

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH



BÀI TẬP LỚN 1  
**TÌM KIẾM (SEARCHING)**

---

**Môn Nhập môn trí tuệ nhân tạo**

---

GVHD: Vương Bá Thịnh  
SV thực hiện: Nguyễn Văn Thi – 1513164  
Nguyễn Tuyết Nga – 1512111

Tp. Hồ Chí Minh, Tháng 5/2018



## Mục lục

<b>1</b>	<b>Depth First Search</b>	<b>2</b>
1.1	Giải thuật . . . . .	2
1.2	Kết quả chi tiết từng bài . . . . .	3
1.3	Cách chạy . . . . .	4
<b>2</b>	<b>Breadth First Search</b>	<b>7</b>
2.1	Giải thuật . . . . .	7
2.2	Kết quả chi tiết từng bài . . . . .	8
2.3	Cách chạy . . . . .	9
<b>3</b>	<b>Best First Search</b>	<b>11</b>
3.1	Giải thuật . . . . .	11
3.2	Hàm lượng giá . . . . .	11
3.3	Kết quả chi tiết từng bài . . . . .	12
3.4	Cách chạy . . . . .	13
<b>4</b>	<b>Hiệu năng về thời gian của Best First Search so với Breadth First Search</b>	<b>15</b>
4.1	So sánh hiệu năng Best First Search so với Breadth First Search . . . . .	15
4.2	Cách chạy . . . . .	16
<b>5</b>	<b>Tham khảo</b>	<b>17</b>

# 1 Depth First Search

## 1.1 Giải thuật

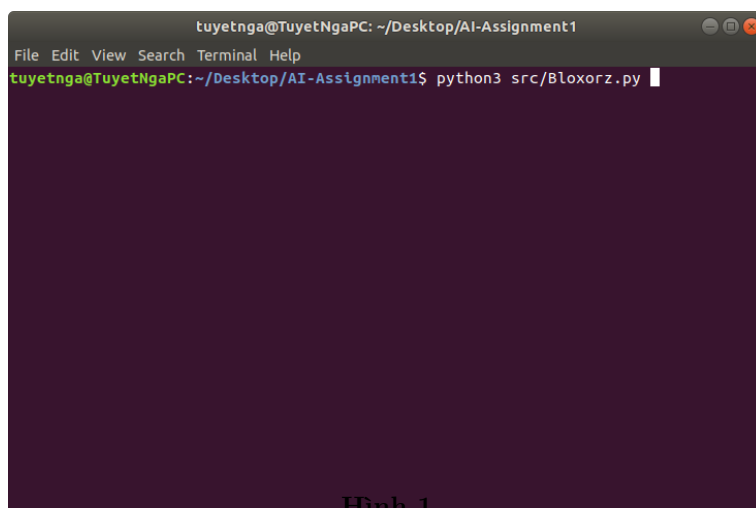
```
def depth_first_search(initState, maxdepth = 50):
    initState.getMap().name += "(depth first search)"
    counter = 0
    stack = list()
    explored = list()
    stack.append(initState)
    while stack:
        state = stack.pop()
        check, value = state.isGoal()
        counter += 1
        if check == True:
            if value == None:
                return [state, counter]
        explored.append(state)
        if state.value > maxdepth:
            continue
        children = nextState(state)
        for i in children:
            if i not in explored:
                stack.append(i)
    return [initState, -1]
```

## 1.2 Kết quả chi tiết từng bài

Bài	Thời gian tìm kiếm kết quả	Chiều cao	Bước chuyển	Trạng thái
1	0.250633 s	50	50	73
2	1.285382 s	100	101	230
3	0.406117 s	50	47	66
4	0.576354 s	50	40	76
5	1.385429 s	100	91	143
6	0.762524 s	100	58	81
7	1.087740 s	100	66	144
8	0.577750 s	50	48	88
9	0.505913 s	100	65	96
10	1.652326 s	100	93	169
11	0.481144 s	50	49	65
12	3.030183 s	100	91	372
13	1.237571 s	100	65	134
14	3.403988 s	200	125	372
15	3.878507 s	100	82	370
16	0.636102 s	50	50	96
17	7.143340 s	300	295	698
18	1.643250 s	100	98	176
19	1.559267 s	100	87	154
20	Not found	0	0	-1
21	1.421129 s	100	72	179
22	2.075379 s	200	124	223
23	57.001248 s	300	301	4060
24	0.941831 s	100	82	123
25	2.173283 s	100	97	209
26	3.384305 s	200	150	310
27	2.548648 s	100	93	255
28	2.133764 s	200	120	214
29	21.487838 s	200	168	1609
30	3.033576 s	200	130	278
31	4.277469 s	200	195	423
32	2.301186 s	200	144	224
33	3.359139 s	200	96	284

