp file **.zip** lên Moodle.

4.2.3 Quy định về mã nguồn

- Các bạn phải đảm bảo là code cho tất cả các yêu cầu của đồ án phải được hoàn thành khi chạy file **run.sh** từ terminal

  ```
  bash run.sh
  ```

- Code các bạn cần xử lý được số lượng bản đồ không cố định

trước, ví dụ khi trong các folder **input/level_1**, **input/level_2**, **input/advance** có 3, 5, hay 10 bản đồ thì khi chạy file **run.sh** thì tất cả các bản đồ đều được xử lý. Các bạn lưu ý yêu cầu này để tránh bị thiếu output (dẫn đến bị trừ điểm) khi các trợ giảng chấm bài.

- Sau khi chạy file **run.sh**, code các bạn cần tạo ra một folder **output** (cùng cấp với folder **input**, trong đó cũng có các folder **level_1**, **level_2**, **level_3**, **advance**) chứa kết quả chạy của các thuật toán cho tất cả các bản đồ. Với một file text input cho 1 bản đồ (ví dụ file **input/level_1/input1.txt**), mỗi thuật toán sẽ có tương ứng một folder output trùng tên với tên thuật toán (ví dụ **output/level_1/input1/bfs**), folder này sẽ chứa kết quả (là file text và các file đồ họa trùng với tên thuật toán, ví dụ **bfs.txt**, **bfs.jpg**) của thuật toán tương ứng khi chạy. Trong trường hợp các bạn sử dụng nhiều heuristic cho cùng 1 thuật toán, các file text và đồ họa được đặt tên theo quy định sau **<tên thuật toán>_heuristic_<ID bắt đầu từ 1>** (ví dụ **astar_heuristic_1**).
- Nếu các bạn sử dụng hệ điều hành Windows, các bạn cần thử chạy lại code của bạn trên máy có hệ điều hành thuộc họ Unix (các bạn có thể upload code và test thử trên Google Colab) để đảm bảo không xảy ra lỗi trong quá trình chấm bài, gây ảnh hưởng điểm số của các bạn (do sẽ có những khác biệt giữa các hệ điều hành về đường dẫn file, câu lệnh, ...).

5 Liên hệ

Mọi câu hỏi về đồ án các bạn vui lòng điền vào link Google Sheets. Với các vấn đề khác, các bạn có thể gửi mail tới các trợ giảng Nguyễn Duy Khánh hoặc Nguyễn Ngọc Băng Tâm.

Chúc các bạn hoàn thành thật tốt đồ án!