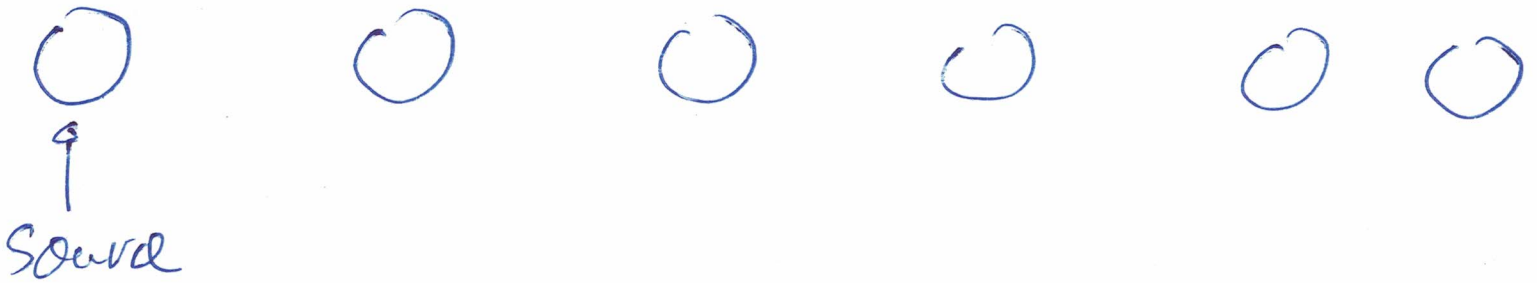
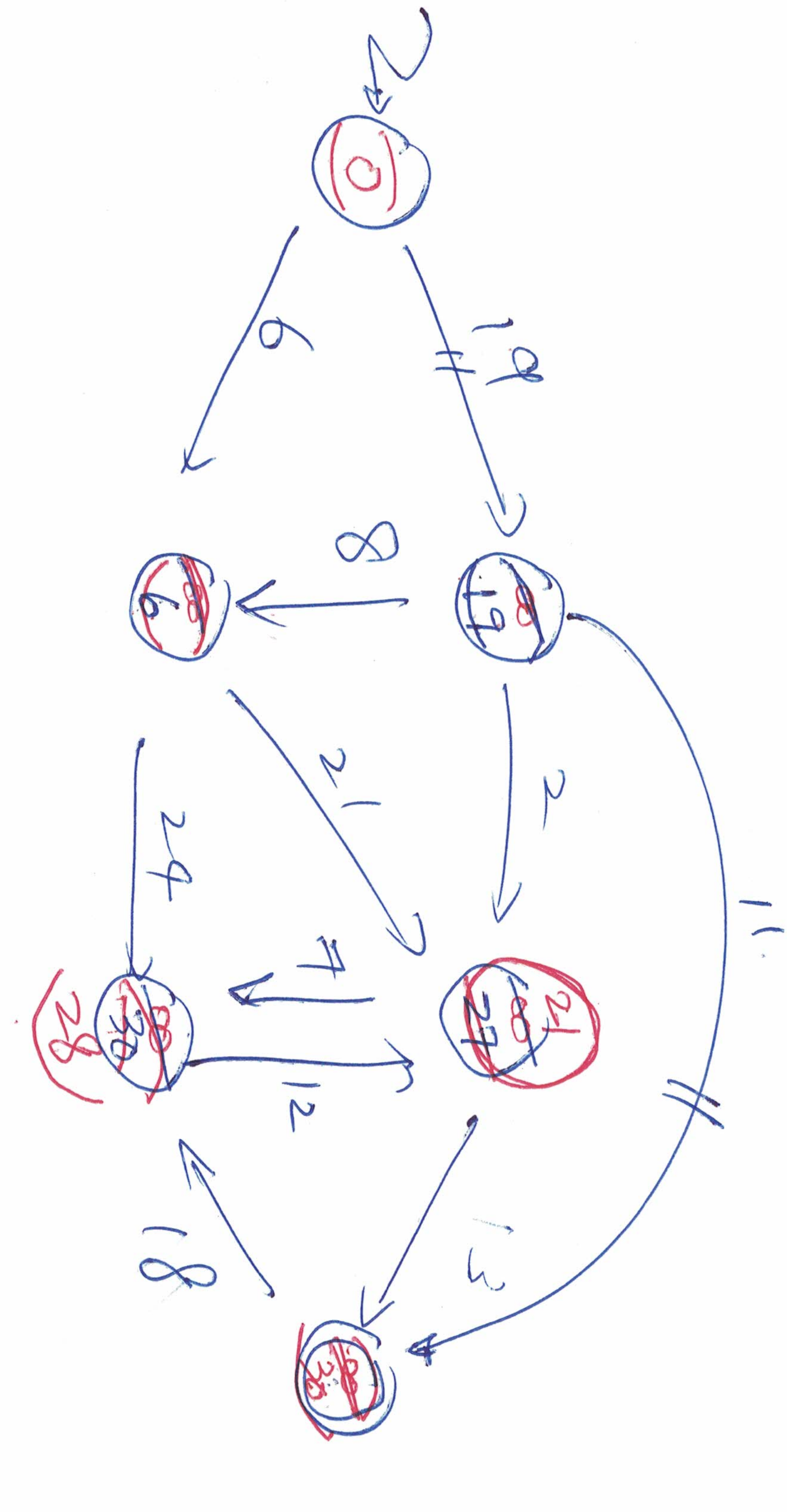
Cpts 515 . 9/2/2020

