Khoa Công nghệ thông tin

Lớp: TH05

Môn: Trí tuệ nhân tạo

**BÁO CÁO ĐỒ ÁN 1**

**ÁP DỤNG THUẬT TOÁN A\* TRÊN BÀI TOÁN 8-PUZZLE**

Sinh viên thực hiện:

* MSSV: 0512175
* Họ tên: Nguyễn Đăng Khoa

# Mục tiêu, nội dung

Chương trình được cài đặt để giải quyết các bài toán tìm kiếm mù bằng phương pháp A\*, cụ thể trong chương trình này là để giải quyết bài toán 8puzzle.

# Cài đặt và hướng dẫn sử dụng

## Cài đặt

* Mô tả cấu trúc dữ liệu chính: thiết kế theo hướng đối tượng, có các lớp chính sau:

Data8puzzle

Information

MinPriorityQueue

AVLTree

BlindSearch

AStar

8puzzle

* + MinPriorityQueue là một hàng đợi ưu tiên được cài đặt bằng cấu trúc heap, được sử dụng cho tập trạng thái OPEN trong thuật toán A\*.
  + AVLTree là một cấu trúc cây AVL, được sử dụng cho tập trạng thái CLOSE trong thuật toán A\*.
  + Lớp Information và BlindSearch là những lớp trừu tượng.
  + BlindSearch cung cấp hàm Run() thuần ảo để các lớp con cài đặt một thuật toán nào đó, cụ thể AStar sẽ cài đặt thuật toán A\* bên trong hàm Run(). BlindSearch có một thuộc tính m\_info kiểu Information\* để nhận và lưu trữ thông tin của bài toán.
  + Information chỉ khai báo các hàm thuần ảo phục vụ cho BlindSearch (chính xác hơn là cho hàm Run() ở các lớp con của BlindSearch, ở đây là hàm Run() của lớp AStar), Information cung cấp 1 số hàm truy xuất dữ liệu cần thiết mà một thuật toán tìm kiếm mù yêu cầu (cụ thể là thuật toán A\*)
  + Data8puzzle cài đặt tất cả các hàm mà nó kế thừa từ Information, những hàm này được cài đặt dựa vào cách thức lưu trữ dữ liệu của lớp Data8puzzle. Data8puzzle chứa đầu vào và đầu ra của bài toán 8puzzle (kiểu int, ma trận 3x3 để mô tả trạng thái đã được biến đổi thành số nguyên), một danh sách chứa thông tin g và trạng thái cha của một trạng thái (được tổ chức bằng cây AVL).
  + AStar chỉ cài đặt hàm Run() để chạy thuật toán A\*.
  + 8puzzle gồm hàm Run() và một số hàm lấy thông tin mà bài toán đặt ra, cụ thể ở đây là xuất ra tập tin các trạng thái trung gian của một cách giải.
* Mô tả các hàm xử lý chính
  + Run() trong lớp AStar: cài đặt thuật toán A\*, nó sử dụng các hàm lấy, cập nhật giá trị g và trạng thái cha của một trạng thái, lấy các trạng thái tiếp theo của một trạng thái thông qua biến m\_Info của lớp BlindSearch.
* Các ý tưởng “đặc biệt” trong quá trình cài đặt
  + Chương trình được thiết kế theo hướng đối tượng để giải quyết các bài toán tìm kiếm mà có thể áp dụng thuật toán A\* mà không cần phải viết lại thuật toán A\* cho phù hợp với điều kiện bài toán. Khi có một bài toán khác, ví dụ như bài toán tìm đường đi trong đồ thị, ta chỉ cần định nghĩa thêm 2 lớp là: một lớp kế thừa lớp Information, định nghĩa lại các hàm cho phù hợp với dữ liệu của bài toán mới; một lớp kế thừa lớp A\*, thêm vào những hàm lấy kết quả mà bài toán mới đặt ra. Trong khi thuật toán A\* đã cài đặt ban đầu vẫn không thay đổi.
  + Nếu có một thuật toán mới thay thế thuật toán A\* thì ta chỉ cần tạo thêm một lớp mới kế thừa lớp BlindSearch và cài đặt thuật toán mới trong hàm Run(), sau đó sử dụng lớp này cho các bài toán tương tự như dùng A\*.
  + Khi lưu trữ các trạng thái của A\*, ta không lưu trực tiếp ma trận 3x3 mà sẽ biến đổi ma trận này thành một số nguyên, lưu số nguyên sẽ ít tốn chi phí bộ nhớ hơn so với lưu ma trận 3x3, quá trình chuyển đổi giữa số nguyên và ma trận 3x3 là đơn giản, và khi lưu với số nguyên thì việc so sánh giữa các trạng thái với nhau là dễ dàng.
  + Đối với trường hợp một trạng thái y của x nằm trong tập Close mà giá trị y mới nhỏ hơn giá trị y trong Close thì ta cập nhật lại y của Close và đưa y này vào lại Open để thực hiện việc sửa đổi các trạng thái con của y. Với cách làm này, ta không cần phải viết thêm hàm để thực hiện việc cập nhật lại tất cả các trạng thái con ở tất cả các cấp của trạng thái y.
* Nguồn tham khảo: có sử dụng source code cài đặt Priority Queue được cung cấp trên moodle.

## Hướng dẫn sử dụng

Chạy bằng dòng lệnh với cấu trúc lệnh như sau:

0512175 <h> <input file> <output file>

h = [1, 2, 3] cho biết hàm heuristic cần thực hiện, các thông số khác là tên tập tin tương ứng.

