Lecture 5 Dynamic Programming 2

Monday, January 27, 2020 2:28 PM

- Longest Increasing Subsequence (LIS)

a[] = {4, 2, 5, 3, 9, 7, 8, 10, 6}

b[] = {2, 5, 7, 8, 10} length = 5

- attempt 1

6 is not in LIS , LIS{4, 2, 5, 3, 9, 7, 8, 10}

6 is in LIS

finding LIS{4, 2, 5, 3, 9, 7, 8, 10}

{2, 5, 7, 8, 10}

does not work because

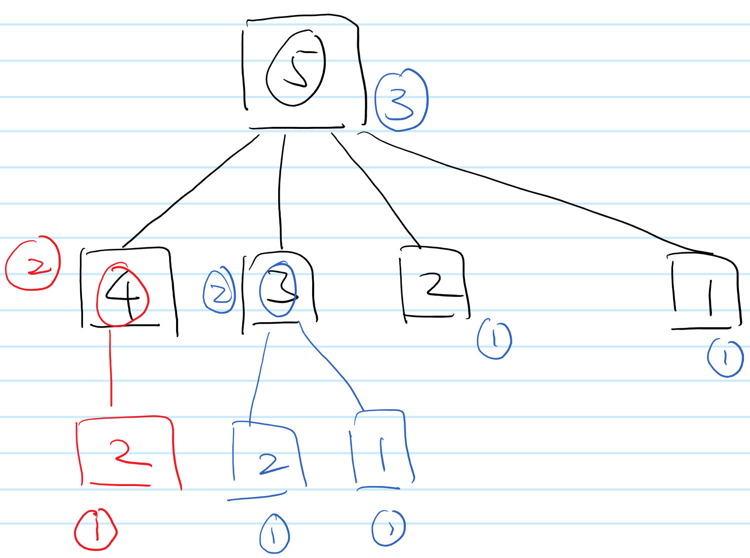
{2, 5, 7, 8, 10, 6} is not an increasing subsequence

want: LIS{4, ... , 10} if $\boxed{\text{every number} < 6}$

- attempt 2

LIS_recursive(k) return length of LIS ending at a[k]

{4, 2, 5, 3, 9}



- recursive search: call LIS_recursive (2) multiple times
- dynamic programming

state: let $f[i]$ be the length of LIS ending at $a[i]$.

transition function:

$$f[i] = \max \begin{cases} 1 & \text{sequence is } a[i] \\ f[j]+1 & \text{for every } j<i,\ a[j]<a[i] \end{cases}$$

sequence is $\{$ LIS ending at $a[j]$, $a[i]\}$

$$\begin{array}{l} \quad\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9 \\ a[\,] = \{4, 2, \underline{5}, \underline{3}, \underline{9}, \underline{7}, 8, 10, 6\} \end{array}$$

i	1	2	3	4	5	6	7	8	9
$f[i]$	1	1	2	2	3	3	4	5	3

set up base case

for $i = 1$ to n

evaluate the transition function at $f[i]$

code for outputting solution (output max $f[i]$)

- analyze running time

running time = # states $\times$ time for evaluating one transition function

LIS: n $\quad O(n)$

running time: $O(n^2)$

Knapsack: nW $\quad O(1)$

running time: $O(nW)$

- Proof for Correctness.

use induction

Induction hypothesis: "smaller subproblems are computed correctly"

before i-th iteration, for every $j < i$, $f[j]$ is length of LIS ending at $a[j]$.

induction: when computing $f[i]$

let $b[]$ be the LIS ending at $a[i]$

case ①: $b[]$ has length 1, considered by the 1st case of transition function

case ②: let $a[j]$ be the second-to-last number in $b[]$, by definition $j < i$
$a[j] < a[i]$

by IH length $b[] \le \underline{f[j]} + 1$

$f[j]+1$ is considered in transition function.

therefore $f[i]$ is also computed correctly.