reduce the space

Space & Transition Optimization

transition time => ↓ T.C

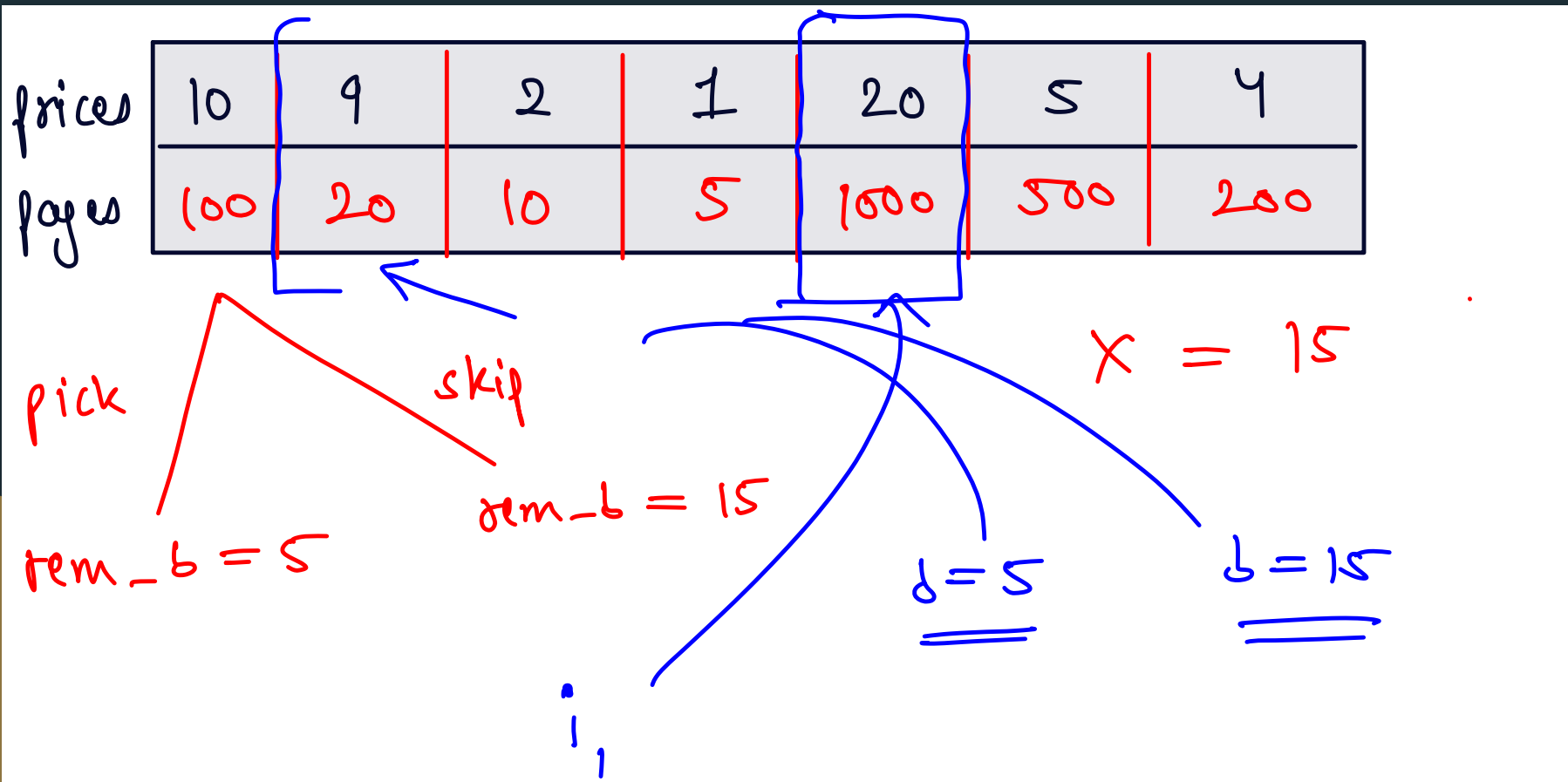
- Priyansh Agarwal

