Introduction to Artificial Intelligence - Exercise 1

209010479, 315922807

2 BFS vs. DFS

2.1 Optimality

1. Both the DFS and the BFS algorithms are guaranteed to return a correct solution, if such exists. The algorithms are also sound and complete, as seen in class. As for optimality,

   - In the Pacman maze problem, the optimal solution is the shortest path from the start point to the target (which results in the least amount of actions done).
     Using **BFS algorithm**, optimality is guaranteed. BFS algorithm explores various paths in the tree simultaneously, expanding all nodes at each depth level, and finishes when one of those paths reaches the target. Thus, once the algorithm reaches the target from any path, it has found the shortest path.
     In **DFS algorithm** on the other hand, the algorithm starts at the root and expands one branch as far as possible into the depth of the search tree before reaching the target or backtracking. As a result of this the algorithm will return the first path that leads to the target regardless of length, which might be suboptimal.
   - In the Blokus Fill problem, the goal is to fit all the given pieces on the board, hence there is no optimality in this case - only success or failure.

2. The results achieved when running the problems:

	BFS	DFS
Blokus Fill	17	17
Pacman Maze	68	130

   - As the results show, for the Pacman maze problem the BFS algorithm found a path with the cost of 68 actions (which is the length of the path), while the DFS algorithm found a path of cost 130 - a longer path, hence not an optimal one. We can't conclude whether BFS's solution is optimal by the results unless we go through every possible path and check if the returned one is indeed the shortest one, but according to theory BFS should return an optimal solution if implemented correctly.
   - For the Blokus Fill problem, each solution is an optimal solution as stated before.

3. The theoretical and empirical results are consistent with each other. In the Pacman maze problem, the results show that BFS algorithm found a solution with a lower score than the DFS algorithm, which means the path found was shorter - as expected according to theory. We have also expected the DFS the return a suboptimal solution. In the Blokus Fill problem as mentioned before any solution is an optimal solution, so it is expected that both algorithms will return the same results.

2.2 Running Time

1. Asymptotically, in the worst case **BFS** has a runtime of O((b^d),where b is the branching factor and d is the shallowest solution's depth. BFS explores all nodes in depth 0 through d-1 in order to ensure it finds the shallowest solution at depth

d. the number of nodes at depth k is b^k, so

$$1 + b + b^2 + \cdots + b^d = O(b^d)$$

For **DFS**, the algorithm has the runtime of $O(b^m)$ in the worst case: the algorithm could expand all nodes up to depth m because it could potentially search every branch of the tree until reaching the deepest level before backtracking in case it didn't found the target. In that case the number of expanded nodes will be

$$1 + b + b^2 + \cdots + b^m = O(b^m)$$

And since $d \leq m \implies b^d \leq b^m$, BFS has a better asymptotic time complexity compared to DFS.

2. The results achieved when running the problems:

	BFS	DFS
Blokus Fill	3119	128
Pacman Maze	269	146

   - In the Pacman maze problem, the DFS algorithm finished with 146 expanded nodes, in contrast to BFS which finished with 269 expanded nodes. DFS found a solution faster, due to the algorithm's behavior where it returns the first path regardless of optimality and only exploring one path at a time until reaching the target, while the BFS algorithm expands all possible paths node by node until it reaches the target in one of the branches.
   - In the Blokus Fill problem, BFS expanded significantly more nodes than DFS, also consistent with what we previously explained about BFS algorithm's expansion of numerous paths, including suboptimal and irrelevant ones.

3. The theoretical and empirical results are not consistent, as according to the empirical results BFS has a higher runtime (more nodes expanded) than DFS in both problems, in contrary to theory that suggests that BFS has a better asymptotical runtime. This discrepancy could be because the theory addreesses worst case scenarios, and not all scenarios, whereas the specific boards we ran in our tests might fit better with DFS. This could occur if the maze is structured in such a way that the optimal path is reached early by the DFS algorithm, leading it to expand only necessary nodes, while BFS explores these relevant nodes plus additional ones, resulting in a higher number of expanded nodes overall. If we tested the algorithm on boards where the optimal path is reached from the last node the DFS explores, we would have seen worse runtimes that align more closely with the theory.

