Best Cow Line

~~A~~ ~~C~~ D ~~B~~ ~~C~~ ~~B~~

A B C B C D

DP[i][j]

Milk Routing

$O(M^2 \log N)$

total number of capacities there can be is M

$O(N^2 \log N)$

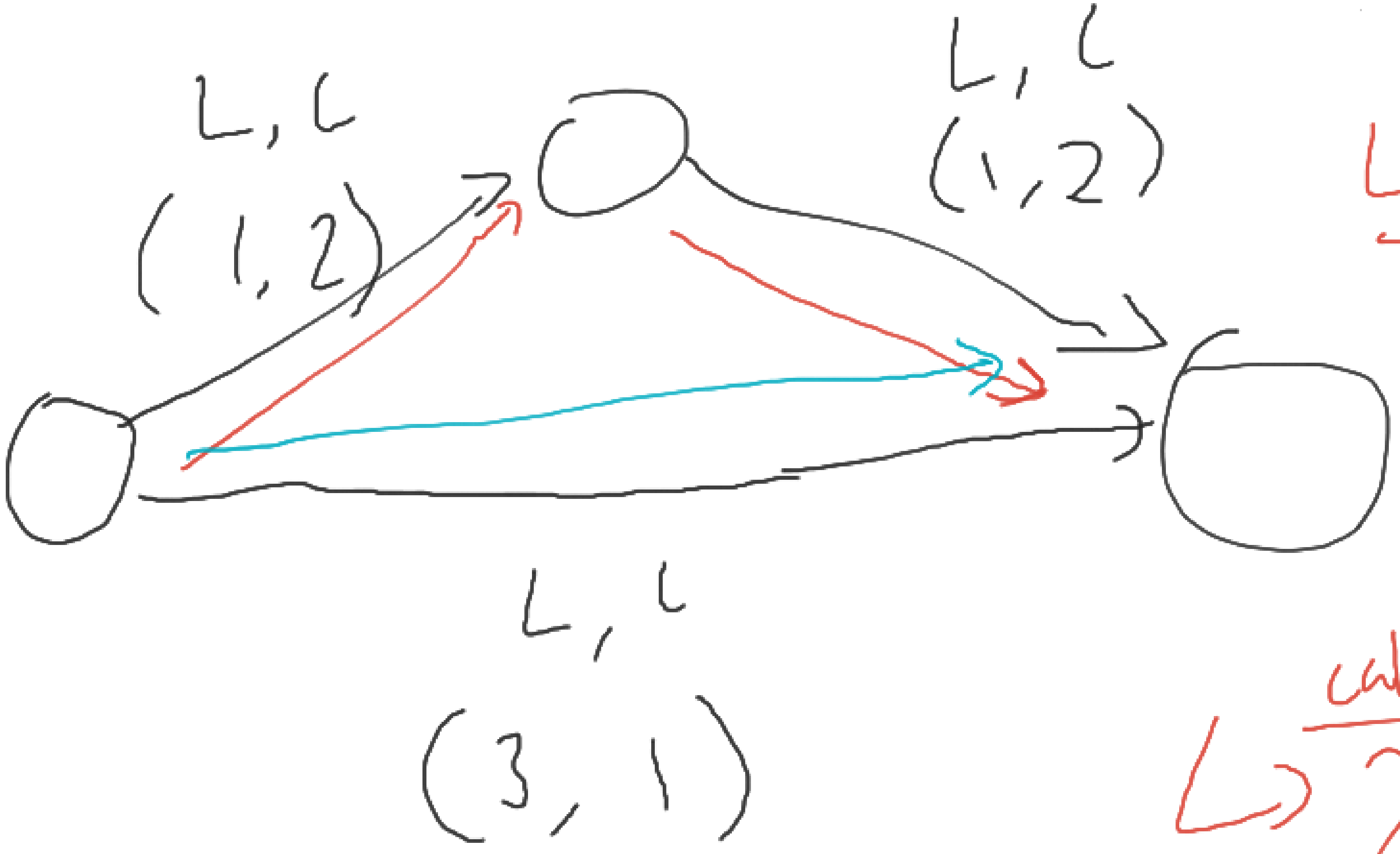
↳ Do dijkstras on the latency using only edges with capacity >= C (iterate through all C)

