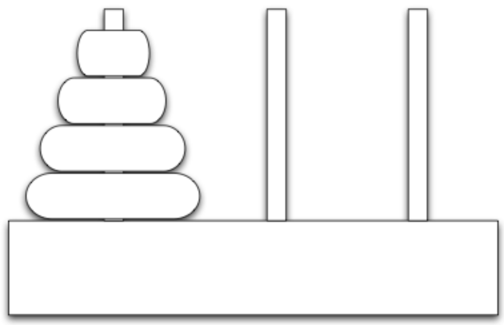
CS 188
Fall 2020

Regular Discussion 1

1 Towers of Hanoi



The Towers of Hanoi is a famous problem for studying recursion in computer science and recurrence equations in discrete mathematics. We start with N discs of varying sizes on a peg (stacked in order according to size), and two empty pegs. We are allowed to move a disc from one peg to another, but we are never allowed to move a larger disc on top of a smaller disc. The goal is to move all the discs to the rightmost peg (see figure).

In this problem, we will formulate the Towers of Hanoi as a search problem.

(a) Propose a state representation for the problem

(b) What is the size of the state space?

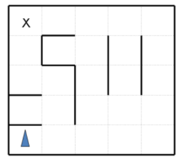
(c) What is the start state?

(d) From a given state, what actions are legal?

(e) What is the goal test?

2 Search and Heuristics

Imagine a car-like agent wishes to exit a maze like the one shown below:



The agent is directional and at all times faces some direction $d \in (N, S, E, W)$. With a single action, the agent can *either* move forward at an adjustable velocity v *or* turn. The turning actions are *left* and *right*, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are *fast* and *slow*. *Fast* increments the velocity by 1 and *slow* decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity (see example below). A consequence of this formulation is that it is impossible for the agent to move in the same nonzero velocity for two consecutive timesteps. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce v below 0 or above a maximum speed $V_{\max}$ is also illegal. The agent's goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if at timestep t the agent's current velocity is 2, by taking the *fast* action, the agent first increases the velocity to 3 and move 3 squares forward, such that at timestep $t + 1$ the agent's current velocity will be 3 and will be 3 squares away from where it was at timestep t. If instead the agent takes the *slow* action, it first decreases its velocity to 1 and then moves 1 square forward, such that at timestep $t + 1$ the agent's current velocity will be 1 and will be 1 squares away from where it was at timestep t. If, with an instantaneous velocity of 1 at timestep $t + 1$, it takes the slow action again, the agent's current velocity will become 0, and it will not move at timestep $t + 1$. Then at timestep $t + 2$, it will be free to turn if it wishes. Note that the agent could not have turned at timestep $t + 1$ when it had a current velocity of 1, because it has to be stationary to turn.

(a) If the grid is M by N, what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

(b) Is the Manhattan distance from the agent's location to the exit's location admissible? Why or why not?

(c) State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

(d) If we used an inadmissible heuristic in A* graph search, would the search be complete? Would it be optimal?

(e) If we used an *admissible* heuristic in A* graph search, is it guaranteed to return an optimal solution? What if the heuristic was consistent? What if we were using A* tree search instead of A* graph search?

(f) Give a general advantage that an inadmissible heuristic might have over an admissible one.