01 _ Given the head of a linked list, remove the nth node from the end of the list.

① FAST, SLOW on ROOT

② FAST ahead of N+1

③ BOTH ahead UNTIL FAST HITS THE END

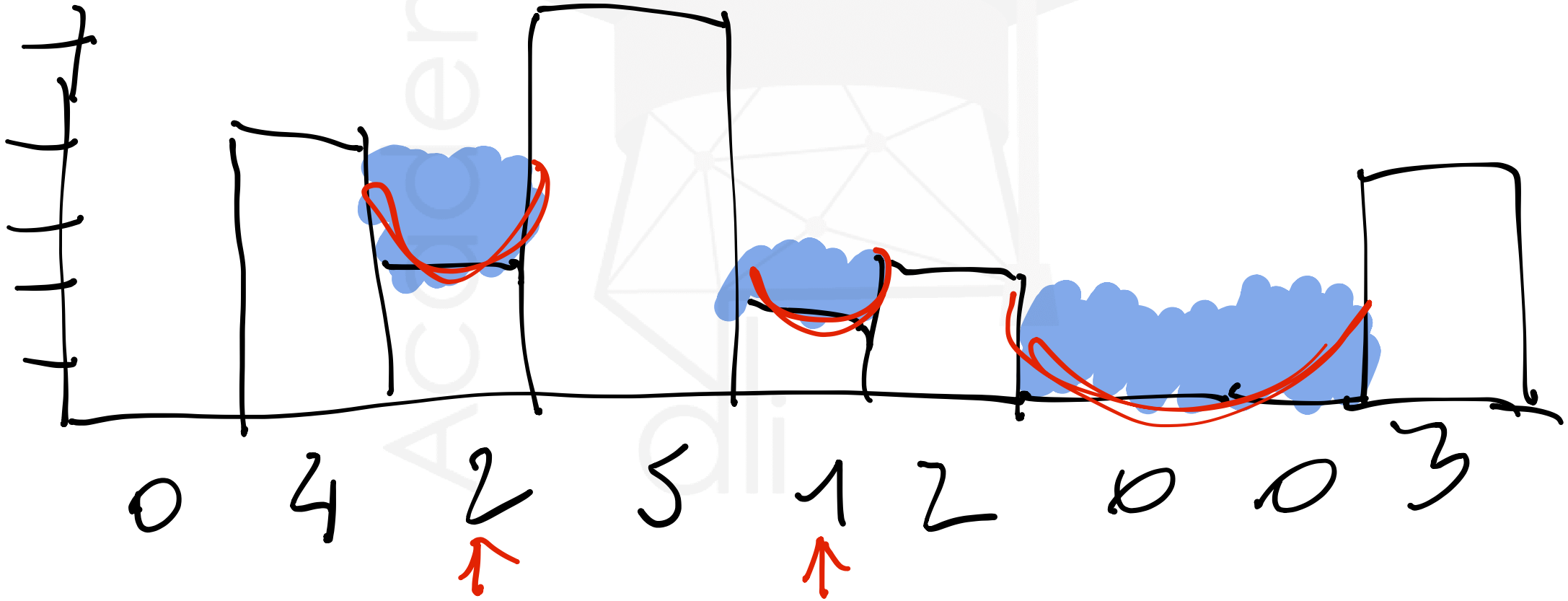
④ SLOW.NEXT =
SLOW.NEXT.NEXT

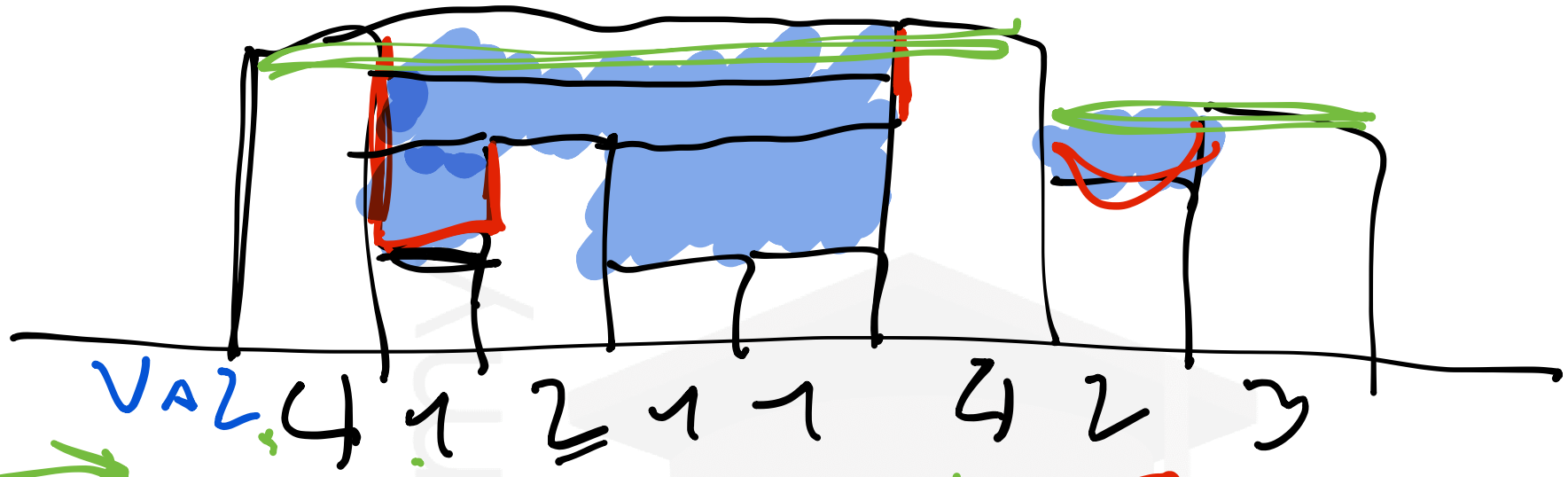
t O(n) // LIST LENGTH

s O(1)

Academy
dli

02 _ Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.



