Submitted by : 315524389, Or Mendel

***I used Manhattan Heuristic Function***The implementation of this function is pretty simple.

We iterate each cell in start state and compare to it to each cell of goal state.

The overall distance is computed by counting the number of moves along the grid that each tile is displaced from its goal position, and summing these values over all tiles in our start state and goal state respectively.

The returned value is 3 times the overall distance and not 4 times, because we can't go back to the previous state (duplicate).

* **Admissible** heuristic:

h is admissible if and only if h never overestimates the true cost to the goal.

Formally:

h(s) = heuristic estimate from s to the goal  
c(s) = optimal cost from s to the goal

h is admissible if and only if for all s in S:

h(s)<=c(s).  
  
Proof:

For all s in S: c(s) >=0, h(s) = 0.

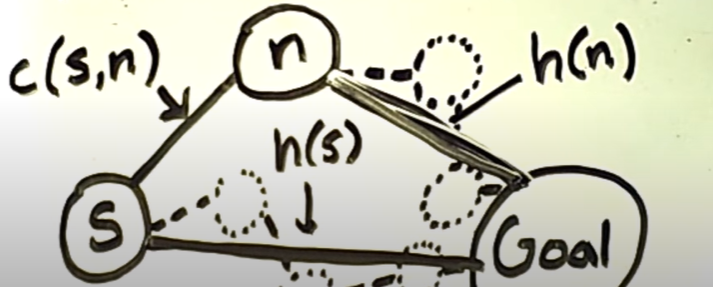
g(s) = cost from start to s

Priority of s = g(s) + h(s) = g(s) + 0 = g(s) (which is equivalent to Uniform Cost Search).

* **Consistent** heuristic:

To show that h is consistent it's the same is to say as h is monotone.

h is consistent if and only if:

1. h(goal)=0
2. h(s) <= c(s,n) + h(n) for all s in S and all n neighbors of s.  
   c(s,n) = step cost from s to n  
   neighbors(n) = set of all states one step from s

This is called global consistency.

[Note: we learnt in lecture that if h is consistent implies that h is admissible.  
Opposite not necessarily true.  
We won't use it here because the proof is for consistency!]

Saying h is consistent is same as saying h is monotonic non-decreasing.

Proving that statement will be enough to show global consistency (and from class we can see both sides of global consistency and local consistency exist)

A function f = g+h is monotonic non-decreasing if for every successor (child) n' of n:  
f(n) <= f(n').

h(n) is consistent if and only if f(n) is monotonic non-decreasing.

Proof:

h(n) <= c(n,n') + h(n')

g(n) + h(n) <= g(n) + c(n,n') + h(n')

g(n) + h(n) <= g(n') + h(n')

f(n) <= f(n').