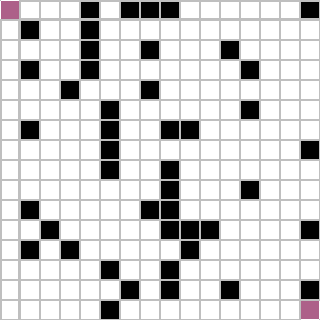
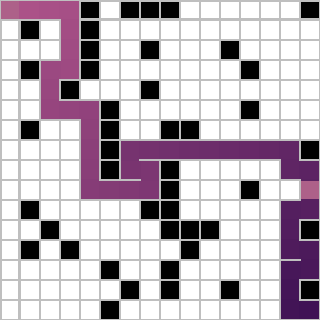
In this question, lots of algorithms have been applied to solve the same maze, because BFS, DFS and A\* have several variants. We have implemented them and compared differences between all the variants.

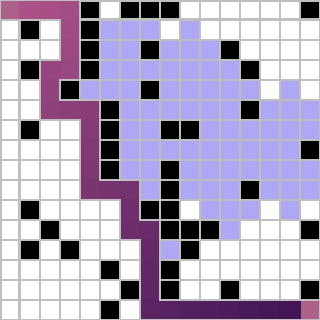
Starting with a small maze whose size is 16 by 16.



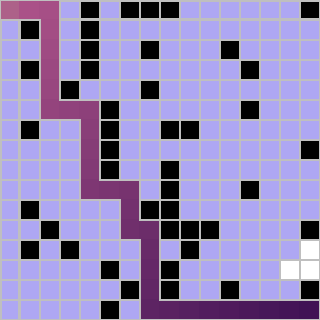
The most common DFS opens 43 blocks and its maximum fringe size is also 43. Coincidentally, it returns a path whose length is still 43. Apparently, its path is not optimal.



Then we have tried Iterative Deepening DFS which opens 2753 blocks (calculated accumulatively). Its maximum fringe size is 34 and path length is 33. Notice that IDDFS do not return an optimal path! (explained in “Modify DFS to Return an Optimal Path.html”)

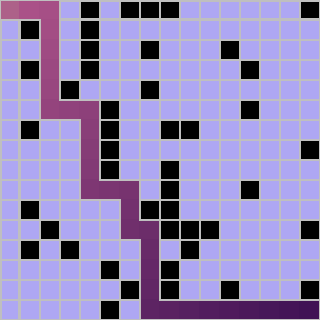


In order to return an optimal path, IDDFS and checkFringe should be set True. CheckFringe means keeping that blocks in the fringe always have the shortest path seen so far, but if we find a new path to this block, which is shorter, update no matter whether this block has been added to closed set or not. In this case, 8453 blocks have been opened, and the maximum fringe size is 29. The optimal path length is 31.

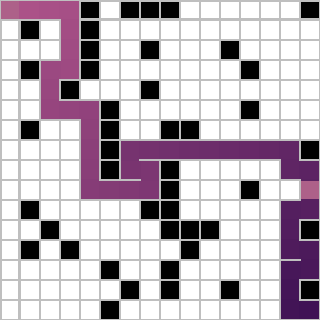


Also, there is another way to find an optimal path. Set keepSearch is True. In this case, when DFS find a path, save it rather than return. Keep searching for a shorter path. However, it takes too much time to return. (A small comparison is recorded in the table.)

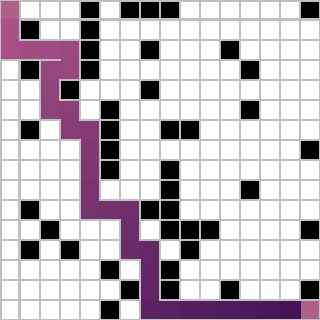
Hence, we can use checkFringe to accelerate keepSearch. By doing so, it opens only 1118 blocks (compared with 8453 of IDDFS and checkFringe), and its fringe size is at most 35. Its path is also optimal.



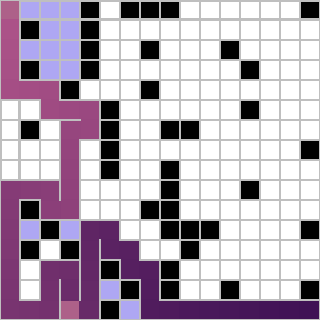
There are many other ways to improve DFS, and one of them is quickGoal, which means return the path the time trying to add the Goal into fringe, instead of the time popping the Goal out of fringe. In this way, DFS opens 43 blocks, but the maximum fringe size is 42 compared with 43 of common DFS. The path is the same as the one common DFS returned. It is interesting to find quickGoal do not improve DFS much. However, it is because DFS’s fringe is a LIFO queue. In this maze runner, it will not make any difference greater than 3.