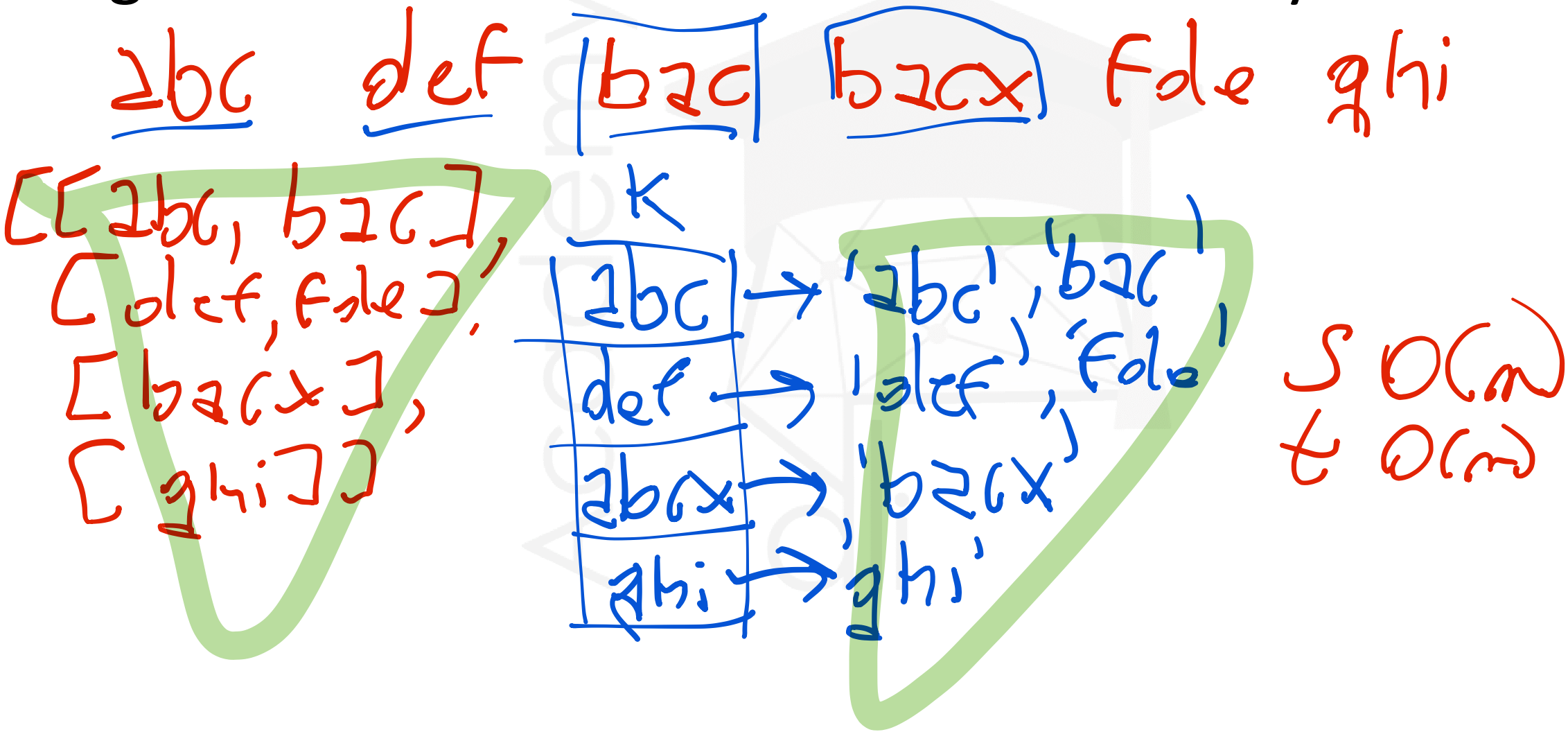


VAL	4	1	2	1	1	4	2	3	
LEFT →	4	4	4	4	4	4	4	4	O(n)
O(n)	4	4	4	4	4	4	3	3	← RIGHT
MAXH	4	4	4	4	4	4	3	3	O(n)
MAXH - VAL	0	3	2	3	3	0	1	0	

T O(n)
S O(1)

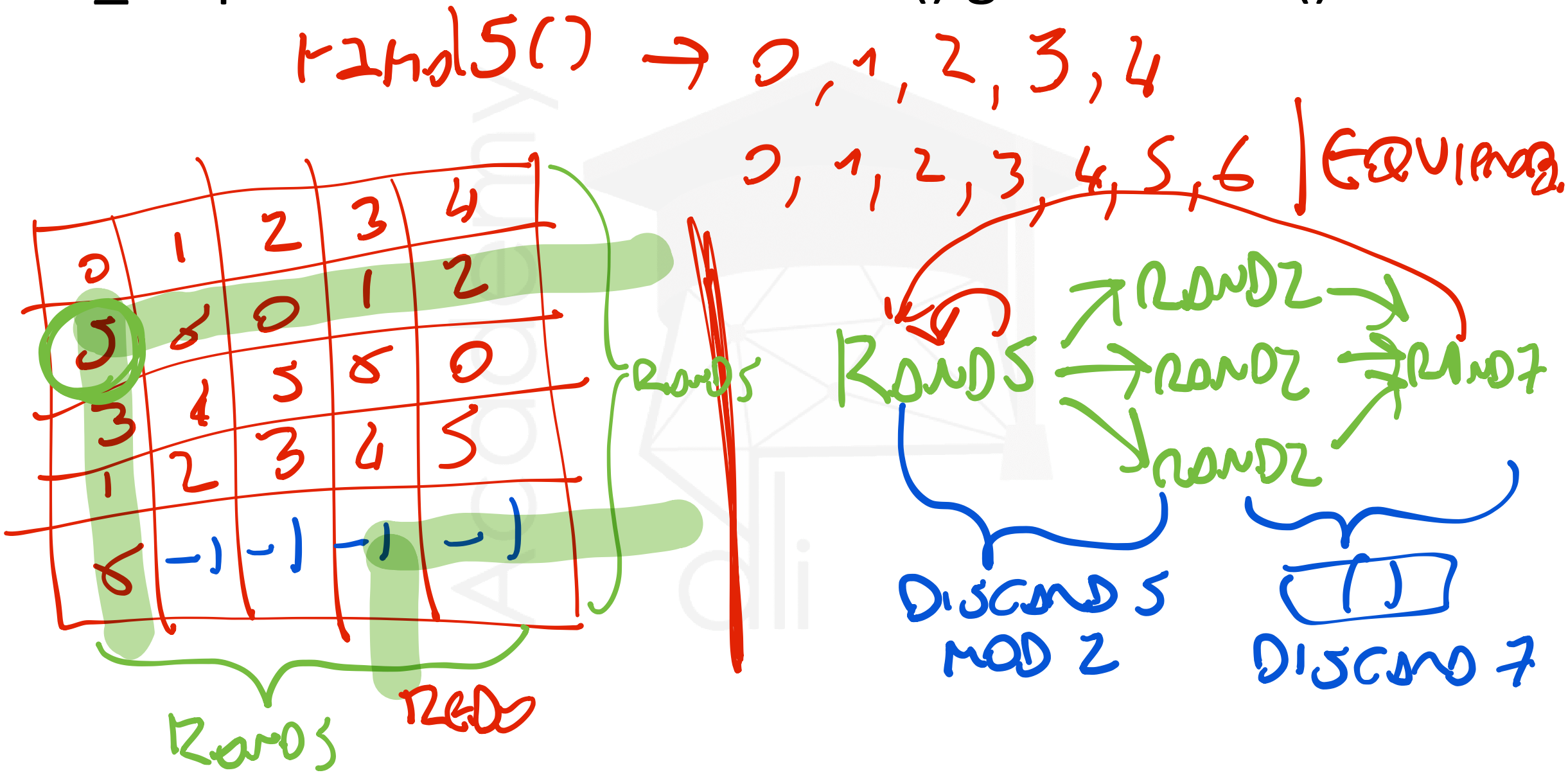
Academy
dli

03 _ Given an array of strings, group the anagrams together. You can return the answer in any order.