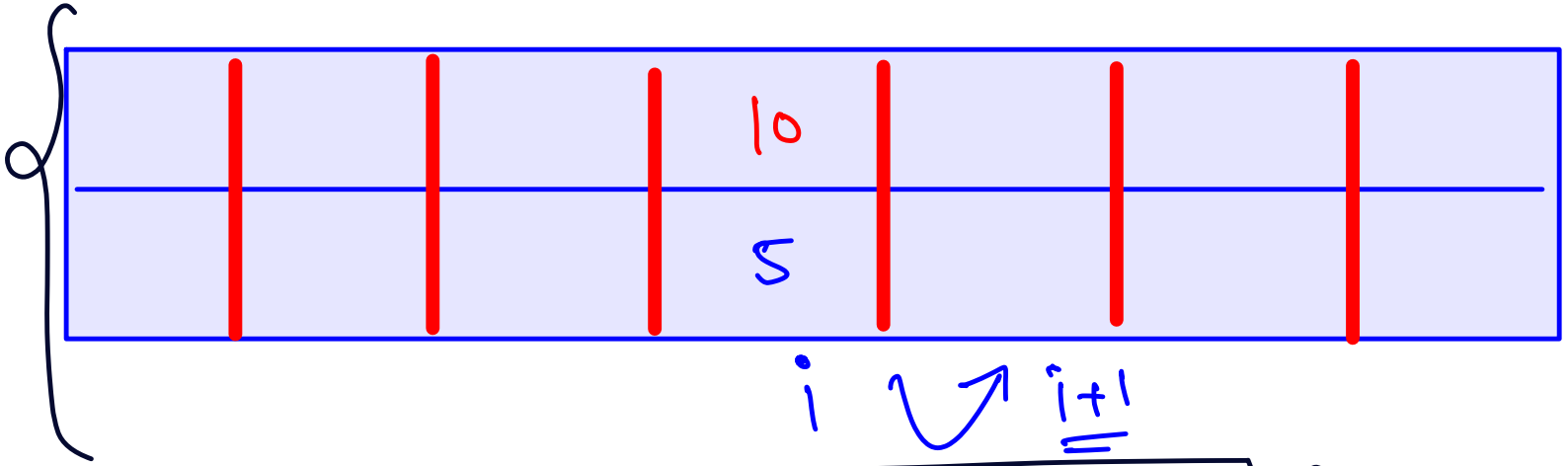
Problem: Book Shop [Knapsack]

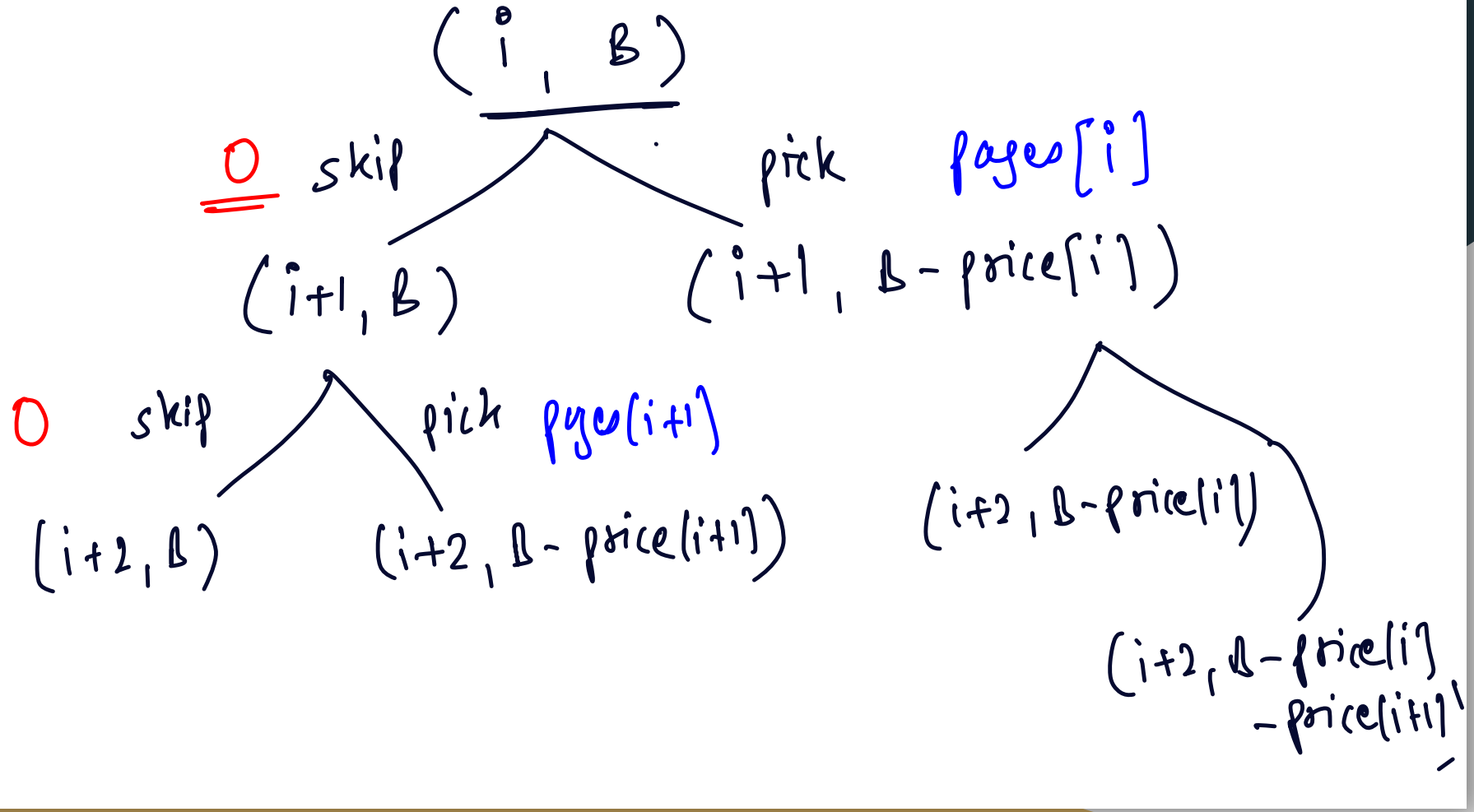
Link: https://cses.fi/problemset/task/1158