Last time: shortest-path ⟵ intuition.

Greedy Alg's key: identify monotonicity.

Shortest path:





Dynamic Programming.

my Intuition: Recursive Programs/Algs are often inefficient!

Eliminate the recursion $\equiv$ Dynamic Programming.

↳ the reverse process of recursion.

LCS Alg. // longest common subseq.

$\alpha =$ AATTTGATGCCTA

$\beta =$ TAAGCTAACTTATTAA

① define subseq. (subseq ≠ substring).

ATCT.

② common subseq. which a subseq of both α and β.

③. LCS(α, β): a longest common subseq.

Studying the problem before designing the ALG.

⇓

Figure out good properties.

empty string

$LCS(\alpha, \wedge) = \wedge$

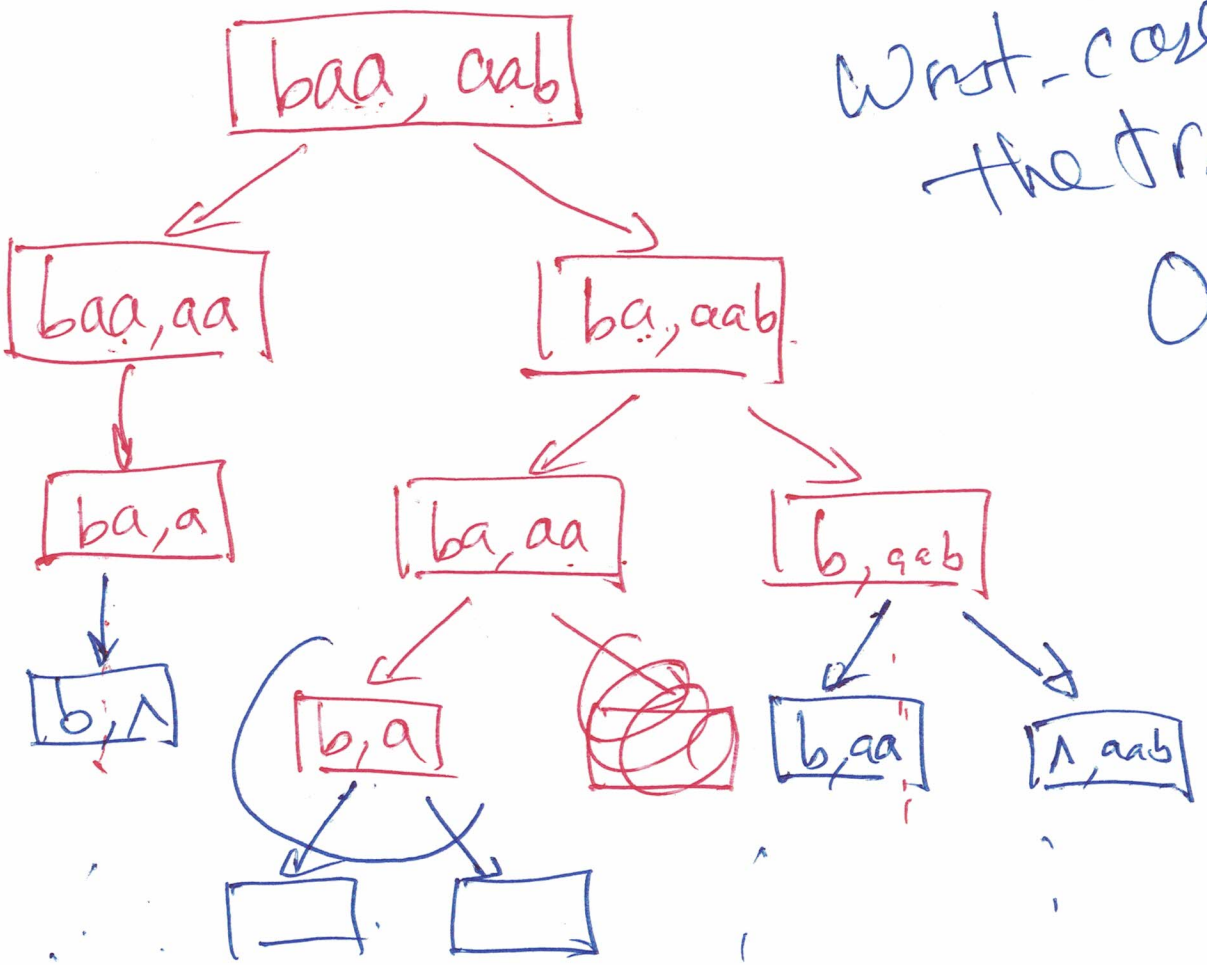
$LCS(\wedge, \beta) = \wedge$

$LCS(\alpha a, \beta a) = LCS(\alpha, \beta)\, a$

$LCS(\alpha a, \beta b) =$ longer of $LCS(\alpha a, \beta)$ and $LCS(\alpha, \beta b)$.

$a \neq b$

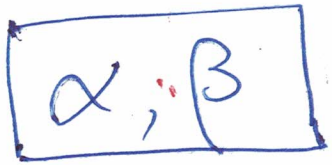
From the 4 properties, we already have recursive alg. Is it efficient? No. Exple:

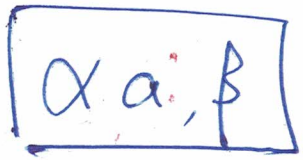


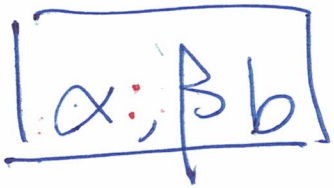
Worst-case of the tree: $O(2^n)$.

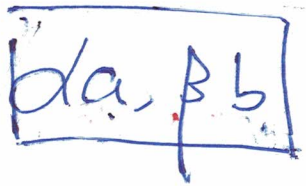
Full binary tree example.

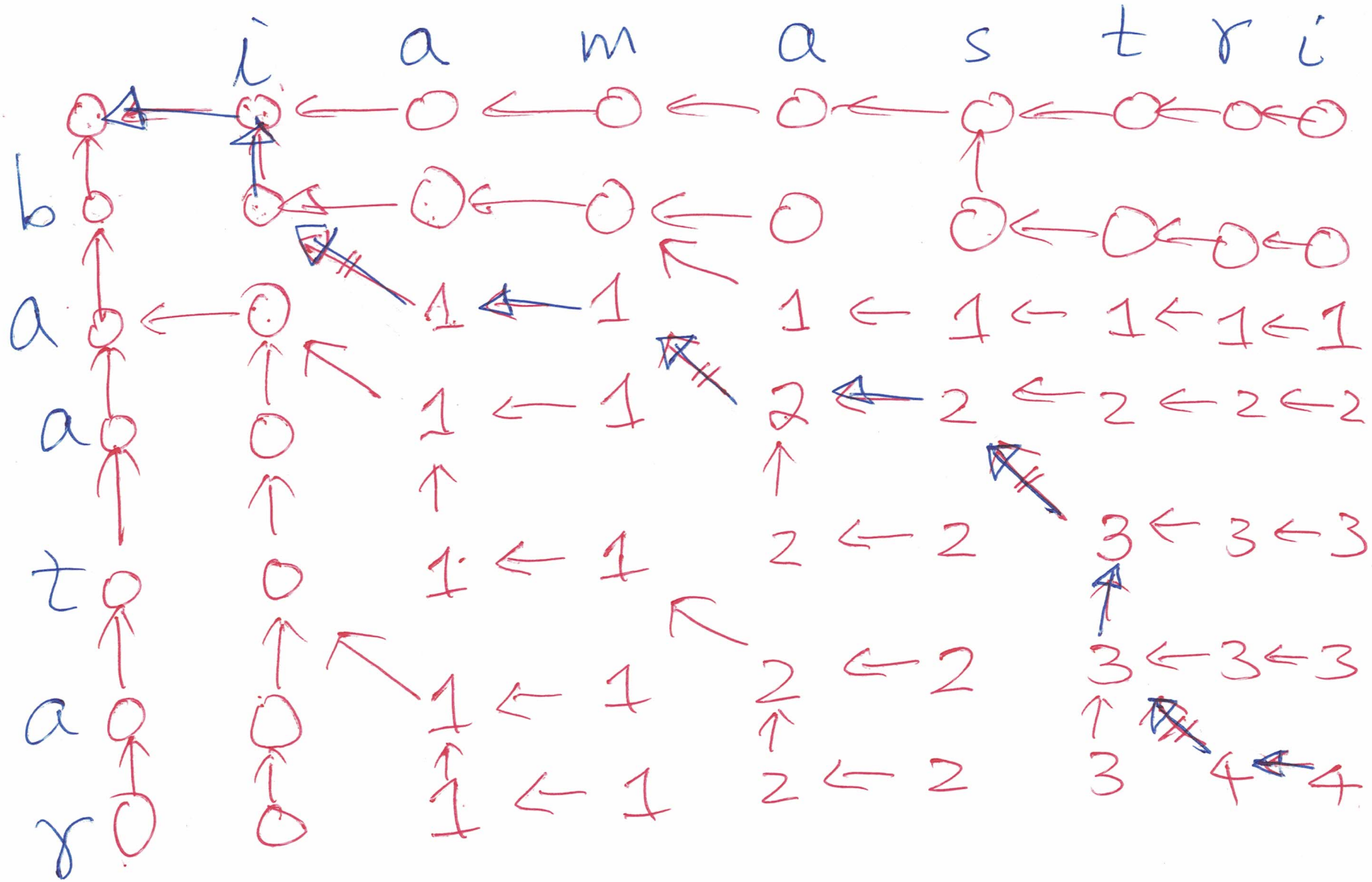
Dynamic programing.









