Thursday, March 10, 2022 10:57 AM

More Dynamic Programming :

- → Define subproblem
- → Recursive Formula
- → Memoization, Evaluation order → Iterative Algo.

Ex 1 : (Subset-Sum)

Given a set of distinct integers: $a_1, a_2, \dots, a_n > 0$
and a target $T > 0$.

Does there exist a subset S of these n integers summing to T?

eg. $3, 5, 10, 19, 33, \boxed{23}$

$T = 48$ $\{5, 10, 33\}$ → True

$T = 17$ → NO.

Intuition: For each number
→ Decide if to pick it or not.
→ If we decide to pick a number a_i then target reduces by a_i. → $(T - a_i)$

→ Define Subproblems: Suppose we consider numbers from a_n to a_1

$C(i, j) =$ True iff $\exists$ a subset of $a_1 \dots a_i$ that sums to j.

$i = 0$ to n , $j = 0$ to T

$i=0 \Rightarrow$ no integer available $\equiv$ Empty prefix

$j=0 \Rightarrow$ Target is reached & do not need to pick any more numbers.

$\Rightarrow$ Answer : $C(n, T)$

$\Rightarrow$ Base cases : $C(i, 0) = \text{True}$ $\quad i = 0, \dots, n$

$C(0, j) = \text{False}$ $\quad j = 1 \dots T$

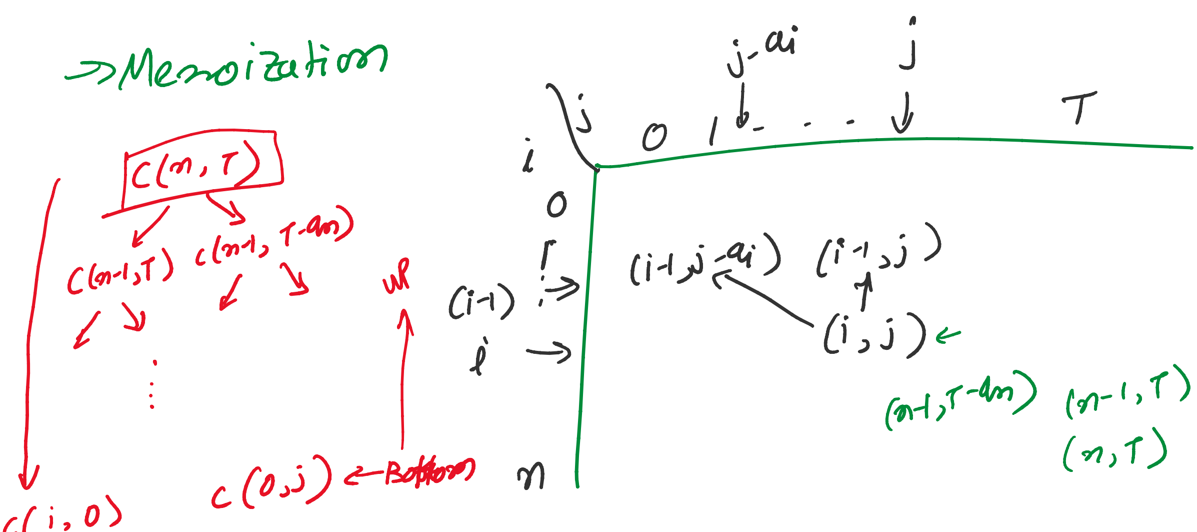
$\Rightarrow$ Recursive Formula : $C(i, j)$

Case I : i is not picked. $\rightarrow C(i-1, j) \rightarrow T/F$

Case II : i is picked $\rightarrow C(i-1, j-a_i)$ if $a_i \le j$. $\rightarrow T/F$

$$\underline{C(i,j)} = \begin{cases} C(i-1, j) \vee C(i-1, j-a_i) & \text{if } a_i \le j \\ C(i-1, j) & \text{o.w.} \end{cases}$$

$\Rightarrow$ Memoization



$\Rightarrow$ Evaluation Order : $i = 0$ to n

For each i, $j = 0$ to T.

→ Evaluation order: $i = 0$ to n
For each i, $j = 0$ to T.

→ Psuedo Code: Exe.

→ Analysis: # subproblems is $O(nT)$.
For each we take $O(1)$
Total time is $O(nT)$
space is $O(nT)$
↪ improve to $O(T)$

Q: Size of the i/p if each of $a_1 \ldots a_n$ & T takes 100 bits.

$$(n+1) \cdot 100 = O(n)$$

Note: Dependence on T is essentially exponential in the size of the i/p, because to represent T we need $\log T$.

