10

June 2, 2016

1 Dynamic Programming

- form subproblems
- get recursive formula to explore all choices at each step
- evaluate formula bottom-up using a table works when total # subproblems is not too big

Ex 1: Evaluate $C(n,k) = C(n-1,k-1) + C(n-1,k)$ (or 1 iff $k = 0$ or $k = n$) that is the binomial theorem.

Naive is $O(2^n)$. Can optimize to $O(n^2)$ if we cache values since there are at most $O(n^2)$ (n,k) combos.

1.1 Coin Changing

Given coin values c_i, target W, find minimum # of coins that sum exactly to W.

1.1.1 First Solution

Define $C(i,j)$ be min # of coins from $\{c_1...c_i\}$ that sum to j. Note the set is ordered.

Then $C(i,j) = \min\{C(i-1,j), C(i,j-c_i)+1\}$ (what'd happen if the last coin we took is c_i).

Of course, we need to test the bounds to ensure $i-1 \geq 0$ and $j - c_i \geq 0$.

Base cases: $C(i,0) = 0$, $C(0,j) = \infty$.

Runtime: $O(nW)$. Space can be reduced to $O(W)$ if we only store last 2 rows. It is also trivial to store "back-pointers" to recover how we got to $C(i,j)$ using $O(nW)$ space.

1.1.2 Second Solution

Define $C(i)$ to be min # of coins to make i sum. $C(i) = \min\{C(i-c_j)+1 | \forall c_j\}$. $C(0) = 0$.

This is still $O(nW)$ since for each $1...W$, for each $c_1...c_n$, we do a constant operation.

2 0/1 Knapsack

Given total weight W, values $v_i > 0$, weights $w_i > 0$, find a subset $S \subseteq \{1...n\}$ s.t. $\sum_{i \in S} v_i$ is maximized, $\sum w_i \leq W$.

Solution: let $f(i,j)$ be the maximal value possible if we're given total weight j and objects $1...i$. Then $f(i,j) = \max\{f(i-1,j), f(i-1,j-w_i)+v_i\}$. Overall $O(nW)$ time complexity.

Base cases: $f(0,\_) = 0$ (nothing to take), $f(\_,0) = 0$ (no space to take anything).