↳ returns min latency given a capacity

↳ calculate the cost

↳ return the minimum cost that we observe

$x = 8$



$L = 1 + 1$

$C = 1$

calculated

↳ $2 + \frac{8}{1} = 10$

actual

$2 + \frac{8}{2} = \boxed{6}$

$C = 1$

Cheese Towers

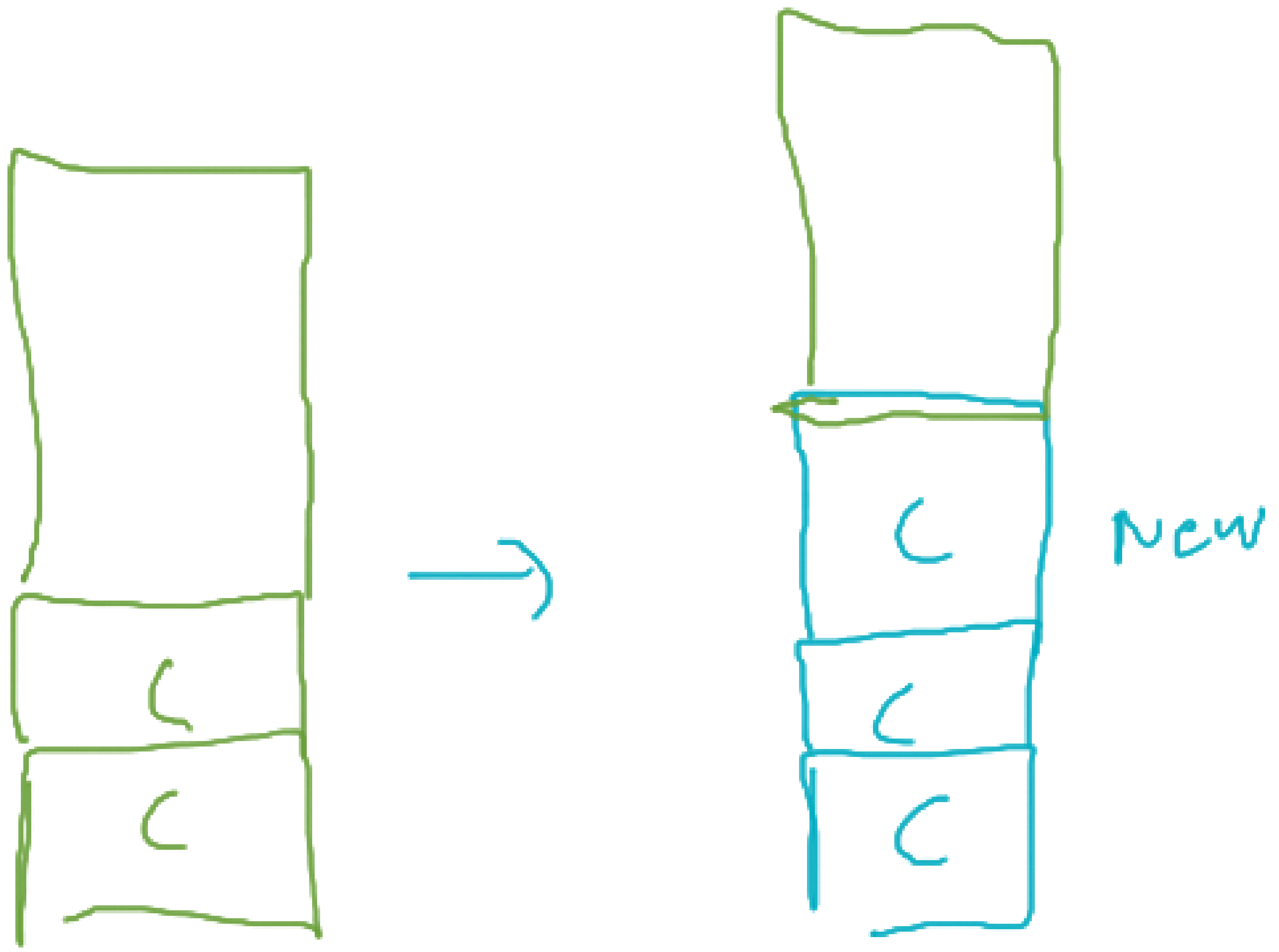
$DP[i][j] = i$ height_$j \in \{0, 1\}$ whether or not we've put big cheesement down