$\Longrightarrow$ find out which subset of books has a total price $\leq X$ and the total no. of pages is maximised.

Brute force $\longrightarrow O(2^n \cdot n)$


prices
10
9
2
1
20
5
4
pages
100
20
10
5
1000
500
200
X = 15
pick
skip
rem_b = 15
rem_b = 5
b = 5
b = 15
i
1


10
5
i
i+1
pick
skip
P
(i, B)
(i+1, B-10)
P1
(i+1, B)
P2
P

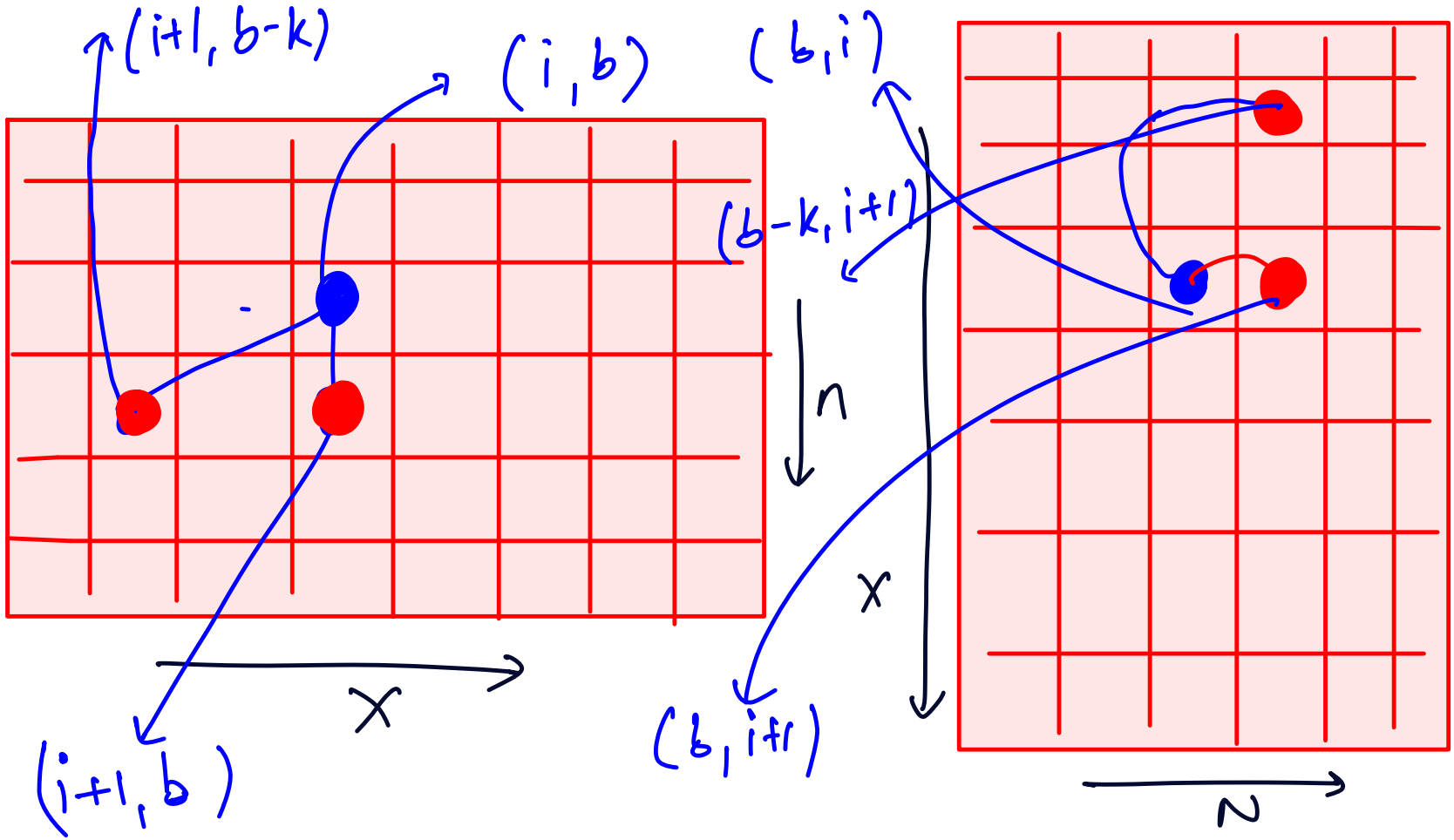

(i, B)
0 skip
pick pages[i]
(i+1, B)
(i+1, B - price[i])
0 skip
pick pages[i+1]
(i+2, B)
(i+2, B - price[i+1])
(i+2, B - price[i])
(i+2, B - price[i] - price[i+1])

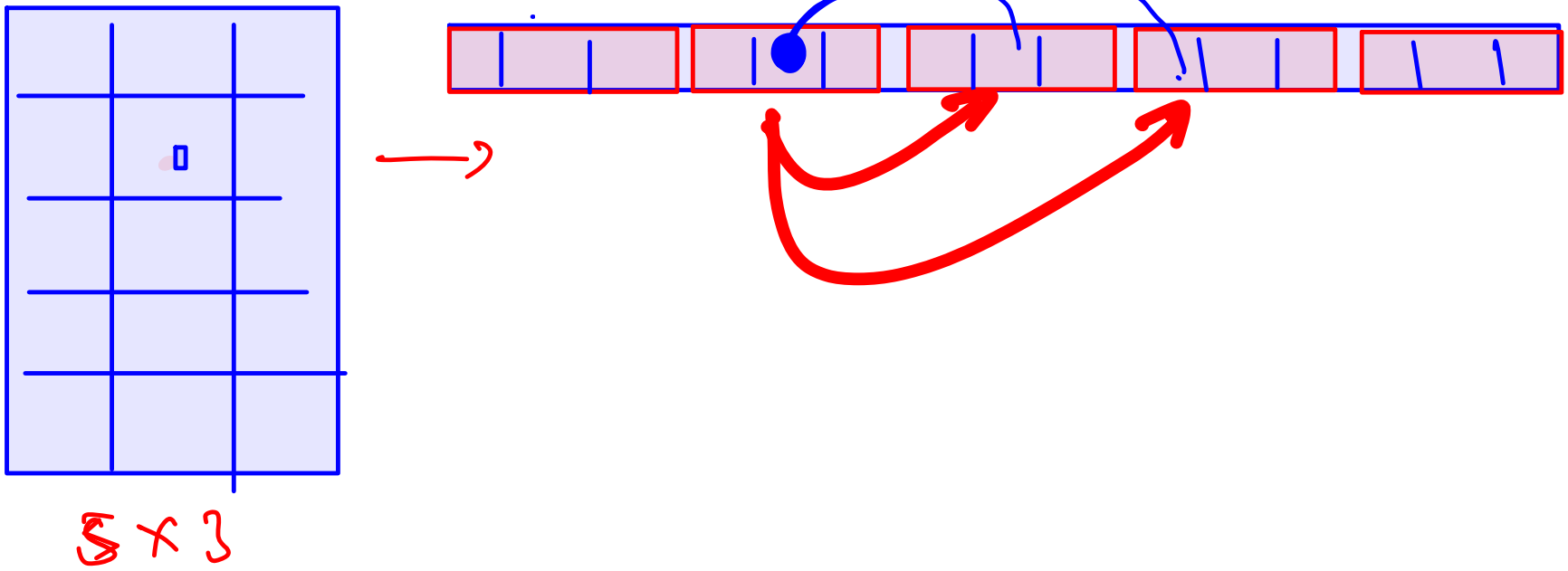
State ⟹ dp[i][b] = max. no. of pages we can read from ith book to n-1th book such that the allowed budget is b.