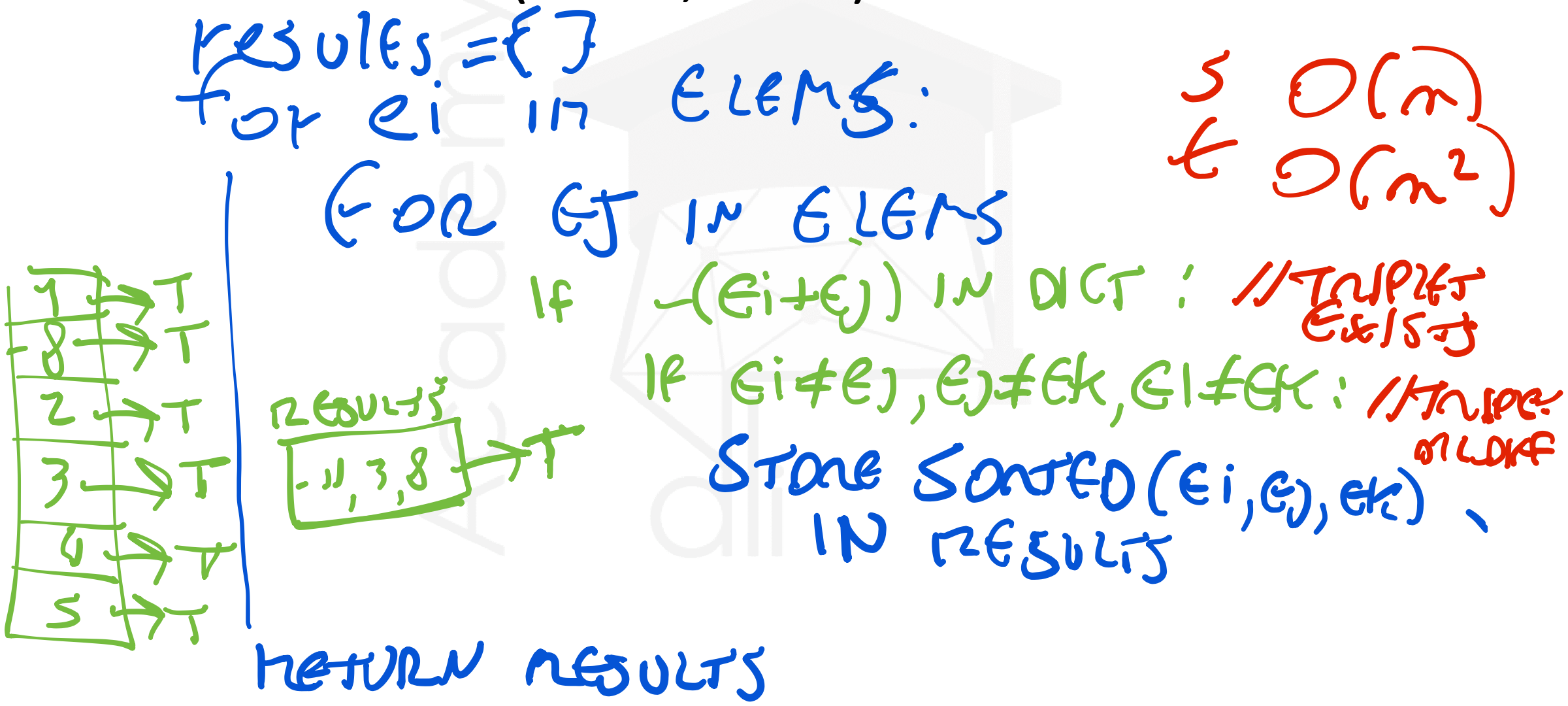
Academy
dli

04 _ Implement a method rand7() given rand5()