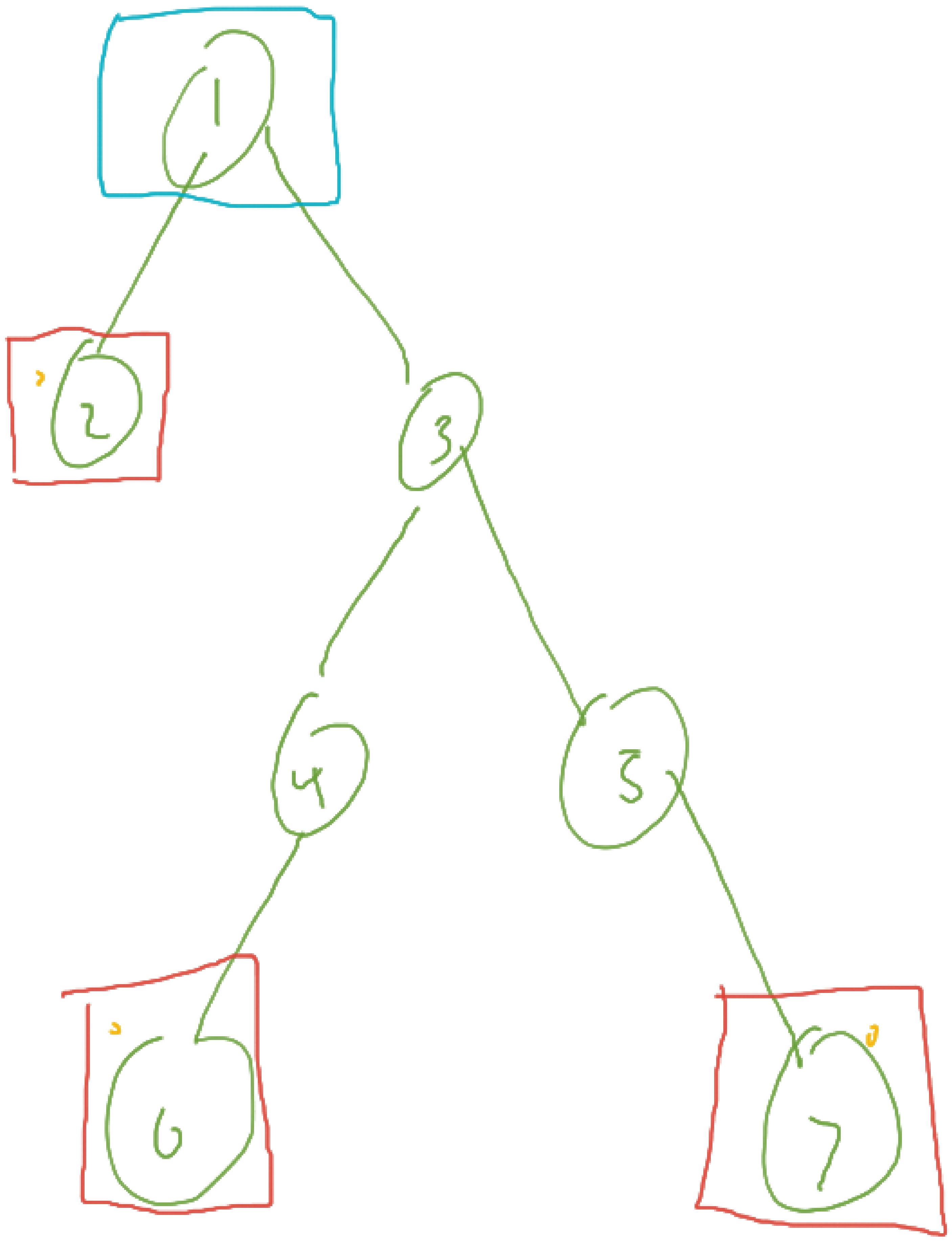
if $H[k] < K$

$$DP[i + H[k]][0] = \max\left(\begin{array}{c} DP[i + H[k]][0] \\ \text{OR} \\ DP[i][0] + V[k] \end{array}\right)$$

