Full name: Cao Lê Minh Hiếu.

Student ID: 18120368.

Class ID: 18_21

The degree of the project's completeness (self-evaluated): 10.

I) Overview Table

Name	Idea	Time Complexity	Space Complexity	Complete	Optimal	Type (Informed or Uninformed)
Breadth-First-Search	explores all of the successors nodes at the present level prior to moving on to the nodes at the next level. The goal test is applied to a node when it	$O(B^d)$	$O(B^d)$	Yes if B is finite	Yes if step costs are all identical.	Uninformed

	is first generated.					
Depth-First-Search	explores the node branch as far as possible before being forced to backtrack and expand other nodes. The goal test is applied to a node when it is first generated.	$O(B^m)$	O(Bm)	No	No	Uninformed
Uniform-Cost-Search	Expands the node with n with the lowest path cost g(n) using the frontier as a priority queue ordered by g. The goal test is applied to a node when it is first selected	$O(B^{1+floor(C*/ep)})$	$O(B^{1+floor(C*/ep)})$	Yes if B is finite and step costs >= ep for positive ep.	Yes	Uninformed

	from the priority queue.					
A*	Expands the node n by evaluating the combination of g(n) (above UCS) and a admissible heuristic function h(n)	$O(B^d)$	$O(B^d)$	Yes	Yes if the heuristic function is admissible or consistent.	Informed
Greedy Best-First-Search	Expands the node that is closest to the goal using a heuristic function f(n) = h(n).	$O(B^m)$	O(Bm)	No	No but often efficient.	Informed
Bidirectional Search	Using BFS for 2 directions simultaneously, one forward search from Start, one backward search from	$O(B^{d/2})$	$O(B^{d/2})$	Yes if B is finite and both directions use BFS	Yes if step costs are all identical and both directions use BFS.	Uninformed

	Goal. The goal test with a check to see whether the frontiers of two searches intersect.					
Iterative-Deepening-Search	DFS with a limited depth from 1 – n. If the goal is not found with the depth in [1:n], then stops to avoid RUN TIME ERROR.	O(B^d)	O(Bd)	Yes if B is finite.	Yes if if step costs are all identical.	Uninformed.
Beam Search	An optimization of Best First Search. Begin with k generated states. At each step, all the successors of all k states	O(km)	O(km)	No	No but often efficient.	Informed.

	which are not explored are generated. If any node is a goal, the algorithm halts. Otherwise, it selects the k best successors from the complete list and repeats.					

Annotation:

B: the branching factor.

d: the depth where a solution is found.

m: the height of the tree.

C*: The cost of the optimal solution path.

K: Beam Width.

II) Notes in the implementation

- All the search algorithms use an extended list to store nodes which are explored. In some algorithms, the backtracking technique is used to guarantee the Completeness.

- All the informed search algorithms 's completeness and optimality depend strongly on the efficient heuristic function.

- The heuristic functions in informed searches are calculated using the Euclidean straight-line-distance from the current state to the goal. I chose this function because the weights of the edges are calculated using the Euclidean distance between two nodes which are adjacent. By doing this, the heuristic function is guaranteed to be CONSISTENT – a strict condition for the OPTIMALITY of A*, etc.
- Beam Width in this implementation is 2. It can be changed by modifying the k variable.
- Bidirectional Search: The Start and Goal are set to Orange color. The intersect Node is set to Purple Color.
- Iterative-Deepening-Search: The output Picture bellow shows the last step in the algorithms.
- BFS, DFS, Bidirectional Search, Iterative-Deepening Search: all step costs are identical and equal to 1.

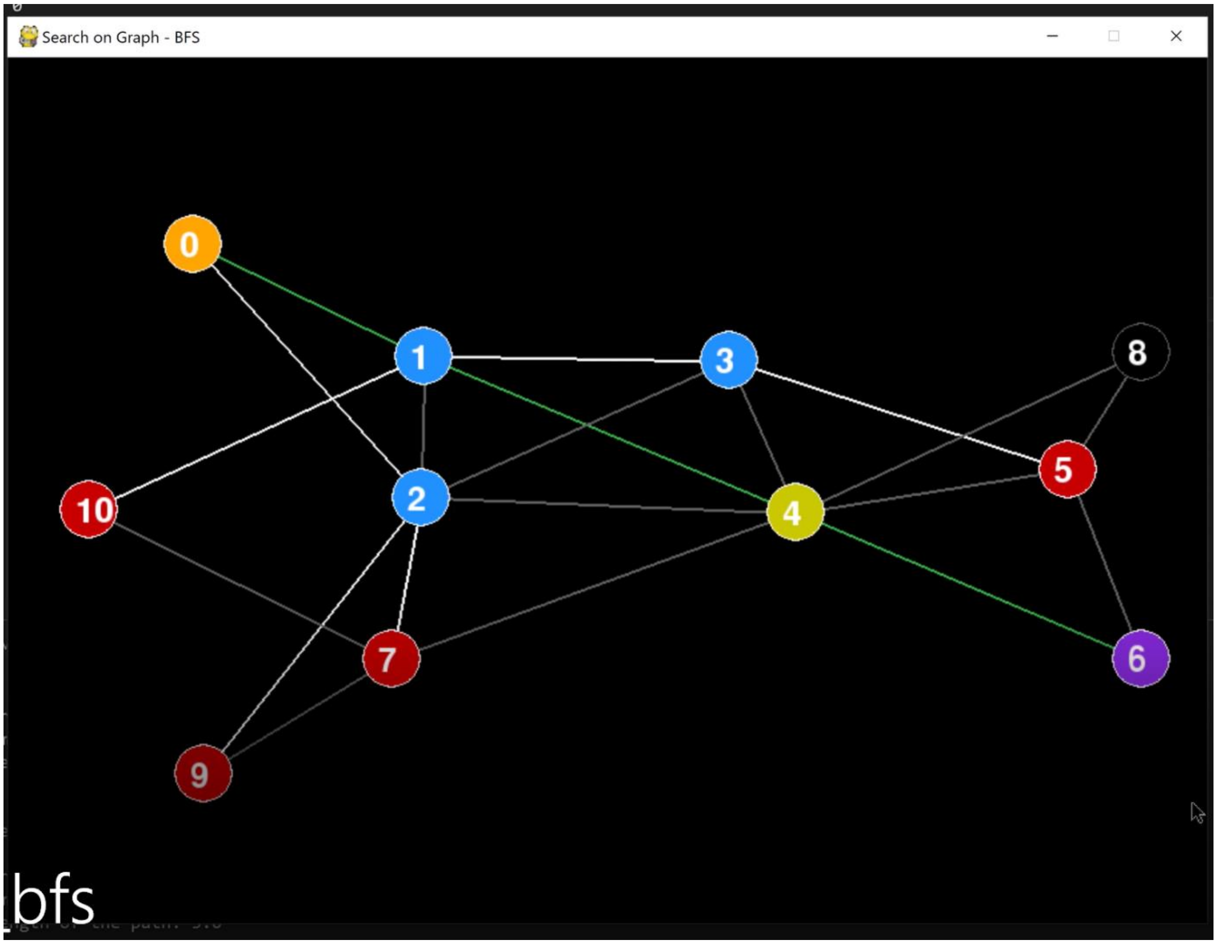
III) The difference between UCS Vs A*.

UCS	A*
An uninformed search that doesn’t use any domain knowledge. It expands the node n using a priority queue ordered by a actual path cost from start to n, f(n) = g(n). It is guaranteed to find the optimal solution path if exists.	A informed search that adds a heuristic function h(n) (admissible or consistent) to estimate how close the current node is to the goal. It expands the node n just like UCS but with the cost function f(n) = g(n) + h(n). Hence A* search has the information to choose the successor which leads to find the goal soon. A* depends strongly on the heuristic function h(n). It is not guaranteed to find the optimal solution path if exists.

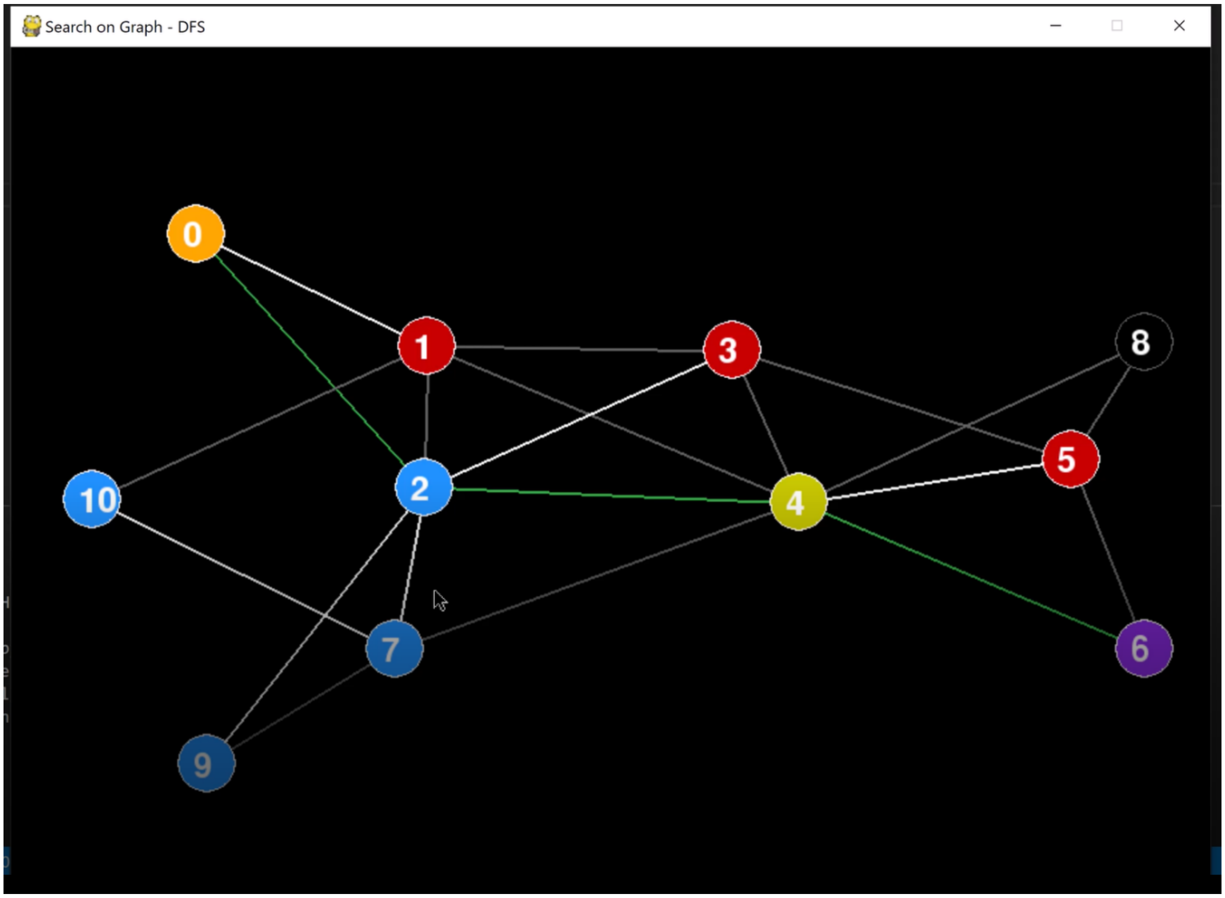
IV) Testing.

Test case 1: is input.txt. This is used to check whether the algorithms run correctly or not.

