CompSci 161
Winter 2023 Lecture 15:
Finishing Dynamic Programming

Iterative Version: Topological Order

- Caution: some recursive calls to *higher* values.
- We can't iterate increasing i and j together.
- $\text{Tree}[i, j]$ will make calls to:
  - $\text{Tree}[i,\ r-1]$ for $i \le r \le j$
  - $\text{Tree}[r+1,\ j]$ for $i \le r \le j$

- For example, $\text{Tree}[2, 5]$ will call:
  - $\text{Tree}[2, 1]$ and $\text{Tree}[3, 5]$ $(r = 2)$
  - $\text{Tree}[2, 2]$ and $\text{Tree}[4, 5]$ $(r = 3)$
  - $\text{Tree}[2, 3]$ and $\text{Tree}[5, 5]$ $(r = 4)$
  - $\text{Tree}[2, 4]$ and $\text{Tree}[6, 5]$ $(r = 5)$

Table looks like

	k_1	k_2	k_3	k_4	k_5	k_6	k_7
k_1	.13						
k_2		.21					
k_3			.11				
k_4				.01			
k_5					.22		
k_6						.08	
k_7							.24

How to get the tree itself?

	k_1	k_2	k_3	k_4	k_5	k_6	k_7
k_1	0.13	0.47	0.69	0.72	1.28	1.52	2.12
k_2		0.21	0.43	0.46	1	1.17	1.73
k_3			0.11	0.13	0.47	0.63	1.19
k_4				0.01	0.24	0.4	0.95
k_5					0.22	0.38	0.92
k_6						0.08	0.4
k_7							0.24

$r=1$ $\quad \sum P_k + \text{Tree}[1, r-1] + T[r+1, j]$

$r=2$

$\vdots$

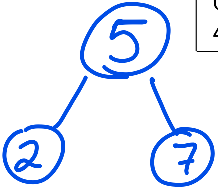
$r=7$

How to get the tree itself?

for $i \leftarrow 1 \ldots n$ **do**
    Tree$[i, i-1] \leftarrow 0$
    Tree$[i, i] \leftarrow p_i$
for $\delta = 1$ to $n - 1$ **do**
    for $i = 1$ to $n - \delta$ **do**
        $j = i + \delta$
        // Tree[i,j] gets filled in here.
        // some value r minimized Tree[i,j] = ...
        roots[i,j] = r

Table With Roots

	k_1	k_2	k_3	k_4	k_5	k_6	k_7
k_1	0.13 1	0.47 2	0.69 2	0.72 2	1.28 2	1.52 2	2.12 5
k_2		0.21 2	0.43 2	0.46 2	1 3	1.17 5	1.73 5
k_3			0.11 3	0.13 3	0.47 5	0.63 5	1.19 5
k_4				0.01 4	0.24 5	0.4 5	0.95 7
k_5					0.22 5	0.38 5	0.92 7
k_6						0.08 6	0.4 7
k_7							0.24 7



Reinforcement: Draw the Tree

- Be able to do it from a table.
  - Great practice for all demonstrated algorithms

- Write a function:
  `Node * printTree(roots, i, j)`

Independent Set on Trees

- *Independent Set* : $V' \subseteq V$, no shared edges.
- Today: input graph is a tree., rooted
- MIS(v) : size largest I.S. subtree rooted at v. (or value, to generalize)

base: nullptr: 0
leaf: value of this

include v = value(v) + $\sum_{\text{grandchild } u}$ MIS(u)

not include v = $\sum_{\text{children } c}$ MIS(c)

return max(include v, not include v)