1. **Exercises:**
   1. **Solving 8-Puzzle with Dijkstra's Algorithm:**
      1. The number of tiles configuration is . Although this enormous number because we want to solve the single pair shortest path problem there are some redundant sate that not belong to the requested path, therefore, we can crate the state online, during Dijkstra iteration, that way we will create only the required state to solve our problem.
      2. *Code is submitted.*
   2. **Solving 8-Puzzle with A\*:**
      1. We define a Manhattan distance on a puzzle state by given examine the minimum number of movements to get from one state to another for example, if number "2" is on position (1,1) the Manhattan distance to move it to (2,3) is (2-1) + (3-1) = 3 and sum for all numbers in the puzzle but "0" because if we arrange all 8 numbers 0 is in its place.
      2. *Code is submitted.*
      3. Yes, the heuristic is matter, because it effects on the number of state the algorithm is going to evolve, the more it is close to real distance it is better. We change the heuristic to "Air distance heuristic", basically it is taking each coordinate delta power it by 2 sums for all coordinates and square root the sum. The heuristic is admissible because, its value is for sure less than Manhattan distance and the Manhattan distance is admissible because is the minimum changes that need to be done to the puzzle without blocking considerations.  
         For Manhattan distance A\* evolves 222 states and for air distance: 327, therefore it is clearly that Manhattan distance is better, it is closer to real distance.
      4. Initial state is:

|  |  |  |
| --- | --- | --- |
| 8 | 7 | 6 |
| 1 | 0 | 5 |
| 2 | 3 | 4 |

Goal state is:

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 4 |
| 8 | 2 | 0 |
| 7 | 6 | 5 |

Actions need to be done:  
d-> l-> u-> r-> r-> d-> l-> u-> u-> r-> d-> d-> l-> u-> u-> l-> d-> d-> r->   
u-> u-> l-> d-> r-> r

Dijkstra solving time: 10.768 seconds and 149,326 states.

A\* solving time: 0.186000 seconds and 2,058 states.

* + 1. The parameter  effects the heuristic function in a way to get it closer to the real distance. For  the algorithm does not use the heuristic and basically behave like Dijkstra, for the algorithm is A\* but with worst and better heuristics. But for we can't claim any more that we hold with admissible heuristic so there is no guarantee that the search algorithm will find the shortest path, but the search might be faster, Notice that the heuristic might stays admissible, then our algorithm will find the shortest path with less evolved states.  
       For the search algorithm assume that for any pair of states the distances is  therefore it will find some path.
    2. For if the original heuristic is admissible therefore the weighted one is also admissible, the algorithm will find the shortest path. Notice that if the original is admissible the weighting is making the algorithm to evolve more states, so it is no needed. But if the original is not admissible the weighting might help to make the heuristic admissible and therefore to find the shortest path. As said before, the search algorithm assume that for any pair of states the distances is  therefore it will find some path and probably not the optimal.
  1. **Solving Cart-Pole with LQR:**