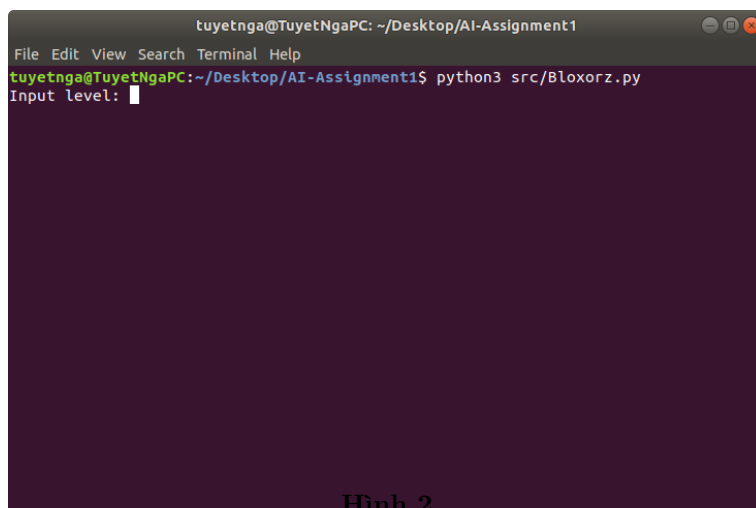
### 1.3 Cách chạy

Bước 1: Chạy lệnh `python3 src/Bloxorz.py` để vào chương trình



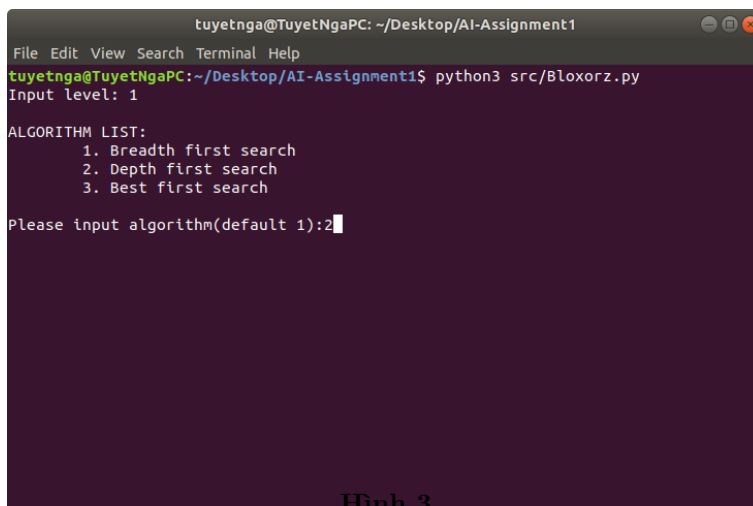
Hình 1

Bước 2: Nhập bàn muốn chơi



Hình 2

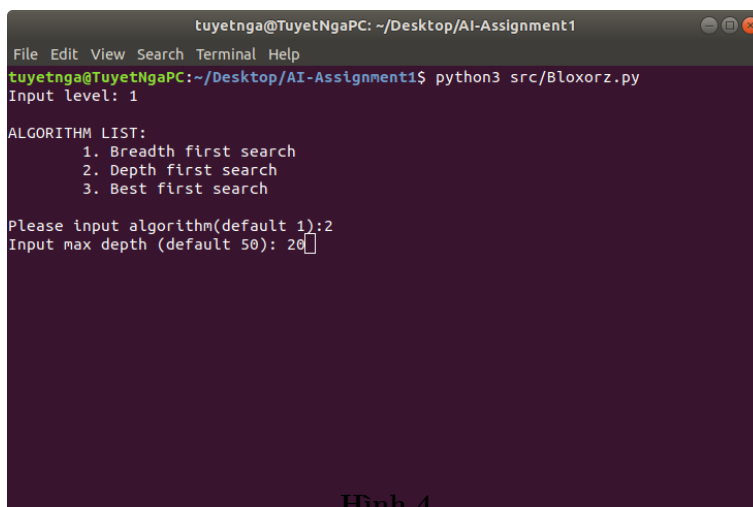
Bước 3: Nhập 2 để chọn giải thuật Depth First Search



```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help
tuyetnga@TuyetNgaPC:~/Desktop/AI-Assignment1$ python3 src/Bloxorz.py
Input level: 1
ALGORITHM LIST:
  1. Breadth first search
  2. Depth first search
  3. Best first search
Please input algorithm(default 1):2
```

