AI Project part A Report

Finally, you must briefly discuss your approach to solving this problem in a separate \_le called report.pdf (to be

submitted alongside your program). Your discussion should be structured around the following 3 questions:

* How have you formulated the game as a search problem? (You could discuss how you view the problem in terms of states, actions, goal tests, and path costs, for example.)
  + Current game environment is:
    - Observable: can see all board and pieces at once
    - Deterministic: We know exactly what the board will be like once a move is performed
    - Static: The board state does not change until a move is made
    - Episodic: Each state can be solved independent of each other
    - Discrete: there is a large – yet finite number of series of moves to achieve a goal state or at all (however with a bad search algorithm it is possible to be stuck in a loop)
  + Each unique board, with number and position of each of its pieces, can be viewed as a unique note or state.
  + Goal is to move each piece we own off the board, or to take the “exit” action for all pieces at the requisite edge of the board
  + Actions: move, jump, exit. We want to move all the pieces off the board in as few moves as possible. Actually that might be more of a goal test.
  + Can be viewed as a search problem in two different states: (??? Is the first one really necessary tho?) 1. The board in its own physical representation of a search graph, with a limited amount of nodes, being each position. Some nodes are inaccessible even given an infinite number of moves, such as spaces with a “block” on them. 2. Initial state is one node. One of the pieces the AI controls is moved, new nodes are created with branching factor = sum of number of unique moves for each piece.
* What search algorithm does your program use to solve this problem, and why did you choose this algorithm? (You could comment on the algorithm's efficiency, completeness, and optimality. You could explain any heuristics you may have developed to inform your search, including commenting on their admissibility.)
  + (We could use BFS for this but until we get that done we focus on A\*)
  + A\* pretty damn efficient, heuristic evaluation function keeps it optimal and not greedy means that we can avoid some local maxima. It is complete eventually.
  + Heuristic is straight line distance? Or just like path cost to goal if there were no blocks in the way.
  + Completely forgot what admissibility is though. Like the clear hex-manhattan distance is always going to be the fastest route
* What features of the problem and your program's input impact your program's time and space requirements? (You might discuss the branching factor and depth of your search tree, and explain any other features of the input which affect the time and space complexity of your algorithm.) Your report can be written using
  + IDK**…..>>>>>????????????????????????????????**