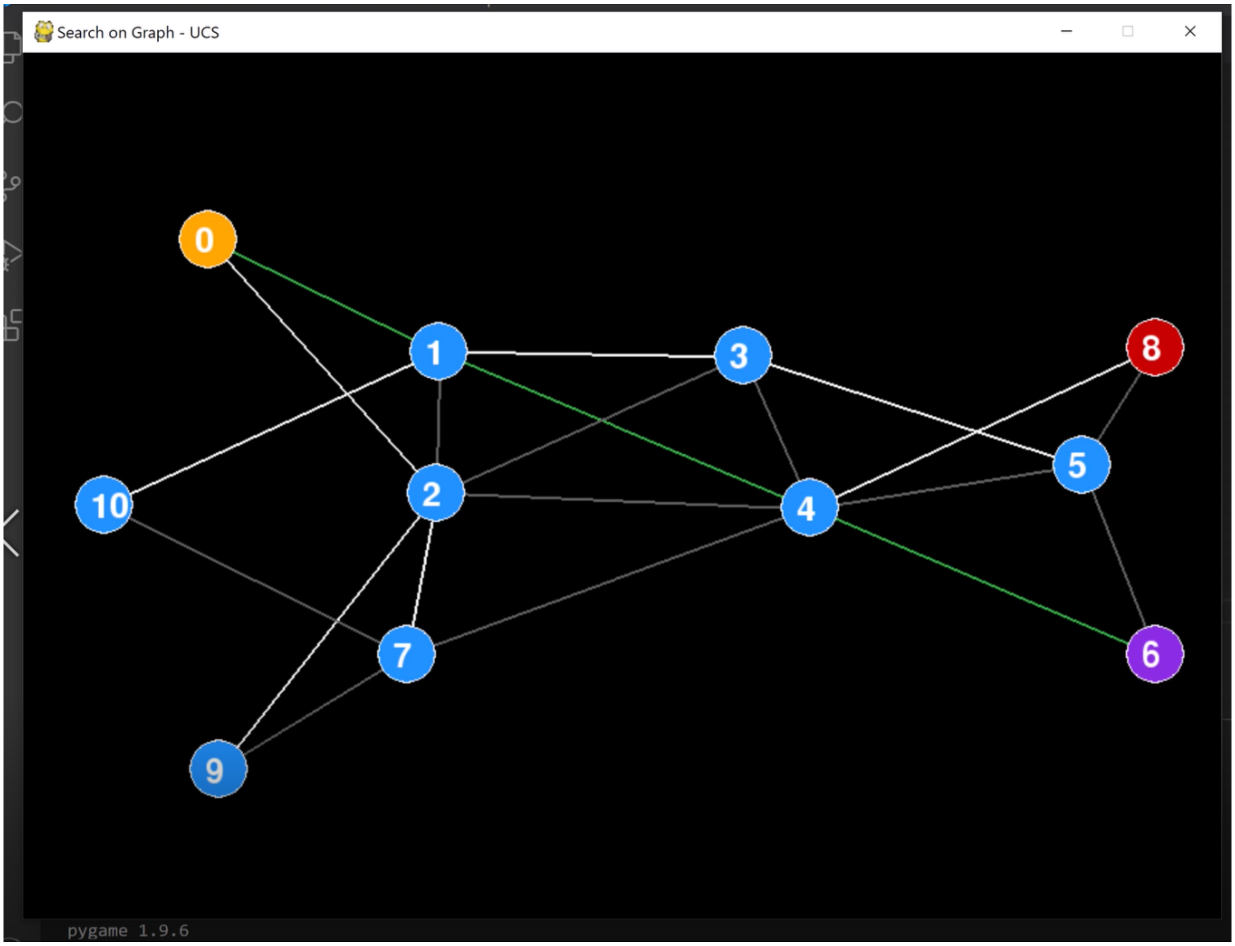
BFS:



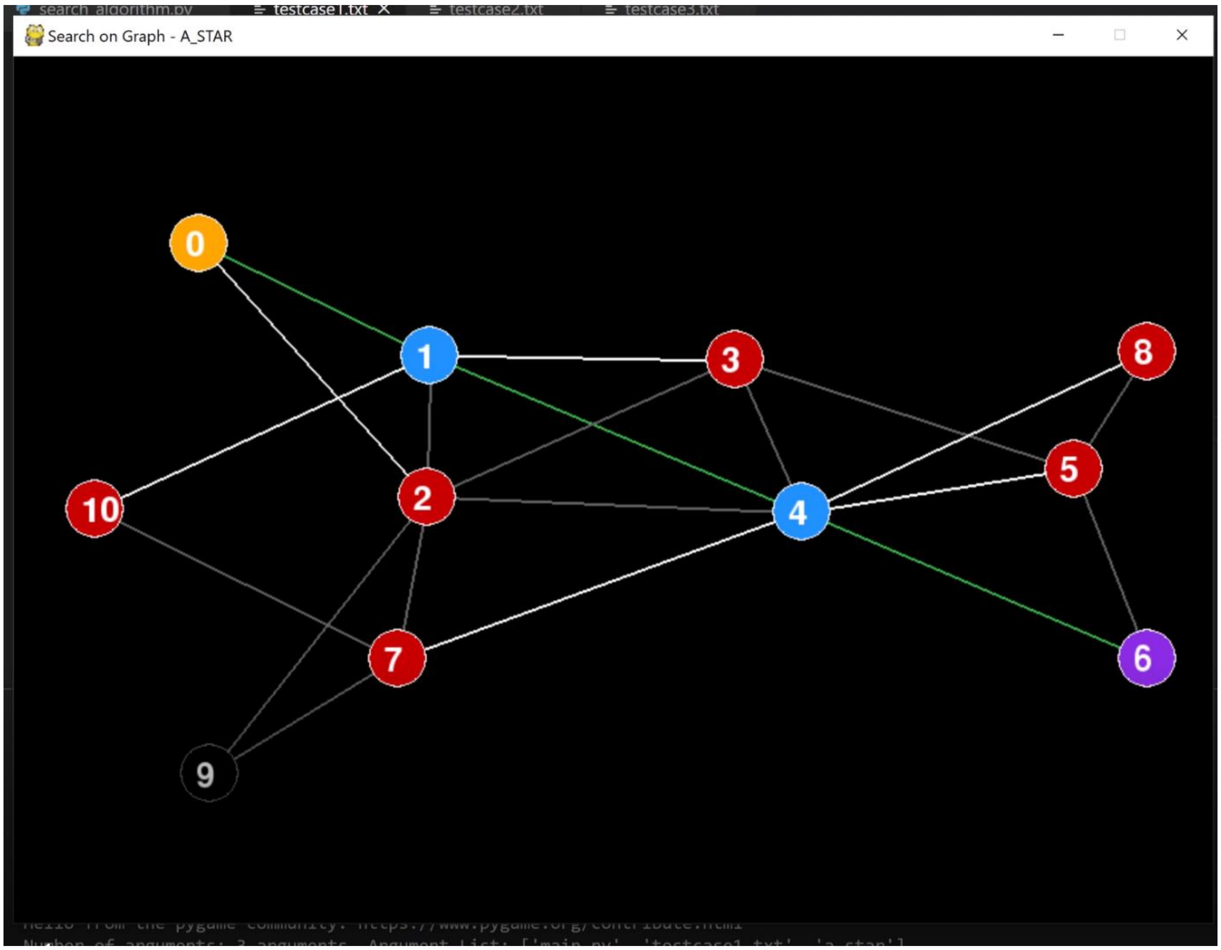
DFS:



UCS:

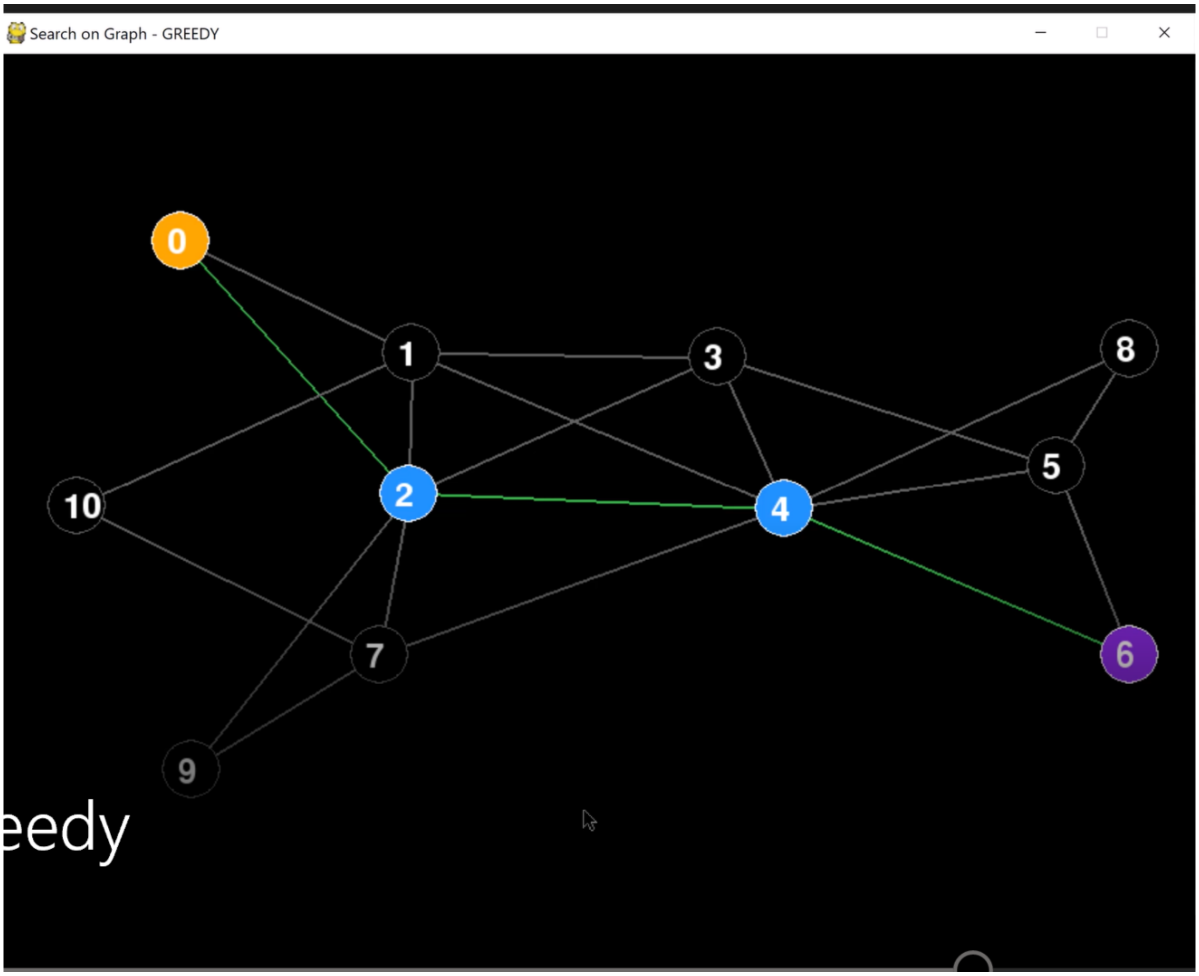


A* search:

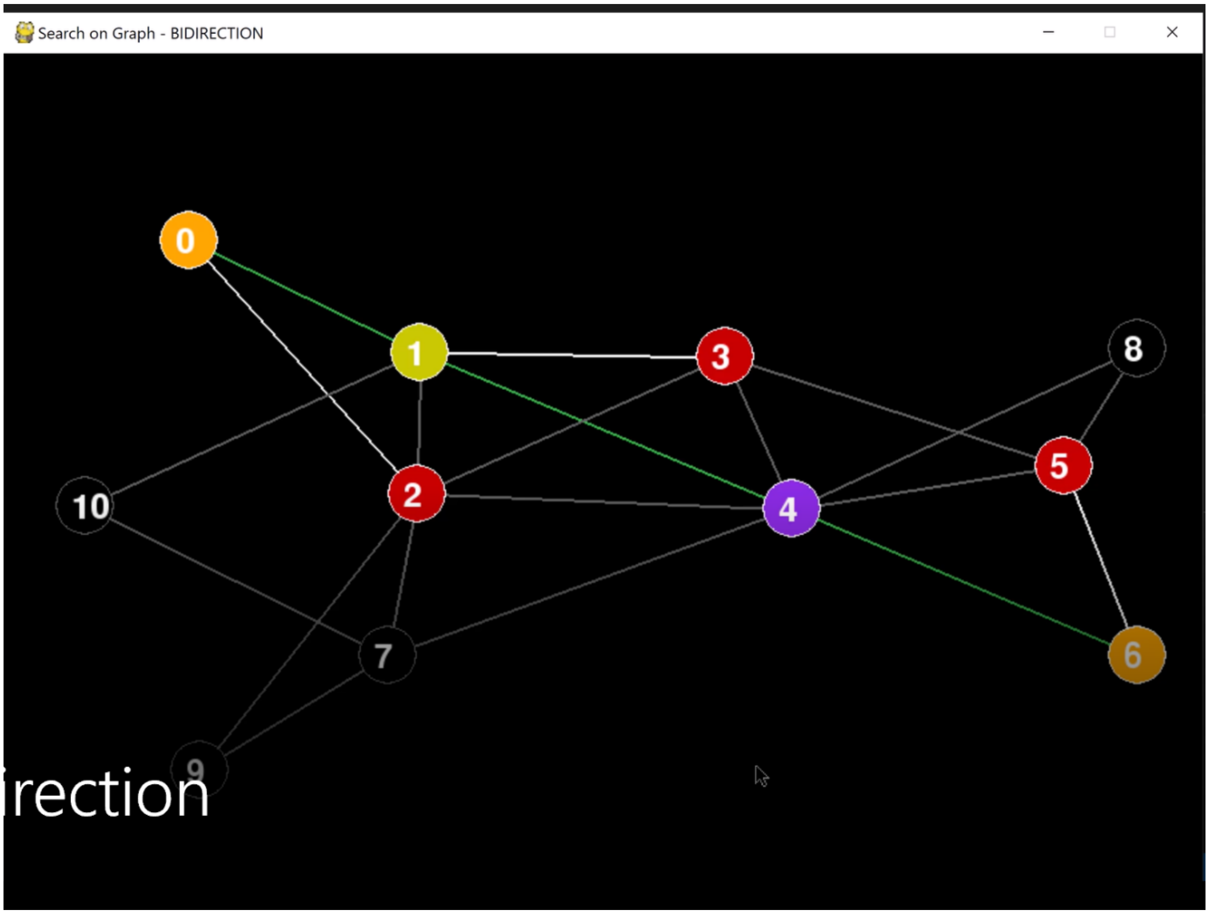


search_algorithm.py
testcase1.txt
testcase2.txt
testcase3.txt
Search on Graph - A_STAR
0
1
3
8
5
10
2
4
7
6
9