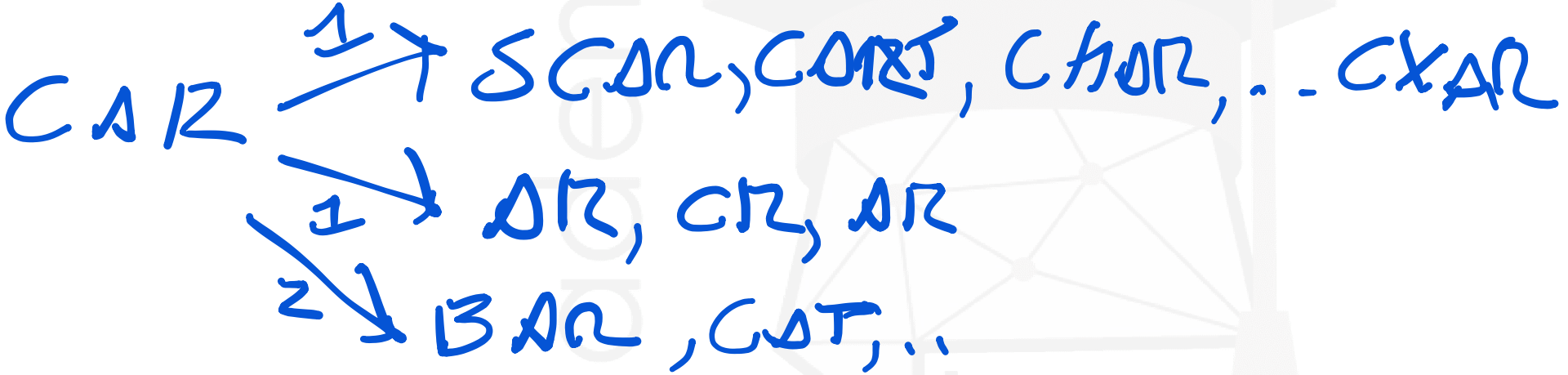
Academy
dli

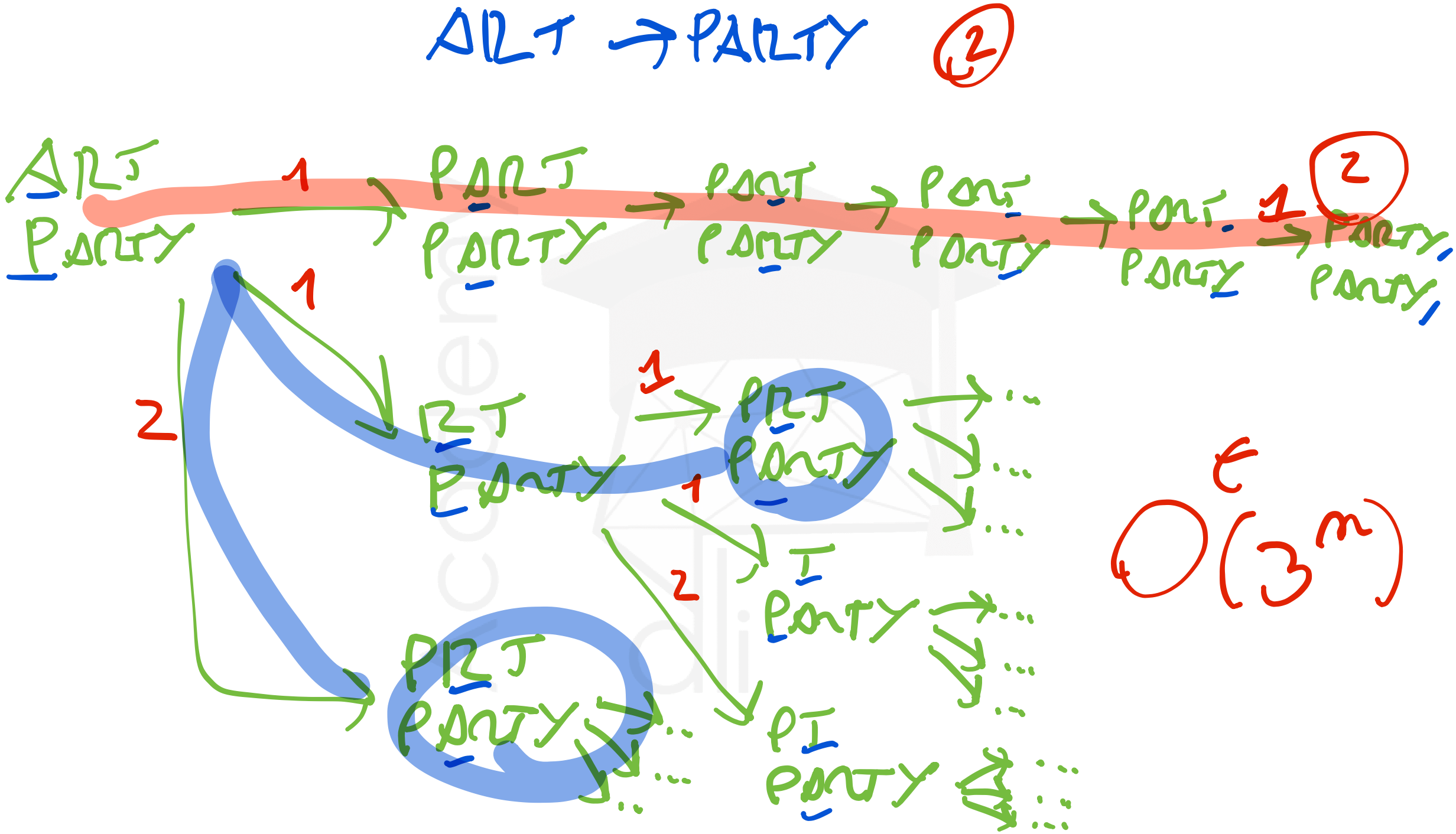
05 _ Given an integer array, return all the triplets of different elements (index,value) that sum to 0