Hình 3

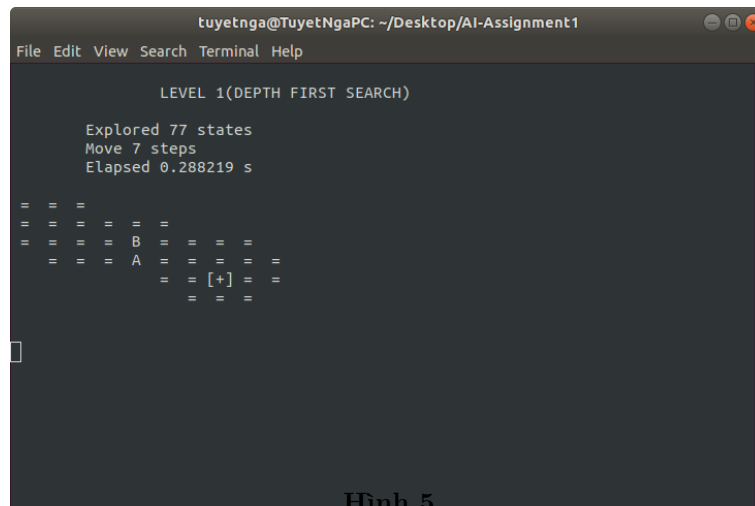
Bước 4: Nhập chiều dài của cây để giới hạn thời gian chạy



```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help
tuyetnga@TuyetNgaPC:~/Desktop/AI-Assignment1$ python3 src/Bloxorz.py
Input level: 1
ALGORITHM LIST:
  1. Breadth first search
  2. Depth first search
  3. Best first search
Please input algorithm(default 1):2
Input max depth (default 50): 20
```

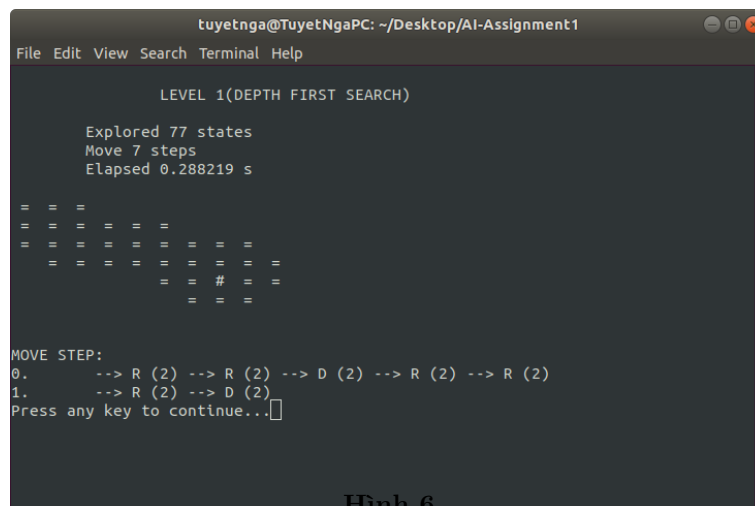
Hình 4

Bước 5: Demo step-by-step, trong đó có tổng các trạng thái tìm được, lời giải tốt nhất và thời gian chạy



Hình 5

Bước 6: Xuất bước chạy rõ hơn sau khi đã có kết quả



Hình 6

## 2 Breadth First Search

### 2.1 Giải thuật

```
def breadth_first_search(initState):
    initState.getMap().name += "(breadth first search)"
    counter = 0
    queue = list()
    explored = list()
    queue.append(initState)
    while queue:
        state = queue.pop(0)
        check, value = state.isGoal()
        counter += 1
        explored.append(state)
        if check == True:
            queue.clear()
            explored.clear()
            if value == None:
                return [state, counter]
        children = nextState(state)
        for i in children:
            if i not in explored:
                queue.append(i)
    return [initState, -1]
```



