ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KH&KT MÁY TÍNH

----------***----------

MÔN TRÍ TUỆ NHÂN TẠO
BÁO CÁO BÀI TẬP LỚN SỐ 1
TÌM KIẾM (SEARCHING)

Giảng viên hướng dẫn:

GS. Cao Hoàng Trụ

ThS. Vương Bá Thịnh

Sinh viên thực hiện:

Huỳnh Tấn Phúc - 51202787

TP Hồ Chí Minh, ngày 10 tháng 04 năm 2015.

MỤC LỤC

I. Yêu cầu

Hiện thực 2 bài toán 2048 và UnBlock Me, và với mỗi bài toán, phải hiện thực bằng 3 giải thuật: Depth-first search, Breadth-first search, và Simple Hill Climbing.

Đánh giá các giải thuật.

II. Quá trình tìm hiểu và hiện thực

1. Tìm hiểu tổng quát về các giải thuật: Depth-first search, Breadth-first search, và Simple Hill Climbing.

1.1.Depth-first search

Depth-first search (DFS): là một thuật toán duyệt hoặc tìm kiếm trên một cây hoặc một đồ thị. Thuật toán khởi đầu tại gốc (hoặc chọn một đỉnh nào đó coi như gốc) và phát triển xa nhất có thể theo mỗi nhánh.

Thông thường, DFS là một dạng tìm kiếm thông tin không đầy đủ mà quá trình tìm kiếm được phát triển tới đỉnh con đầu tiên của nút đang tìm kiếm cho tới khi gặp được đỉnh cần tìm hoặc tới một nút không có con. Khi đó giải thuật quay lui về đỉnh vừa mới tìm kiếm ở bước trước. Trong dạng không đệ quy, tất cả các đỉnh chờ được phát triển được bổ sung vào một ngăn xếp LIFO.

1.2.Breadth-first search

Breadth-first search (BFS): là một thuật toán tìm kiếm trong đồ thị trong đó việc tìm kiếm chỉ bao gồm 2 thao tác: thăm một đỉnh của đồ thị; thêm các đỉnh kề với đỉnh vừa thăm vào danh sách có thể thăm trong tương lai. Có thể sử dụng thuật toán tìm kiếm theo chiều rộng cho hai mục đích: tìm kiếm đường đi từ một đỉnh gốc cho trước tới một đỉnh đích, và tìm kiếm đường đi từ đỉnh gốc tới tất cả các đỉnh khác. Trong đồ thị không có trọng số, thuật toán tìm kiếm theo chiều rộng luôn tìm ra đường đi ngắn nhất có thể. Thuật toán BFS bắt đầu từ đỉnh gốc và lần lượt thăm các đỉnh kề với đỉnh gốc. Sau đó, với mỗi đỉnh trong số đó, thuật toán lại lần lượt thăm các đỉnh kề với nó mà chưa được thăm trước đó và lặp lại.

Gọi b là hệ số phân nhánh của bài toán, tức số trạng thái tiếp theo tối đa của 1 trạng thái; d là độ sâu của lời giải, tức khoang cách từ trạng thái khởi đầu đến trạng thái mục tiêu, tính theo số tầng trên cây trạng thái; m là độ sâu tối đa của cây trạng thái. Ta có bảng so sánh 2 giải thuật tìm kiếm như sau:

Tiêu chí	BFS	DFS
Thời gian	b^d	b^m
Không gian	b^d	$b \times m$
Tính tối ưu	Có	Không
Tính đầy đủ	Có	Không

1.3.Simple Hill Climbing

Để tránh sự bùng nổ về số lượng các trạng thái phải duyệt như 2 giải thuật trên, dẫn đến độ phức tạp thời gian hàm mũ. Chiến lược tìm kiếm theo kinh nghiệm ra đời điển hình là giải thuật leo đồi đơn giản (Simple Hill Climbing). Giải thuật này sẽ dùng 1 hàm lượng giá để đánh giá xem 1 trạng thái bất kì gần với mục tiêu như thế nào. Bước đi nào dẫn đến trạng thái gần với mục tiêu hơn so với trạng thái hiện tại sẽ được chọn để thực hiện.