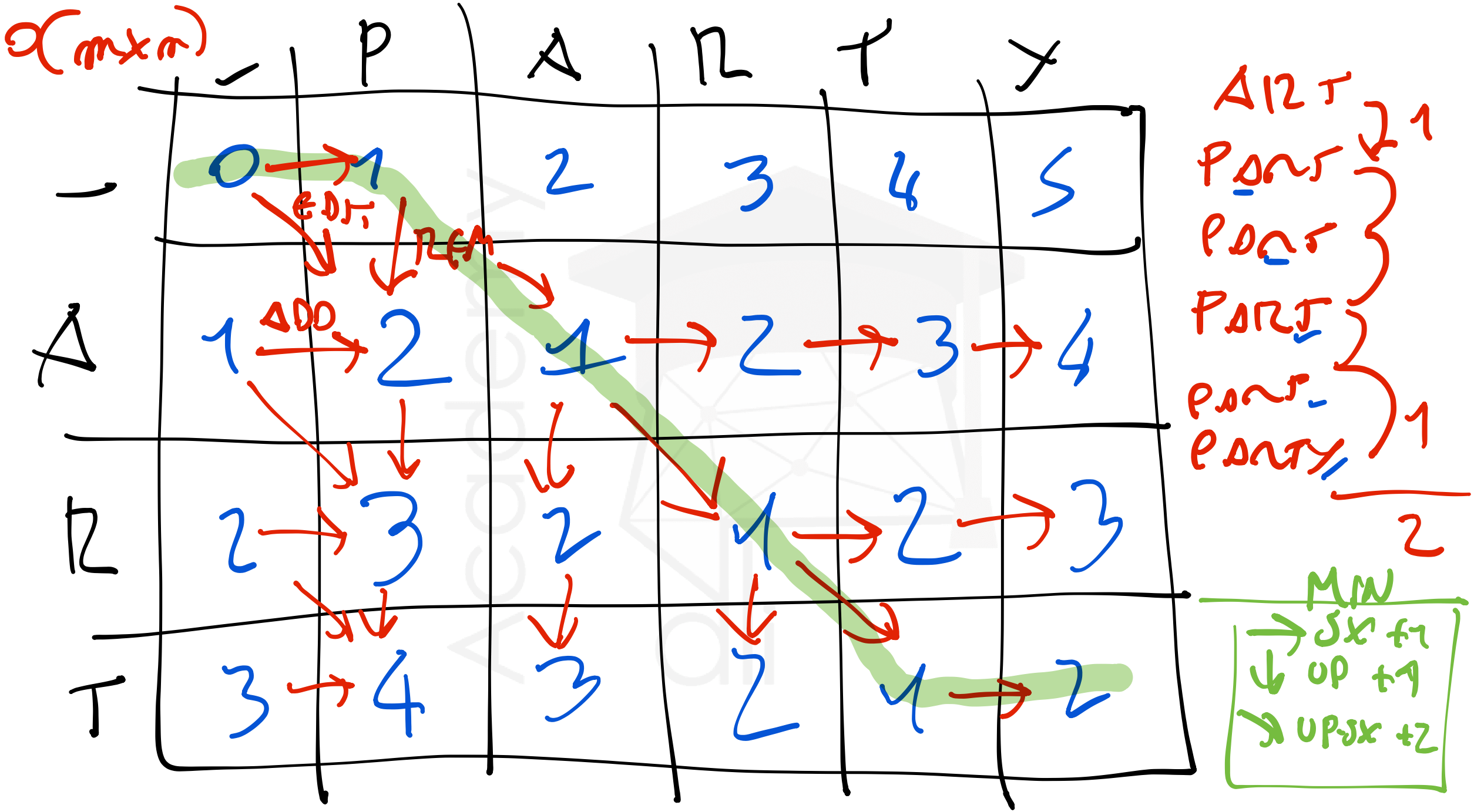


Academy
dli

06 _ Given a source string and a target string, find the minimum edit distance between the two strings. Assume you can either insert, remove, or edit a character with costs of 1,1 and 2.

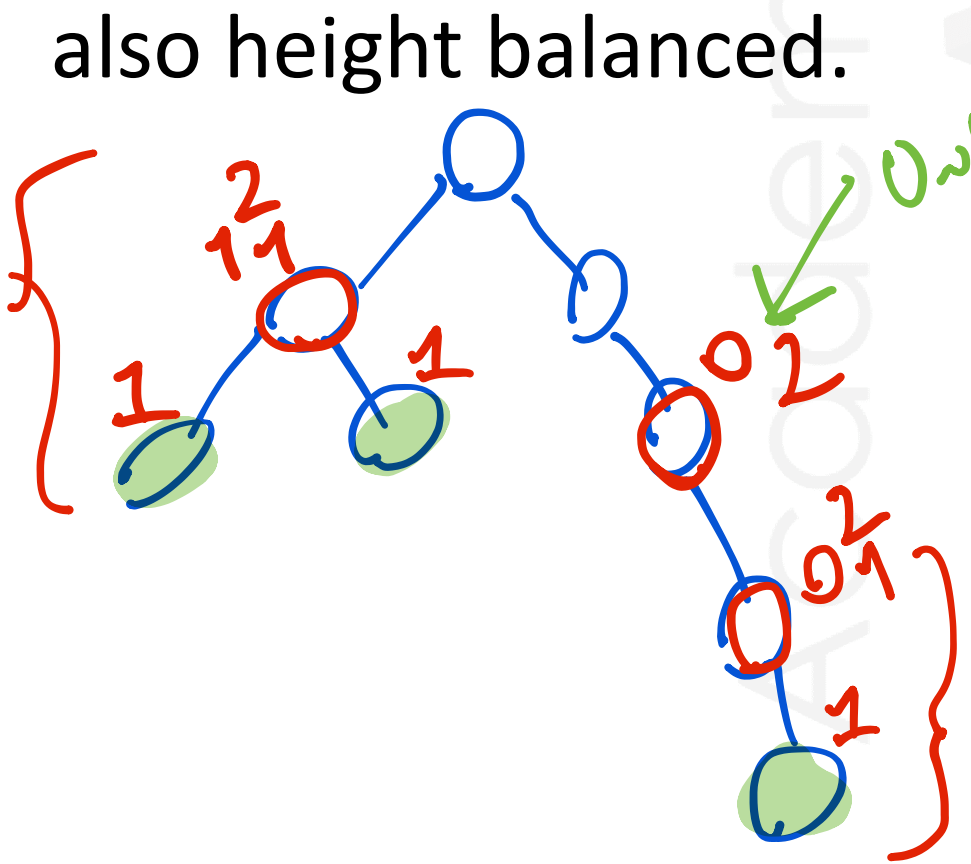



ART → PARTY
2
ART
PARTY
1
PART
PARTY
PART
PARTY
PART
PARTY
PART
PARTY
1
2
PARTY
PARTY
1
2
RT
PARTY
PRT
PARTY
1
1
I
PARTY
PRT
PARTY
PI
PARTY
O(3^n)


O(m x n)
_
P
A
R
T
Y
0
1
2
3
4
5
EDIT
REM
ADD
1
2
1
2
3
4
2
3
2
1
2
3
3
4
3
2
1
2
ALT
PART
PART
PART
PART
1
1
2
MIN
→ SX +1
↓ UP +1
↘ UP-SX +2

Academy
dli

07 _ Check if a binary tree is height-balanced; that is, if the height of the left subtree and right subtree of any node does not differ by more than 1 and both the left and right subtrees are also height balanced.

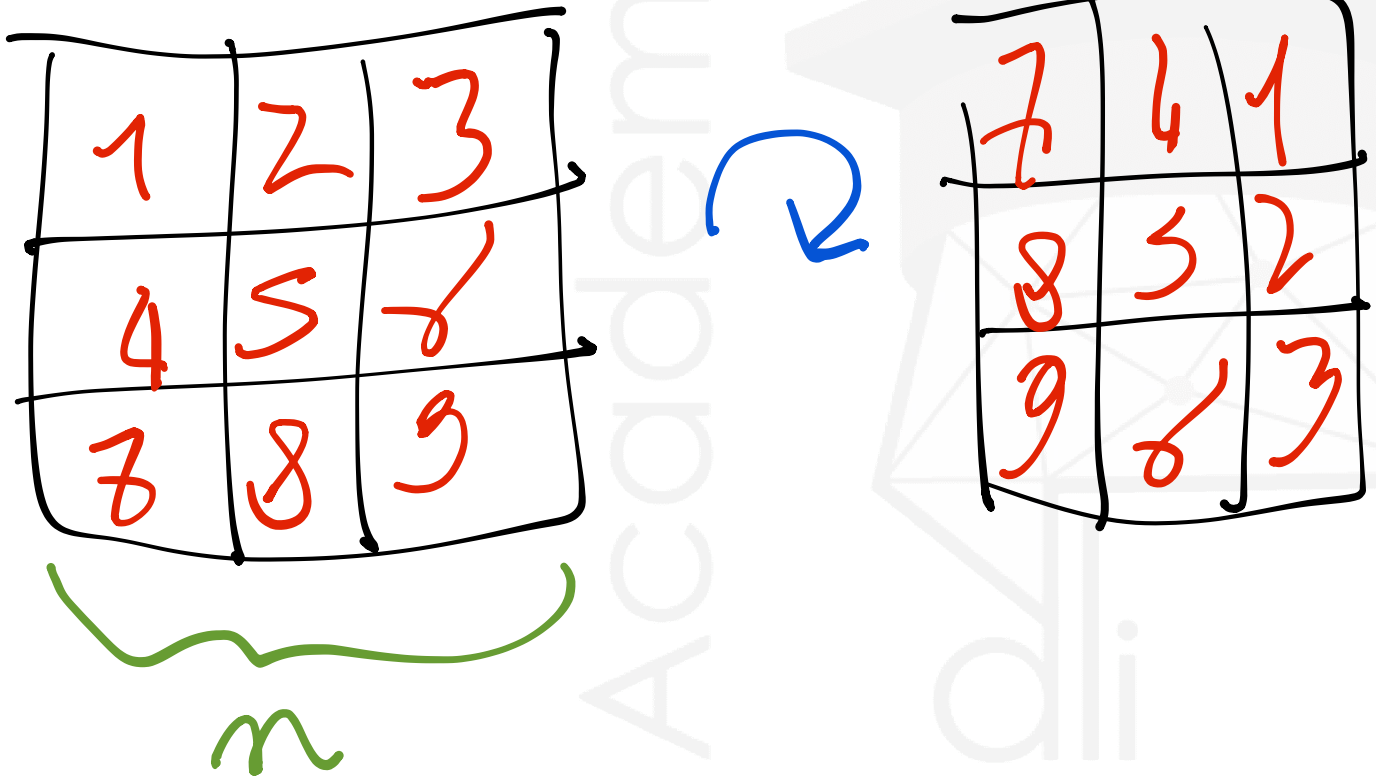



```
IF LEAF:
    RETURN 1
ELSE
    LB, LD ← RECURSION LEFT
    RB, RD ← RECURSION RIGHT
    D ← MAX(LD, RD)+1
    IF LD-RD IN [-1, 0, 1]: // OK
        RETURN TRUE, D
    ELSE
        RETURN FALSE, D
```