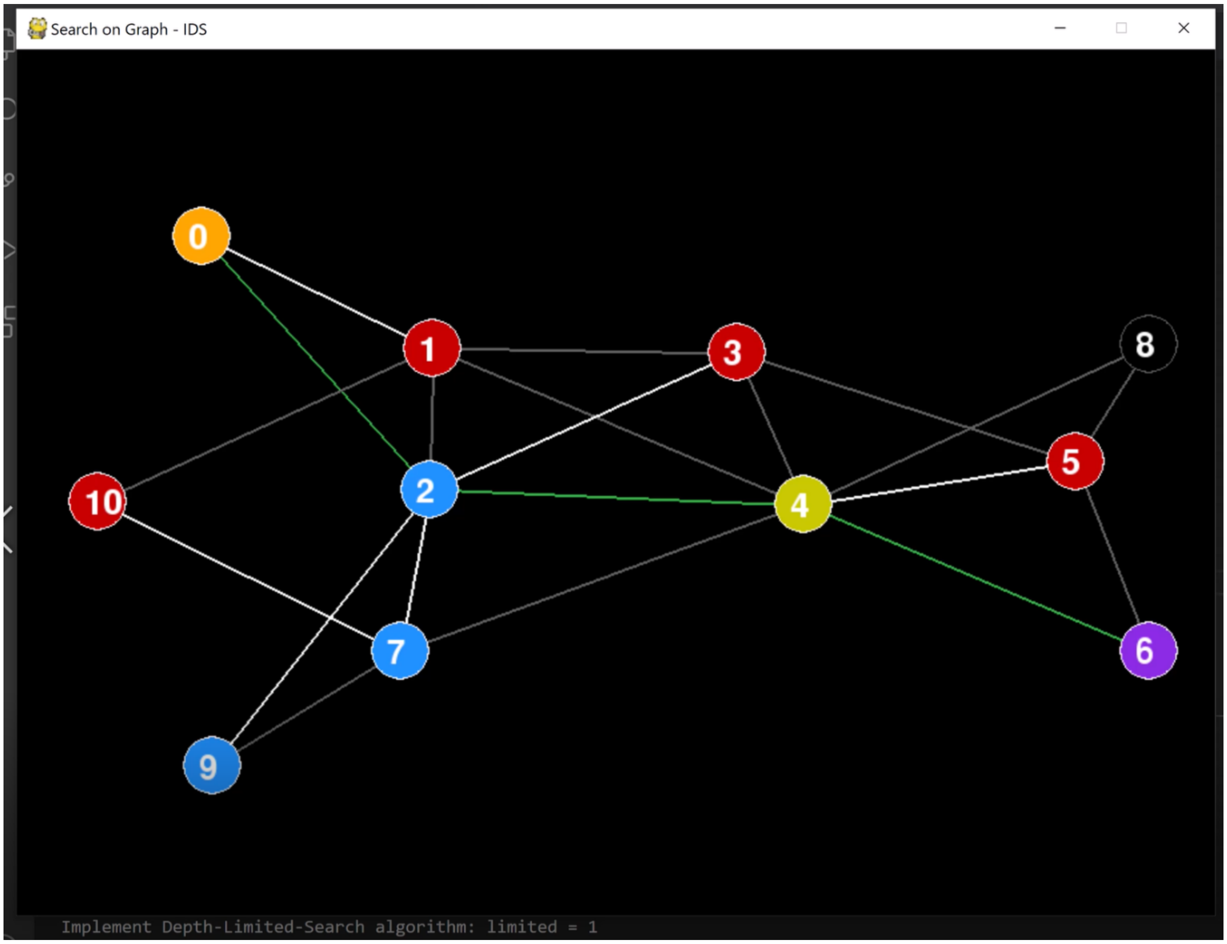
Greedy Best First Search:



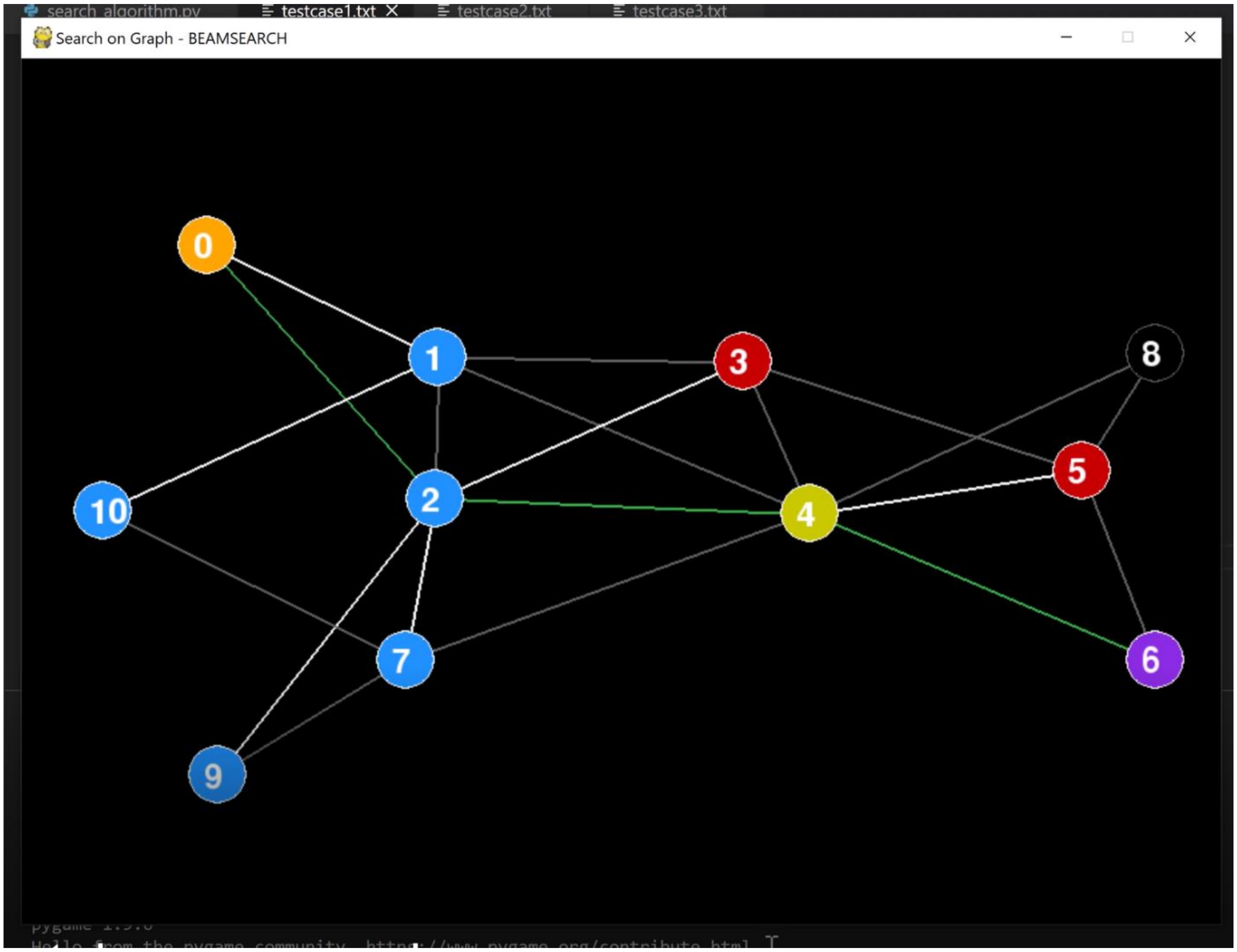
Bidirectional Search:



Iterative-Deepening-Search:



Beam Search:


testcase1.txt
testcase2.txt
testcase3.txt
Search on Graph - BEAMSEARCH
0
1
2
3
4
5
6
7
8
9
10

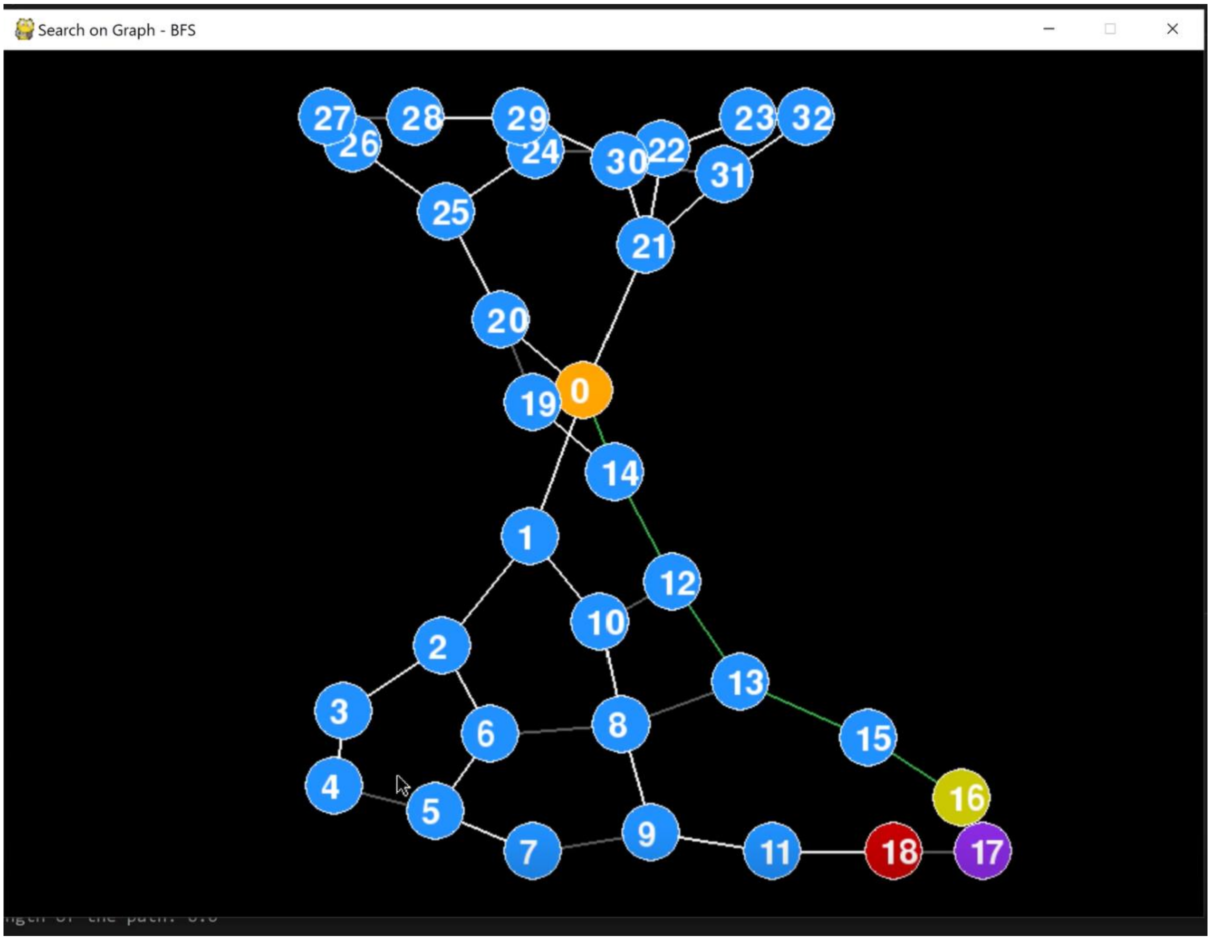
Comments:

1. BFS vs DFS: in this case is quite the same. But DFS is more efficient than BFS because it uses less memory.
2. UCS vs A* vs Greedy: UCS and A* show the optimal path, but A* is much faster (8 blue nodes in UCS vs 2 blue nodes in A*). This is because the A*'s heuristic function is consistent. On the other hand, Greedy Search is quick but close to the optimal path because it uses only the heuristic function.
3. DFS vs IDS: in this case the results of the algorithms are the same and IDS is much slower because it runs iteratively the depth from 1 to 3(3 Depth-Limited-Searches are called).
4. BFS vs Bidirectional Search: Obviously, Bidirectional Search finds the Goal faster as the result of running two BFSs, 1 from Start and 1 from Goal, intersects at 4.
5. Beam Search: It uses Beam Width = 2, as you can see two nodes 3 and 4 are only preserved. 5 is red because 4 is searching.

