Programing Assignment #1

CSCE 625 - Artificial Intelligence
Spring 2015

Name: Xiaoshu Zhang

Email: kallen5208@gmail.com

On managing frontier, BFS uses a FIFO queue, DFS uses a LIFO queue, GBFS uses a priority queue sorted on straight-line distance heuristic.
Ten pairs of departure/destination combination have been chosen to compare three algorithms' performances.
Parts of transcripts are shown in Appendix A, Appendix B, and Appendix C.
The completed transcripts are in "BFS_Search.txt", "DFS_Search.txt", and "GBFS_Search.txt".
The summaries are shown in Table 1, Table 2, and Table 3.
The average comparison is shown in Table 4.

Table 1 BFS's performance

departure	destination	total iterations	max frontier size	vertices visited/275	path length
(1,1)	(4,4)	9	7	16	3
(1,1)	(20,20)	258	21	270	28
(1,1)	(1,20)	159	21	180	19
(1,1)	(20,1)	172	21	186	19
(1,20)	(20,1)	268	21	271	33
(7,6)	(13,6)	259	18	269	28
(5,13)	(16,10)	228	20	238	25
(6,13)	(7,10)	53	19	67	8
(15,12)	(15,8)	101	16	117	14
(8,10)	(15,12)	189	20	200	19
average		169.6	18.4	181.4	19.6

Table 2 DFS's performance

departure	destination	total iterations	max frontier size	vertices visited/275	path length
(1,1)	(4,4)	3	5	10	3
(1,1)	(20,20)	31	35	67	31
(1,1)	(1,20)	173	67	207	33
(1,1)	(20,1)	50	52	103	50
(1,20)	(20,1)	38	36	75	38
(7,6)	(13,6)	58	49	106	53
(5,13)	(16,10)	45	40	86	41
(6,13)	(7,10)	86	67	151	65

(15,12)	(15,8)	164	92	248	90
(8,10)	(15,12)	232	85	268	40
average		88.0	52.8	132.1	44.4

Table 3 GBFS's performance

departure	destination	total iterations	max frontier size	vertices visited/275	path length
(1,1)	(4,4)	3	5	10	3
(1,1)	(20,20)	31	35	67	31
(1,1)	(1,20)	19	20	40	19
(1,1)	(20,1)	19	20	40	19
(1,20)	(20,1)	63	51	114	47
(7,6)	(13,6)	58	33	94	31
(5,13)	(16,10)	62	31	97	25
(6,13)	(7,10)	12	8	21	8
(15,12)	(15,8)	22	20	43	14
(8,10)	(15,12)	38	26	64	19
average		32.7	24.9	59.0	21.6

Table 4 Average comparison of BFS, DFS, and GBFS

	total iterations	max frontier size	vertices visited/275	path length
BFS	169.6	18.4	181.4	19.6
DFS	88.0	52.8	132.1	44.4
GBFS	32.7	24.9	59.0	21.6

1. Which algorithm is fastest (finds goal in fewest iterations)?

GBFS.

2. Which is most memory efficient (smallest max frontier size)?

BFS.

3. Which visits the fewest vertices?

GBFS.

4. Which generates the shortest path length? (Is any of them optimal?)

BFS.

5. Are the performance differences what you expected based on the theoretical complexity analysis?

Yes.

I expected DFS should have the best space-complexity $O(bm)$, linear with its path, compared to BFS $O(b^d)$. Besides, BFS should have better time-complexity $O(b^d)$, compared to DFS $O(b^m)$. However, in this project, we have an opposite result. The reasons are analyzed below.

For BFS, the frontier expands with nearly same distance from the departure, it likes a circle. So the max frontier size likes the perimeter of a circle. In this graph, the max perimeter of a circle is less than 40.

For DFS, the frontier expands as the path digging deeper. In this graph, almost every vertex is

connected. If the path gets side-tracked, it may blindly dig rather deep before reaching the destination. In that case, the path has no upper boundary, so does the frontier.

Since solutions are dense, DFS can be faster than BFS.

For GBFS, time and space complexities are both $O(b^m)$, seems bad. However, the path's direction is guided by straight-line distance heuristic. It always goes a right direction, unless it is block by an obstacle. Even if it is blocked, it will be guided to another right direction again. This good heuristic can give GBFS dramatic improvement.

To sum up, these performance differences result from graph's characteristic and vertices pairs' positions.

6. Does BFS always find the shortest path?

Yes.

7. Does GBFS always go "straight" to the goal, or are there cases where it gets side-tracked?

No. It plans to go "straight" to the goal. However, when it is blocked by an obstacle, it will return to an accessible vertex which is second nearest to the goal. And then it plans a new path going "straight" to the goal. The procedure repeats. In this case it gets side-tracked.