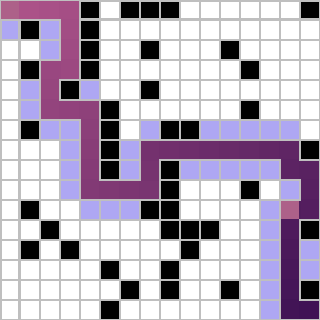
Another method is randomWalk. In common DFS, the priority of 4 directions is Left > Down > Right > Up. But randomWalk will randomly pick up a direction between left and down and between right and up. Notice that left and down is still preferable to right and up. Sometimes, it works. For example, in this case, the fringe size is 35, and block count is 33. The path is also shorter than common DFS, which is 33-block length.



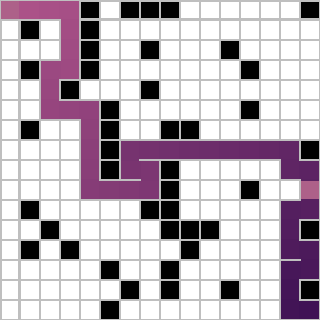
Yet it will not be a good idea to totally randomly pick up a direction. Here is the case. Set randomWalkPlue true. It opens 70 blocks. Fringe size goes up to 44. And the path is 53-block length. The reason is that totally random will cause agents go around in a small region, which is contradictory to the idea of DFS. Also, since the Start and the Goal locate at the ends of diagonal, the whole path direction should be southeast.



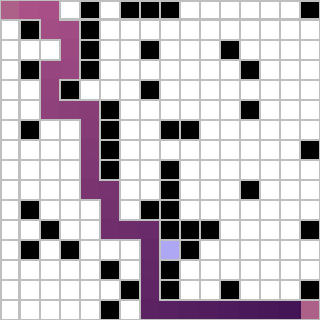
Besides, chechFringe itself can also make the path a little bit shorter. Setting checkFringe true returns a path whose length is 39 shorter than common DFS, though it is not the optimal.



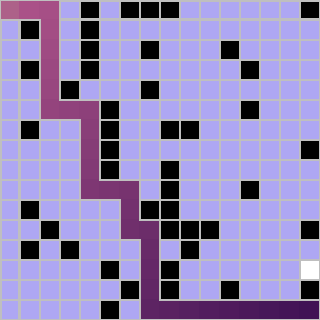
Except checkFringe, we can also use distinctFringe to stop duplicated block being pushed into fringe. But once this block is closed, it will never re-open even if we might find a shorter path. In this case, the fringe size, block count and path length is the same as common DFS, but we can see the difference later.



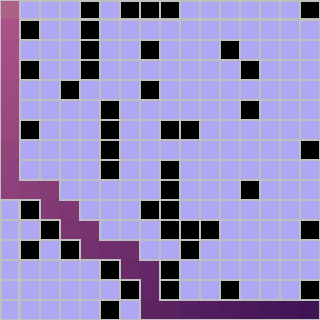
Certainly, we can combine those options, here is an example. Set quickGoal randomWalk and distinctFringe true. The block count, path length and fringe size are 34, 33 and 31.



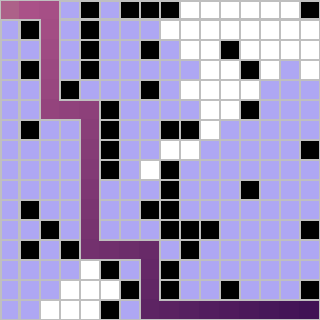
The most common BFS opens 113708 blocks, and its fringe size is 27898, because there are lots of duplicated blocks in its fringe. But it returns an optimal path.



To fix this issue, we can use checkFringe. Now it opens only 210 blocks and its fringe size decreases to 16. The path is absolutely optimal.

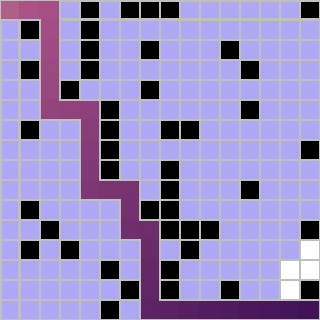


Also, Bi-Directional BFS can be helpful, whose fringe size is 2468 compared to 27898 of common BFS. Furthermore, it only opens 3528 blocks. Since it is a variant of BFS, the path must be optimal. When there is no path, Bi-Directional search usually returns faster than common ones because an empty fringe of either side, the Start or the Goal, indicates there is no path.