O(n) t
O(1) s

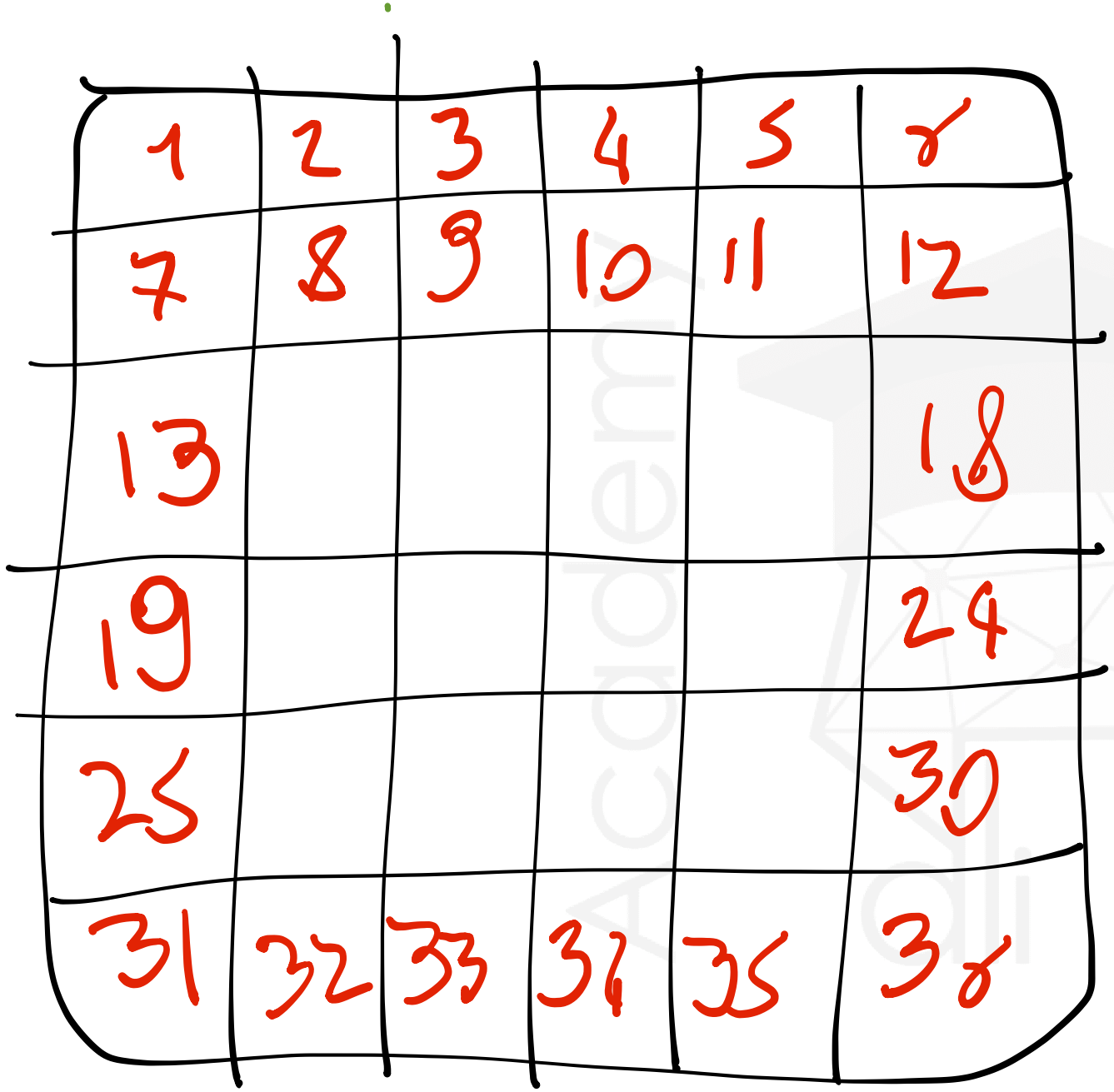
Academy
dli

08 _ Given a matrix representing an image, rotate the image by 90 degrees (clockwise).