# Thử nghiệm

* Bảng kết quả yêu cầu 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Trạng thái đầu, trạng thái đích** | **Hàm Heuristic** | **m** | **l** | **b** | **t (s)** |
| |  |  |  | | --- | --- | --- | | 2 | 8 | 1 | | 0 | 4 | 3 | | 7 | 6 | 5 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 8 | 0 | 4 | | 7 | 6 | 5 | | H1 | 568 | 10 | 1,8852 | 0,063 |
| H2 | 58 | 10 | 1,4983 | 0,016 |
| H3 | 28 | 10 | 1,3904 | 0 |
| |  |  |  | | --- | --- | --- | | 2 | 8 | 1 | | 4 | 6 | 3 | | 0 | 7 | 5 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 8 | 0 | 4 | | 7 | 6 | 5 | | H1 | 3271 | 13 | 1,8636 | 0,391 |
| H2 | 164 | 13 | 1,4797 | 0,016 |
| H3 | 47 | 13 | 1,3425 | 0 |
| |  |  |  | | --- | --- | --- | | 2 | 4 | 0 | | 1 | 8 | 5 | | 3 | 6 | 7 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 0 | | H1 | 423211 | 27 | 1,6158 | 74,312 |
| H2 | 48079 | 27 | 1,4908 | 6,484 |
| H3 | 3061 | 27 | 1,3462 | 0,375 |
| |  |  |  | | --- | --- | --- | | 8 | 4 | 7 | | 5 | 0 | 6 | | 1 | 3 | 2 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 0 | | H1 | 470076 | 27 | 1,6221 | 85,328 |
| H2 | 65929 | 27 | 1,5083 | 9,078 |
| H3 | 3766 | 27 | 1,3566 | 0,453 |
| |  |  |  | | --- | --- | --- | | 5 | 6 | 7 | | 4 | 0 | 8 | | 3 | 2 | 1 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 8 | 0 | 4 | | 7 | 6 | 5 | | H1 | 521885 | 31 | 1,5291 | 103,641 |
| H2 | 209777 | 31 | 1,4848 | 32,766 |
| H3 | 1470 | 31 | 1,2652 | 0,156 |

Nhận xét:

* + Cả 3 hàm heuristic trên đều cho ra lời giải tối ưu (nếu có lời giải)
  + Khi độ sâu của bài toán tăng lên thì số trạng thái phải xét tăng.
  + Trạng thái mở phải xét nhìều hay ít là phụ thuộc vào hàm H.
  + Thời gian thực thi phụ thuộc vào số trạng thái mở, tức là phụ thuộc vào H.
  + Khi độ sâu của bài toán càng tăng thì hàm H3 càng chiếm ưu thế hơn hai hàm còn lại.
  + Đối với H1: Số phân nhảnh trung bình b = 1,7031.
  + Đối với H2: Số phân nhảnh trung bình b = 1,4924.
  + Đối với H3: Số phân nhảnh trung bình b = 1,0709.
  + Từ số liệu trên cho thấy hàm H3 chạy nhanh hơn H1 và H2 trong mọi trường hợp. (Các số liệu này phụ thuộc vào cách cài đặt thuật toán A\* của mỗi người, nên số trạng thái phải xét của mỗi cách cài đặt là khác nhau).
* Bảng kết quả yêu cầu 3, nhận xét

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Trạng thái đầu, trạng thái đích** | **w** | **m** | **l** | **b** | **t (s)** |
| |  |  |  | | --- | --- | --- | | 2 | 4 | 0 | | 1 | 8 | 5 | | 3 | 6 | 7 |  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 4 | 5 | 6 | | 7 | 8 | 0 | | 0 | 305 | 71 | 1,0839 | 0,047 |
| 0,1 | 399 | 47 | 1,1358 | 0,031 |
| 0,2 | 608 | 39 | 1,1786 | 0,079 |
| 0,3 | 471 | 27 | 1,2559 | 0,047 |
| 0,4 | 1634 | 29 | 1,2906 | 0,187 |
| 0,5 | 3061 | 27 | 1,3462 | 0,375 |
| 0,6 | 22503 | 27 | 1,4494 | 2,938 |
| 0,7 | 91921 | 27 | 1,5270 | 13,406 |
| 0,8 | 118829 | 27 | 1,5416 | 17,766 |
| 0,9 | 239871 | 27 | 1,5822 | 39,641 |
| 1 | 423211 | 27 | 1,6158 | 74,859 |

Nhận xét:

* 0 <= w < 0,5 không tìm được lời giải tối ưu (trường hợp w = 0,3 có lời giải tối ưu là do may mắn).
* 0,5 <= w <= 1 tìm được lời giải tối ưu.
* w càng tăng thì m, b, t càng tăng.
* w càng tiến về 1 thì số trạng thái phải xét nhiều, tăng theo cấp số nhân, gần về với bài cách giải vét cạn.

# Kết luận

* Đã cài đặt được thuật toán A\*, áp dụng thành công vào bài toán 8puzzle.
* Thực hiện được các yêu cầu mà bài tập đưa ra.
* Do làm theo cách tổng quát cho các bài toán A\* nên tốc độ sẽ không nhanh bằng làm trực tiếp thuật toán A\* lên bài 8puzzle.
* Hướng phát triển: cài đặt giao diện đồ họa để thể hiện thuật toán A\*, mô tả từng bước chọn trạng thái của A\* để minh họa cho bài toán.
* Tốc độ vẫn còn chậm do sinh ra nhiều trạng thái trung gian, số trạng thái sinh ra > 9! = 362800 là do trong quá trình chạy sinh ra các trạng thái trung gian. Chưa khắc phục được nhược điểm này. Và đối với những test không có cách đi thì nó phải vét hết tất cả các trạng thái nên tốn nhiều thời gian hơn những test có kết quả.