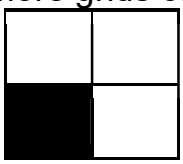
Ex1

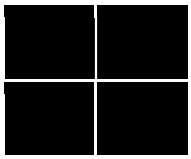
Answers for the understanding questions

Part 2.

- 2.1.
  - 1. Theoretically bfs and dfs both are uninformed search methods, which means that none of these methods guarantee optimal solution for every cost function, let's look at each method separately:
    BFS : finds the shallowest path to the solution so on the pacman it does find the optimal solution.
    In the other hand on the filling blokus board problem it does not necessarily finds the optimal solution for example consider a board of 2 rows and 2 cols where the first solution returned by bfs is that: (assuming that the goal state is that there is no more grids can be filled)

    Where the optimal solution is :

    DFS : finds the first solution searching in depth which means that it would find the optimal solution of the pacman in a very specific structure and does not find an optimal solution in general, also same as the BFS (with the same counter example), DFS it does not guarantee optimal solution for filling blokus board problem.

  - 2. The results for fill blokus :
    With BFS : we can not conclude or reject optimality or from the results.
    with DFS : same as BFS we can not conclude or reject optimality based on the results.

    The results for Pacman :
    With BFS :
    we can see that the pacman took the shortest path and that the time it takes to finish is directly proportional to the score so we can conclude that this score is optimal.

With DFS :
we can reject optimality based on the BFS search that got a better score.

	Pacman score	Pacman cost	Fill blokus score
BFS	68	442	17
DFS	246	264	17

- 3. Yes the BFS found the shallowest path therefore it was optimal in the pacman problem while DFS also as in theory it was likely to find the non optimal path, both BFS and DFS did not promise optimality with the fill blokus problem although we go optimal score based on the pisces set (tiny_set.txt)

- 2.2.
  - 1. DFS has better or equal complexity than BFS : it is O(b^m) for DFS and O(b^d) for BFS where b is the branching factor and m is the depth of the first solution.
    Pacman problem :
    b- the max number of different directions that we can take from any place of the maze,
    d- is the number of junctions in the maze.
    m- is the is the number of junctions for the first solution.

    Fill blokus problem:
    b- the number of the pieces on in the game (different orientation considered as different piece)
    d- the max number of pieces that can be placed on the board.
    m- the number of pieces on the board of the first solution.

	Number of expanded nodes
BFS	b^d
DFS	b^m

  - BFS can go through all the nodes (when the solution is on the last level)

- DFS runtime depends on the depth of the first solution. That is m can be equal to d and the two algorithms can have the same runtime

- 2.

	Pacman expanded nodes	Fill blokus expanded nodes
BFS	269	3119
DFS	269	156

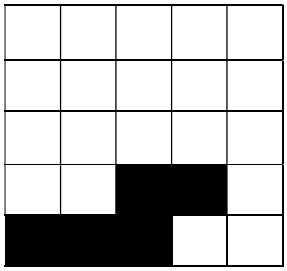
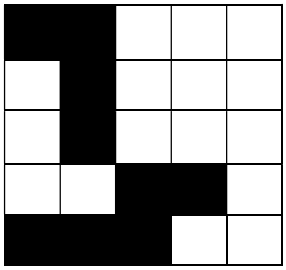
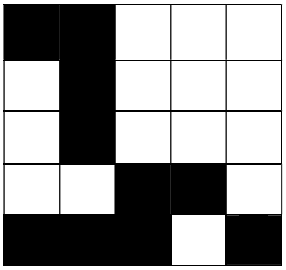
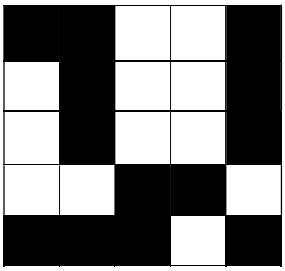
  - DFS is faster than BFS in the fill blokus problem and the same in Pacman
  - Intuitively the similarity of runtime in Pacman maze is due to the fact that DFS need to get through many junctions on its way to the solution and that b is less or equal to 3.

- 3. Yes the theoretical and the empirical results does consistent with each other, we saw that DFS runtime can be bigger than BFS and they can be similar as explained above.

Part 5.

- 5.1.
  - The heuristic is : for every state h(state) is the number of the unfilled corners.
  - It is admissible because for every state v0and for every path from v0, h(v0) less or equal to the number of unfilled corners in that state, and the cost of filling m corners is greater or equal than m so h(v0) m ic(vi) where m is the number of filled corners form a state v0
  - Read only if you would consider this heuristic trivial : another heuristic is the sum over the corners of minimal manhattan distances between a filled grid in the board and the corner divided by 2.
  - It is admissible because for every state v0 and for every path v0,v1,...,vn the cost of the path is greater or equal than h(v0) hence we divide by 2.

- 5.2.
  - Another non admissible heuristic is the sum over the corners of minimal manhattan distances between a filled grid in the board and the corner.
  - Consider the following example :

	Real distance	First admissible	Second admissible	Non admissible
V0	8	3	3.8	11
V1	4	2	1.6	5
V2	3	1	1	4
V3	0	0	0	0

In this example we see that the non-admissible heuristic better approximate the distance but we also see that it over exaggerate the path cost (11 > 8) so it is not admissible.