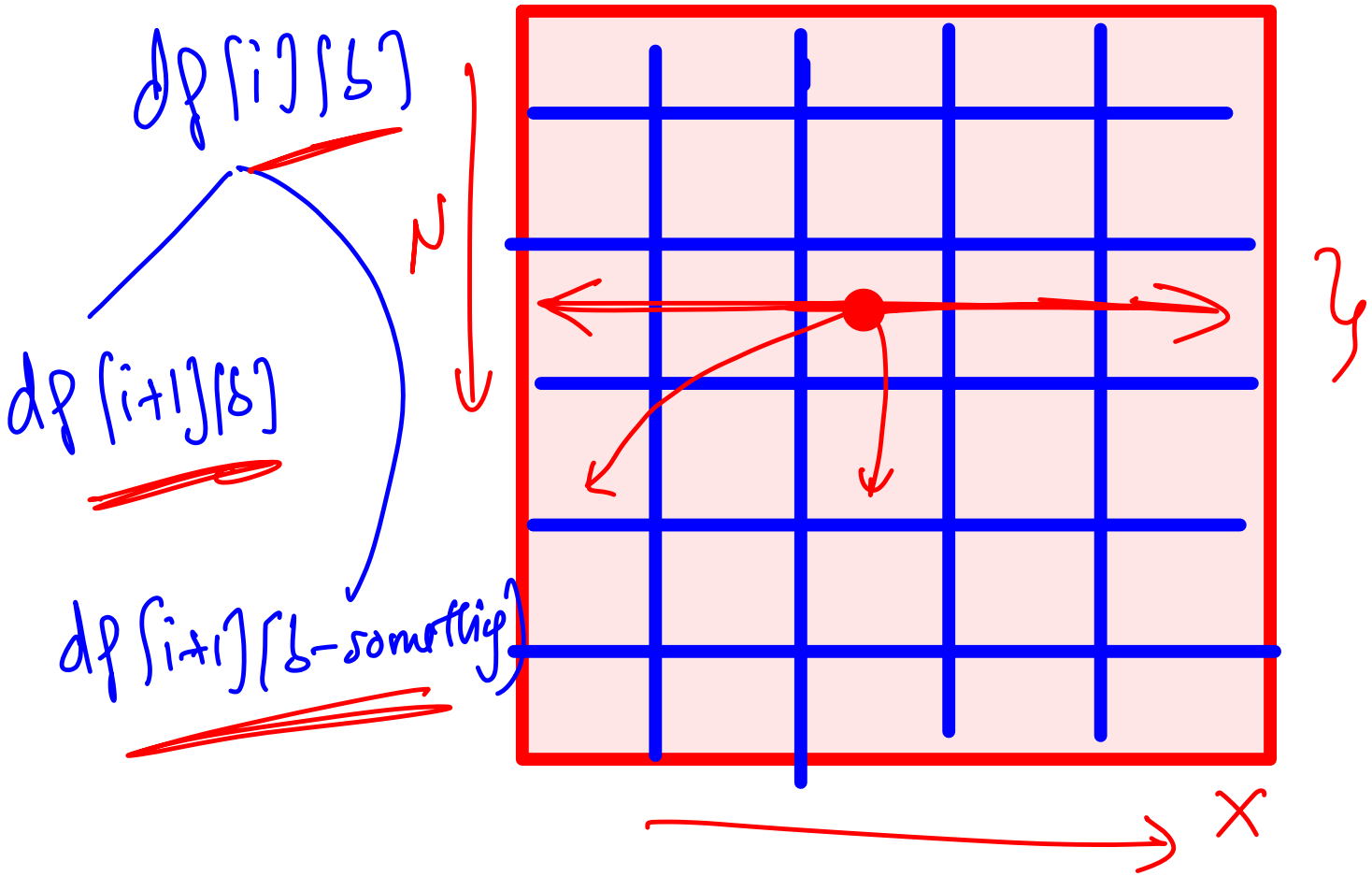
Transition ⟹ dp[i][b]
- pick ⟹ pages[i] + dp[i+1][b-price[i]]
- skip ⟹ 0 + dp[i+1][b]