$O(n^2)$ t

$O(1)$ s

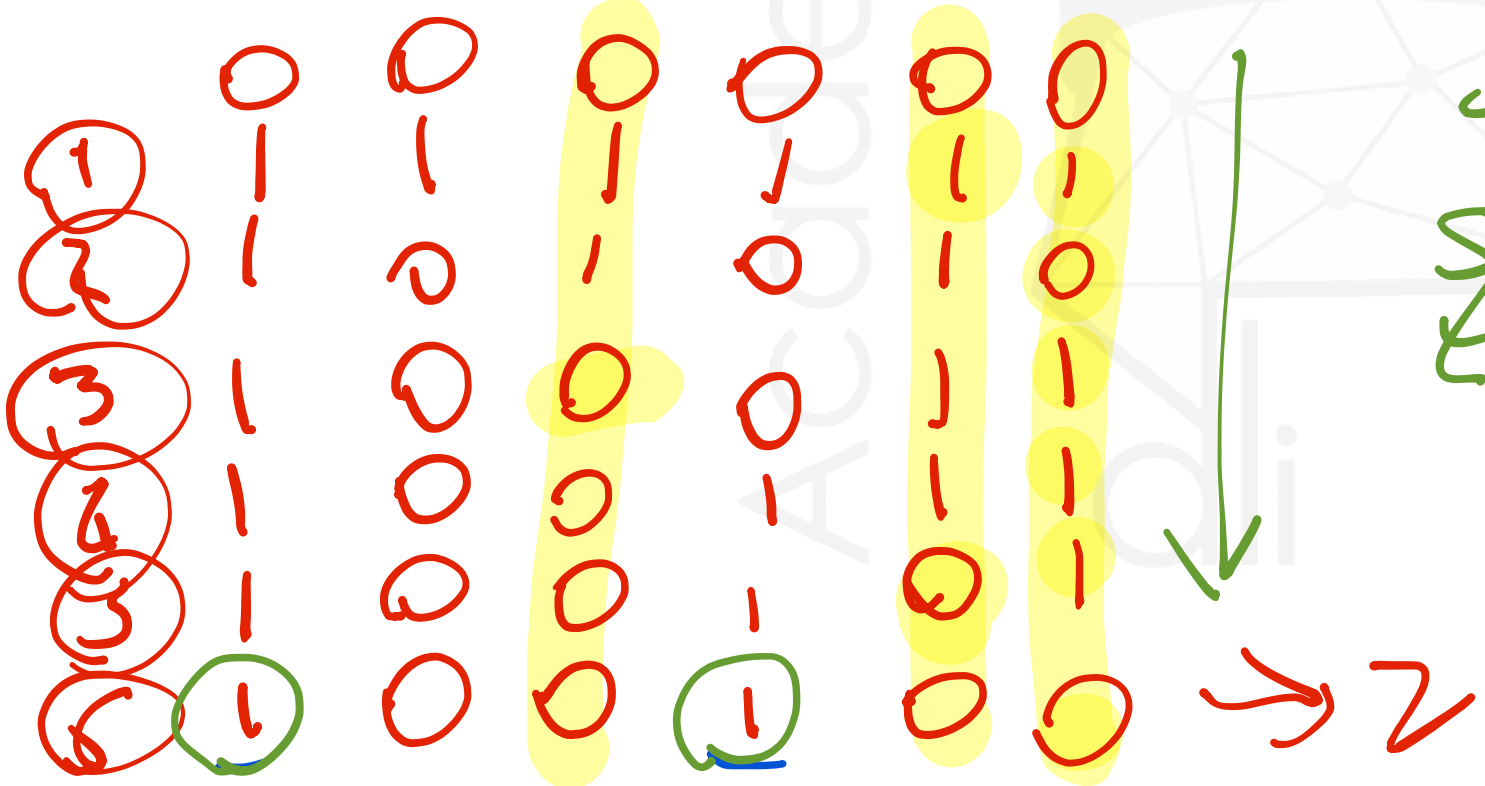



```
for i in range(n//2):
    for j in range(i, n-1-i):
        temp = M[i,j]
        M[i,j] = M[n-1-j, i]
        M[n-1-j, i] = M[n-1-i, n-1-j]
        M[n-1-i, n-1-j] = M[j, n-1-i]
        M[j, n-1-i] = temp
```

n-1=5	1,2	3,1	4,3	2,4
i=1				
j=2	9	20	28	17

Academy
dli

09 _ There are n bulbs that are initially off. You first turn on all the bulbs, then you turn off every second bulb. On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the ith round, you toggle every i bulb. For the nth round, you only toggle the last bulb. Return the number of bulbs that are on after n rounds.



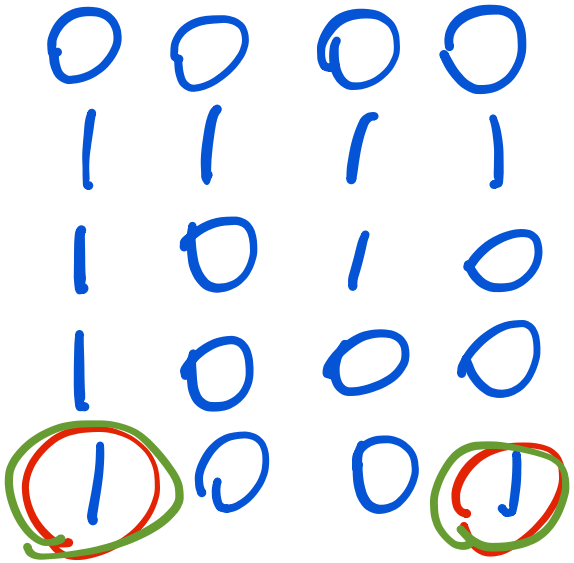
SIMULATION

$\leq O(n)$

$\leq O(n^2)$

SQUARES!

$O(1)$ $O(1)$



4 → 2

BULB 6 → [1, 2, 3, 6]
BULB 5 → [1, 5]

NUM_OF_SWITCHES → NUM DIVS!

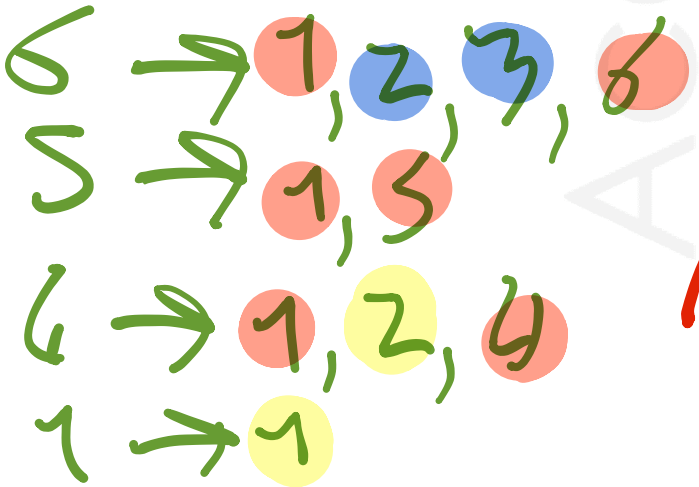
ON_AT_END = ODD TOGGLES!

ON_AT_END = ODD NUM DIVS!

SQUARES!