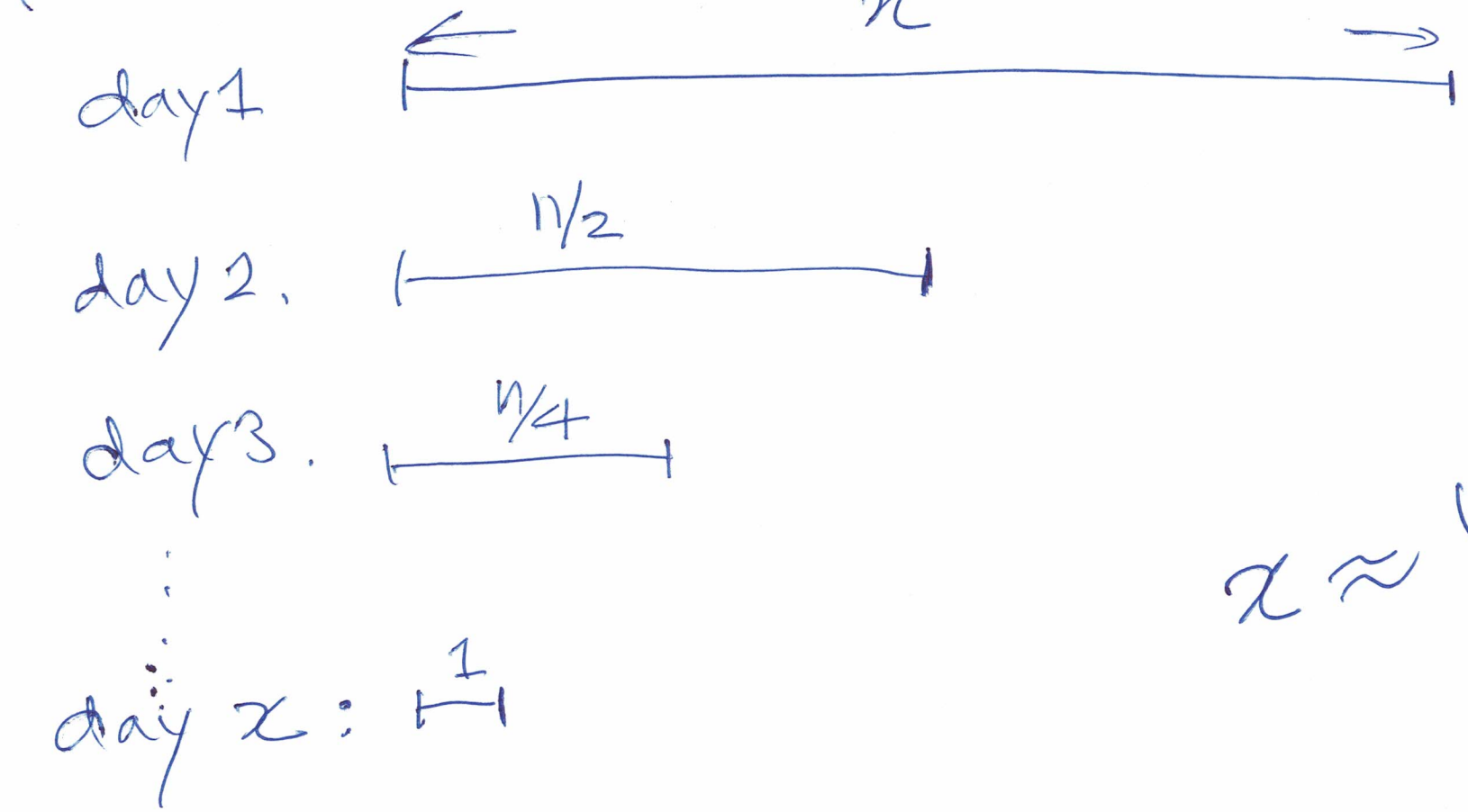


aatr is a LCS.

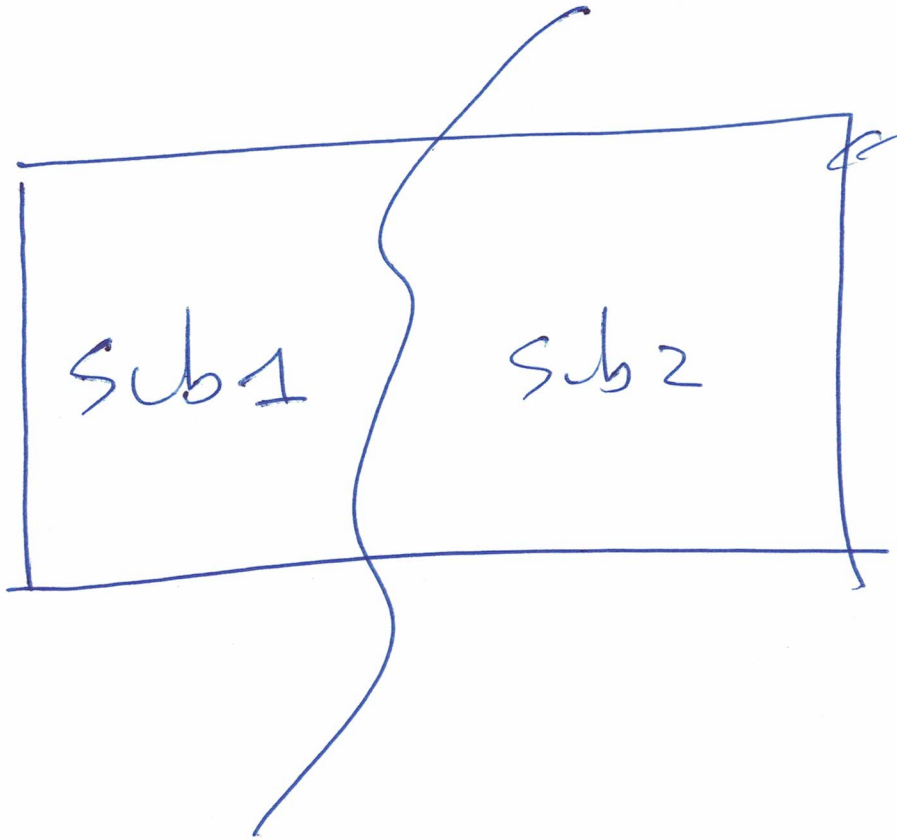
Greedy.

Dynamic Programing.

Divide & Conquer. ⟶ $\log n$.



$x \approx \log_2 n$.



