March 5, 2019 10:58 AM

More Dynamic Programming

Ex1 (Subset Sum)

Given n integers $a_1, .., a_n > 0$ and T, does there exist subset summing to T?

Define subproblems $(i=0,..,n,\ j=0,..,T)$

$C(i,j) =$ true iff $\exists S \subseteq \{a_1, ..., a_i\}$ summing to j

Final Ans: $C(n,T)$.

Base cases: $C(0,0) =$ true

$C(0,j) =$ false $\quad j \geq 1$

Recursive formula:

Consider opt sol'n for $C(i,j)$

Case 1. not use a_i $\rightarrow C(i-1,j)$

Case 2. use a_i $\rightarrow C(i-1, j-a_i)$

$$\boxed{C(i,j) = \begin{cases} C(i-1,j) \vee C(i-1, j-a_i) & \text{if } j \geq a_i \\ C(i-1,j) & \text{if } j < a_i \end{cases}}$$

evaluate in increasing order of i

$\Rightarrow$ # subproblems $O(nT)$

time per subproblem $O(1)$

$\Rightarrow$ # subproblems

time per subproblem $O(1)$

$\Rightarrow$ $\boxed{O(nT)}$ time

Retrieving opt sol'n:

```
OutputAns(i, j):
    if i = 0 return
    if C(i-1, j) true
        OutputAns(i-1, j)
    else { OutputAns(i-1, j-a_i), print a_i }
Call OutputAns(n, T).
```

extra time $O(n)$

Rmk: unfortunately T can be big!

polynomial in n? not likely!

[$\tilde{O}(\sqrt{n}\,T)$ by Koiliaris, Xu '16]

Variations:

- what if a number can be used more than once?

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

(or drop i: $C(j) = \bigvee_{\substack{1 \le i \le n \\ a_i \le j}} C(j-a_i)$)

subproblems $O(T)$

time per subprob. $O(n)$ $\Rightarrow$ $O(nT)$ time)

- what if we want smallest-size sol'n?

define $C(i,j)$ = ~~size~~ weight of smallest subset of $\{a_1, \ldots, a_i\}$ summing to j

formula: $C(i,j) = \min\{C(i-1,j),\ C(i-1, j-a_i) \cancel{+1} + w_i\}$

formula: $C(i,j) = \min\{C(i-1,j),\ C(i-1,j-a_i)\ \cancel{+1}\}$

- what if we want smallest-weight sol'n where each element a_i is given a weight w_i?

- what if we want subset to have size exactly k?

define $C(i,j,\ell)$ = true iff $\exists S \subseteq \{a_1,..,a_i\}$ of size ℓ summing to j

formula: $C(i,j,\ell) = C(i-1,j,\underline{\ell}) \vee C(i-1,j-a_i,\underline{\ell-1})$

$\Rightarrow O(knW)$ time

⋮

Ex 2 (Edit distance)

Given strings $a_1 \cdots a_m$, $b_1 \cdots b_n$,

find min # edits ← insert/delete/change a char

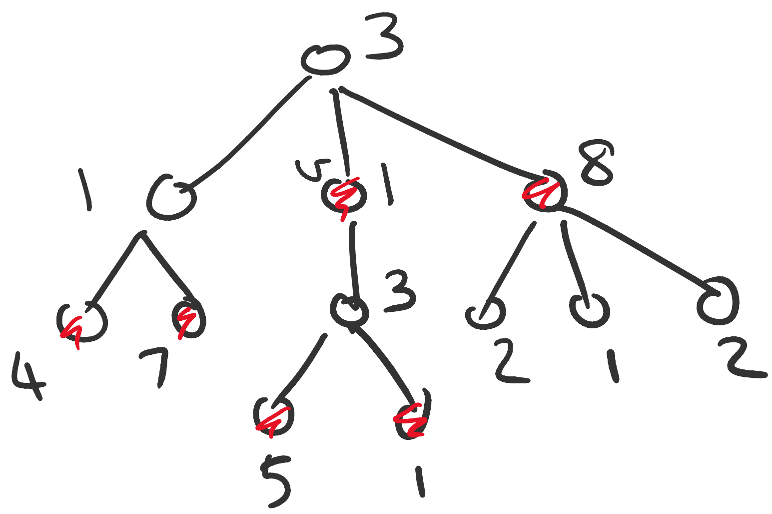
Define $D(i,j)$ = edit dist between $a_1 \cdots a_i$ & $b_1 \cdots b_j$

Formula:

$$D(i,j) = \begin{cases} \min\{\underbrace{D(i-1,j)+1}_{\text{delete } a_i},\ \underbrace{D(i,j-1)+1}_{\text{insert } b_j},\ \underbrace{D(i-1,j-1)+1}_{\text{change } a_i \text{ to } b_j}\} & \text{if } a_i \neq b_j \\ \min\{D(i-1,j)+1,\ D(i,j-1)+1,\ D(i-1,j-1)\} \end{cases}$$

$$\min\{D(i-1,j)+1,\ D(i,j-1)+1,\ D(i-1,j-1)\}\quad \text{if } a_i = b_j$$

Ex3: max-weight Independent Set for a Tree



each vertex v has weight $w(v) > 0$

$8 + 7 + 4 + 5 + 1 + 1 = 26$

Define subproblems:

$A(v) =$ weight of max-weight indep set for subtree at v.

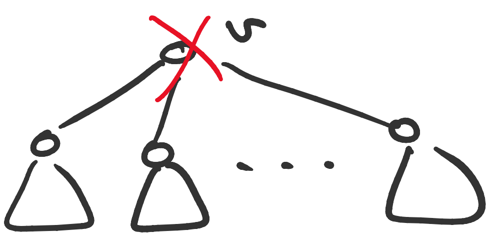
Final Ans: $A(\text{root})$.

Base cases: if v leaf, $A(v) = w(v)$.

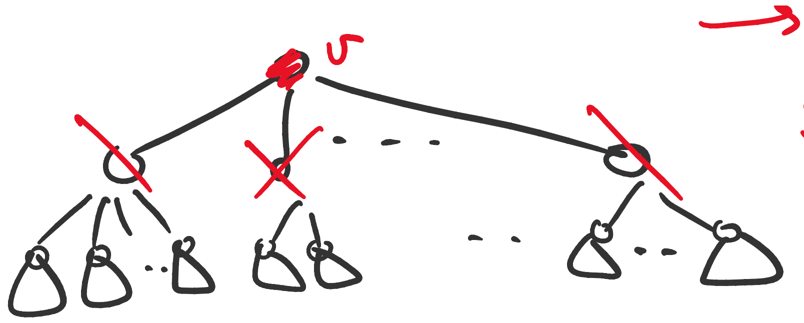
Recursive formula:

Consider opt sol'n for $A(v)$.

Case 1. not use v $\rightarrow \sum_{\text{child } u \text{ of } v} A(u)$



Case 2. use v $\rightarrow \sum A(u) + w(v)$



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

$$A(v) = \max \left\{ \sum_{\substack{\text{child } u \\ \text{of } v}} A(u), \left(\sum_{\substack{\text{grand-child} \\ u \text{ of } v}} A(u) \right) + w(v) \right\}$$

Evaluation order: postorder traversal of tree

Analysis: # subproblems $O(n)$

time per subproblem $O(n)$

$\Rightarrow O(n^2)$ total time

Better analysis: each node u examined twice (at its parent & grandparent)

$\Rightarrow \boxed{O(n)}$ time