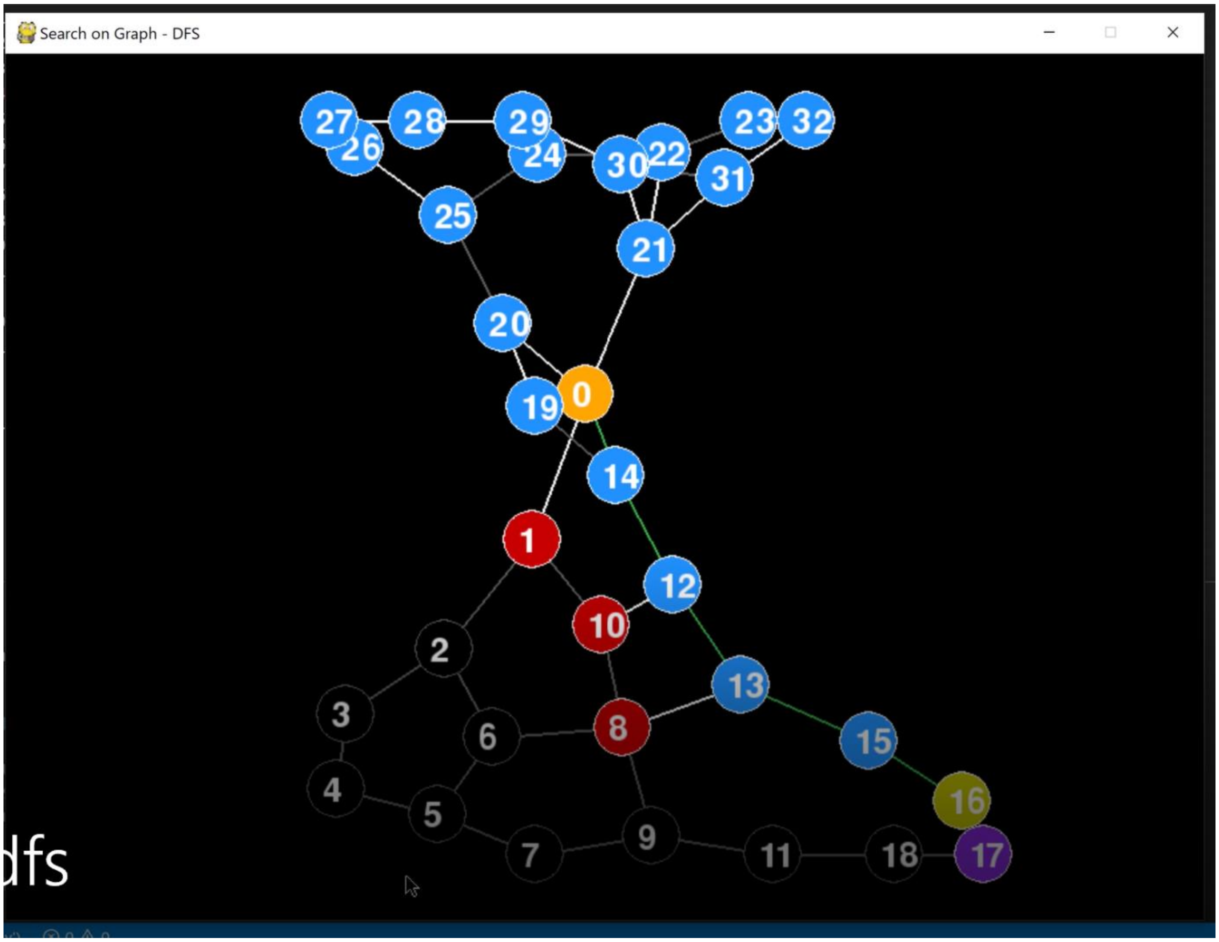
****
****

Test case 2: a large-scaled graph which has a lot of edges and vertices. This is used to show the pros and cons of the algorithms.

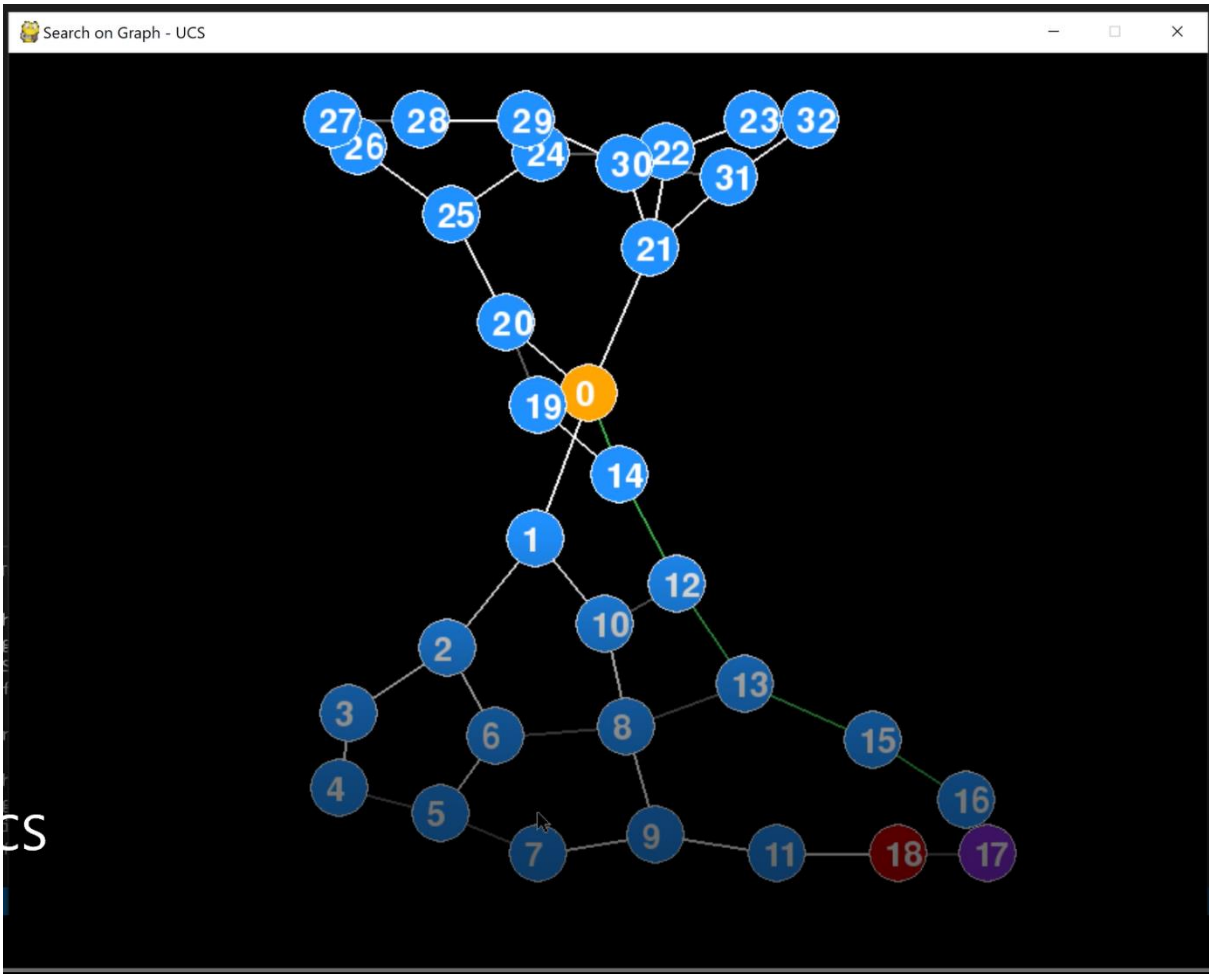
BFS:


Search on Graph - BFS
27
28
29
26
24
30
22
23
32
31
25
21
20
0
19
14
1
12
10
2
13
3
6
8
15
4
5
16
7
9
11
18
17

DFS:

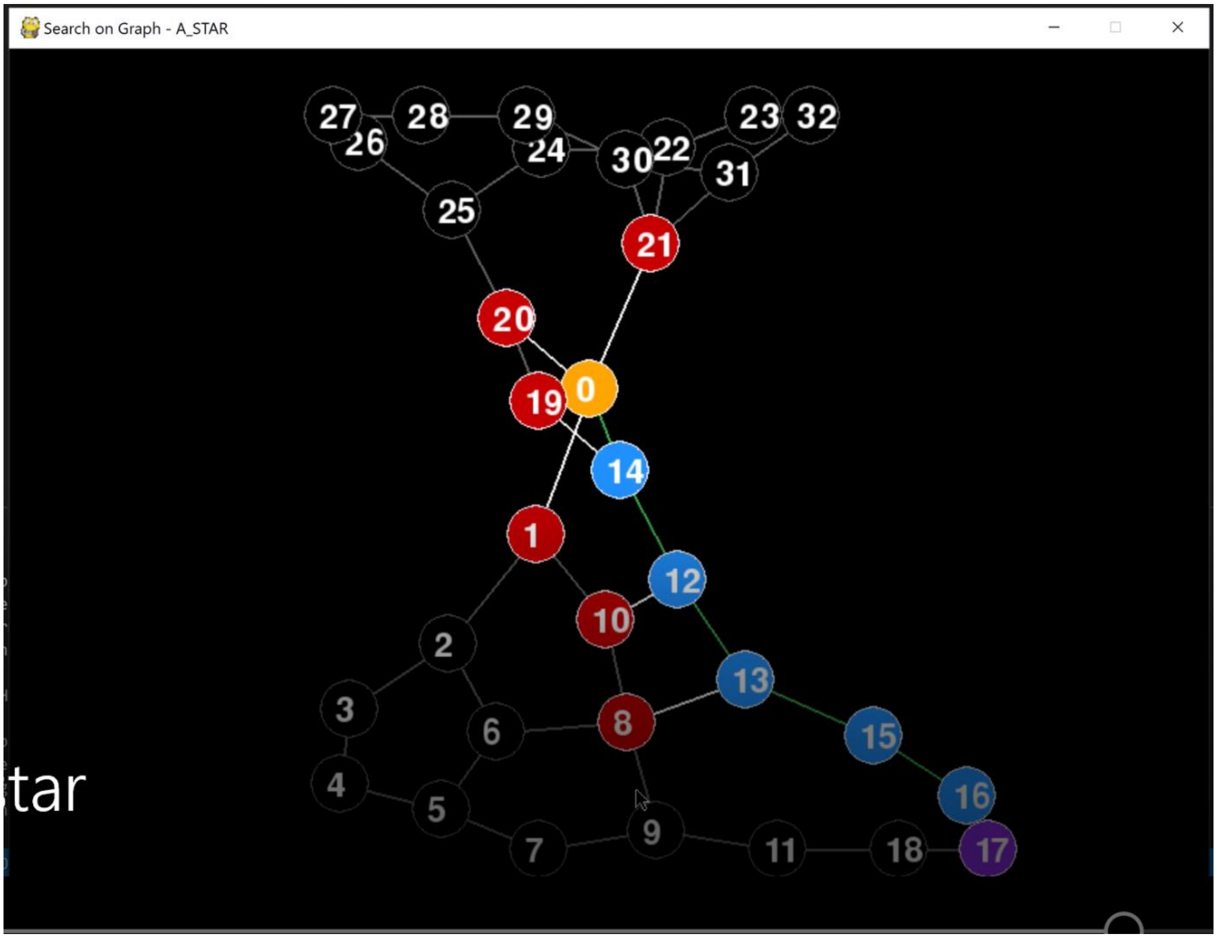

Search on Graph - DFS
27
28
29
26
24
30
22
23
32
31
25
21
20
0
19
14
1
12
10
2
13
3
6
8
15
4
5
16
9
7
11
18
17
dfs

UCS:



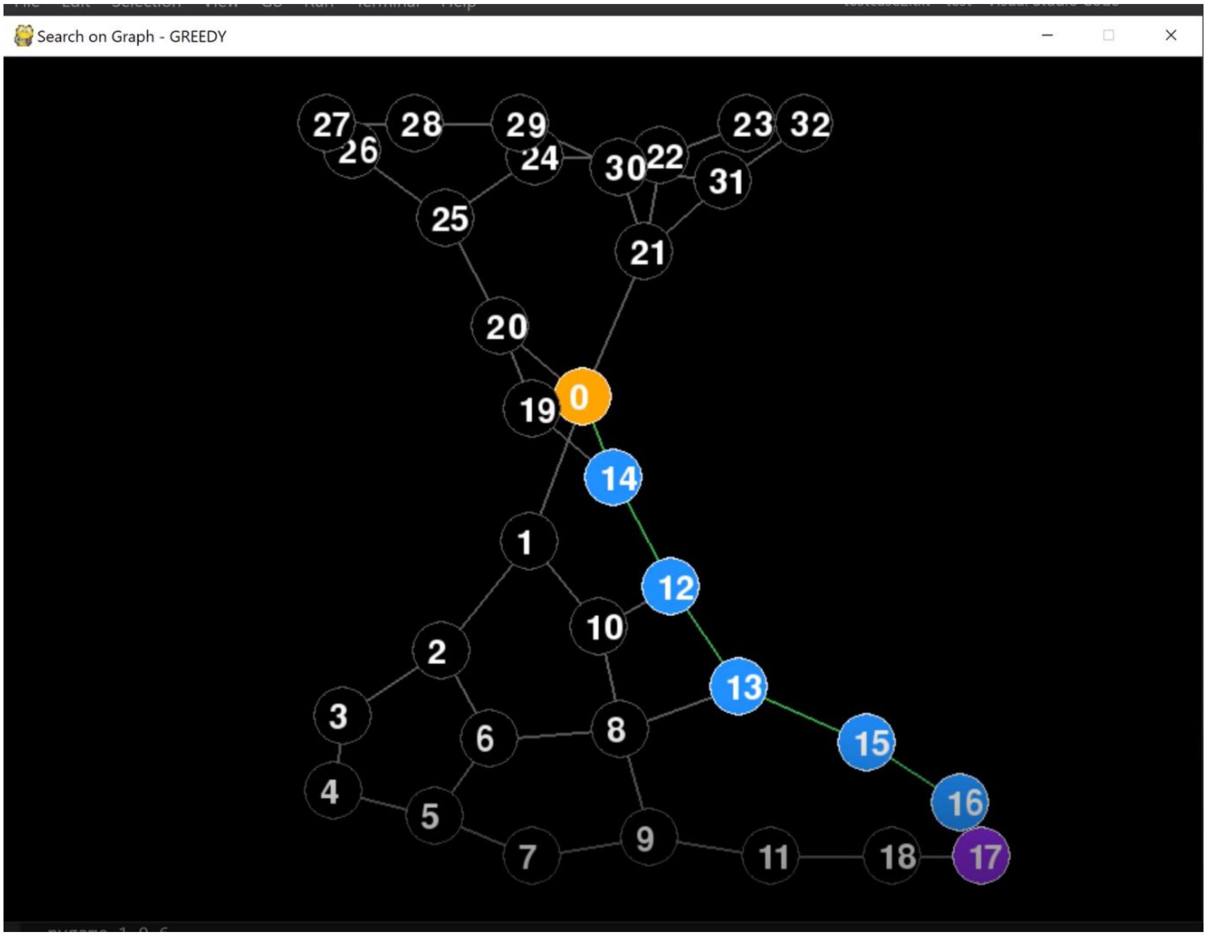
Search on Graph - UCS
27
28
29
23
32
26
24
30
22
31
25
21
20
0
19
14
1
12
10
2
13
3
6
8
15
4
5
16
CS
7
9
11
18
17