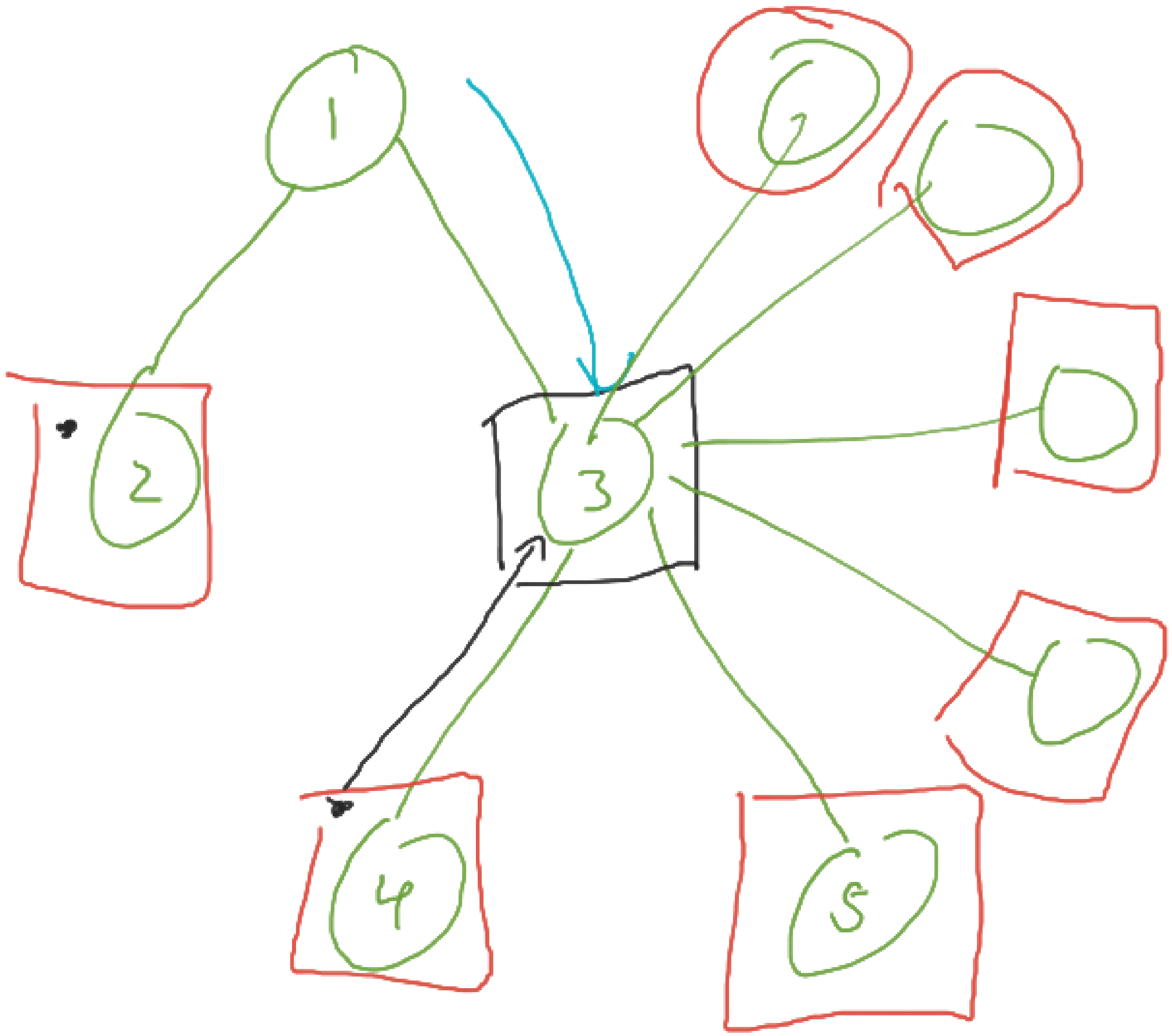
$$DP[i + H[k]][1] = \max\left(\begin{array}{c} \text{itself} \\ \text{OR} \\ DP[i][1] + V[k] \end{array}\right)$$