---------- There are appendixes in the next pages. ----------

Appendix A: BFS's performance on (6,13) to (7,10)

```
vertices=275, edges=641
start=(6,13), goal=(7,10), vertices: 91 and 106
iter=1, frontier=0, popped=91 (6,13), depth=0, dist2goal=3.2
  push 79 (5,13)
  push 92 (6,14)
iter=2, frontier=1, popped=79 (5,13), depth=1, dist2goal=3.6
  push 80 (5,14)
iter=3, frontier=1, popped=92 (6,14), depth=1, dist2goal=4.1
  push 93 (6,15)
  push 107 (7,15)
iter=4, frontier=2, popped=80 (5,14), depth=2, dist2goal=4.5
  push 81 (5,15)
iter=5, frontier=2, popped=93 (6,15), depth=2, dist2goal=5.1
  push 94 (6,16)
  push 108 (7,16)
. . .
. . .
iter=51, frontier=12, popped=158 (12,19), depth=7, dist2goal=10.3
  push 171 (13,19)
  push 172 (13,20)
iter=52, frontier=13, popped=159 (12,20), depth=7, dist2goal=11.2
iter=53, frontier=12, popped=122 (8,12), depth=7, dist2goal=2.2
  push 106 (7,10)
  push 121 (8,11)
-----------------
  vertex 91 (6,13)
  vertex 92 (6,14)
  vertex 107 (7,15)
  vertex 124 (8,15)
  vertex 142 (9,15)
  vertex 141 (9,14)
  vertex 140 (9,13)
  vertex 122 (8,12)
  vertex 106 (7,10)

search algorithm = BFS
total iterations = 53
max frontier size= 19
vertices visited = 67/275
path length      = 8
```

Appendix B: DFS's performance on (6,13) to (7,10)

```
vertices=275, edges=641
start=(6,13), goal=(7,10), vertices: 91 and 106
iter=1, frontier=0, popped=91 (6,13), depth=0, dist2goal=3.2
  push 79 (5,13)
  push 92 (6,14)
iter=2, frontier=1, popped=92 (6,14), depth=1, dist2goal=4.1
  push 80 (5,14)
  push 93 (6,15)
  push 107 (7,15)
iter=3, frontier=3, popped=107 (7,15), depth=2, dist2goal=5.0
  push 108 (7,16)
  push 124 (8,15)
  push 125 (8,16)
iter=4, frontier=5, popped=125 (8,16), depth=3, dist2goal=6.1
  push 126 (8,17)
iter=5, frontier=5, popped=126 (8,17), depth=4, dist2goal=7.1
  push 109 (7,17)
  push 127 (8,18)
. . .
. . .
iter=84, frontier=66, popped=142 (9,15), depth=66, dist2goal=5.4
iter=85, frontier=65, popped=123 (8,13), depth=65, dist2goal=3.2
iter=86, frontier=64, popped=122 (8,12), depth=64, dist2goal=2.2
  push 106 (7,10)
-----------------
  vertex 91 (6,13)
  vertex 92 (6,14)
  vertex 107 (7,15)
  . . .
  . . .
  vertex 119 (8,9)
  vertex 137 (9,10)
  vertex 138 (9,11)
  vertex 139 (9,12)
  vertex 122 (8,12)
  vertex 106 (7,10)

search algorithm = DFS
total iterations = 86
max frontier size= 67
vertices visited = 151/275
path length      = 65
```

Appendix C: GBFS's performance on (6,13) to (7,10)

vertices=275, edges=641
start=(6,13), goal=(7,10), vertices: 91 and 106
iter=1, frontier=0, popped=91 (6,13), depth=0, dist2goal=3.2
  push 79 (5,13)
  push 92 (6,14)
iter=2, frontier=1, popped=79 (5,13), depth=1, dist2goal=3.6
  push 80 (5,14)
iter=3, frontier=1, popped=92 (6,14), depth=1, dist2goal=4.1
  push 93 (6,15)
  push 107 (7,15)
iter=4, frontier=2, popped=80 (5,14), depth=2, dist2goal=4.5
  push 81 (5,15)
iter=5, frontier=2, popped=107 (7,15), depth=2, dist2goal=5.0
  push 108 (7,16)
  push 124 (8,15)
  push 125 (8,16)
iter=6, frontier=4, popped=93 (6,15), depth=2, dist2goal=5.1
  push 94 (6,16)
iter=7, frontier=4, popped=124 (8,15), depth=3, dist2goal=5.1
  push 142 (9,15)
iter=8, frontier=4, popped=81 (5,15), depth=3, dist2goal=5.4
  push 69 (4,15)
  push 82 (5,16)
iter=9, frontier=5, popped=142 (9,15), depth=4, dist2goal=5.4
  push 141 (9,14)
iter=10, frontier=5, popped=141 (9,14), depth=5, dist2goal=4.5
  push 140 (9,13)
iter=11, frontier=5, popped=140 (9,13), depth=6, dist2goal=3.6
  push 122 (8,12)
  push 123 (8,13)
  push 139 (9,12)
iter=12, frontier=7, popped=122 (8,12), depth=7, dist2goal=2.2
  push 106 (7,10)
  push 121 (8,11)
---
  vertex 91 (6,13)
  vertex 92 (6,14)
  vertex 107 (7,15)
  vertex 124 (8,15)
  vertex 142 (9,15)
  vertex 141 (9,14)
  vertex 140 (9,13)

vertex 122 (8,12)
vertex 106 (7,10)

search algorithm = GBFS
total iterations = 12
max frontier size= 8
vertices visited = 21/275
path length = 8