A* search:

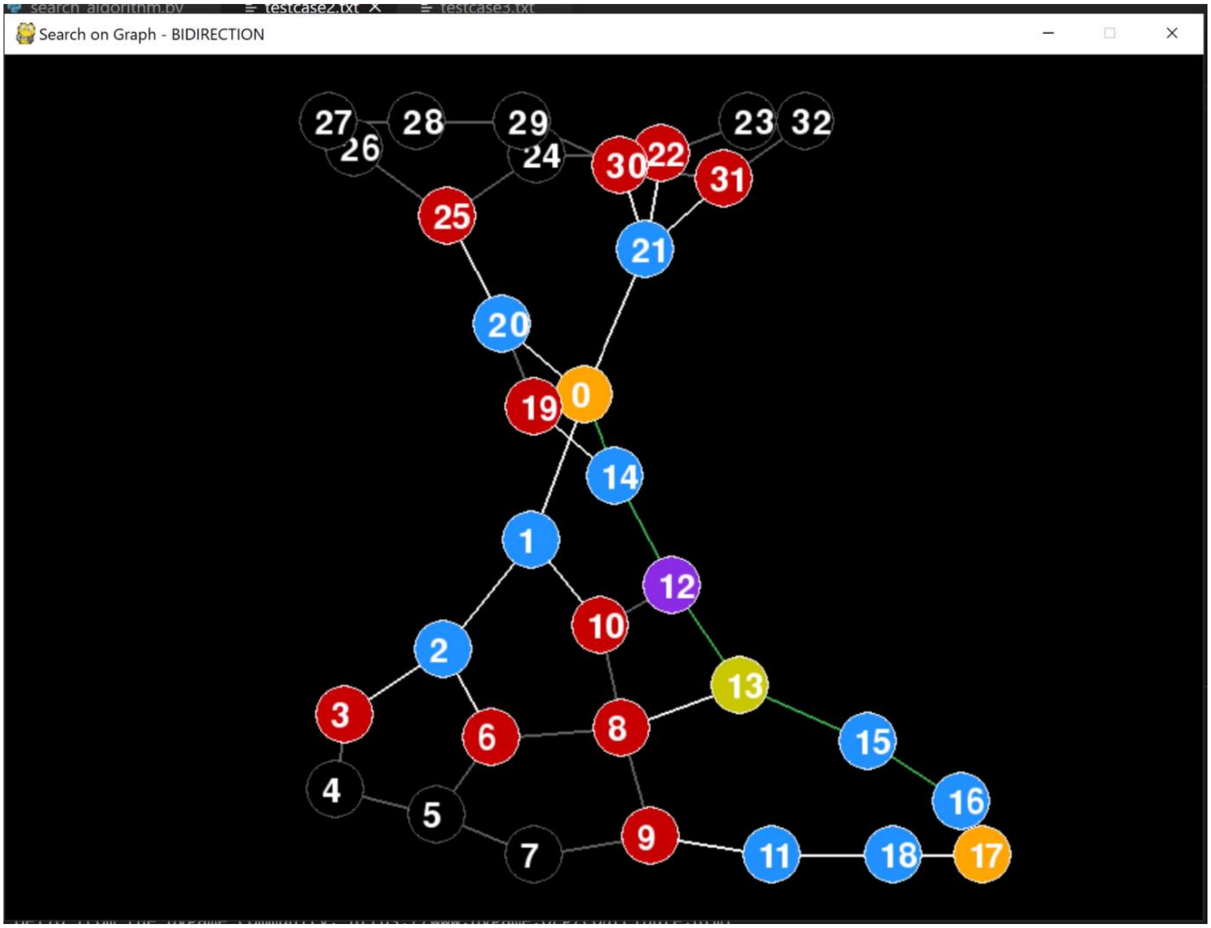


Search on Graph - A_STAR
27
28
29
26
24
30
22
23
32
31
25
21
20
19
0
14
1
12
10
2
13
3
6
8
15
4
5
7
9
11
18
16
17
tar

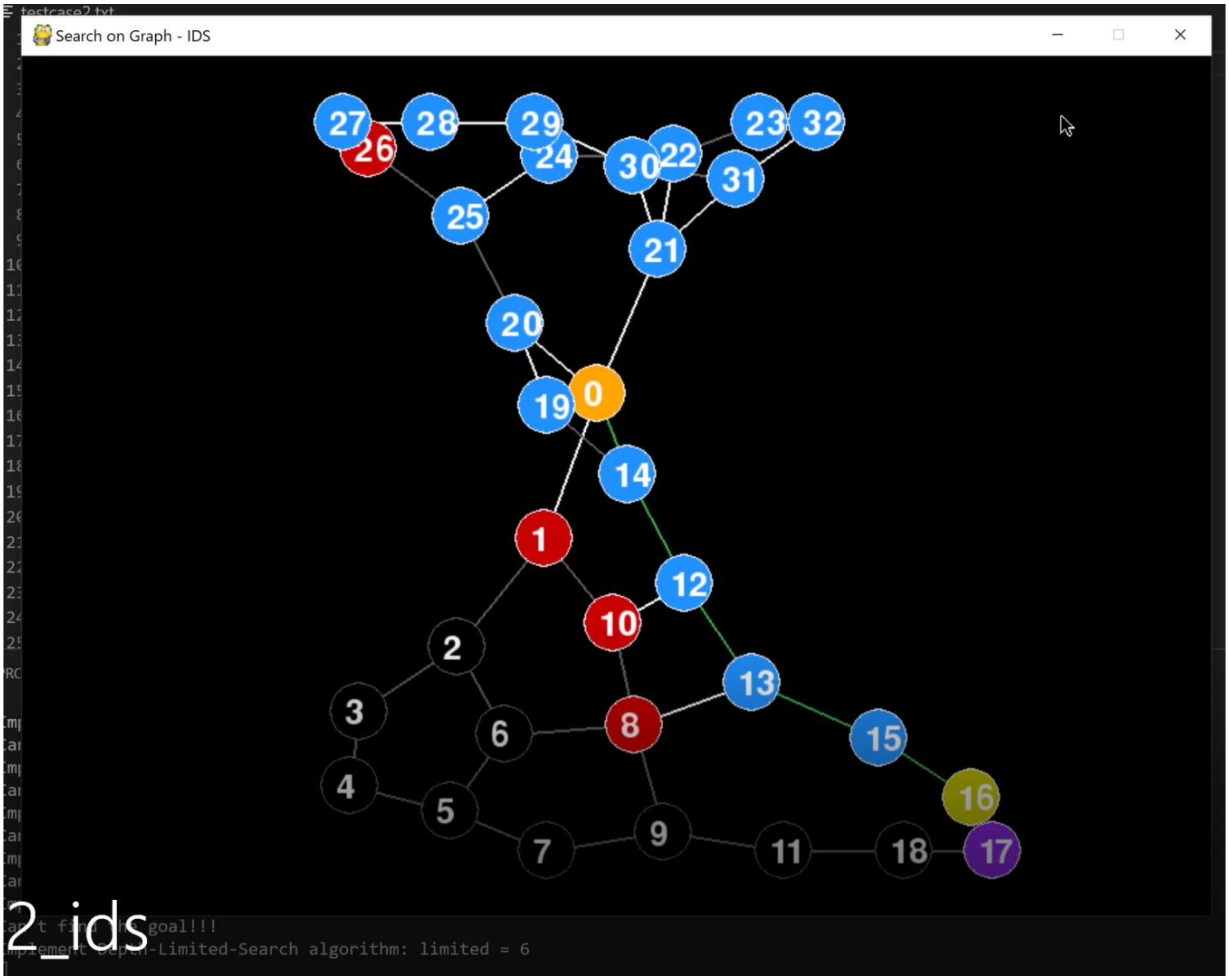
Greedy Best First Search:



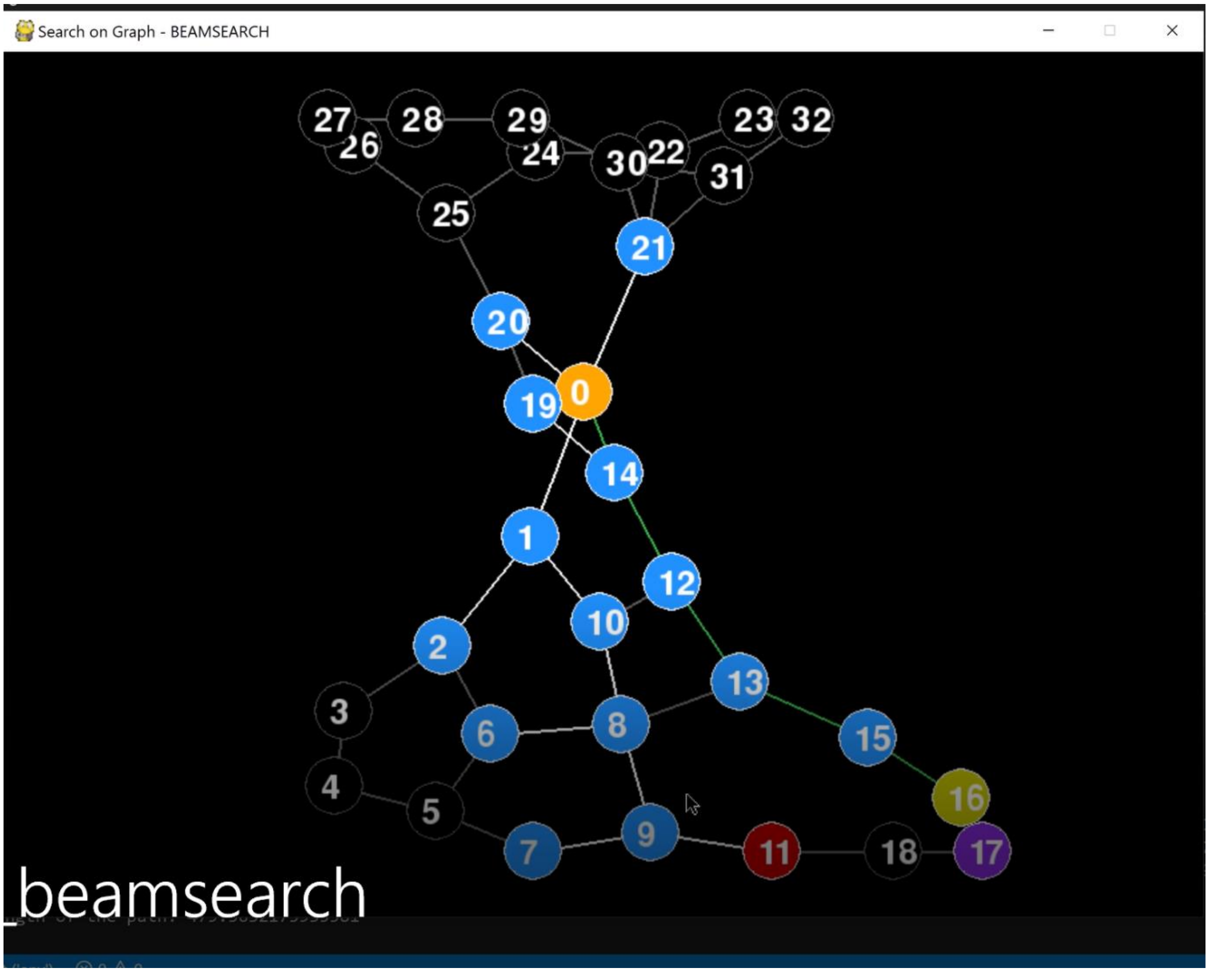
Bidirectional Search:



Iterative-Deepening-Search:



Beam Search:


Search on Graph - BEAMSEARCH
27
28
29
26
24
30
22
23
32
31
25
21
20
0
19
14
1
12
10
2
13
3
6
8
15
4
5
16
7
9
11
18
17
beamsearch

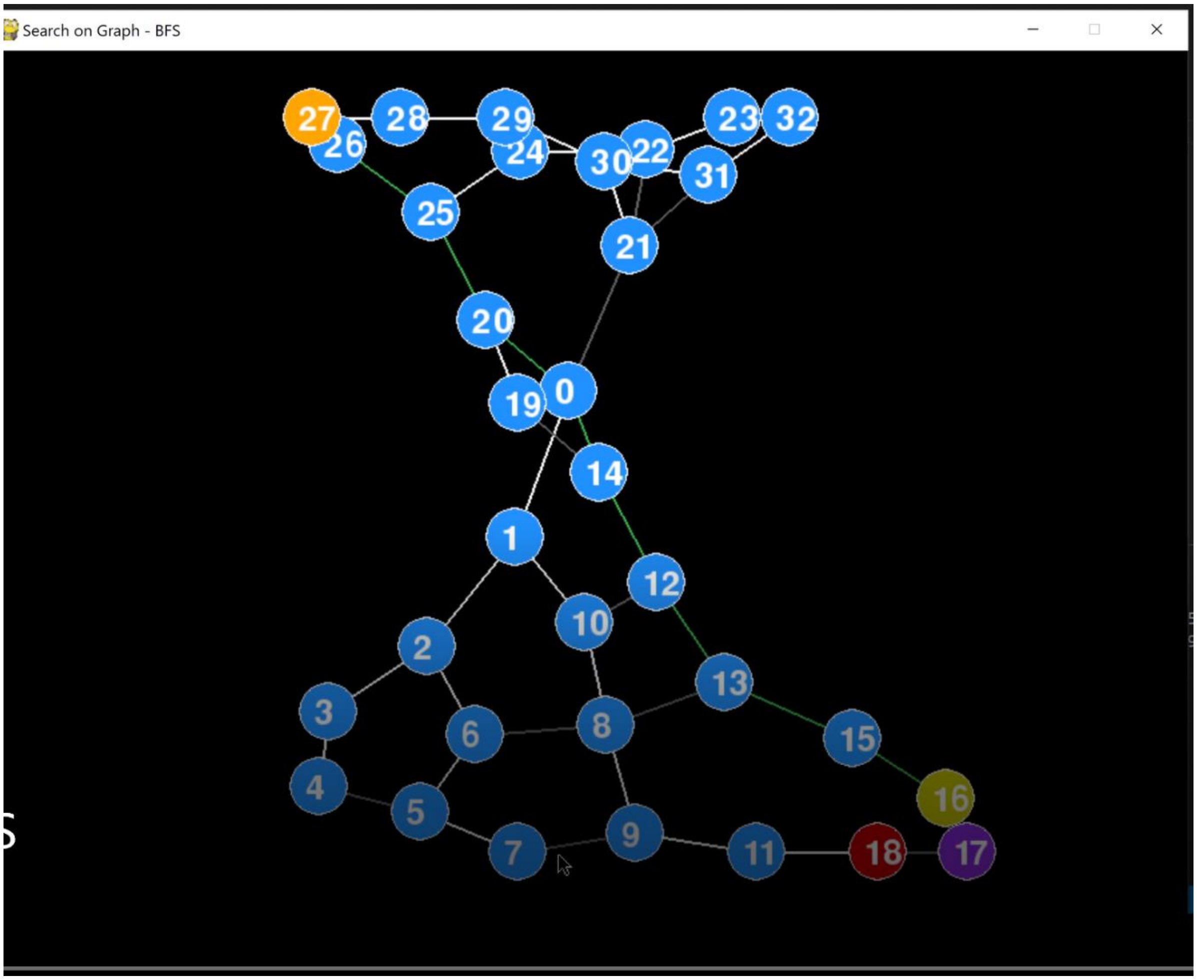
Comments:

1. BFS vs DFS: in this case DFS is better BFS, both run time and memory.
2. UCS vs A* vs Greedy: UCS, A*, Greedy search show the optimal path, but A* and Greedy Search is extremely more efficient than UCS (almost the tree is blue in UCS vs nearly 5 blue nodes in A* and Greedy Search). This implies that Greedy Search is often efficient.
3. DFS vs IDS: in this case the results of the algorithms are the same and IDS is much slower because it runs iteratively the depth from 1 to 6 (6 Depth-Limited-Searches are called).
4. BFS vs Bidirectional Search: Obviously, Bidirectional Search finds the Goal faster than half of the time as the result of running two BFSs, 1 from Start and 1 from Goal, intersects at 12.
5. Beam Search: It uses Beam Width = 2, as you can see two nodes 11 and 16 are only preserved.

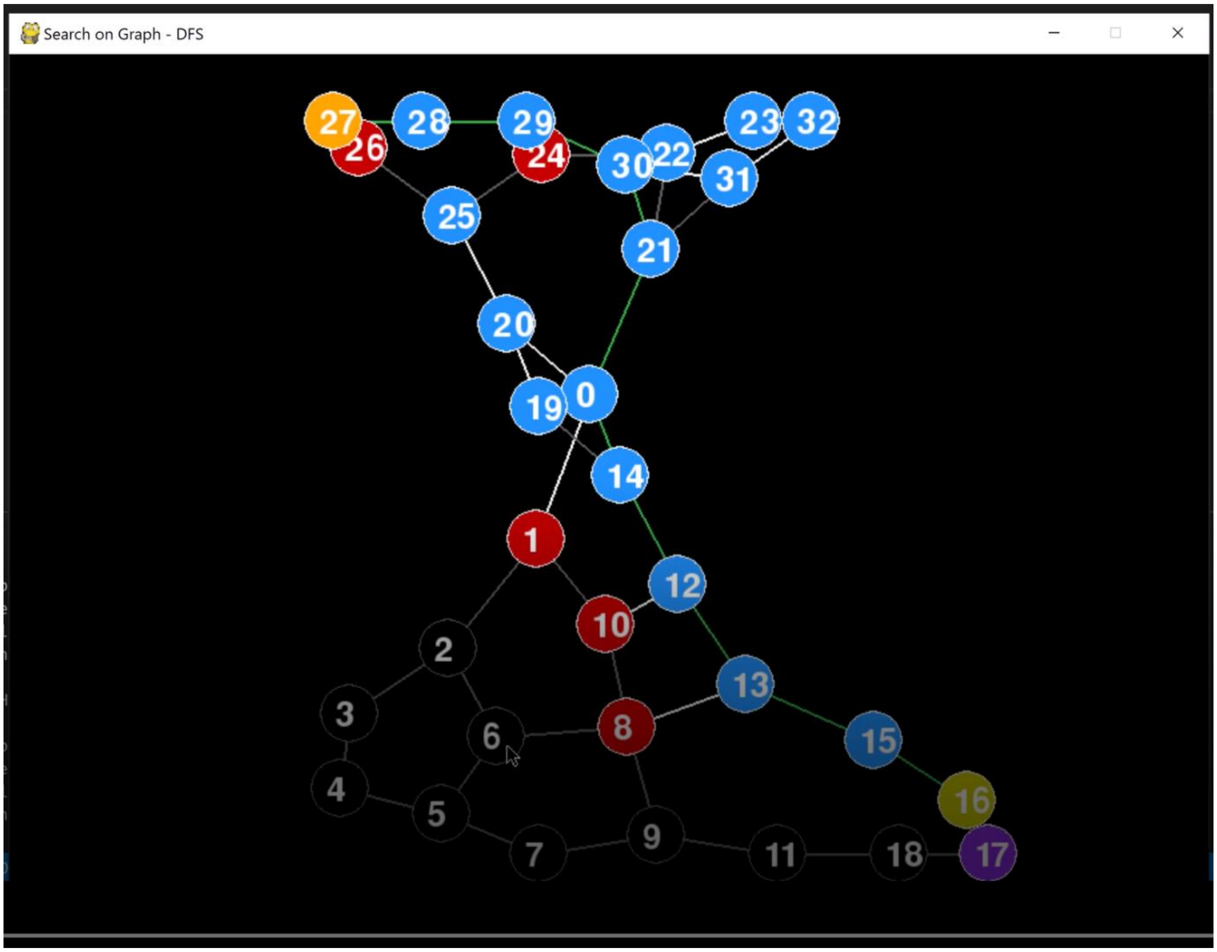
****
****

Test case 3: the graph used is the same at test case 2. But with a different start state.

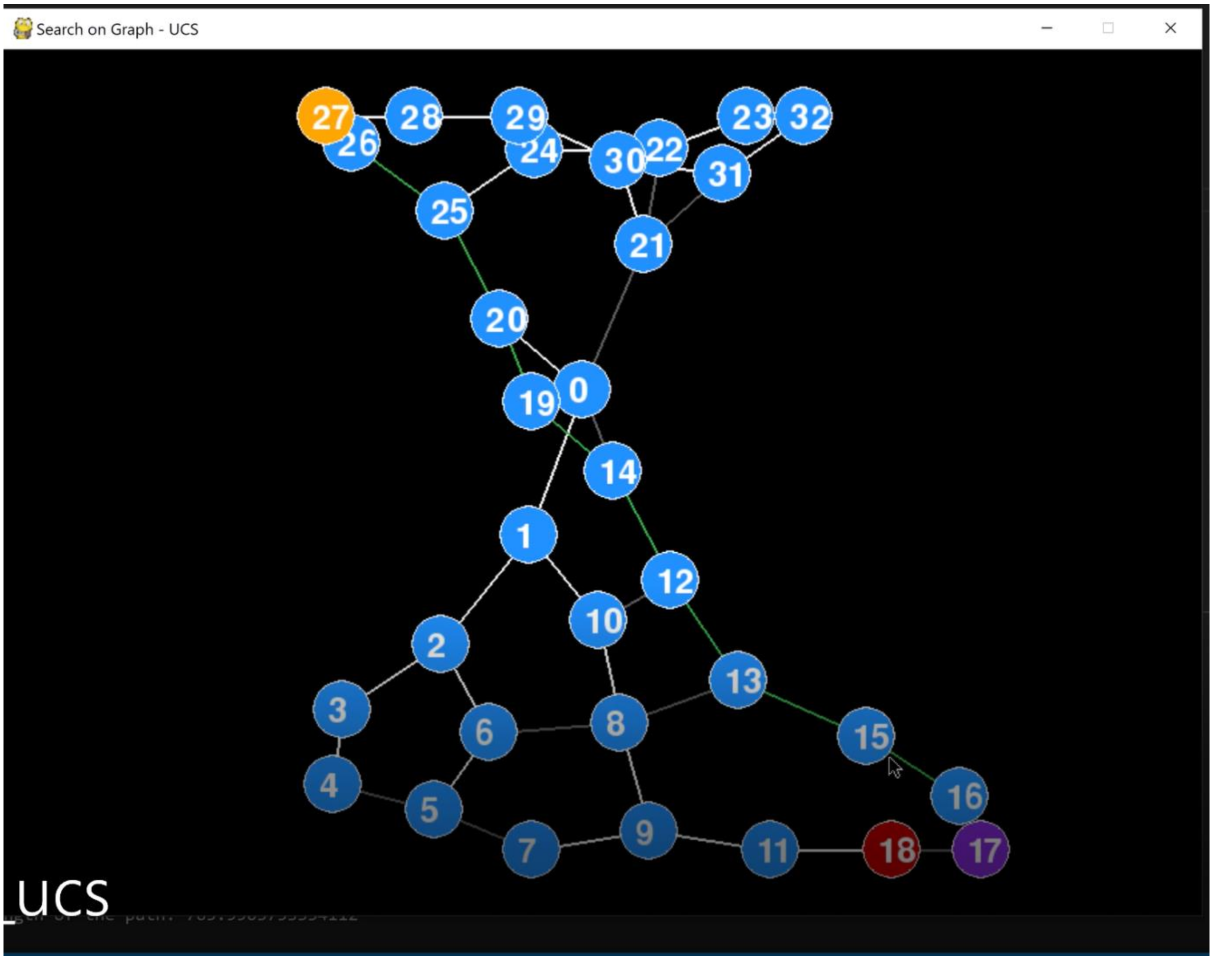
BFS:


Search on Graph - BFS
27
28
29
26
24
30
22
23
32
31
25
21
20
19
0
14
1
12
10
2
13
3
6
8
15
4
5
16
7
9
11
18
17

