Ethan Davis
April 30, 2021
Analysis of Algorithms

Unbound Knapsack (Dynamic Programming): Max profit with weight capacity with any item used unbound times. Let c be a weight capacity, n be a length of profits and weights, p be an array of profits and w be an array of weights

$$\begin{aligned} f(c,n,p,w) &= 0 && c = 0 \\ &= max(p_i + f(c - w_i, n, p, w)) && c \geq w_i \end{aligned}$$

Make Change (Dynamic Programming): Min coins for amount. Let c be an array of coins and a be an amount

$$\begin{aligned} f(c,a) &= 0 && a = 0 \\ &= min(1 + f(c, a - c_i)) && a \geq c_i \end{aligned}$$

Cut Rod (Dynamic Programming): Max profit for cuts. Let n be a rod length and p be an array of rod length profits

$$\begin{aligned} f(n,p) &= 0 && n = 0 \\ &= max(p_{i-1} + f(n - i, p)) && n \geq i \end{aligned}$$

Staircase (Dynamic Programming): Start at step 0 or 1, each step take 1 or 2 steps, find least cost. Let s be an array of steps with cost, n be a length of steps and i be an array index

$$\begin{aligned} f(s,n,i) &= 0 && i \geq n - 1 \\ &= min(s_i + f(s,n,i+1), s_{i+1} + f(s,n,i+2)) && \text{else} \end{aligned}$$

Combination Sum (Backtracking): Given an ordered list of numbers n and a number x, find the sublists which sum is x. Let n be a list of numbers, r be a remainder, s be a start index, b be a cache of solutions and t be a temporary solution

$$\begin{aligned} f(n,r,s,b,t) &= b.push(copy(t)) && r = 0 \\ &= t.push(n_i) && r \geq n_i \\ & \quad f(n, r - n_i, i, t, b) \\ & \quad t.pop() \end{aligned}$$

0/1 Knapsack (Dynamic Programming): Max profit with weight for capacity with any item used at most once. Let c be a weight capacity, n be a length of profits and weights, p be an array of profits, w be an array of weights and i be an array index

$$
\begin{aligned}
f(c,n,p,w,i) &= 0 && i = n \\
&= max(p_i + f(c - w_i, n, p, w, i+1), f(c,n,p,w,i+1)) && c \geq w_i \\
&= f(c,n,p,w,i+1) && c < w_i
\end{aligned}
$$

Longest Common Subsequence (Dynamic Programming): Longest common subsequence of (non-)contiguous characters. Let $s1$ be string one, $s2$ be string two, i be an index of $s1$ and j be an index of $s2$

$$
\begin{aligned}
f(s1,s2,i,j) &= 0 && i = len(s1) \text{ or } j = len(s2) \\
&= 1 + f(s1,s2,i+1,j+1) && s1_i = s2_j \\
&= max(f(s1,s2,i+1,j), f(s1,s2,i,j+1)) && \text{else}
\end{aligned}
$$

Powerset (Backtracking): Powerset for a set. Let s be an array, n be an array length, i be an array index, b be a cache of solutions and t be a temporary solution

$$
\begin{aligned}
f(s,n,i,b,t) &= b.push(copy(t)) && i = n \\
&= t.push(n_i) && \text{else} \\
& \quad f(s,n,i+1,b,t) \\
& \quad t.pop() \\
& \quad f(s,n,i+1,b,t)
\end{aligned}
$$

Activity Selection (non-Greedy): Max activities given start and end times. Let s be an array of start times, e be an array of end times, n be a length of start and end times, i be an array index and c be a current end time

$$
\begin{aligned}
f(s,e,n,i,c) &= 0 && i = n \\
&= max(1 + f(s,e,n,i+1,e_i), f(s,e,n,i+1,c)) && c \leq s_i \\
&= f(s,e,n,i+1,c) && \text{else}
\end{aligned}
$$