$DP[i + 4/5 \cdot H(t)][1]$

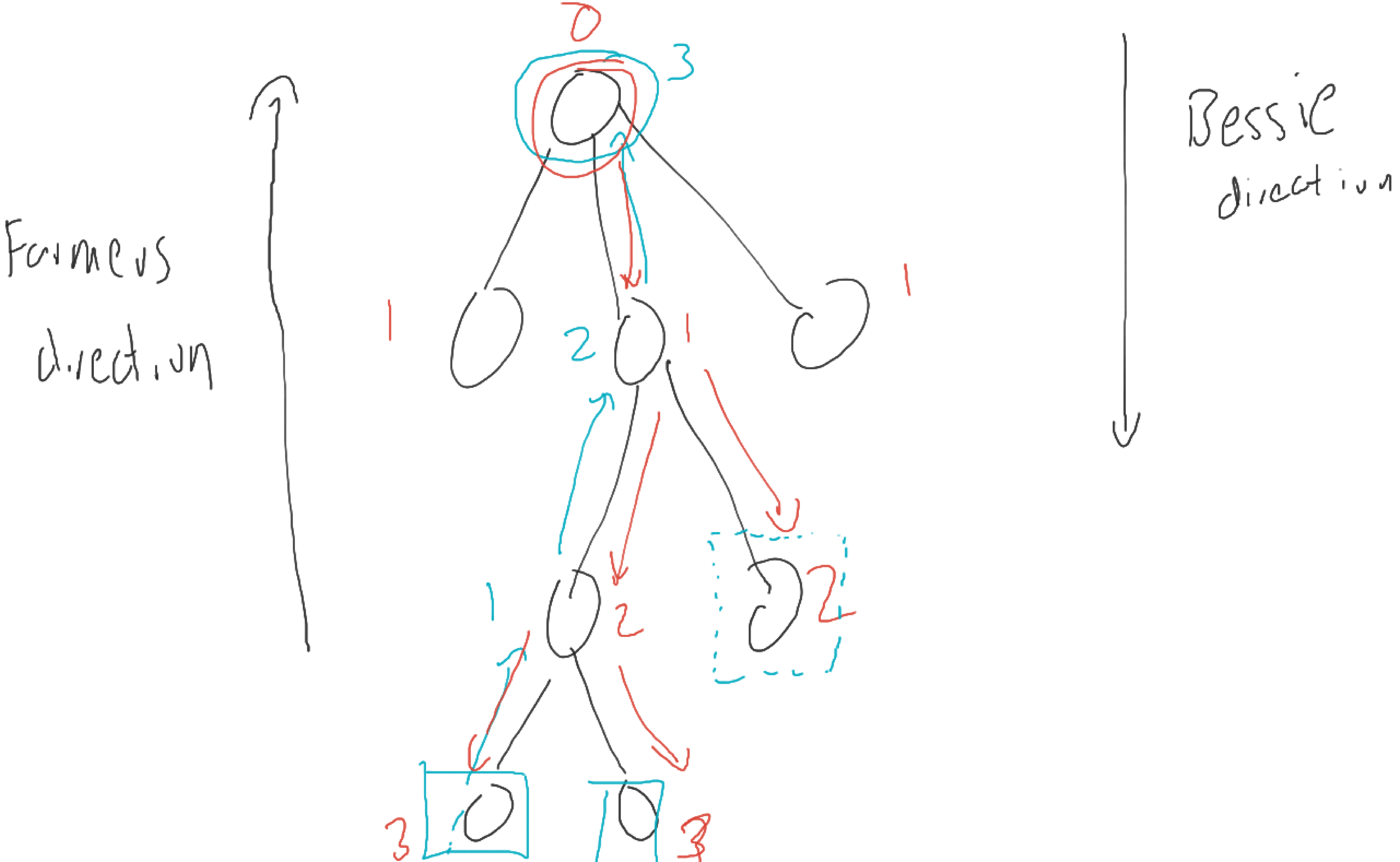


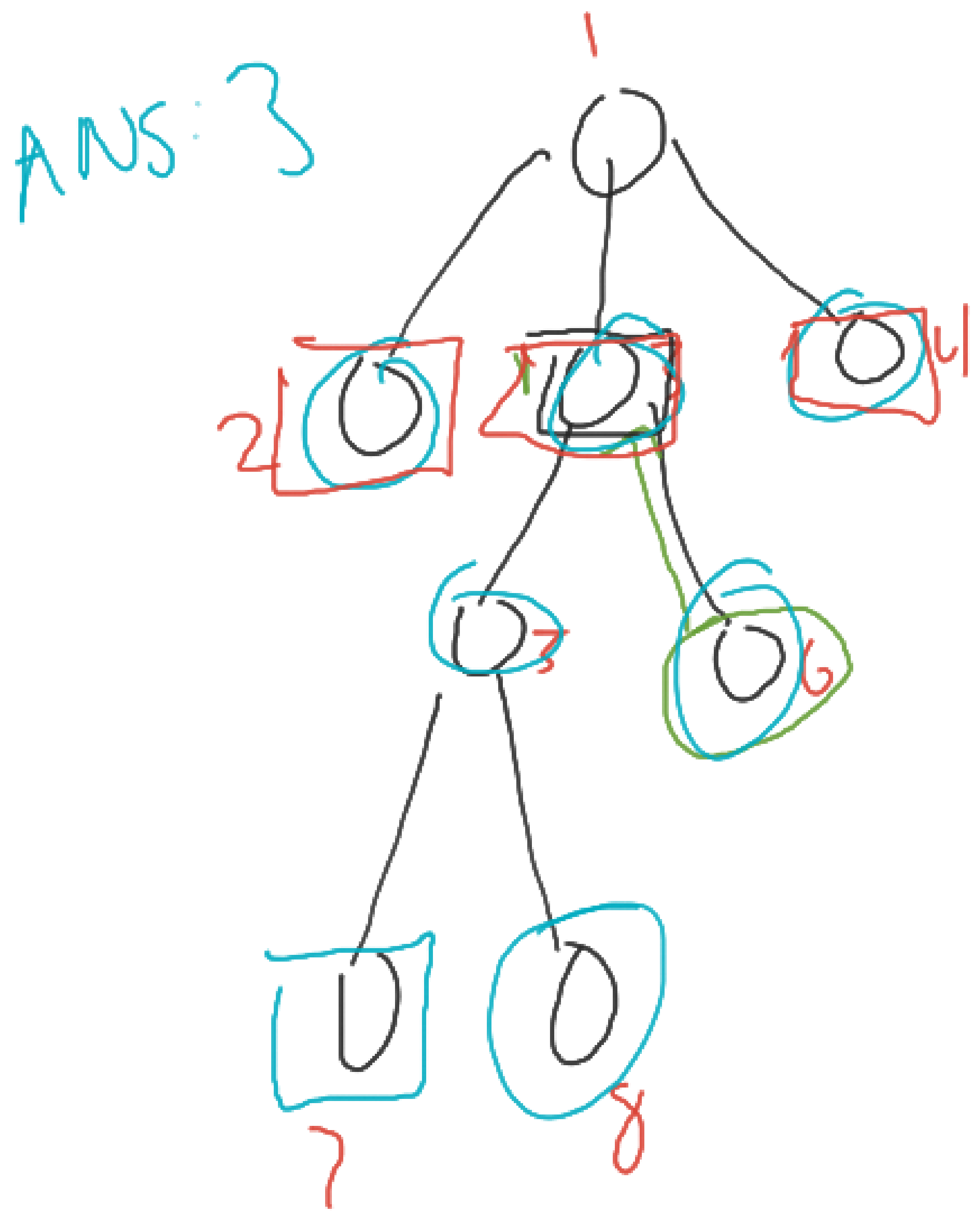


1
2
3
4
5
6
7



1
2
3
4
5


Farmers direction
Bessie direction



BLOCKED IF given node and farmer

$$time_{farmer} \le time_{Bessie}$$

BLOCKED IF (given just node)

$$\min(\underbrace{time_{farmer}}_{\text{Multisource BFS}}) \le \underbrace{time_{Bessie}}_{\text{DFS}}$$

