Last Time

- Huffman Coding
Today

- Optimization Slack and Practice
Optimization is a unique class of problems. Many solutions exist. Fundamentally a search problem.
Both rely on an optimal substructure.

The efficiency of Dynamic Programming comes from the presence of repeated subproblems.

This may lead to the calculations of subproblems that do not get used in a globally optimal solution.

In general, you cannot know which local decision will lead to a global optimum.

However, if you can prove that an optimal local choice leads to a global optimum, you can calculate the solution to many fewer subproblems.
Knights Moves.

Given a starting position $x$ and an goal position $y$, count how many unique paths of length 3 there are for a knight to take from point $x$ to point $y$.

In chess, a knight moves two spaces forward and one to the side. This gives 8 possible moves.
Knights Moves
Frog Moves.

A frog can jump three spaces forward or two backwards. How many ways can a frog reach a point 10 spaces ahead in exactly 6 jumps? How do we set up the dynamic programming table?
Frog Moves.

A frog can jump three spaces forward or two backwards. How many ways can a frog reach a point 10 spaces ahead in exactly 6 jumps? How do we set up the dynamic programming table?

What if we asked how to move 10 spaces in less than 6 jumps?
A quick peek at Graphs

A* Search.
For each location, we define a heuristic “distance” function, guessing how far it is from the destination.

Also, we have an actual distance \( d(x,y) \) between each pair of locations.

If \( h(x) \leq d(x, y) + h(y) \), we can estimate the total distance to the goal by the distance to the current point plus the heuristic distance.

We then search paths, exploring those paths with smallest heuristic distance first.

Also we ignore paths that lead to already visited nodes, and those further than we’ve already traveled.
Bye

- HW 8 is posted.
- Next time (11/10)
  - Graphs...finally.