2. Áp dụng

2.1.Bài toán 2048

2.1.1. Mã hóa bài toán

Nhận xét bài toán gồm 16 ô vuông có các số chẵn trên đó. Từ đó ta tiến hành mã hóa.

Class Tile dùng để biểu diễn 1 ô vuông với biến _value là số trên ô vuông. Ta quy ước _value bằng 0 nếu là ô trống.

Class State dùng biểu diễn trạng thái hiện tại gồm 16 ô vuông chỉ số từ 0->15 cùng với giá trị của chúng.

Class ChashMap dùng để lưu vết từ trạng thái ban đầu đến trạng thái hiện tại mục đích là để in dữ liệu ra cho đúng thứ tự.

2.1.2. Hiện thực

Depth-first search: ta đặt trạng thái hiện tại vào stack, sau đó pop ra, đặt các trạng thái con sau khi thực hiện phép move (nếu được) đồng thời cũng sinh tất cả các khả năng của số 2 và 4 rồi cho vào stack, trong khi stack chưa rỗng ta pop 1 state ra khỏi stack, đặt các trạng thái con vào…

Breadth-first search: tương tự Depth-first search chỉ khác là ta thay stack bằng queue.

Simple Hill Climbing: Đầu tiên ta có các nhận xét tổng quát sau:

Gọi b' và b là số ô trống sau khi thực hiện 1 phép move, ta lun có $b' \geq b$. *Nếu ta chọn hàm lượng giá* $f = b' - b > 0$ *(với mong muốn mình sau mỗi bước move thì số ô trông sinh ra lun lớn hơn lúc đầu vì số ô trống càng nhiều càng khó dẫn đến trạng thái chết). Với cách chọn này thì giải thuật không hoạt động nếu b'= b, điều này xảy ra khi không có cách move nào sao cho số ô trống sau khi move > số ô trống trước đó, suy ra hàm lượng giá này khó hoạt động lúc mới bắt đầu game.*

Còn nếu gọi s' và s tổng các giá trị trên các ô vuông sau và trước khi move ta lun có s'= s.

Nếu gọi m và m' là tổng số bước move cho tới trạng thái hiện tại và trạng thái sau, ta lun có m' > m.

Nếu ta chọn hàm lượng giá là tổng sự chênh lệch giữa các giá trị 2 ô kề nhau của tổng tất cả các hàng và cột.

0	0	0	0
0	2	4	0
0	0	0	0
0	0	0	0

22

Ta lun có tổng lúc trước lun lớn (p) hơn hay bằng tổng lúc sau (p'). Giả sử ta chọn p – p' > 0 làm hàm lượng giá thì giải thuật chỉ chạy được lúc đầu nhưng kết quả không cao.

Tóm lại ta có được các hàm lun đúng sau:

$b' - b \geq 0$

$p - p' \geq 0$

$m' - m > 0$

$s - s' = 0$

Câu hỏi liệu có thể suy ra 1 hàm lun đúng f(x) > 0 từ 4 hàm trên khi dùng log, e mũ, nghịch đảo…Tôi không làm được.

Vì thế tôi không quan sát giá trị nữa mà nhìn tổng quát như sau: nếu ta bỏ đi 1 phép move nào đó chỉ thực hiện 3 phép move còn lại theo 1 cách nào đó thì giá trị càng lớn sẽ càng được dồn về phía đối diện với phía mà mình đã bỏ không move, khi đó nhìn chung những số nhỏ sẽ xuất hiện bên phía đối diện với bên lớn tạo điều kiện cho ta dễ sắp xếp. Do đó theo kinh nghiệm tôi chọn hàm lượng giá: nếu move right + 5, move up + 4, move down +3, move left -6.

$f = x' - x > 0$ với x' và x là số điểm trước và sau khi move.

Nếu chọn hàm lượng giá như vậy cao nhất có thể đạt 512, trung bình 256, và 128.

2.2.Bài toán UnBlock Me

2.2.1. Mã hóa bài toán

Nhận xét bài toán 1 tấm gỗ gồm nhiều thanh ngang, dọc mỗi thanh chỉ có chiều dài 2 hay 3 và có 1 thanh màu đỏ, và thanh ngang chỉ di chuyển theo chiều ngang, thanh dọc chỉ di chuyển dọc. Mục dích cuối là đưa thanh đỏ ra ngoài.