Base Case ⟹ dp[n][anything] = 0

final problem ⟹ dp[0][x]


(i+1, b-k)
(i, b)
(b, i)
(b-k, i+1)
n
x
(i+1, b)
(b, i+1)
x
n


5 x 3


dp[i][j]
dp[i+1][j]
dp[i+1][j-1]
N
X

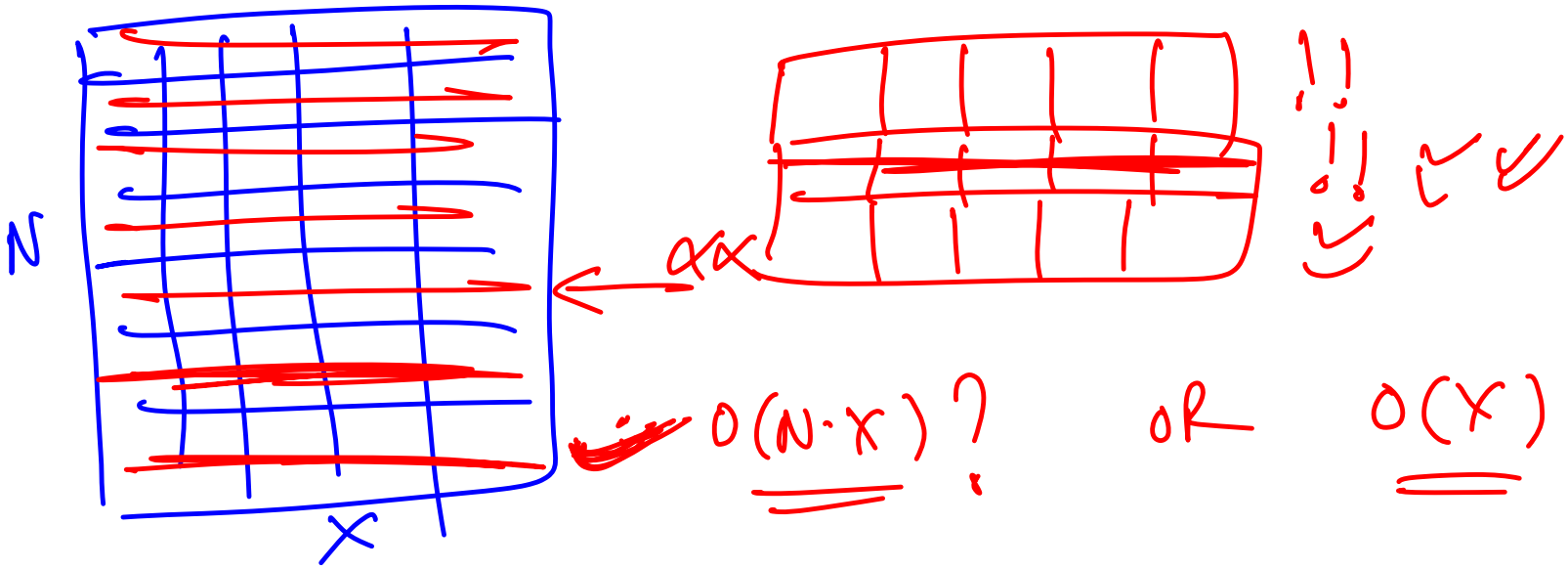
Space Optimization

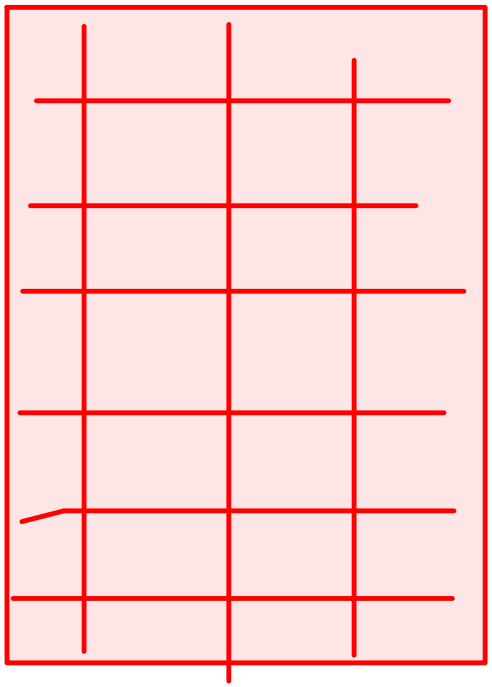
- What other state does our current state depend on?
- Do we need answers to all subproblems at all times?
- Well, let's store only relevant states then!
