Continue selection problem ....

let $t(n)$ = avg. # of array comp. by RandSelect(A, 1, n, i).

Assume all permutations of $A[1 \ldots n]$ are equally likely, so q is equally likely to be any $1 \le q \le n$.

$$t(n) = \frac{\sum_{q=1}^{n} \left((n-1) + P(i<q)\,t(q-1) + P(i>q)\,t(n-q) \right)}{n}$$

where

$$P(i<q) = \frac{n-i}{n} \quad , \quad P(i>q) = \frac{i-1}{n}$$

$i < q \le n$ $\qquad$ $1 \le q < i$

so

$$t(n) = (n-1) + \frac{1}{n}\sum_{q=1}^{n}\left(\left(\frac{n-i}{n}\right)t(q-1) + \left(\frac{i-1}{n}\right)t(n-q)\right)$$

$$= (n-1) + \frac{1}{n^2}\left(\sum_{q=1}^{n}(n-i)\,t(q-1) + \sum_{q=1}^{n}(i-1)\,t(n-q)\right)$$

$$= (n-1) + \frac{1}{n^2}\left(\sum_{q=1}^{n-1}(n-i)\,t(q) + \sum_{q=1}^{n-1}(i-1)\,t(q)\right)$$

$$= (n-1) + \frac{1}{n^2}\cdot(n-i+i-1)\cdot\sum_{q=1}^{n-1}t(q)$$

hence

$$t(n) = (n-1) + \left(\frac{n-1}{n^2}\right)\cdot\sum_{q=1}^{n-1}t(q)$$

Exercise: $t(n) = \Theta(n)$

Handout: Dynamic Programming.

Problem: Compute binomial coefficients

$$\binom{n}{k} = (\#\text{ of } k\text{-subsets of an } n\text{-set})$$

formula: $$\binom{n}{k} = \frac{n!}{k!(n-k)!} \qquad (0 \le k \le n)$$

Pascal's identity:

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

Generate Pascal Δ:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

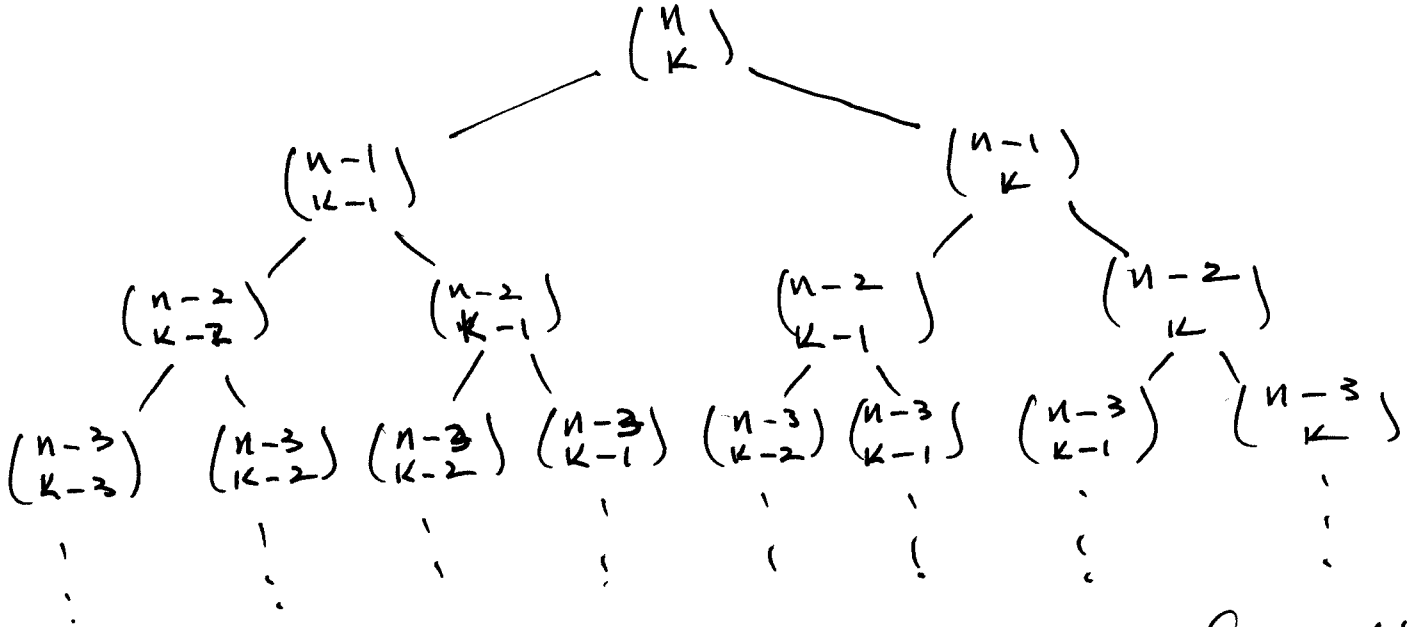
⋮ ⋮ ⋮ ⋮ ⋱

Pseudo-code

Binomial Coeff(n, k)

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

recursion tree:



many sub-instances are solved multiple times.

Runtime

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

↑ if $n=2k$

↗ exponential runtime.

More efficient approach:
maintain a table of intermediate results. when a subinstance value is needed, first look up in table to see if it's already computed. If yes, return it, otherwise compute and save in table.

Iterative implementation is called Dynamic Programming.

see DynBinCoeff(n, k) in handout.

$$\text{Cost} = 1 + \Theta(k) + \Theta(nk) = \Theta(nk)$$

If $n = 2k$ we have $k = \frac{n}{2}$

$$\text{cost} = \Theta(2k^2) = \Theta(n^2)$$

Exercise

write recursive version.

Problem: Coin Changing.

We have coins in n denominations

$$\{d_1, d_2, d_3, \ldots, d_n\}$$

where each $d_i \geq 1$. Assume have unlimited supply of coins in each denomination. Goal: Pay out N units using least # of coins.

Two questions.

- Value of optimal solution: what is the least # of coins needed.
- optimal solution: exactly how many coins in each denomination should be disbursed.

for i^{th} question, define $C[i,j]$

for $1 \le i \le n$, $0 \le j \le N$.

$C[i,j]$ = min # of coins necessary to pay j units using only coins $\{d_1, d_2, \ldots, d_i\}$