Từ đó tôi chọn cách mã hóa sau:

Class Tile dùng biểu diễn 1 thanh với các thuộc tính _horizontal, _vertical, _goal được khởi tạo là false nếu thanh ngang và là thanh màu đỏ thì _horizontal true, _goal true,…

Class State dùng để biểu diễn tấm gỗ đó, với _numTiles chỉ số lượng thanh, _tiles là array chứa các tile.

Class ChashMap dùng để lưu vết từ trạng thái ban đầu đến trạng thái hiện tại mục đích là để in dữ liệu ra cho đúng thứ tự.

Thanh ngang chiều dài 2, 3 lần lượt -- và ---

Thanh doc chiều dài 2, 3 lần lượt là | và |

| |

|

```
Thanh đỏ là $$, những chỗ trống là .

Khi thanh đỏ đến mép ngoài là thành công.
```

2.2.2. Hiện thực

Depth-first search: ta đặt trạng thái hiện tại vào stack, sau đó pop ra, đặt các trạng thái con sau khi thực hiện phép move (nếu được) vào stack, trong khi stack chưa rỗng ta pop 1 state ra khỏi stack, đặt các trạng thái con vào…

Breadth-first search: tương tự Depth-first search chỉ khác thay stack bằng queue.

Simple Hill Climbing: Đầu tiên ta có nhận xét tổng quát sau: để thanh màu đỏ có thể ra ngoài ta chọn trạng thái move sao cho cố gắng tạo ra khoản trống để thanh đỏ có thể ra ngoài.

Từ đó ta có hàm lượng giá, nếu phép move nào tạo khoản trống sau thanh đỏ ta cộng 1 ngược lại trừ 1.

III. Đánh giá

1. Bài toán 2048

1.1.Thời gian thực thi (s)

Bảng III.1.1

Input	Depth-first search	Breadth-first search	Simple Hill Climbing
1	0.006	0.004	0.006
2	0.006	0.355	0.003
3	0.007	0.546	0.002
4	0.024	Tràn bộ nhớ	0.011
5	0.026	Tràn bộ nhớ	0.011
6	0.051	Tràn bộ nhớ	0.019
7	0.054	Tràn bộ nhớ	0.041
8	0.100	Tràn bộ nhớ	Không chạy tới được
9	0.463	Tràn bộ nhớ	Không chạy tới được
10	45.226	Tràn bộ nhớ	Không chạy tới được

1.2.Sự tiêu tốn bộ nhớ (MB)

Bảng III.1.2

Input	Depth-first search	Breadth-first search	Simple Hill Climbing
1	0.306	0.309	0.271
2	0.374	51.226	0.271
3	0.382	60.149	0.271
4	1.053	Tràn bộ nhớ	0.271
5	1.000	Tràn bộ nhớ	0.273
6	1.158	Tràn bộ nhớ	0.271
7	1.260	Tràn bộ nhớ	0.272
8	1.990	Tràn bộ nhớ	Không chạy tới được
9	3.915	Tràn bộ nhớ	Không chạy tới được
10	3.429	Tràn bộ nhớ	Không chạy tới được

1.3.Đánh giá

Từ input 1 đến 10 độ phức tạp của input tăng dần do đích đến càng lớn => số node duyệt qua càng nhiều. Do tính chất của input nên đối với DFS thì nhìn chung thời gian thực thi và bộ nhớ dùng đều tăng, BFS cũng vậy nhưng so với DFS thì tốn thời gian và bộ nhớ nhiều hơn do nó duyệt theo chiều ngang (từ gốc nó phải duyệt qua 32 nốt, rồi 32x32,… còn DFS đi theo chiều sâu nên ít bị bùng nổ bộ nhớ. Hơn nữa khi BFS đã tìm được lời giải thì đó là lời giải đầy đủ và tối ưu, còn lời giải của DFS thì không tối ưu (dễ thấy khi quan sát các file output). Simple Hill Climbing do tính chất của input nên thời gian và độ tiêu tốn bộ nhớ nhìn chung tăng, nhưng so với DFS và BFS thì

đều ít hơn, do ta đã dùng 1 hàm lượng giá để chọn lối đi cho nó, nhưng đổi lại do hàm lượng giá chỉ là heuristic nên nhiều input ko đạt đến goal.