should a node be blocked

↳ block nodes whose parent cannot be blocked

Work on problem 5 and 6 until 3:35

Cow Frisbee Team

$1 \le R_i \le 100000$

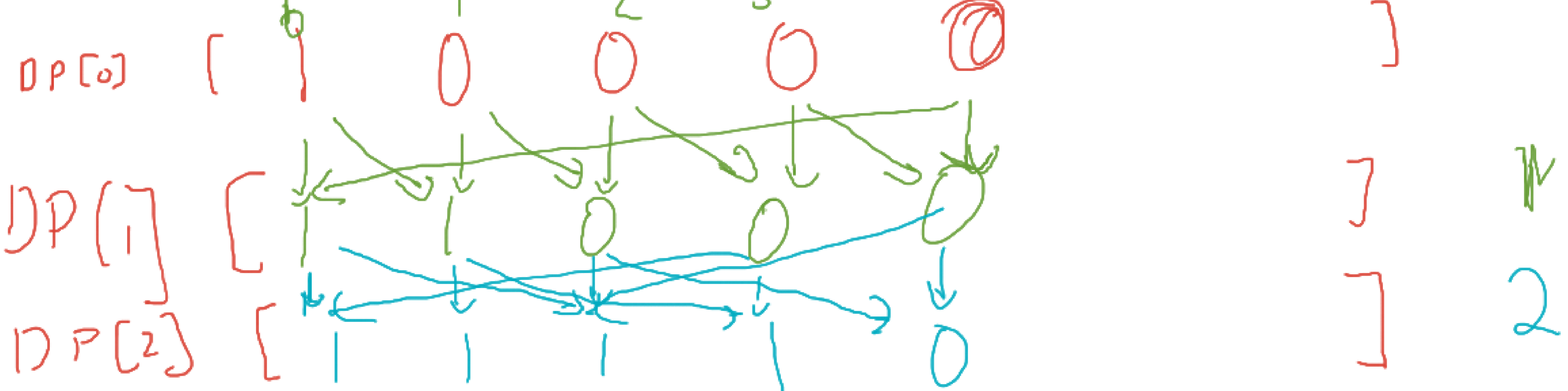
Sample Input:
4 5
1
2
8
2

$R_i' = R_i \% F$

Teams
{}
{1}
{2}
{1, 2}

DP[i][j] = number of ways to create a team with rating%f == j using the first i cows

BASE CASE: DP[0][0] = 1



$R[i]$ - curr element

$$DP[i][j] = DP[i-1][j] + DP[i-1][\underline{\quad\quad}]$$

$$(j - R[i] + F) \% F$$

Return Value

$$DP[n][0] - 1$$

River Crossing

DP[i] = min cost to ship i crws

↳ and return to start

Transitions

DP[k] + (cost(k+1..i) + m)

subtract m at the end (from DP[N])

O(N²)

$-5 + 7 = 2$

B E

0 0 0 1 0 | 1 0 0 0 1

1 -2 3

0 0 0 1 1 | 0 0 0 0 1

0 0

0 1 0 1 0 3 inversions

0 1 0 0 1

Observation 1:

only swap 0 and 1, not 0 and 0 or 1 and 1

every swap will only change the number of inversions by one if its on the same half

0 0 0 1 0 | 1 0 0 0 1

A B

abs(A − B)

Swap in the middle

0 1 0 0 1 | 0 0 0 0 1

A − (# of 1s on this side)

B − (# of 0s on this side) O_B

→ N − (# of 0s on this side) O_A

$\delta = A - B$

$$\Delta\delta = A - N + O_A - B + O_B$$
$$= (A - B) - N + (O_A + O_B = O_T)$$

$0 \quad 1$

$$\boxed{\Delta\delta = -N + O_T}$$

$1 \quad 0$

$$\Delta\delta = N - O_T$$

we should only need to do one type of these

$$\begin{array}{ccc|ccc} 0 & & & 2 & & \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}$$

$N = 3 \qquad O_T = 4$

Pick one of the types of swaps. Search over how many of those swaps (0, 1, .. N) or however many are possible. Using the formulas + our two pointer for finding where to bring the 0/1 from, in linear time, we can compute the difference between A and B given that we do S swaps. Take the minimum value of |A-B| over all of these.

Runtime

O(N)

Memory

O(N)