Dynamic Programming

Thursday, October 8, 2020 1:43 PM

Subset Sum Revisited.

Given an array of integers A, where $|A| = n$,

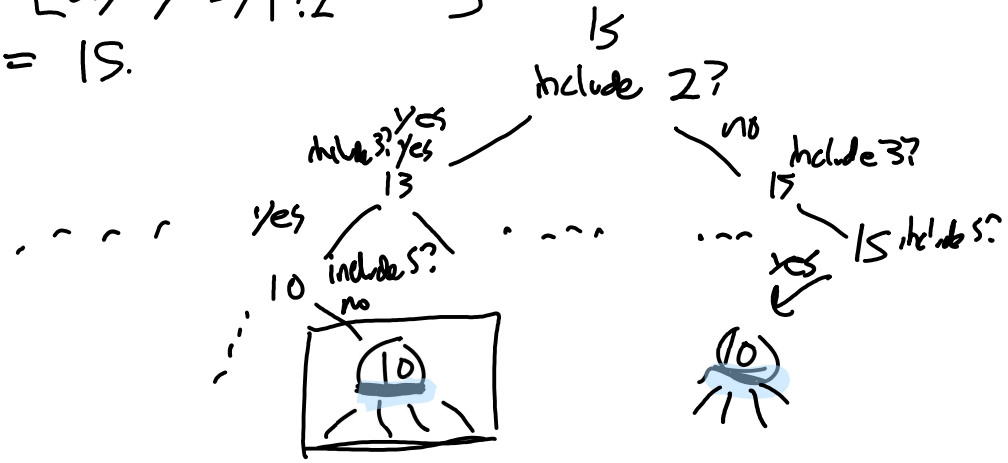
* such that every element $A[i]$ satisfies $0 \le A[i] \le n$,

decide whether there exists a subset that sums to t.

$$SSum(A, t) = \begin{cases} \text{if } t = 0, T \text{ otherwise} & A = [\,] \\ SSum(A', t - x) & A = x, A' \text{ and } t - x \ge 0 \\ \text{or } SSum(A', t) & \\ SSum(A', t) & A = x, A' \text{ and } t - x < 0 \end{cases}$$

ex: $A = [2, 3, 5, 7, \ldots]$

$t = 15$.



Change the arguments to be integer indices.

$SSum(A, t) \Rightarrow SSum(i, t)$

array, integer → integers.

$SSum(i, t)$ will decide the solution to $SSum(A[i..n-1], t)$

$SSum(i, t)$: ← target

$t = 0$ ... $i = n$ // end of the array

OR $ssum(i+1, t - A[i])$ — $i < n$, $t - A[i] \geq 0$ — $A = x, A'$

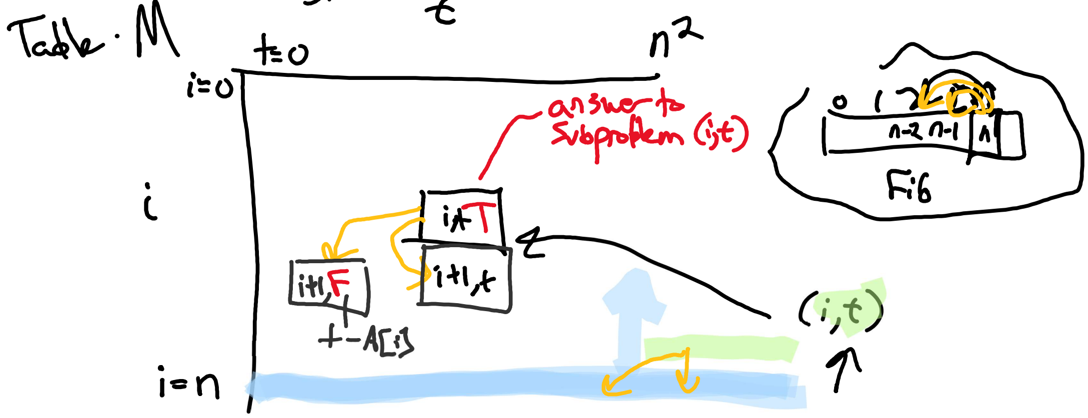
$ssum(i+1, t)$

$ssum(i+1, t)$ — $i < n$, $t - A[i] < 0$ — $A[i], A[i+1, ...]$

Each subproblem is named (i, t)

meaning, is there subset of $A[i \ldots n-1]$ that adds up to t.

Table: M



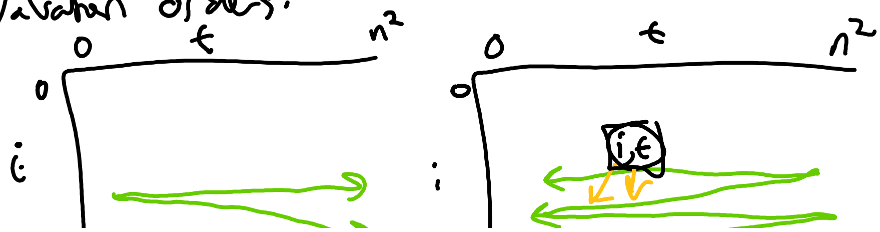
- To store answers to all subproblems, we'll need a $n \times n^2$ table.
- Evaluation order: each i,t depends on $i+1$, and $\leq t$
  ... decreasing values of i, decreasing values of t
- Running time: # of subproblems × time to compute each
  $O(n^3) \times O(1) = O(n^3)$
- Algorithm for pseudo code:
  Iteratively fill in values in table.

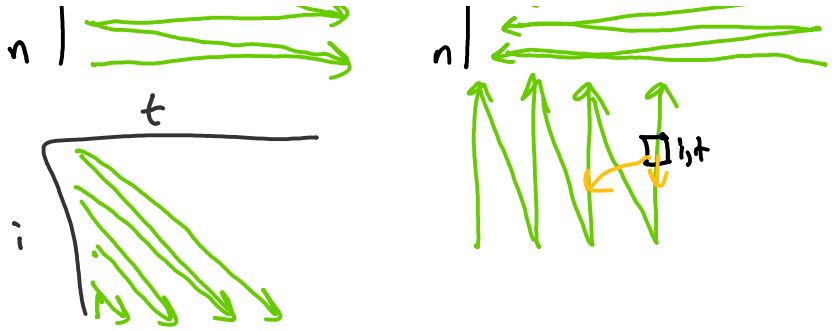
$O(n^3)$

```
for i = n down to 0 // from 0 to n
    for t = 0 up to n^2 // from 0 to n^2
        if i = n: M[i,t] = T if t=0, F otherwise.
        else: if t - A[i] >= 0:   // if we include A[i]
            M[i,t] = OR  M[i+1, t-A[i]]
                         M[i+1, t)   // if we do not
```

$O(1)$

Evaluation orders: (Many are acceptable)





Implicit memoization.

In this class
We'll use explicit memoization

Next time: Longest Increasing Sequence.