2. UnBlock Me

2.1.Thời gian thực thi (s)

Bảng III.2.1

Input	Depth-first search	Breadth-first search	Simple Hill Climbing
1	0.008	0.007	0.008
2	0.015	0.019	0.032
3	0.035	0.008	0.009
4	0.146	Quá lâu	Không tìm ra lời giải
5	23.307	Quá lâu	Không tìm ra lời giải
6	0.027	0.024	Không tìm ra lời giải
7	0.057	Quá lâu	Không tìm ra lời giải
8	105.302	Quá lâu	Không tìm ra lời giải
9	200.428	Quá lâu	Không tìm ra lời giải
10	378.901	Quá lâu	Không tìm ra lời giải

2.2.Sự tiêu tốn bộ nhớ (MB)

Bảng III.2.2

Input	Depth-first search	Breadth-first search	Simple Hill Climbing
1	0.273	0.273	0.272
2	0.282	0.367	0.276
3	0.282	0.287	0.274
4	1.355	Quá lâu	Không tìm ra lời giải
5	7.755	Quá lâu	Không tìm ra lời giải
6	0.303	0.409	Không tìm ra lời giải
7	0.509	Quá lâu	Không tìm ra lời giải
8	0.585	Quá lâu	Không tìm ra lời giải
9	0.700	Quá lâu	Không tìm ra lời giải
10	0.550	Quá lâu	Không tìm ra lời giải

2.3.Đánh giá

Ta thấy giải thuật DFS ở các input trên lun tìm ra lời giải, nhưng khi BFS tìm ra lời giải thì BFS có lời giải tối ưu hơn. Nhưng do BFS bị bùng nổ không gian trạng thái nên với những input có không gian trạng thái lớn thì lời giải quá lâu hoặc tràn bộ nhớ. Còn giải thuật leo đồi đơn giản do hàm lượng giá không tốt nên chỉ tìm được lời giải của những input dễ. Hơn nữa khi BFS đã tìm được lời giải thì đó là lời giải đầy đủ và tối ưu, còn lời giải của DFS thì không tối ưu (dễ thấy khi quan sát các file output).

IV. Kết quả

1. Bài toán 2048

Input 7

2 0 0 0

0 0 0 0 Goal 256

0 2 0 0

0 0 0 0

Output 7

DFS	BFS	Simple HillClimbing
Program is runing ... This is solution: ---------step:1---------- 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 ---------step:2---------- 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 4 ---------step:3---------- 2 0 0 0 0 0 0 0 2 0 0 0 4 0 0 4 ---------step:4---------- 0 0 0 2 0 0 0 0 0 0 0 2 0 0 4 8 ---------step:5---------- 2 0 0 0 0 0 0 0 2 0 0 0 4 8 0 4 ---------step:6---------- 0 0 0 2 0 0 0 0 0 0 0 2 4 4 8 4 ---------step:7---------- 0 0 0 2 0 0 0 0 0 0 0 2 4 8 8 4 … ---------step:99---------- 0 0 4 8 4 8 32 4 16 64 16 32 8 64 128 8 ---------step:100---------- 4 8 0 4 4 8 32 4 16 64 16 32 8 64 128 8 ---------step:101---------- 4 4 8 4 4 8 32 4	Tràn bộ nhớ	Program is runing ... Value of this state: 10 ---------step:1---------- 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 Value of this state: 14 ---------step:2---------- 2 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 Value of this state: 19 ---------step:3---------- 4 0 4 4 0 0 0 0 0 0 0 0 0 0 0 0 Value of this state: 24 ---------step:4---------- 4 0 4 8 0 0 0 0 0 0 0 0 0 0 0 0 Value of this state: 29 ---------step:5---------- 4 0 8 8 0 0 0 0 0 0 0 0 0 0 0 0 Value of this state: 34 ---------step:6---------- 4 0 4 16 0 0 0 0 0 0 0 0 0 0 0 0 Value of this state: 39 … ---------step:107---------- 4 32 64 128 16 16 32 64 8 8 16 32 4 4 0 0 Value of this state: 489 ---------step:108---------- 4 32 64 128 4 32 32 64 0 16 16 32 0 0 0 8 Value of this state: 493

```
16      64      16      32
8       64      128     8
---------step:102----------
4       8       8       4
4       8       32      4
16      64      16      32
8       64      128     8
---------step:103----------
4       4       16      4
4       8       32      4
16      64      16      32
8       64      128     8
---------step:104----------
4       8       16      4
4       8       32      4
16      64      16      32
8       64      128     8
---------step:105----------
0       0       16      0
8       4       32      8
16      16      16      32
8       128     128     8
---------step:106----------
0       0       0       16
8       4       32      8
0       16      32      32
4       8       256     8
Solution found!
It takes 0.054 s
Used memory is bytes:
1321224.0 bytes
Used memory is megabytes:
1.2600173950195312 MB
```

```
----------step:109----------
8       64      64      128
4       16      32      64
0       0       16      32
0       0       0       8
Value of this state: 498
----------step:110----------
4       8       128     128
4       16      32      64
0       0       16      32
0       0       0       8
Value of this state: 502
----------step:111----------
8       8       128     128
4       16      32      64
0       0       16      32
0       0       0       8
Value of this state: 507
----------step:112----------
4       0       16      256
4       16      32      64
0       0       16      32
0       0       0       8
Solution found!
It takes 0.041 s
Used memory is bytes:
284704.0 bytes
Used memory is megabytes:
0.271514892578125 MB
```