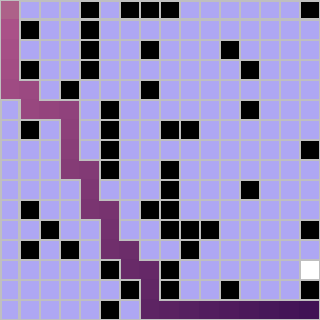
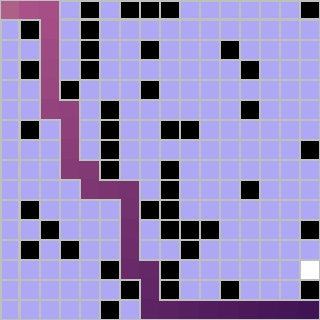


Now let us examine the performance of quickGoal of BFS. 93752 blocks have been opened, which is much fewer than common BFS. The fringe size also dramatically decreases to 19956. Because the fringe of BFS is a FIFO queue, quickGoal will return the path much earlier.

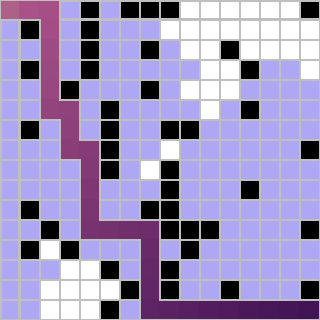
.



However, randomWalk and randomWalkPlus will not work regarding of BFS. Due to the fact that BFS is searching blocks level by level. The block count, path length, and fringe size of randomWalk is 113708, 31, and 27898. 113707, 31 and 27898 is the same data of randomWalkPlus.

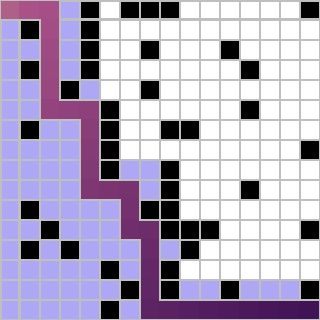
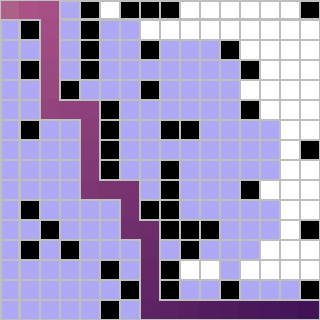
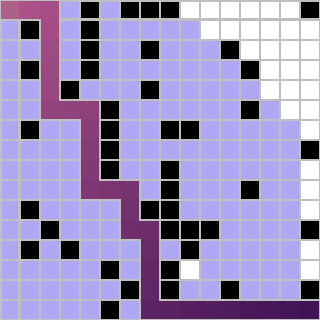


Again, we can use many options at the same time. Here is an example. Set BDBFS, quickGoal and randomWalk true. The block count, path length and fringe size becomes 151 31 and 25.

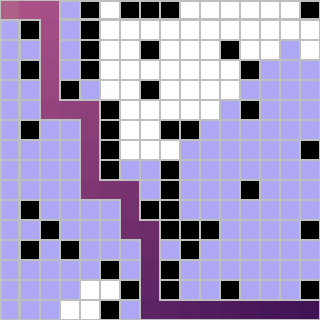
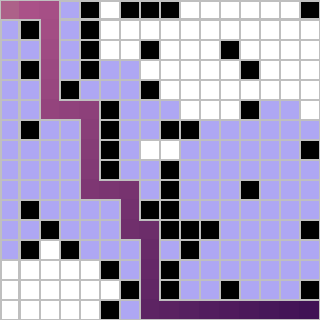
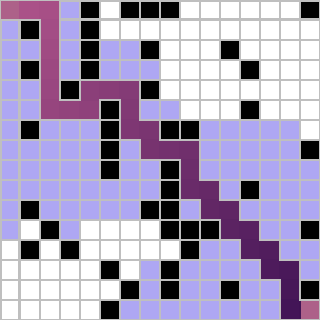


Speaking of A\*, there are mainly 3 dimensions to improve it. The heuristic function, Bi-Directional A\* and another kind of priority queue.