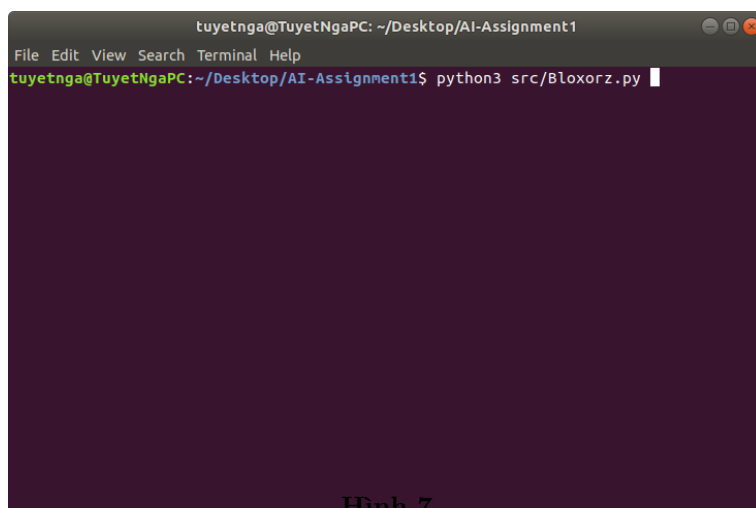


## 2.2 Kết quả chi tiết từng bài

Bài	Thời gian tìm kiếm kết quả	Bước chuyển	Trạng thái
1	0.346270 s	7	85
2	4.163358 s	17	626
3	1.073998 s	19	174
4	0.904767 s	28	99
5	3.377141 s	34	319
6	2.277367 s	35	204
7	2.427308 s	45	271
8	0.906574 s	11	108
9	1.094362 s	24	199
10	27.803828 s	57	2219
11	0.751215 s	47	89
12	4.972121 s	60	556
13	2.421999 s	46	235
14	6.123488 s	67	618
15	4.726335 s	64	490
16	2.966731 s	40	342
17	14.307251 s	106	1379
18	3.665601 s	92	364
19	3.103062 s	67	296
20	3.309927 s	61	347
21	3.996603 s	71	395
22	6.003093 s	77	610
23	23.857360 s	75	2066
24	1.112096 s	57	150
25	3.084451 s	55	295
26	8.722139 s	110	956
27	4.643386 s	71	403
28	11.883237 s	100	1102
29	9.317949 s	110	947
30	13.321815 s	114	1077
31	8.140459 s	86	759
32	9.089880 s	129	1051
33	3.757065 s	65	362

