Review week 4

* Search algorithm expands nodes in order of increasing f value
* For graph search, the f-values ​​will produce the surrounding coverage curve, but the point.
* contour lines will surround but the value is the same
* example: contour line 380, the points inside this line all have f value less than or equal to 380
* use algorithm A\* to solve 8 puzzle
* The average solution to an 8 puzzle is 22 steps to change the state from star to goal
* The branching factor b is about 3
* Have 2 heuristics to do this puzzle is h1, h2.
* h1 is the number of misplaced tiles.
* h2 is the sum of the distances of the tiles from their goal positions.
* Use tree-search of Mahattan
* Memory-bound heuristic search
* In some bad cases, A\* memory still grows exponentially
* difference with IDS:
* instead we use depth, we use f-value (g + h)
* If it fails, then the new cutoff value is the value of the smallest f-value that is beyond the cut off of the previous iteration
* IDA algorithm, the f-value will increase gradually
* if solution is non-null then return solution
* if f \_limit = ∞ then return failure
* IDS solution after increasing depth to find goal
* IDA use f-value and any node whose f-value exceeds f-limit will cut off
* The bad thing about the IDA algorithm is that every time it fails, it goes back to the top and then it continues searching.
* So Recursive best-first search (RBFS) algorithm is an improved algorithm over IDA algorithm
* for BFS algorithm, at each node will keep track f-value 2nd best and it only reverses when the current node exceeds f \_limit
* When it runs backwards, replace the f -value of each value along the path with the best value f (n) its children
* The f -limit value for each recursive call is shown on top of each current node, and every node is labeled with its f -cost.
* Optimality: optimal if h(n) is admissible
* Time complexity: Depends on accuracy of h(n) and how often best path changes
* Space complexity: Linear time: O(bd)
* Simplified Memory-bound A\* (SMA\*) is similar to A\*
* If memory is full, it wil delete the worst node
* expands the new leaf and deletes the old leaf
* in case f-value is equal then it will delete the oldest nodes first
* Heuristic functions: Effective branching factor b characterize the quality of a heuristic.
* N + 1 = 1 + (b∗) + (b∗)^2 + ... + (b∗)^d
* A well-designed heuristic would have a value of b∗ close to 1 ⇒ fairly large problems solved at reasonable cost
* The composite heuristic function is defined as h(n) = max{h1(n), . . . , hm(n)} is consistent and dominates all component heuristics
* A problem with fewer restrictions on the actions is called a relaxed problem.
* the cost of an optimal solution to a relaxed problem is an admissible
* heuristic for the original problem.
* Using Uniform-Cost-Search to find the shortest path from G to the other states
* The final g values can be considered as optimal h∗ values
* Pattern databases (PDB) where store the exact solution costs for every possible subproblem instance
* Disjoint pattern databases: Count only moves of the pattern tiles, ignoring
* non-pattern moves.