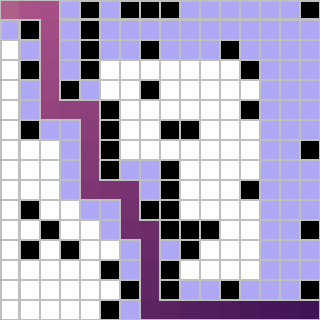
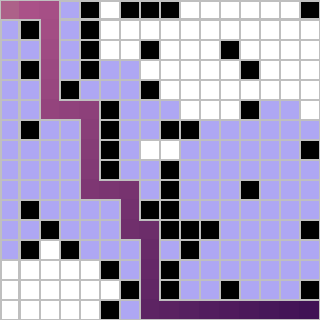
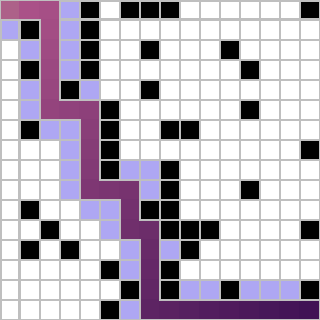
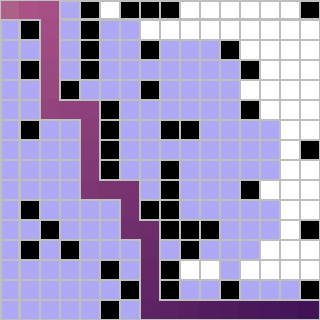
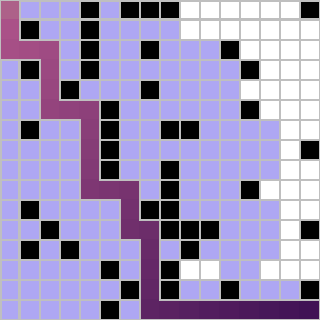
To begin with, we have chosen 3 distance function to be compared: Euclidean, Manhattan, and Chebyshev. All of them return an optimal path. But the preformation of Manhattan > Euclidean > Chebyshev. The reason is Manhattan is more closed to the real cost than Euclidean and Chebyshev.



In addition, we can also implement Bi-Directional A\*. However, BDA\* may not return an optimal path. Because the meet point has shortest path to the Start and Goal, yet the shortest path from the Start to the Goal may not go through the meet point.



Speaking of priority queue. 2 elements with the same priority can be LIFO or FIFO. The FIFO ones are more common, but we also have tried a LIFO one. LIFO ones preforms much like DFS, especially the maze is nearly empty (no walls).



All the comparison is recorded in the table. Notice that BDA\* LIFO with Manhattan is extraordinary except for non-optimal.

|  |  |  |  |
| --- | --- | --- | --- |
|  | size = 8 | | |
| block | path | fringe |
| DFS | 17 | 15 | 14 |
| keepSearch | 5326 | 15 | 14 |
| keepSearch checkFringe | 124 | 15 | 12 |
| quickGoal randomWalk | 15 | 15 | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | size = 16 | | |
| block | path | fringe |
| DFS | 43 | 43 | 43 |
| IDDFS | 2753 | 33 | 34 |
| IDDFS checkFringe | 8453 | 31 | 29 |
| keepSearch checkFringe | 1118 | 31 | 35 |
| quickGoal | 43 | 43 | 42 |
| randomWalk | 33 | 33 | 35 |
| randomWalkPlus | 70 | 53 | 44 |
| distinctFringe | 43 | 43 | 43 |
| checkFringe | 41 | 39 | 35 |
| quickGoal randomWalk | 34 | 33 | 35 |
| quickGoal randomWalk distinctFringe | 34 | 33 | 31 |
| BFS | 113708 | 31 | 27898 |
| BDBFS | 3528 | 31 | 2468 |
| quickGoal | 93752 | 31 | 19956 |
| randomWalk | 113708 | 31 | 27898 |
| randomWalkPlus | 113707 | 31 | 27898 |
| checkFringe | 210 | 31 | 16 |
| BDBFS quickGoal randomWalk checkFringe | 151 | 31 | 25 |
| A\* c | 159 | 31 | 24 |
| A\* c LIFO | 142 | 31 | 26 |
| A\* e | 139 | 31 | 23 |
| A\* e LIFO | 134 | 31 | 28 |
| A\* m | 91 | 31 | 11 |
| A\* m LIFO | 37 | 31 | 26 |
| BDA\* c | 116 | 33 | 22 |
| BDA\* c LIFO | 103 | 33 | 25 |
| BDA\* e | 123 | 31 | 25 |
| BDA\* e LIFO | 123 | 31 | 25 |
| BDA\* m | 133 | 31 | 29 |
| BDA\* m LIFO | 69 | 31 | 59 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | size = 256 | | |
| block | path | fringe |
| DFS | 1265 | 953 | 803 |
| distinctFringe | 1230 | 953 | 803 |
| quickGoal randomWalk | 32557 | 915 | 8521 |
| quickGoal randomWalk distinctFringe | 24866 | 847 | 8943 |
| BDBFS quickGoal randomWalk checkFringe | 36570 | 521 | 314 |
| A\* c | 39889 | 521 | 343 |
| A\* c LIFO | 39836 | 521 | 363 |
| A\* e | 39426 | 521 | 388 |
| A\* e LIFO | 39399 | 521 | 389 |
| A\* m | 14678 | 521 | 1688 |
| A\* m LIFO | 14573 | 521 | 3866 |
| BDA\* c | 20471 | 531 | 345 |
| BDA\* c LIFO | 20245 | 531 | 369 |
| BDA\* e | 23392 | 523 | 488 |
| BDA\* e LIFO | 23392 | 523 | 488 |
| BDA\* m | 4313 | 521 | 697 |
| BDA\* m LIFO | 3799 | 523 | 1551 |