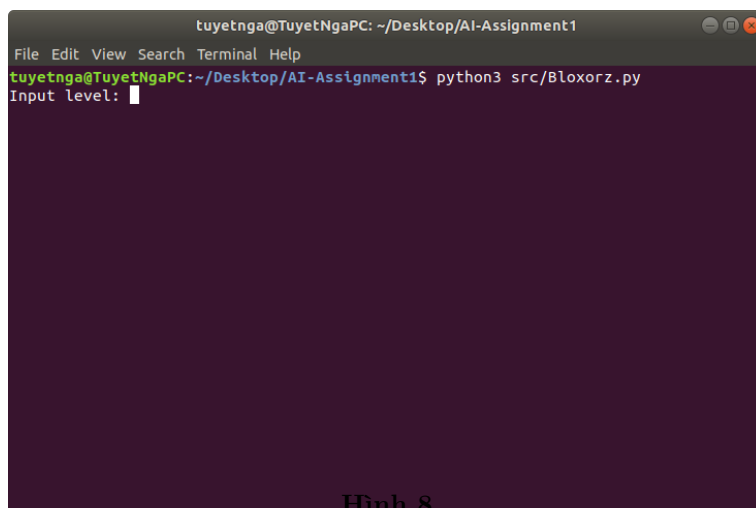
## 2.3 Cách chạy

Bước 1: Chạy lệnh `python3 src/Bloxorz.py` để vào chương trình



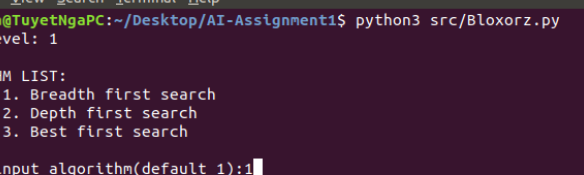
Hình 7

Bước 2: Nhập bàn muốn chơi



Hình 8

Bước 3: Nhập 1 để chọn giải thuật Breadth First Search



```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help
tuyetnga@TuyetNgaPC:~/Desktop/AI-Assignment1$ python3 src/Bloxorz.py
Input level: 1

ALGORITHM LIST:
1. Breadth first search
2. Depth first search
3. Best first search

Please input algorithm(default 1):1
```

## Hình 9

Bước 4: Xem demo step-by-step và xuất bước chạy rõ hơn sau khi tìm được kết quả

```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help

LEVEL 1(BREADTH FIRST SEARCH)

Explored 85 states
Move 7 steps
Elapsed 0.375520 s

= = =
= = = = =
= = = = = = =
= = = = = = =
= = = = = # =
= = =

MOVE STEP:
0. --> R (2) --> D (2) --> D (2) --> R (2) --> R (2)
1. --> D (2) --> R (2)
Press any key to continue...
```

### Hình 10

## 3 Best First Search

### 3.1 Giải thuật

```
def best_first_search(initState):
    initState.getMap().name += "(best first search)"
    counter = 0
    queue = PriorityQueue()
    explored = list()
    queue.put(initState)
    while not queue.empty():
        state = queue.get()
        check, value = state.isGoal()
        counter += 1
        if check == True:
            if value == None:
                return [state, counter]
        explored.append(state)
        children = nextState(state)
        for i in children:
            if i not in explored:
                queue.put(i)
    return [initState, -1]
```

### 3.2 Hàm lượng giá

Hàm lượng giá tính khoảng cách từ điểm hiện tại tới mục tiêu cần tìm

```
def distance(self):
    A = self.block.A
    g = self.lsGoal[0]
    if self.block.control == A and self.goalA:
        g = self.goalA[0]
    elif self.block.control == self.block.B and self.goalB:
        A = self.block.B
        g = self.goalB[0]
    if g.type == CHANGECONTROL:
        g = self.lsGoal[0]
    vectorAG = [g.x - A.x, g.y - A.y]
    return sqrt(vectorAG[0]**2 + vectorAG[1]**2)
```

