Exercise 1

1.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| SAD1 | BFS | 0,23 | 25,77 MB | 19 | 80 |

1.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| SAD2 | BFS | 196.85 | 12312 MB | Not Found | 90203 |

In SAD2.lvl we have 3 extra boxes. Adding more boxes to the level creates a larger state space and agent can take more action when it is attached to a box. The fact that all boxes are close to the agent leads to a big number of child-states generated by BFS strategy on early stages and leads to a huge number of nodes in our frontier as the algorithm. Concluding, the big number of possible actions along with the increased number of possible states make BFS strategy unable to solve this problem with the given resources.

1.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| SAD1 | DFS | 0,14 | 25,39 MB | 31 | 76 |
| SAD2 | DFS | 0,14 | 27,10 MB | 29 | 88 |

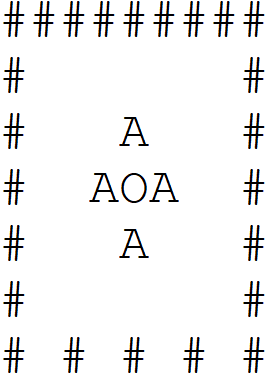
Changing from BFS to DFS we change the way the algorithm explores new nodes from FIFO to LIFO. The result is that we found a solution with bigger length than the optimal but in the given example (SAD2.lvl) we find the solution faster because the algorithm does not “stuck” on exploring all possible actions for every step.

1.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| FriendOfBFS | BFS | 0,74 | 106.84 MB | 3 | 697 |
| FriendOfBFS | DFS | 160,77 | 12289,12 MB | Not found | 90370 |

On the level FriendOfDFS DFS strategy performs better because in the state graph there are many nodes that they represent the goal state. Thus, applying DFS strategy will lead us to a solution relatively fast. On the other hand, BFS strategy ultimately will find the optimal solution of length n but finding the solution is costly. That happens because BFS algorithm creates every node of the state-tree up to depth n-1. In addition to that, FriendOfDFS contains many boxes so push and pull actions happen more often producing more child nodes. In detail, in the worst case we have 12 possible actions (figure 1) so we can set the upper bound of explored nodes by BFS to . In our example n=8 so we have an upper limit of explored nodes bigger than 108.

Figure 1



1.5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| FriendOfDFS | BFS | 122,59 | 10740,84 MB | 8 | 77760 |
| FriendOfDFS | DFS | 0,12 | 34,91 MB | 30 | 149 |

On the level FriendOfBFS BFS strategy performs better because the length of the optimal solution is small and as discussed above the explored nodes have upper bound of 123 which is a reasonable number. On the contrary, DFS strategy will underperform on this example because the size of the level is big and the DFS algorithm is most likely to walk away from the goal state ending up exploring a big number of nodes running out of memory.

1.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Level | Strategy | Time (in s) | Memory Used | Solution length | Nodes Generated |
| SAFirefly | BFS | 178,55 | 12288,11 MB | Not found | 89284 |
| SAFirefly | DFS | 119,30 | 12288,20 MB | Not found | 87763 |
| SACrunch | BFS | 134,65 | 12288,41 MB | Not found | 88498 |
| SACrunch | DFS | 106,17 | 12288,24 | Not found | 88999 |

Exercise 3

3.1

The shortest solution is *14*. And it is given if we push the box right every step.

The length of the shortest solution is *Θ(n).* More precise the shortest solution is always n-2.

The state space is *Θ(n2).* More precise, the *ith* step creates *i+1* new possible states.

3.2

Same as the previous example, the shortest solution is *14*. And it is given if we push the box right every step.

The length of the shortest solution is *Θ(n).* More precise the shortest solution is always n-2.

The state space is *Θ(n4).* Each state is described by the agent’s and box’s position. We have possible non-goal positions for the box and for each one we have possible positions for the agent.

3.3