Can we get poly in n? (unlikely).
(Best known: $\tilde{O}(\sqrt{n}\, T)$ by Koiliaris & Xu '16)

→ Output the subset:

Variations:

→ what if a number can be used multiple time?

$$C(i,j) = \begin{cases} C(i-1, j) \vee C(i, j-a_i) & \text{if } a_i \le j \\ \ldots & \text{o.w.} \end{cases}$$

$$c(i,j) = \begin{cases} c(i-1,j) \vee c(i, \ldots & \\ c(i-1,j) & \text{o.w.} \end{cases}$$

$c(j) =$ true iff $\exists$ a multi-set of $a_1 \ldots a_n$ that sums to j

$$c(j) = \bigvee_{\substack{i=1 \ldots n \\ a_i \leq j}} c(j-a_i)$$

→ what if we want smallest subset summing to T

$c(i,j) =$ size of the smallest subset of $a_1 \ldots a_i$ summing to j.

$$c(i,j) = \min\{c(i-1,j), \boxed{1+} c(i-1, j-a_i)\}$$

$\boxed{w_i +}$

→ we want min-weight subset summing to T. where each element a_i has weight w_i.

$$\text{weight}(S) = \sum_{a_i \in S} w_i$$

→ what if we want check if $\exists$ a subset of size exactly k summing to T.

$$c(i,j,l) = c(i-1,j,l) \vee (i-1, j-a_i, l-1)$$

Ex 2: Max-Weight Independent set on Trees.

Given a Tree undirected $T=(V,E)$ (no cycles), $w(v) > 0$ weight for each $v \in V$

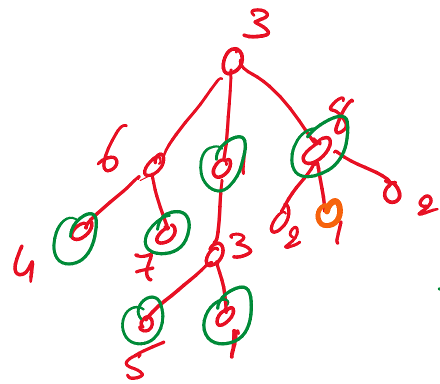
find subset $S \subseteq V$ of max-weight

Given a tree $T(V,E)$ ... for each $v \in V$

Find subset $S \subseteq V$ of max-weight s.t. $u, v \in S \Rightarrow (u,v) \notin E$.

Weight of $S = \sum_{v \in S} w(v)$

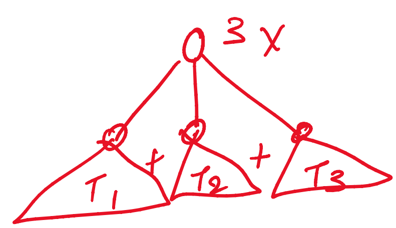
e.g.



$$\boxed{4+7+5+1+1+8}$$

→ Define subproblems:

Intuition



$\boxed{OV}$ = $T(v)$ $A(v)$ = weight of max-weight independent set of sub-tree rooted at node $v = T(v)$

→ Answer: $A(\text{root})$

→ Base cases: For each leaf v, $A(v) = w(v)$

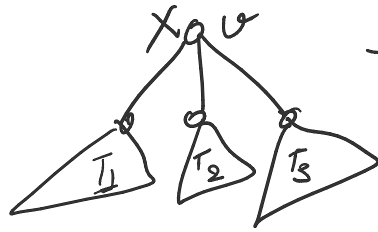
$A(\emptyset) = 0$

→ Recursive Formula: $A(v)$

case I: Not pick v.

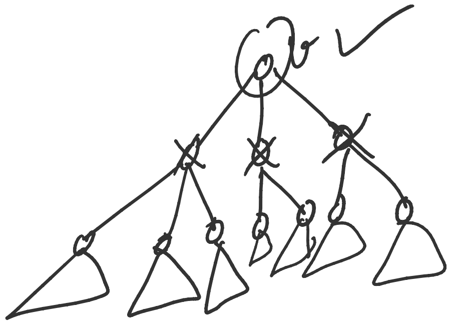
$T(v)$  $\rightarrow \sum A(u)$

$T(V)$



$\longrightarrow \sum_{\text{child } u \text{ of } v}$

Case II: Pick v.

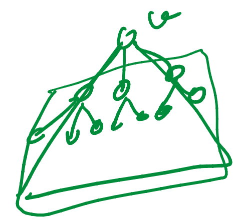


$\longrightarrow w(v) + \sum_{\text{grandchild } u \text{ of } v} A(u)$

$|V| = n$

→ Memoization $\quad A[v_1] \ldots \ldots A[v_n]$

$$A(v) = \max \begin{cases} \sum_{\text{child } u \text{ of } v} A(u), \\ w(v) + \sum_{\text{grandchild } u \text{ of } v} A(u) \end{cases}$$



→ Evaluation Order: Post. order traversal.

→ Pseudo code

MIS($T=(V,E)$)

$[v_1 \ldots v_n]$ = post order traversal of T.

root

for $i = 1..n$

Base case ← if v_i is leaf then $A[i] = w(v_i)$

else

$$A[i] = \max \begin{cases} \sum_{v_k \text{ child of } v_i} A[k], \\ w(v_i) + \sum_{v_k \text{ grandchild of } v_i} A[k] \end{cases}$$

$\lfloor w(v_i)$

v_k grand child of v_i

Return $A[n]$.

→ Analysis : # subproblem $O(n)$.
For each subproblem time taken
to compute the two summations
can be $O(n)$.

$O(n^2)$ running time

$O(n)$ space.

Can we improve?

→ Better Analysis :

Q: How many times does $A(u)$
appear in the L.H.S. summation?

A: At most twice.
once for its parent &
once for its grandparent

$\Downarrow$

$O(n)$ running time