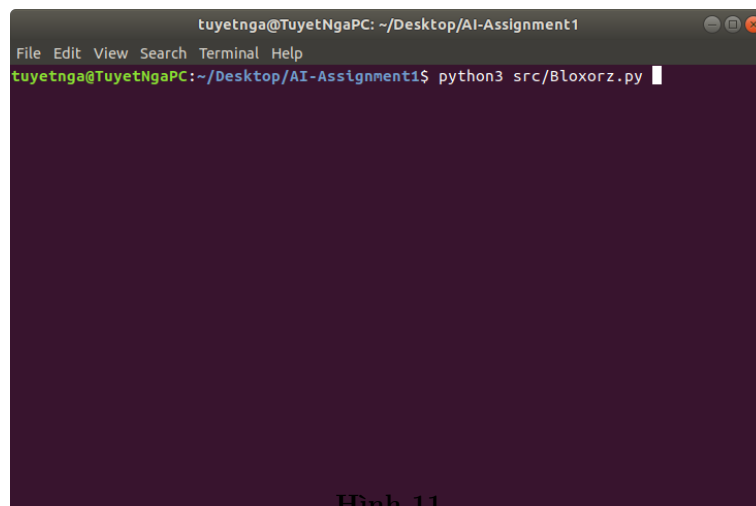


### 3.3 Kết quả chi tiết từng bài

Bài	Thời gian tìm kiếm kết quả	Bước chuyển	Trạng thái
1	0.030900 s	7	9
2	0.221260 s	18	39
3	0.298219 s	30	53
4	0.846096 s	39	86
5	2.576809 s	34	218
6	0.893582 s	40	87
7	1.480511 s	59	186
8	0.106365 s	11	14
9	6.626361 s	42	1013
10	31.932690 s	58	2336
11	0.609098 s	47	80
12	3.656315 s	66	355
13	0.858556 s	48	87
14	4.849752 s	72	497
15	6.472636 s	84	609
16	0.769421 s	38	112
17	6.513172 s	109	602
18	1.998310 s	90	225
19	1.351635 s	67	143
20	5.236975 s	0	-1
21	1.302634 s	71	139
22	2.916878 s	81	316
23	5.284086 s	12820	7268
24	1.962009 s	82	251
25	2.109535 s	55	234
26	83.173022 s	0	-1
27	2.490003 s	71	254
28	4.411172 s	110	381
29	29.840586 s	127	2156
30	4.848795 s	114	481
31	8.243943 s	92	770
32	5.548439 s	129	625
33	2.770999 s	80	277

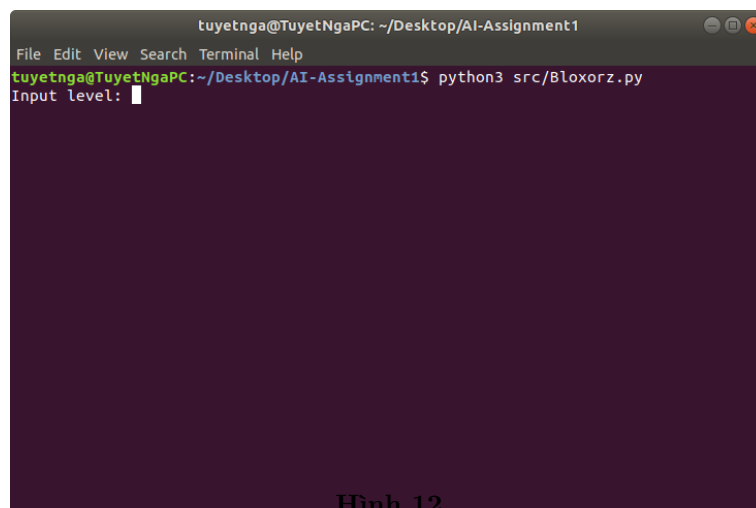
### 3.4 Cách chạy

Bước 1: Chạy lệnh `python3 src/Bloxorz.py` để vào chương trình



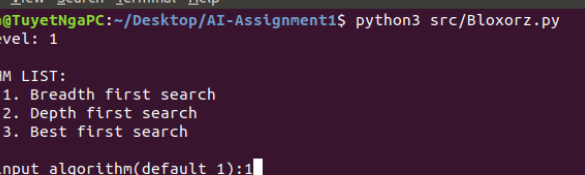
Hình 11

Bước 2: Nhập bàn muốn chơi



Hình 12

Bước 3: Nhập 3 để chọn giải thuật Best First Search



```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help
tuyetnga@TuyetNgaPC:~/Desktop/AI-Assignment1$ python3 src/Bloxorz.py
Input level: 1

ALGORITHM LIST:
1. Breadth first search
2. Depth first search
3. Best first search

Please input algorithm(default 1):1
```

### Hình 13

Bước 4: Xem demo step-by-step và xuất bước chạy rõ hơn sau khi tìm được kết quả

```
tuyetnga@TuyetNgaPC: ~/Desktop/AI-Assignment1
File Edit View Search Terminal Help

LEVEL 1(BEST FIRST SEARCH)

Explored 9 states
Move 7 steps
Elapsed 0.063536 s

= = =
= = = = =
= = = = = = =
= = = = = = =
= = = = = # =
= = =

MOVE STEP:
0. --> R (2) --> R (2) --> D (2) --> R (2) --> R (2)
1. --> R (2) --> D (2)
Press any key to continue...
```

