COP 3503 11/12/26

More Dynamic Programming

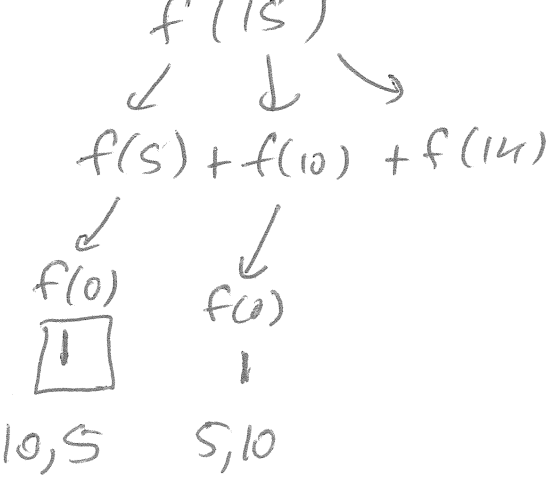
1. # ways of making change
2. fewest # of coins to make change
3. subset sum
4. 0-1 knapsack (CSES: Book Shop)
5. CSES: Grid Paths
6. Kattis: Using Digits

ways make change

Input: Denominations
1, 5, 10, 25 (sorted) $]\ d_0, d_1, d_2, \ldots d_{n-1}$
n = 15¢ (# cents)

10, 5
10, 1, 1, 1, 1, 1
5, 5, 5
5, 5, 1, 1, 1, 1, 1
5, 1, 1, 1, 1, 1, 1, 1, 1, 1





$f(n, i)$ = # ways to make change for n cents using denomination i or less.

all ways with coin i.

$$f(n,i) = f(n-d_0, 0) + f(n-d_1, 1) + f(n-d_2, 2) + \dots + f(n-d_i, i)$$

$f(n,i) = f(n-d_i, i) + f(n, i-1)$

all ways w/o coin i.

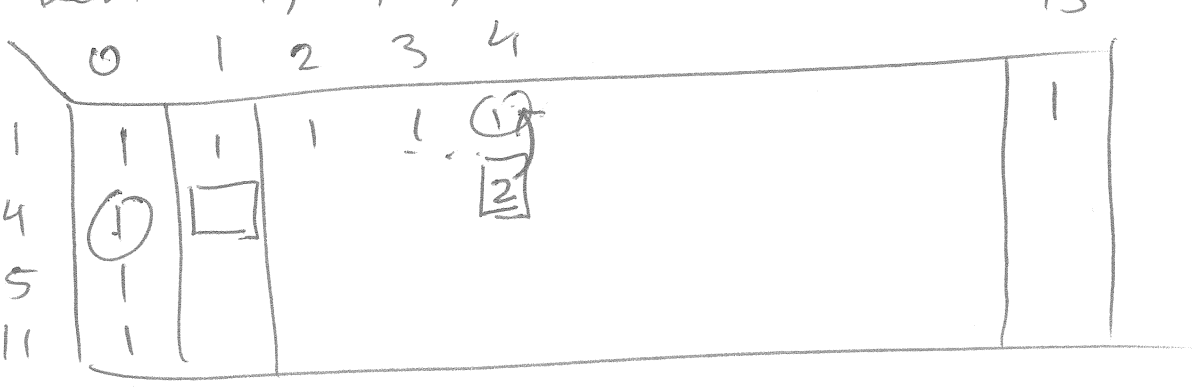
memo[n+1][k]

↑ 0 to n cents ↳ den 0 to k-1

memo[k][n+1]

```
int dp[k][n+1];
//Assume d0 = 1
Arrays.fill(dp[0], 1); // 1 way to make change for any amt using only pennies
for (int i=1; i<k; i++) dp[i][0] = 1;
for (int i = 1; i < k; i++) {
    for (int cents = 1; cents <= n; cents++) {
        dp[i][cents] = dp[i-1][cents];
        if (cents >= den[i]) dp[i][cents] += dp[i][cents - den[i]];
    }
}
return dp[k-1][n];
```

Den: 1, 4, 5, 11 n = 15



Suggested exercise: take my code + remove the loop also play around w/ different denominations.

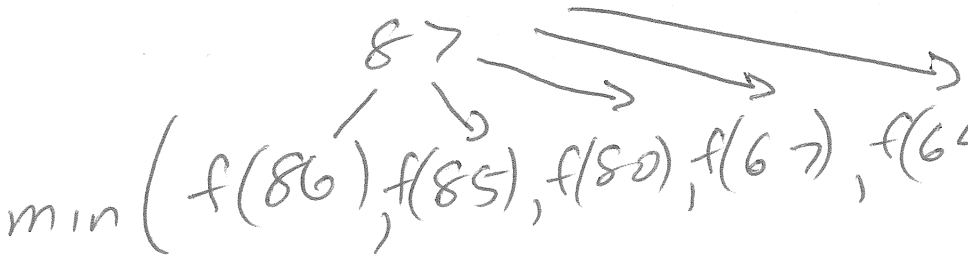
Fewest Coins

Same Input as previous problem

den: 1, 2, 7, 20, 23

n = 87 cents

What is the fewest # coins we need to use



f(0) = 0

```
int fewest(int n) {
if (n == 0) return 0;
if (memo[n] != -1) return memo[n];
int res = -1;
for (int i=0; i<k; i++) {
    if (den[i] > n) break;
    int tmp = fewest(n - den[i]);
    if (tmp == -1) continue;
    if (res == -1 || tmp+1 < res)
        res = tmp+1;
}
return memo[n] = res;
```

fewest coins

Suggested Exercise: code memoized version add to class example.

Subset sum → pos ints

$S = \{3, 18, 5, 27, 13\}$

$t_1 = 50$

$t_2 = 37$

$O(nT)$ ← target value; n ↑ # items set

Consider items 1 by 1.

boolean array hasSubset[i] = true if there exists a subset of size i, false otherwise

```
hasSubset[T+1]; // stores ans for all int from 0 to T inclusive
hasSubset[0] = true; // rest false

for (int i = 0; i < n; i++) { //go through item
    //val is subset I'm trying to make
    for (int val = T; val >= S[i]; val--)
        hasSubset[val] || = hasSubset[val-S[i]]
```

0-1 Knapsack (SS is a instance of knapsack)

Max weight of knapsack W (Similar to T subsetsum)

Set has items

i	wi	vi
0	5	9
1	2	3
2	4	7
3	6	13
4	3	5

W = 10

Knapsack array

dp[j] = max value of knapsack with max weight j.

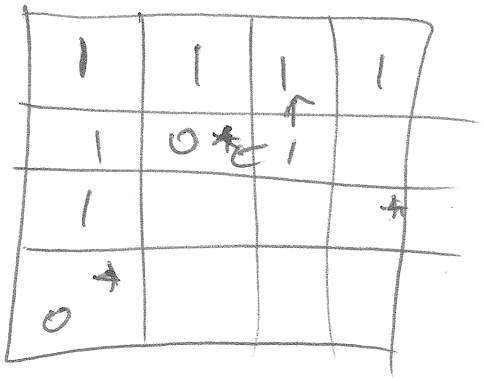
Do I want object W=4, V=7

Currently dp[5] = 9

dp[9] = max(dp[9], dp[5] + S[i].v)

9 - S[i].w

Grid Paths




```
if   g[i][j] == '*'
        dp[i][j] = 0
else
     g[i][j] = g[i-1][j] + g[i][j-1]
```