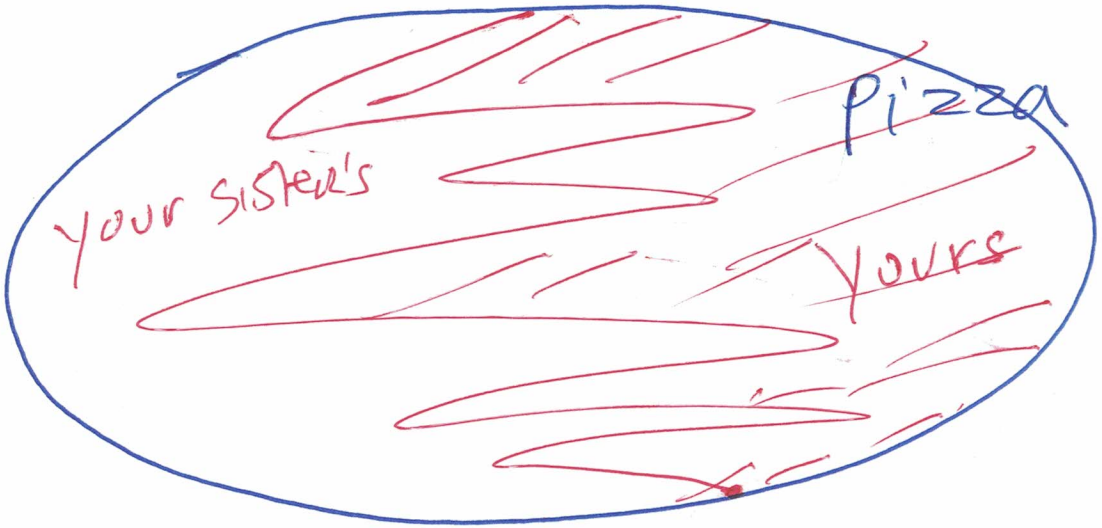
① Cut the whole problem into two subprobs.

Solve

② ~~Resolve~~ the subproblems recursively

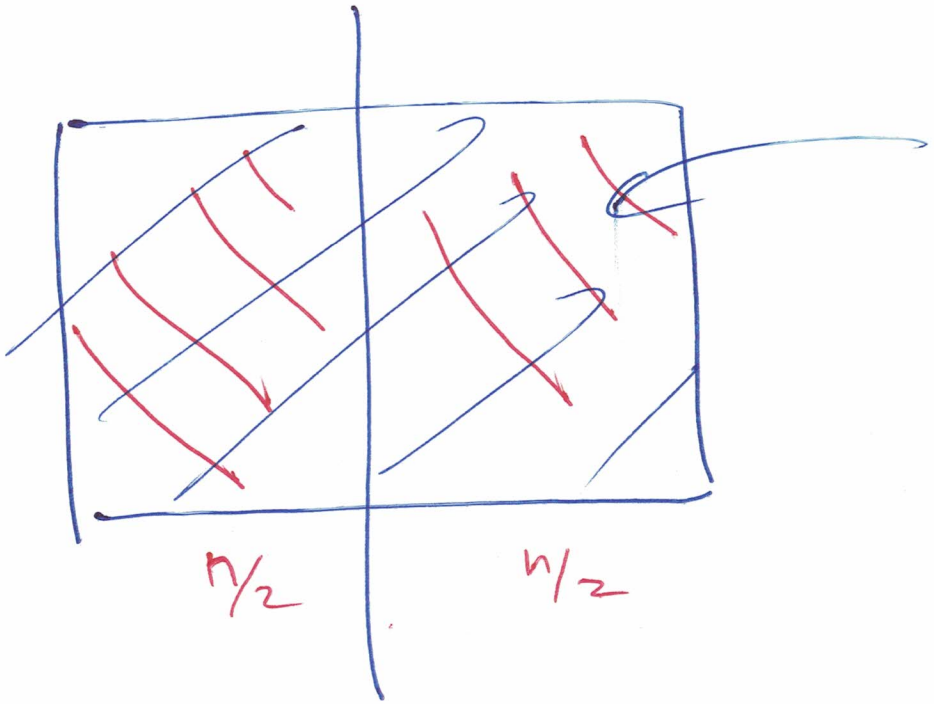
③ Assemble the solutions into a solution of the whole problem.

(1): subproblem is easier than whole problem. X



(2). Always cut the whole problem into 2 pieces (summon > 2) most diretly.

(3). Divide & Conquer Should Work!



(*) $T(n) = T(n/2) + T(n/2) +$ overhead of assembly.

if $T(n)$ is linear time.

$T(n) \sim O(n)$.

then: (*) will not give you a better time!

if $T(n)$ is NOT linear, e.g.

$T(n) \sim O(n^3)$ ← what you thought originally

then: From (*), you can get a much smaller $T(n)$ dep. on the overhead.