### Hình 14

#### 4 Hiệu năng về thời gian của Best First Search so với Breadth First Search

Bài	Breadth First Search			Best First Search			Hiệu năng
	Thời gian	Bước	Trạng thái	Thời gian	Bước	Trạng thái	
1	0.346270 s	7	85	0.030900 s	7	9	8.923672
2	4.163358 s	17	626	0.221260 s	18	39	5.314460
3	1.073998 s	19	174	0.298219 s	30	53	27.767183
4	0.904767 s	28	99	0.846096 s	39	86	93.610841
5	3.377141 s	34	319	2.576809 s	34	218	76.301493
6	2.277367 s	35	204	0.893582 s	40	87	39.237505
7	2.427308 s	45	271	1.480511 s	59	186	60.993949
8	0.906574 s	11	108	0.106365 s	11	14	11.732633
9	1.094362 s	24	199	6.626361 s	42	1013	605.499917
10	27.803828 s	57	2219	31.932690 s	58	2336	114.849977
11	0.751215 s	47	89	0.609098 s	47	80	81.081714
12	4.972121 s	60	556	3.656315 s	66	355	73.536324
13	2.421999 s	46	235	0.858556 s	48	87	35.448239
14	6.123488 s	67	618	4.849752 s	72	497	79.199175
15	4.726335 s	64	490	6.472636 s	84	609	136.948312
16	2.966731 s	40	342	0.769421 s	38	112	25.934977
17	14.307251 s	106	1379	6.513172 s	109	602	45.523574
18	3.665601 s	92	364	1.998310 s	90	225	54.515208
19	3.103062 s	67	296	1.351635 s	67	143	43.558105
20	3.309927 s	61	347	5.236975 s	0	-1	0
21	3.996603 s	71	395	1.302634 s	71	139	32.59353
22	6.003093 s	77	610	2.916878 s	81	316	48.589585
23	23.857360 s	75	2066	5.284086 s	12820	7268	22.148662
24	1.112096 s	57	150	1.962009 s	82	251	176.424427
25	3.084451 s	55	295	2.109535 s	55	234	68.39256
26	8.722139 s	110	956	83.173022 s	0	-1	0
27	4.643386 s	71	403	2.490003 s	71	254	53.624726
28	11.883237 s	100	1102	4.411172 s	110	381	37.120963
29	9.317949 s	110	947	29.840586 s	127	2156	320.248437
30	13.321815 s	114	1077	4.848795 s	114	481	36.397405
31	8.140459 s	86	759	8.243943 s	92	770	101.271231
32	9.089880 s	129	1051	5.548439 s	129	625	61.039739
33	3.757065 s	65	362	2.770999 s	80	277	73.754353



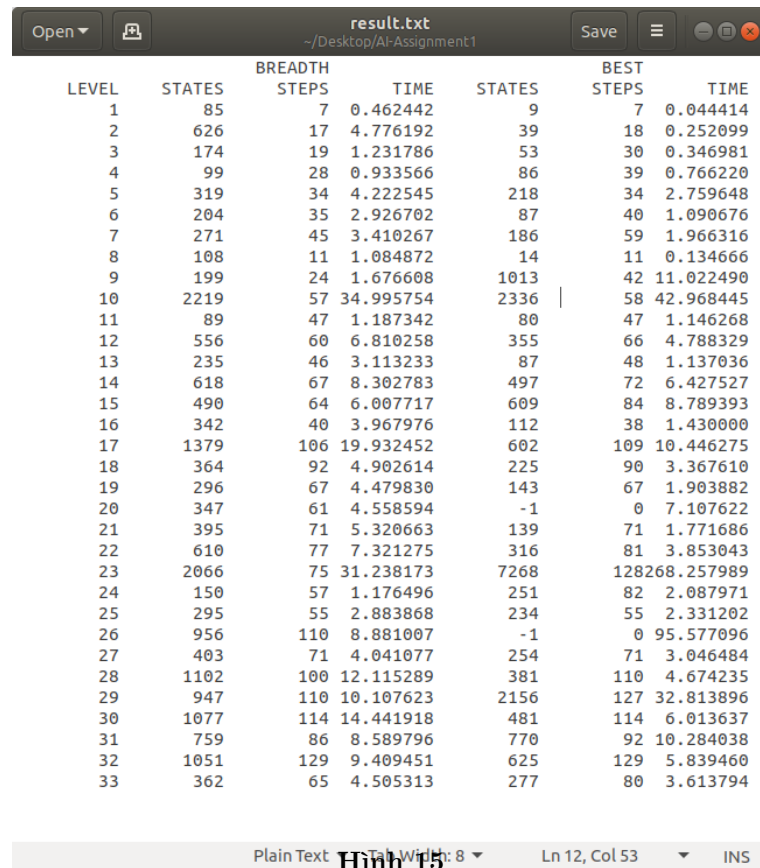
## 4.1 So sánh hiệu năng Best First Search so với Breadth First Search

