CSR 102 11-2-21

Dynamic Programming (Chap 15)

Problem: compute $\binom{n}{k}$ ← Binomial Coefficient

defn: $\binom{n}{k}$ = # k-element s/sets of an n-set where $0 \le k \le n$.

Thm: $\binom{n}{k} = \dfrac{n!}{k!(n-k)!}$

Thm: (Pascal's Thm)

$$\binom{n}{k} = \begin{cases} 1 & k = 0, n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & 0 < k < n \end{cases}$$

Pseudo-code:

BC(n, k)

1. if k = 0 or k = n
2. return 1
3. else
4. return BC(n-1, k-1) + BC(n-1, k)

Basic op: +

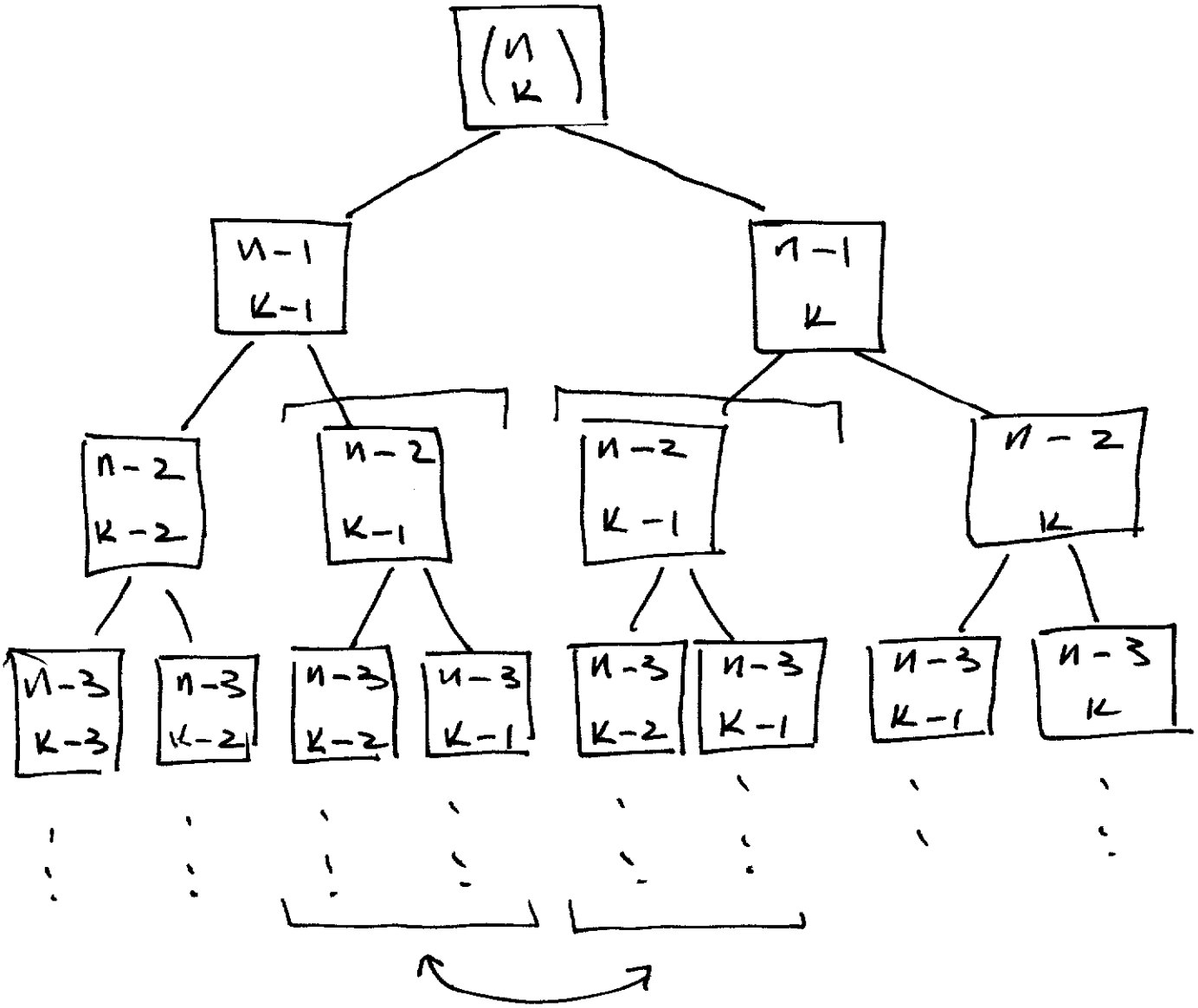
basic ops = $\binom{n}{k}$

Recall if $n = 2k$, then

$$\binom{n}{k} = \binom{2k}{k} = \Theta\left(\frac{4^k}{\sqrt{k}}\right) = \Theta\left(\frac{2^n}{\sqrt{n}}\right)$$

what's the problem?

Box trace:



identical subtrees

Two ways around this:

- memoization: Passing a partially filled table of $\binom{n}{k}$ values, and only recur if don't have value in table, otherwise compute & insert in table.

- Dynamic Programming.

  Iterative Version

Pascal's Δ as a table

	0	1	2	3	...	K-1	K
0	1	0	0	0	..	0	0
1	1	1	0	0	..	0	0
2	1	2	1	0	..	0	0
3	1	3	3	1	..	0	0
⋮	⋮	⋮	⋮	⋮		⋮	⋮
n-1	1					$\binom{n-1}{K-1}$	$\binom{n-1}{K}$
n	1						$\binom{n}{K}$ ← goal

Pseudo code:

DynBC(n, k)

```
1.  C[0] = 1
2.  for i = 1 to k
3.      C[i] = 0
4.  for i = 1 to n
5.      for i = k down-to 1
6.          C[i] = C[i-1] + C[i]
7.  return C[k]
```

Exercise: figure out how to minimize further.

Runtime: $\Theta(nk)$

Typically, Dynamic Programming is is used to solve optimization Problems.

Problem: Coin changing

Suppose have unlimited supply of coins in denom. $\{d_1, d_2, d_3, \ldots, d_n\}$.

Pay N units of money with fewest number of coins. Two questions:

- what is least # of coins needed to pay N units? [Value of opt. soln].
- exactly which coins are to be paid? [the optimal solution].

Create a 2-dim table

$$C[\underbrace{1 \ldots n}_{\text{rows}}; \underbrace{0 \ldots N}_{\text{columns}}]$$

define (for $1 \le i \le n$, $0 \le j \le N$)

$C[i,j]$ = min # of coins needed to pay j units using denom. set $\{d_1, \ldots, d_i\}$

Goal: find $C[n,N]$

To solve sub problem for cell (i,j), we have 2 choices

(1) use no coins of value d_i (even though it allowed.) least # coins in this case is $C[i-1,j]$

or

(2) use at least one coin of value d_i. In this case, the least # of coins is

$$1 + C[i, j-d_i]$$

Thus

$$C[i,j] = \min\left(C[i-1, j],\ 1 + C[i, j-d_i]\right)$$

<u>note</u>:

if $i=1$ or $j < d_i$, then one of values falls outside table, regard such values as ∞. If both i & j are out of bounds, the table value $C[i,j] = \infty$.

→ see handout on Dynamic Prog.

Pages: 3 and 4.