2. UnBlock Me

Int put 3:

```
-  -  -  -  -  -
-  -  -  -  -  -
.  $  $  .  .  .
.  .  .  |  .  .
-  -  -  |  .  .
.  .  .  .  .  .
```

Out put 3:

DFS

```
Program is runing ...
This is solution:
---------step:1---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  $  $  .  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
```

BFS

```
Program is runing ...
This is solution:
---------step:1---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  $  $  .  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
```

Simple Hill Climbing

```
Program is runing ...
---------step:1---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  $  $  .  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
```

```
 .  .  .  .  .  .
---------step:2---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  $  $  .  .  .
 .  .  .  .  .  .
 -  -  -  |  .  .
 .  .  .  |  .  .
---------step:3---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  $  $  .  .
 .  .  .  .  .  .
 -  -  -  |  .  .
 .  .  .  |  .  .
---------step:4---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  $  $  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:5---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  $  $  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:6---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  $  $  .
 .  .  .  .  .  .
 -  -  -  |  .  .
 .  .  .  |  .  .
---------step:7---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  .  $  $
 .  .  .  .  .  .
 -  -  -  |  .  .
 .  .  .  |  .  .
Solution found!
It takes 0.035 s
Used memory is bytes:
295296.0 bytes
Used memory is
megabytes:
0.2816162109375 MB
```

```
 .  .  .  .  .  .
---------step:2---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  $  $  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:3---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  $  $  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:4---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  .  $  $
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
Success!
It takes 0.008 s
Used memory is bytes:
301096.0 bytes
Used memory is
megabytes:
0.28714752197265625 MB
```

```
---------step:2---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  $  $  .  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:3---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  $  $  .
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
---------step:4---------
 -  -  -  -  -  -
 -  -  -  -  -  -
 .  .  .  .  $  $
 .  .  .  |  .  .
 -  -  -  |  .  .
 .  .  .  .  .  .
Success!
It takes 0.009 s
Used memory is bytes:
286808.0 bytes
Used memory is
megabytes:
0.27352142333984375 MB
```

Ta thấy output BFS tối ưu hơn DFS.

Do các output quá lớn, nên ở đây mỗi giải thuật chỉ liệt kê 1 ouput điển hình. Xem thêm các output khác ở file đính kèm.

V. Tham khảo

Tài liệu học tập của môn học

http://en.wikipedia.org/wiki/Depth-first_search

http://en.wikipedia.org/wiki/Breadth-first_search

http://en.wikipedia.org/wiki/Hill_climbing

Play Store trên Android

Chi tiết code và file output xem ở file đính kèm.