black

- Thời gian trung bình là 80.3509964, thời gian tìm kiếm lời giải ít hơn so với Breadth First Search do nó luôn tìm đường đến mục tiêu gần nhất  $\Rightarrow$  Thời gian tối ưu hơn
  - Trạng thái trung bình là 20.1818182, đa số trạng thái của Best First Search ít hơn so với Breadth First Search nên về bộ nhớ sẽ chứa ít hơn, tuy nhiên do vài bài bị dính vấn đề tối ưu cục bộ nên trạng thái nhiều hơn dẫn đến tổng số trạng thái trung bình của Best First Search sẽ nhiều hơn Breadth First Search  $\Rightarrow$  Nhìn tổng quát thì Best First Search vẫn sẽ tốt hơn
  - Mà nhìn chung Breadth First Search sẽ tối ưu hơn so với Depth First Search
- $\Rightarrow$  Vậy Best First Search tối ưu về thời gian và bộ nhớ hơn so với Breadth First Search và Depth First Search

## 4.2 Cách chạy

Chạy lệnh `python3 src/Bloxors.py runall` để chạy hết các màn game 1 lượt để xuất ra kết quả các trạng thái, bước chuyển và thời gian chạy giữa Best First Search và Breadth First Search (Riêng Depth First Search do phải giới hạn chiều dài của cây nên khó so sánh hết các màn 1 lần được). Kết quả sẽ hiện trên file `result.txt` như hình dưới:



The screenshot shows a text editor window titled 'result.txt' with the file path '~\Desktop\AI-Assignment1'. The window contains a table with 7 columns: LEVEL, STATES, BREADTH STEPS, TIME, STATES, BEST STEPS, and TIME. The table lists search results for levels 1 through 33, comparing different search methods (Breadth-First Search and Best-First Search) across various states. The status bar at the bottom indicates 'Plain Text', 'Tab Width: 8', 'Ln 12, Col 53', and 'INS'.

LEVEL	STATES	BREADTH STEPS	TIME	STATES	BEST STEPS	TIME
1	85	7	0.462442	9	7	0.044414
2	626	17	4.776192	39	18	0.252099
3	174	19	1.231786	53	30	0.346981
4	99	28	0.933566	86	39	0.766220
5	319	34	4.222545	218	34	2.759648
6	204	35	2.926702	87	40	1.090676
7	271	45	3.410267	186	59	1.966316
8	108	11	1.084872	14	11	0.134666
9	199	24	1.676608	1013	42	11.022490
10	2219	57	34.995754	2336	58	42.968445
11	89	47	1.187342	80	47	1.146268
12	556	60	6.810258	355	66	4.788329
13	235	46	3.113233	87	48	1.137036
14	618	67	8.302783	497	72	6.427527
15	490	64	6.007717	609	84	8.789393
16	342	40	3.967976	112	38	1.430000
17	1379	106	19.932452	602	109	10.446275
18	364	92	4.902614	225	90	3.367610
19	296	67	4.479830	143	67	1.903882
20	347	61	4.558594	-1	0	7.107622
21	395	71	5.320663	139	71	1.771686
22	610	77	7.321275	316	81	3.853043
23	2066	75	31.238173	7268	128268	257989
24	150	57	1.176496	251	82	2.087971
25	295	55	2.883868	234	55	2.331202
26	956	110	8.881007	-1	0	95.577096
27	403	71	4.041077	254	71	3.046484
28	1102	100	12.115289	381	110	4.674235
29	947	110	10.107623	2156	127	32.813896
30	1077	114	14.441918	481	114	6.013637
31	759	86	8.589796	770	92	10.284038
32	1051	129	9.409451	625	129	5.839460
33	362	65	4.505313	277	80	3.613794

Hình 15

## 5 Tham khảo

- Tài liệu học tập của môn học
- [http://en.wikipedia.org/wiki/Depth-first\\_search](http://en.wikipedia.org/wiki/Depth-first_search)
- [http://en.wikipedia.org/wiki/Breadth-first\\_search](http://en.wikipedia.org/wiki/Breadth-first_search)
- [http://en.wikipedia.org/wiki/Hill\\_climbing](http://en.wikipedia.org/wiki/Hill_climbing)
- <http://www.coolmath-games.com/0-bloxorz>