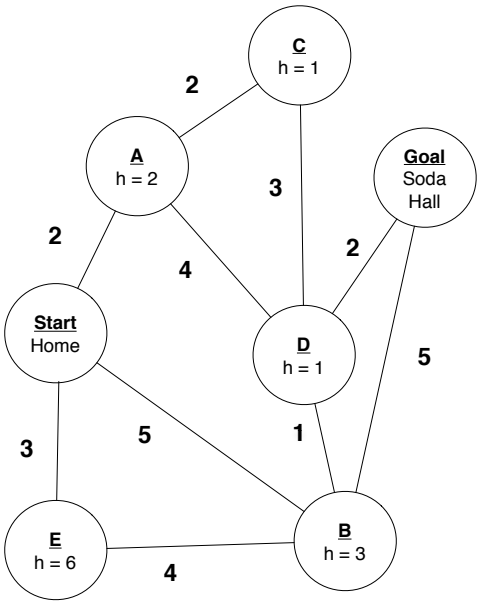
# CS 188: Section #2 Handout

## 1 Search Nodes vs. Search States

What's the difference between a search state and a search node? What kind of information would you have in a search node that you wouldn't have in a search state?

## 2 Uniform-Cost vs. A* Search




Our intrepid hero, Search Agent, is late for her artificial intelligence class and needs to get there fast! The graph below represents how long it takes Search Agent to walk between different parts of campus.

**a)** Use uniform-cost search to help Search Agent get from *Home* to *Soda Hall*. Keep track of the nodes on the priority queue for each step of the algorithm. Assume ties are broken alphabetically.

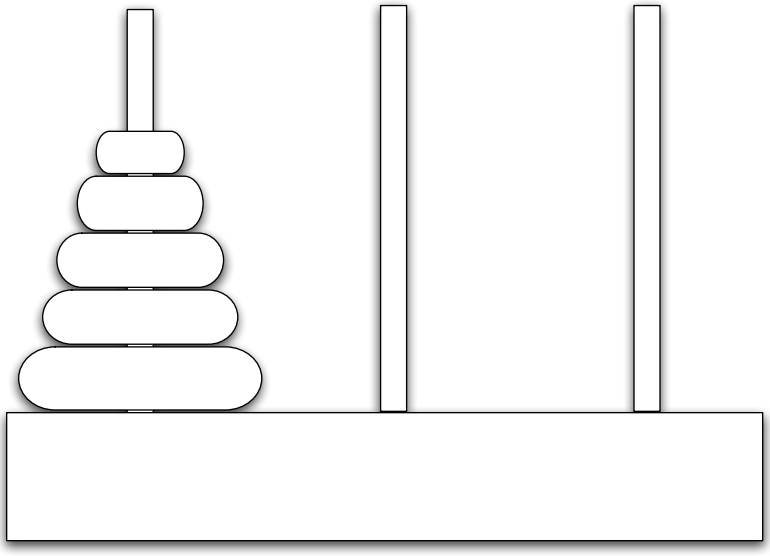


Figure 1: Diagram of *Tower of Hanoi* problem for $N = 5$

**b)** Now imagine Search Agent shows her graph to some friends and asks them to give her an *estimate* of how long it takes them to walk from that spot to Soda hall. Search Agent writes this information as heuristic guesses ($h$ values) in each of the states. Use A* to help Search Agent get from *Home* to *Soda Hall*. Keep track of the nodes on the priority queue for each step of the algorithm. Assume ties are broken alphabetically.

**c)** In general, what do we have to know about Search Agent's friend's estimates in order to be sure that A* is giving us a lowest cost solution?

# 3 Designing an A* Heuristic for Towers of Hanoi

The *Towers of Hanoi* is a famous problem for studying recursion in computer science and recurrence equations in discrete mathematics. There are $N$ discs of varying sizes on a peg, and two empty pegs. We are allowed to move a disc from one peg to another as long as we never move a larger disc on top of a smaller disc. The goal is to move all the discs to the rightmost peg (see Figure 1 for an illustration). There is a very interesting, albeit surely apocryphal, story regarding the origin of the problem:

*It is said that after creating the world God set on Earth three rods made of diamond and 64 rings of gold. At the creation they were threaded on one of the rods in order of size, the largest at the bottom and the smallest at the top. God also created a monastery close to the rods. The monks' task in life is to transfer all the rings onto another rod. The only operation permitted consists of moving a single ring from one rod to an- other, in such a way that no ring is ever placed on top of another smaller one. When the monks have finished their task, according to the legend, the world*

*will come to an end.*

In this problem we'll formulate the problem as a search problem and develop a heuristic for solving it. [1]

**a)** Formalize the Towers of Hanoi as a search problem

- What is the search state
- At a given state, what are the actions we can take and what are the resulting states
- What is the cost of an action
- What is the goal test

**b)** Develop a heuristic for the Towers of Hanoi. From a given state, can you tell at least how many actions it will take to reach the goal state? Is your heuristic admissible? Why?

**c) Programming (and Optional) Exercise:** Write a recursive solution to the Tower of Hannoi problem in python.

**Solution**

```
"""
Tower Of Hanoi:
  n: Number of disks on leftmost peg

Returns number of moves needed to move "n" pegs
to the rightmost peg and prints the solution
"""
def TowersOfHanoiHelper(start, end, aux, n):
  if n == 1:
    print 'Move from %s peg to %s peg' % (start,end)
    return 1
  else:
    moveAllToAux =  TowersOfHanoiHelper(start, aux,  end, n-1)
    moveTopDisc =  TowersOfHanoiHelper(start, end,  aux, 1)
    moveAuxToDst = TowersOfHanoiHelper(aux, end,  start, n-1)
    return moveAllToAux + moveTopDisc + moveAuxToDst

def TowersOfHanoi(n):
  return TowersOfHanoiHelper('left', 'right', 'middle', n)
```

[1] The CS 188 staff in no way endorses the hastening of the apocalypse through algorithmic or other means