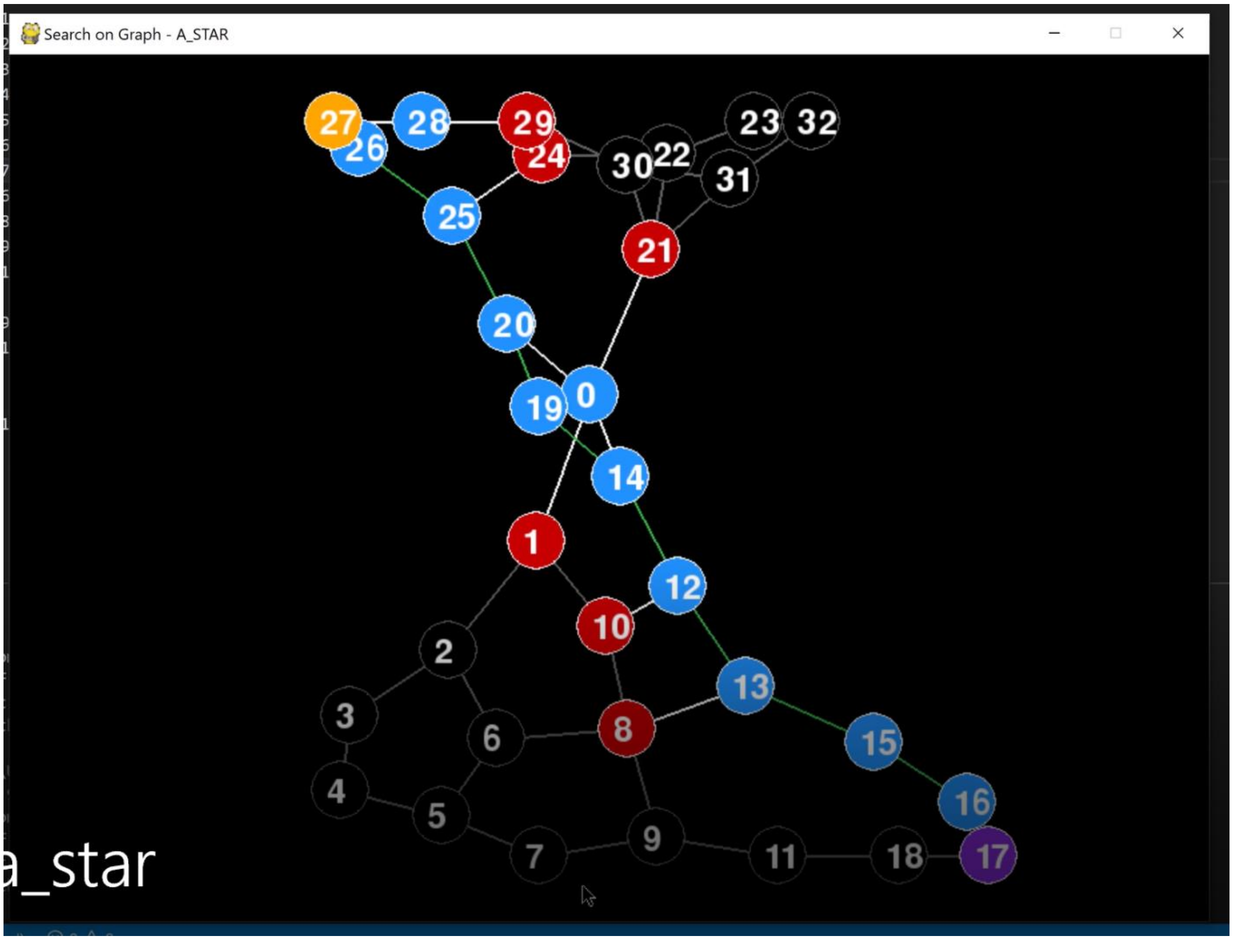
DFS


Search on Graph - DFS
27
28
29
26
24
30
22
23
32
31
25
21
20
19
0
14
1
12
10
2
13
3
6
8
15
4
5
16
9
7
11
18
17

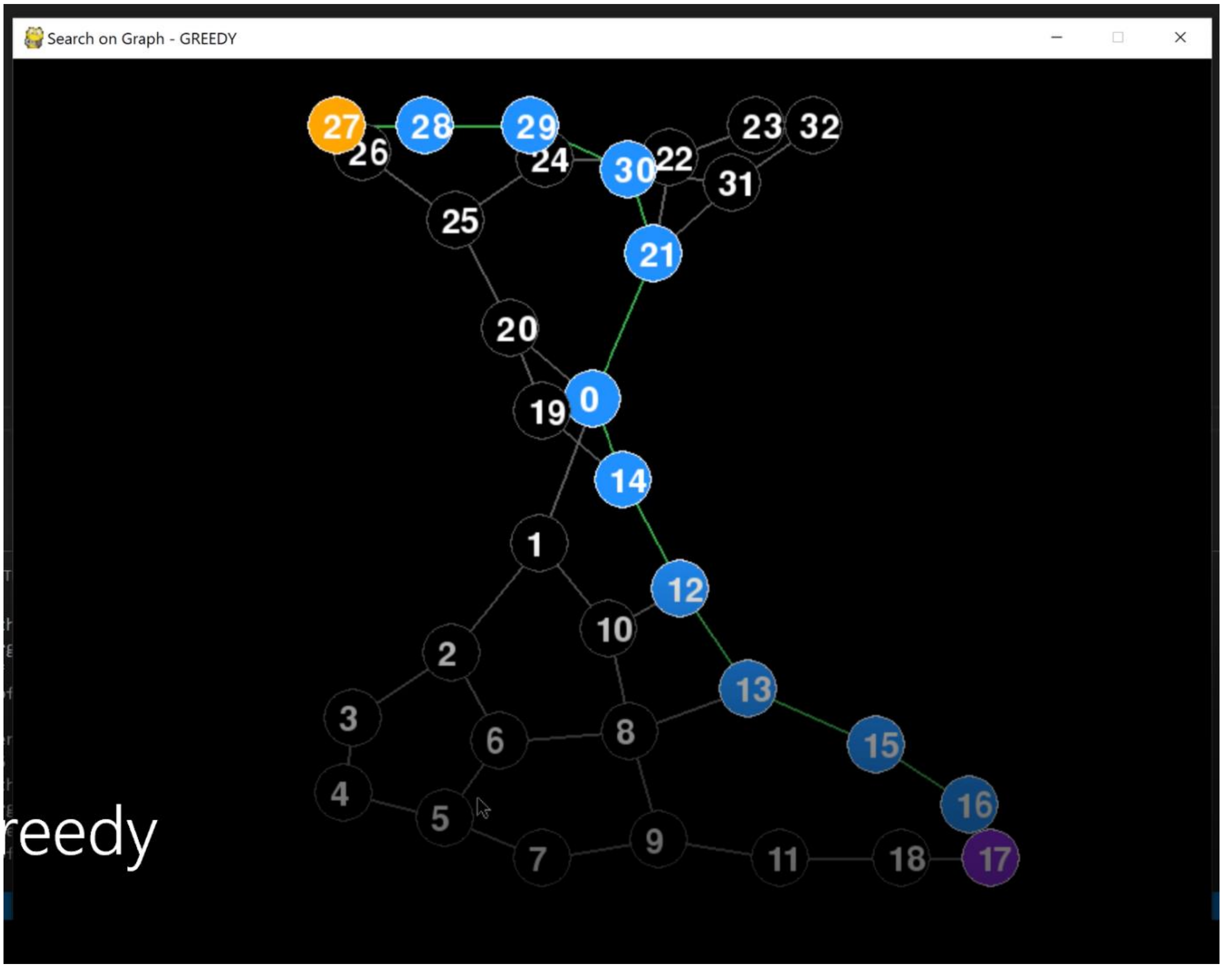
UCS:


Search on Graph - UCS
27
28
29
23
32
26
24
30
22
31
25
21
20
0
19
14
1
12
10
2
13
3
6
8
15
4
16
5
7
9
11
18
17
UCS

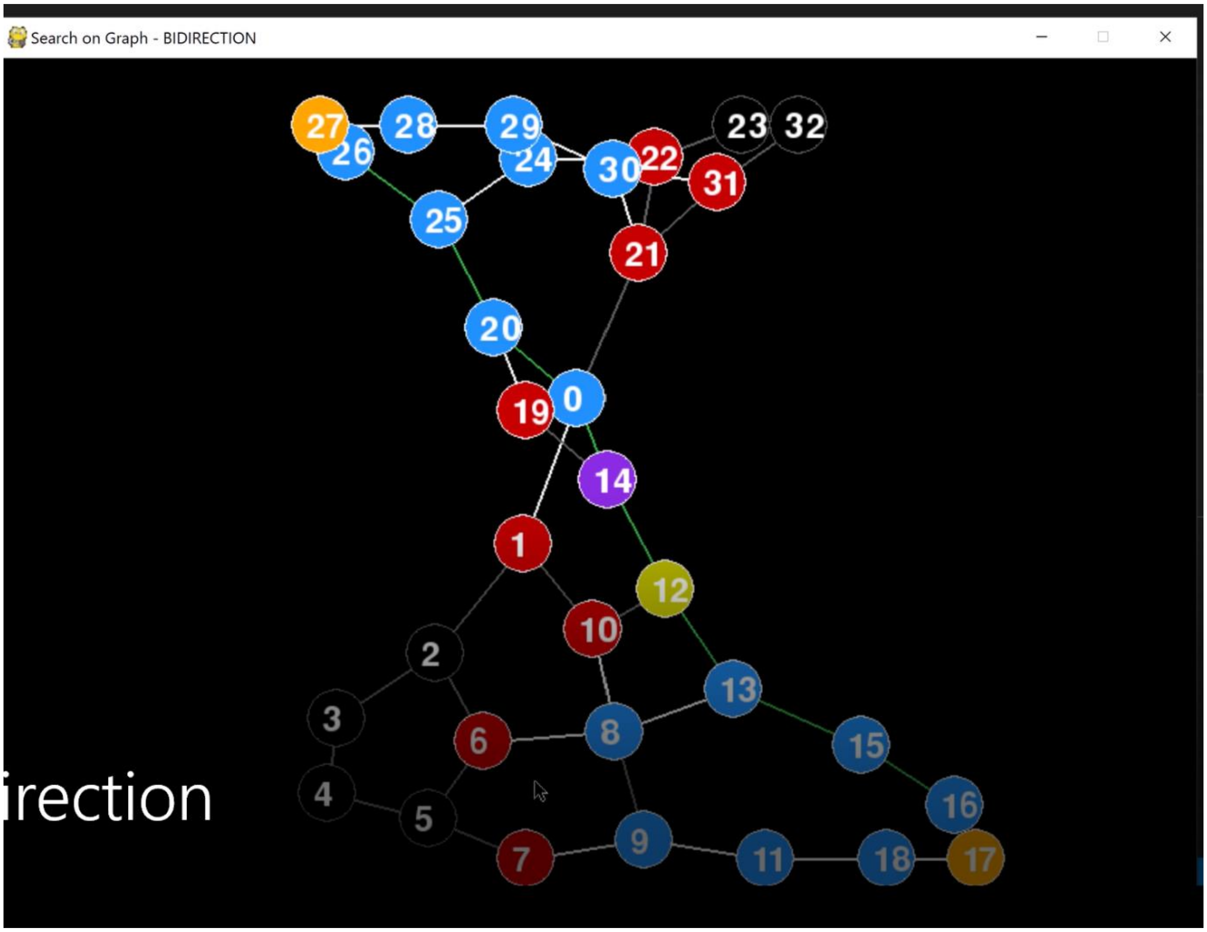
A* search:


Search on Graph - A_STAR
27
28
29
26
24
30
22
23
32
31
25
21
20
19
0
14
1
12
10
2
13
3
6
8
15
4
5
16
7
9
11
18
17
a_star