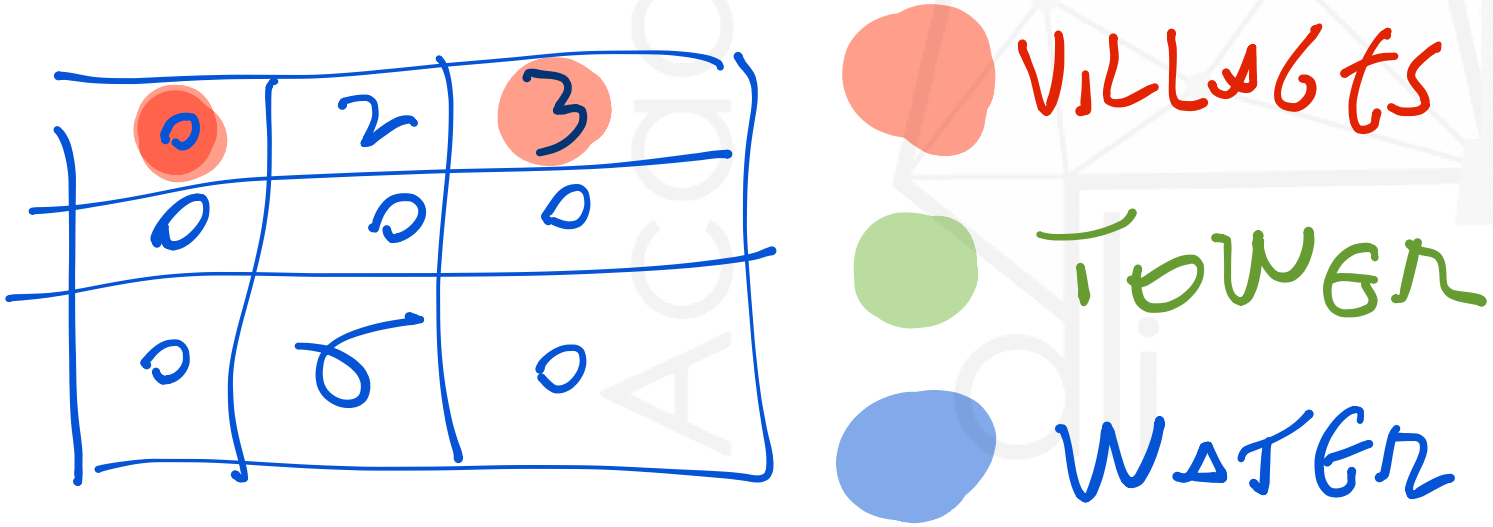
NUM_ON_AT_END = SQUARES ≤ N

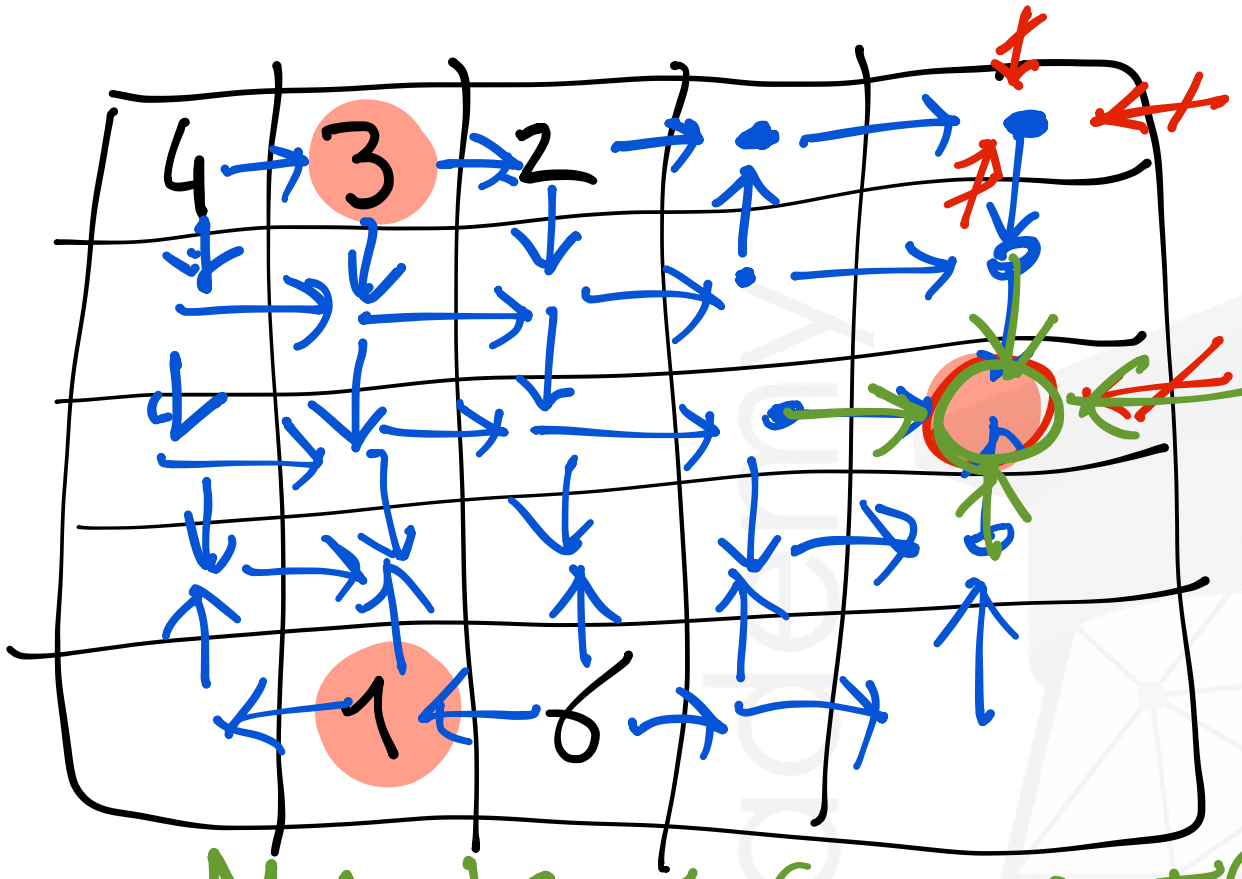
⌊SQRT(N)⌋



Academy
dli

10 _ Given a m*n matrix of positive ints, the value of each point in the matrix is the height of that point. Assume that there are K given villages in the matrix that need water. Assume a water tower can be built anywhere, but the water can only flow from high to low or equal to equal.
Find a point to build a single water supply tower so that the maximum number of villages can be reached.





$O(k \cdot m \cdot n)$

$O(m \cdot n)$

t

COMPUTE_BASIN
~FILL_POINTS

MARK AS CANDIDATE
FOR ADJ IN UP, DOWN, LEFT, RIGHT:
IF IN_MATRIX AND UNEXPLORED:
RECURSIVE_CALL(ADJ)