Dynamic Programming

October 18th 2016

Fibonacci

- How do we calculate the n^{th} fibonacci number, efficiently?
- Brute Force Method takes $O(2^n)$ time approximately

Fibonacci Solution

- We can solve the solution in O(n) time
- Use memoization to store the values in an array
  - No need to recompute

FibonacciList = {0,1}

def fib(n)

    If the value is in our list, return the value

    Otherwise, the value is fib(n-1)+fib(n-2), store in the list, return it

General Dynamic Programming (DP)

- Find subproblems
- Store the subproblem values
- Recursion

Coin Problem

- Let's say we have coin values a_1 a_2...a_k (say 1,2,7,9)
- How many coins do we need to make change for n cents
- Naive solution - Keep subtracting the largest coin value
  - For 12, subtract 9, then subtract 2, then subtract 1 = 3 total coins (1+2+9)
  - Doesn't work for 15 (7,7,1) vs. (9,2,2,2)

Coin Problem Solution

- Solve it via recursion
- Let f(n) be the number of coins needed for n cents
- f(n) = min(f(n-1),f(n-2),f(n-7),f(n-9)) for (1,2,7,9) coin set
- Memoize the values of f(n) in a list

Edit Distance

- Given two strings (“abc” and “bcd”) find the edit distance
  - Only can add, remove, replace
  - For this example, is 2 (remove a, add d)
- Think about a solution

Edit Distance Solution

- Find the edit distance between the first m characters (of 1st string) and n characters of 2nd string
- If they're equal, f(m,n) = f(m-1,n-1)
- Otherwise, find the maximum of
  - Adding the n^{th} character, so 1+f(m,n-1)
  - Delete the m^{th} character, so 1+f(m-1,n)
  - Replace the m^{th} character with the n^{th} character, so 1+f(m-1,n-1)

Knapsack

- Given a list of items, each with a weight and a value, then find the maximum value of items, where the total weight<w (The weight limit)

Knapsack Solution

- We store a state, of using the first m items, with n weights left
- Use the recursion f(m,n) =
  - If we can't use the m^{th} item, then its just f(m-1,n)
  - If we can use it
    - max(f(m-1,n-w_m)+v_m,f(m-1,n)), maximum of should we use it or not

Additional Problems

1. Coin Change problem. Given coins $a_1, a_2 \ldots a_n$, determine the numbers of ways to make change for n cents
2. Subset Sum problem. Given a list of numbers, determine whether any subset sums to 0