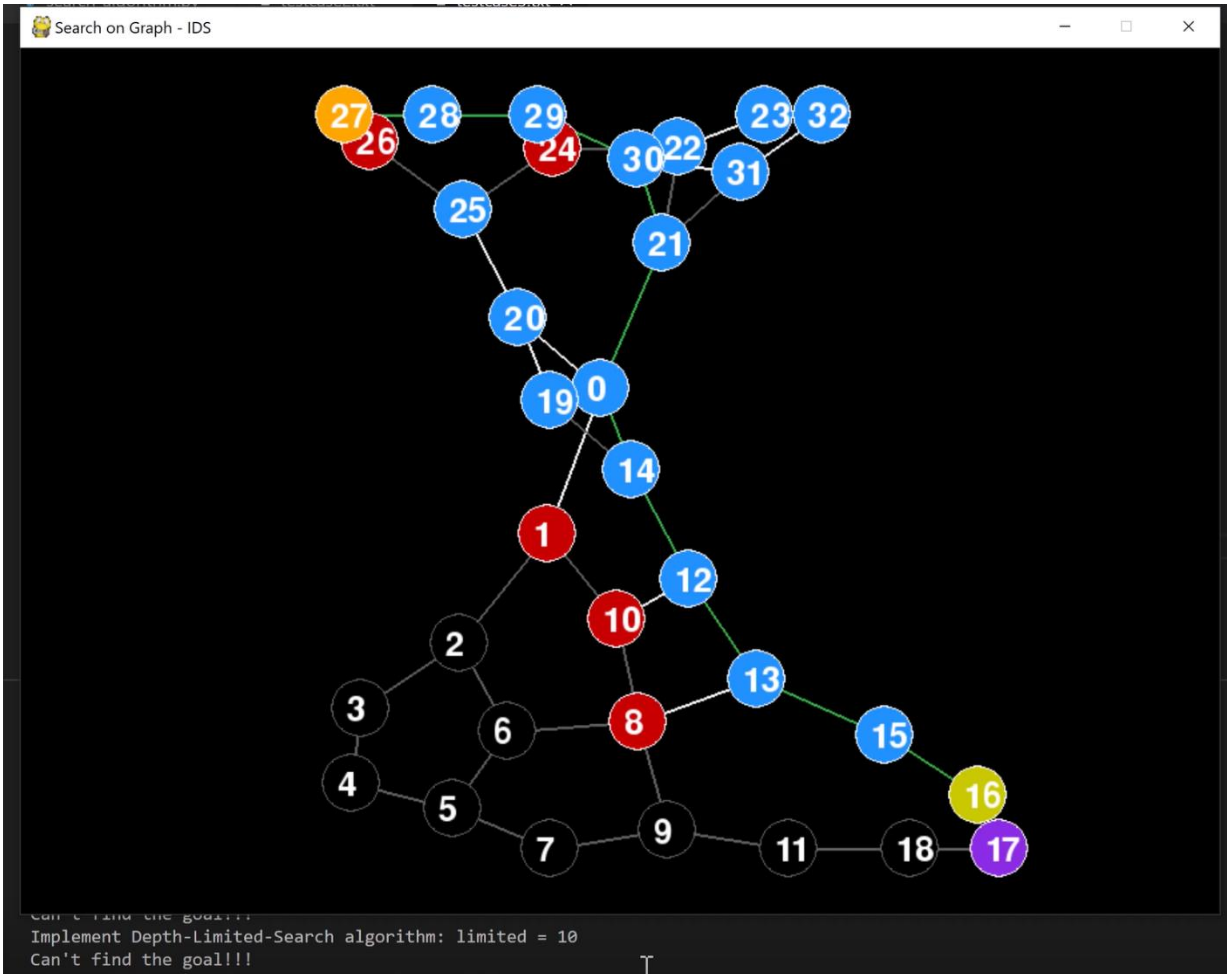
Greedy Best First Search:



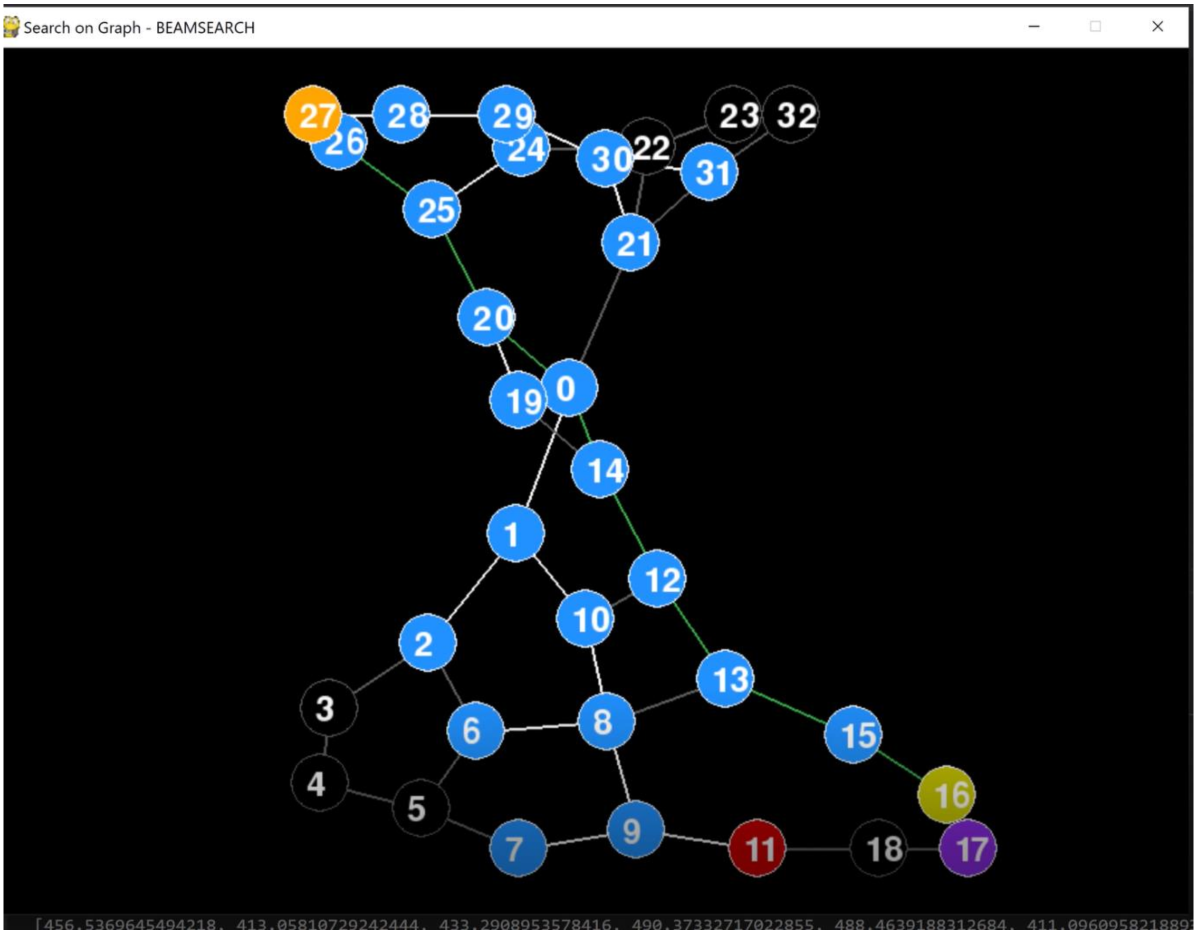
Bidirectional Search:



Iterative-Deepening-Search:



Beam Search:



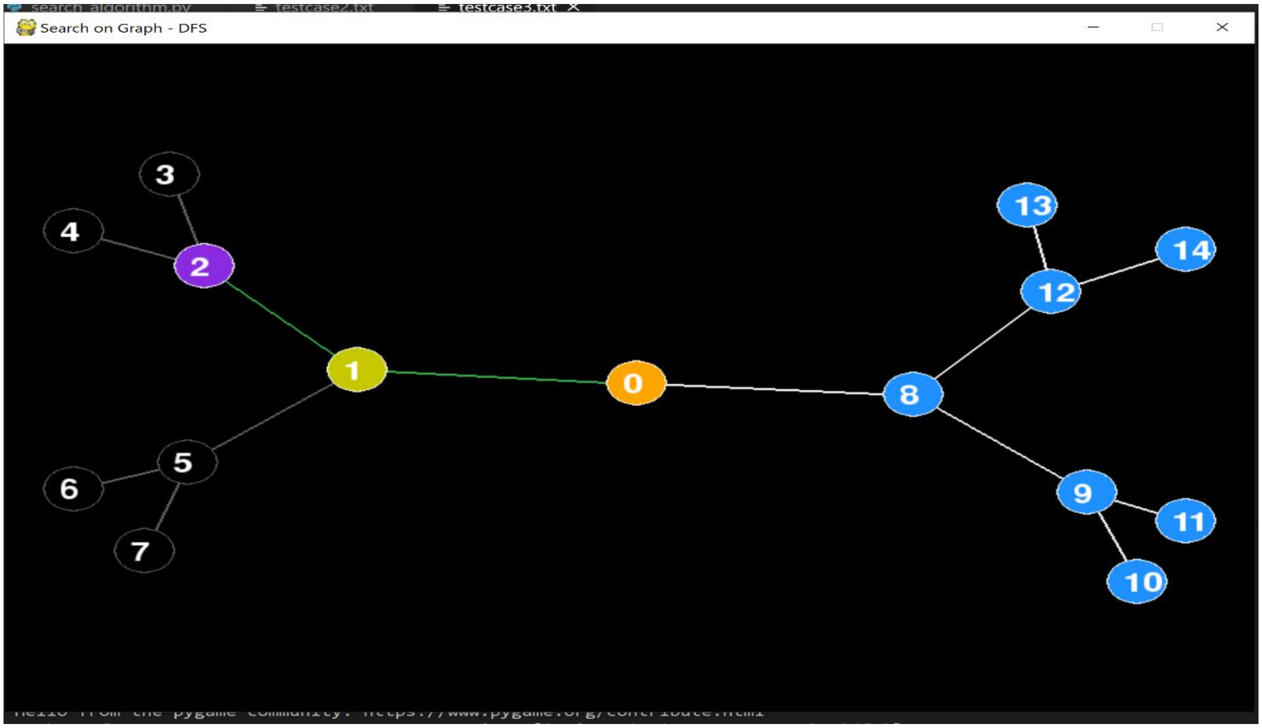
Comments:

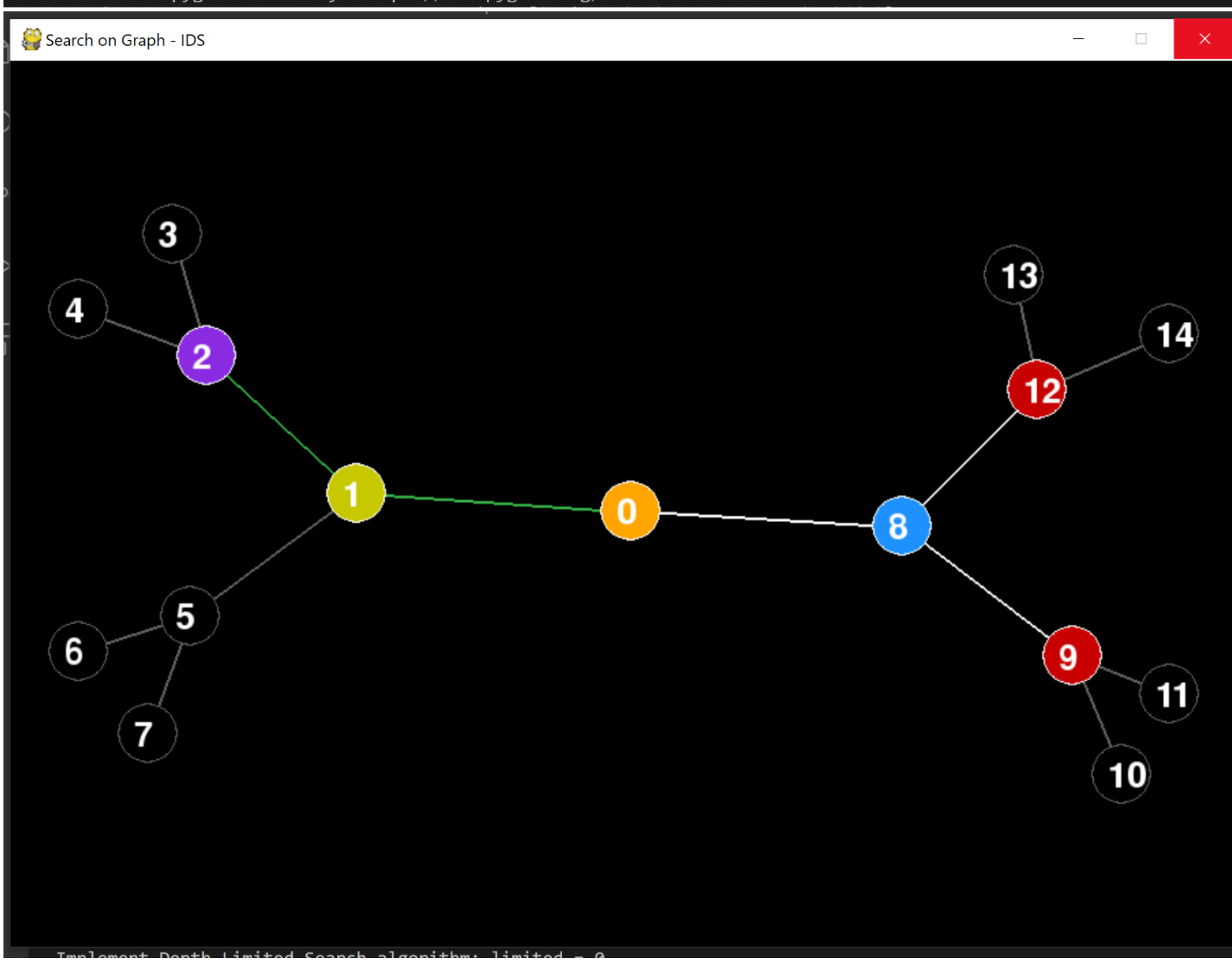
The same with the test case 2. But there are some points to notice:

1. The Greedy Search is not optimal and not complete in general.
2. BFS will be much more efficient than DFS if DFS determines the left most unexplored branches and the Goal is in the right most branch. Hence Iterative-Deepening-Search is used to solve this problem and take the advantage of DFS in Space memory.
3. Bidirectional Search is optimal if the backward BFS from the goal is easy to compute. In practice, the backward BFS cannot be operated.

****
****

One example is to show the efficiency of Iterative Deepening Search compared with DFS.


Search on Graph - DFS
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14


Search on Graph - IDS
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14