5 Admissible and Non-Admissible Heuristics

5.1 a Valid Heuristic

1. An admissible and consistent heuristic for the Blokus corner's problem is Chebyshev distance to the nearest corner from any occupied tile on the board, where Chebysev distance is defined by

$$D(x, y) := \max_i(|x_i - y_i|)$$

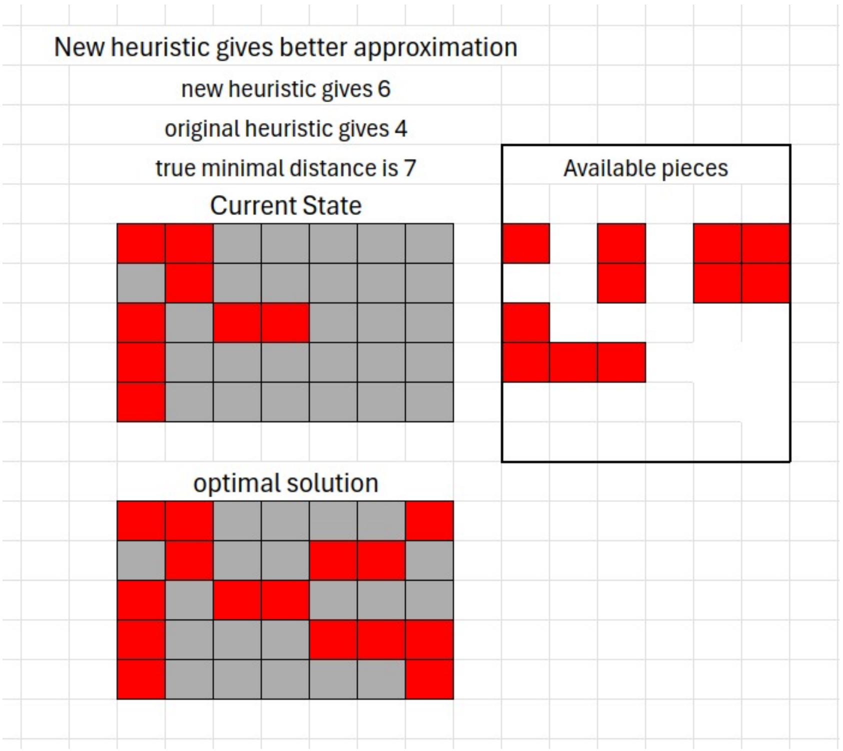
2. The heuristic is admissible since chebyshev distance calculates the number of tiles between the occupied tile to the corner, and takes into account both tiles that share an edge and a corner to the occupied tile, like the move a king piece can make on a Chess board. Because of this, Chebyshev distance is the minimal amount of tiles that can be covered in order to win if no limitations are presented. When presenting limitations, like specific piece shapes that might cover extra tiles that aren't necessary, the cost will always be higher than the Chebyshev distance, and thus it will never overestimate.

5.2 a New Heuristic

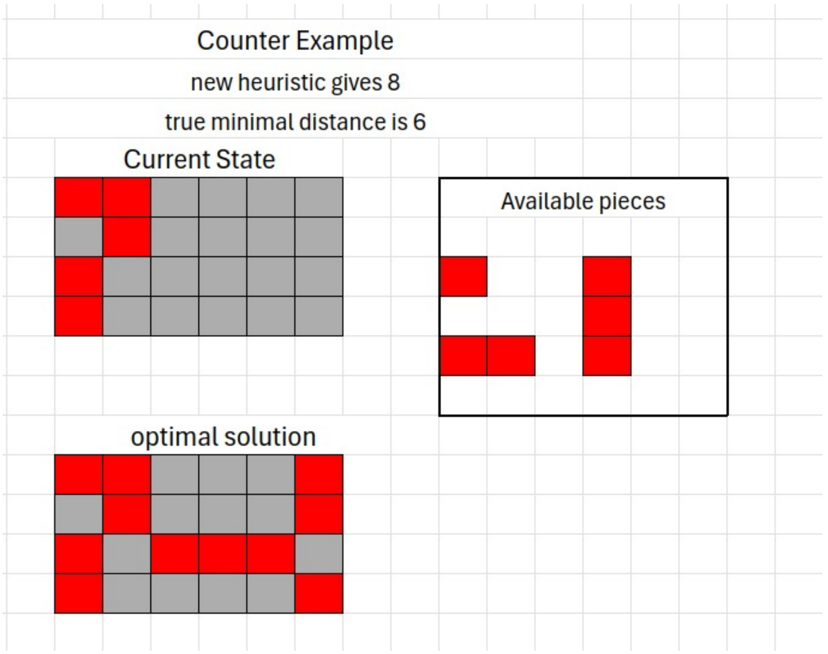
1. A non admissible heuristic is the following heuristic:

$$\sum_{c \in \text{uncovered corners}} \left(\operatorname*{Min}_{b \in \text{covered tiles}} \left(\text{Chebysev distance}(b, c) \right) \right)$$

2. The heuristic is better at approximating the minimal distance in the following case:


New heuristic gives better approximation
new heuristic gives 6
original heuristic gives 4
true minimal distance is 7
Current State
Available pieces
optimal solution

Yet it is not admissible, as shown by this example:


Counter Example
new heuristic gives 8
true minimal distance is 6
Current State
Available pieces
optimal solution