- But wait, does this always work?
  - What if the final subproblem requires all the states?
  - What if we need to backtrack? [more on this in later lectures]

dp[i][s] ← dp[i+1][s]

dp[i+1][s-something]

dp[i][anything] → dp[i+1][something]

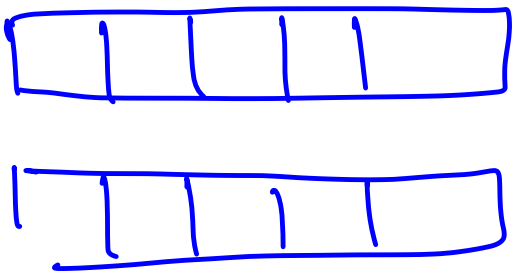




$$dp[i][b] = \max \begin{cases} dp[i+1][b] \\ pay[i] + dp[i+1][b - price[i]] \end{cases}$$

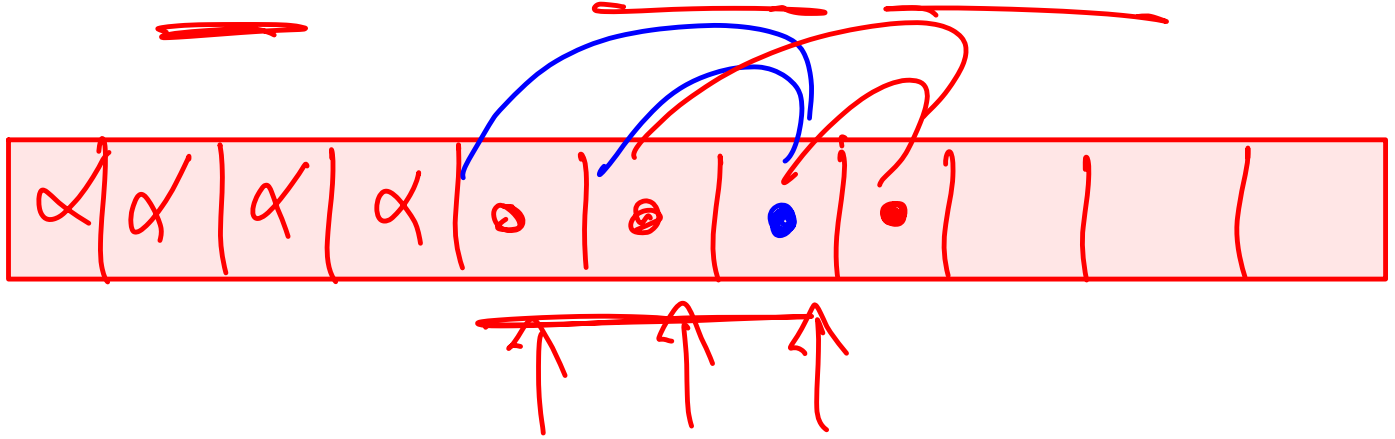
$dp(i)(anything) = curr(anything)$

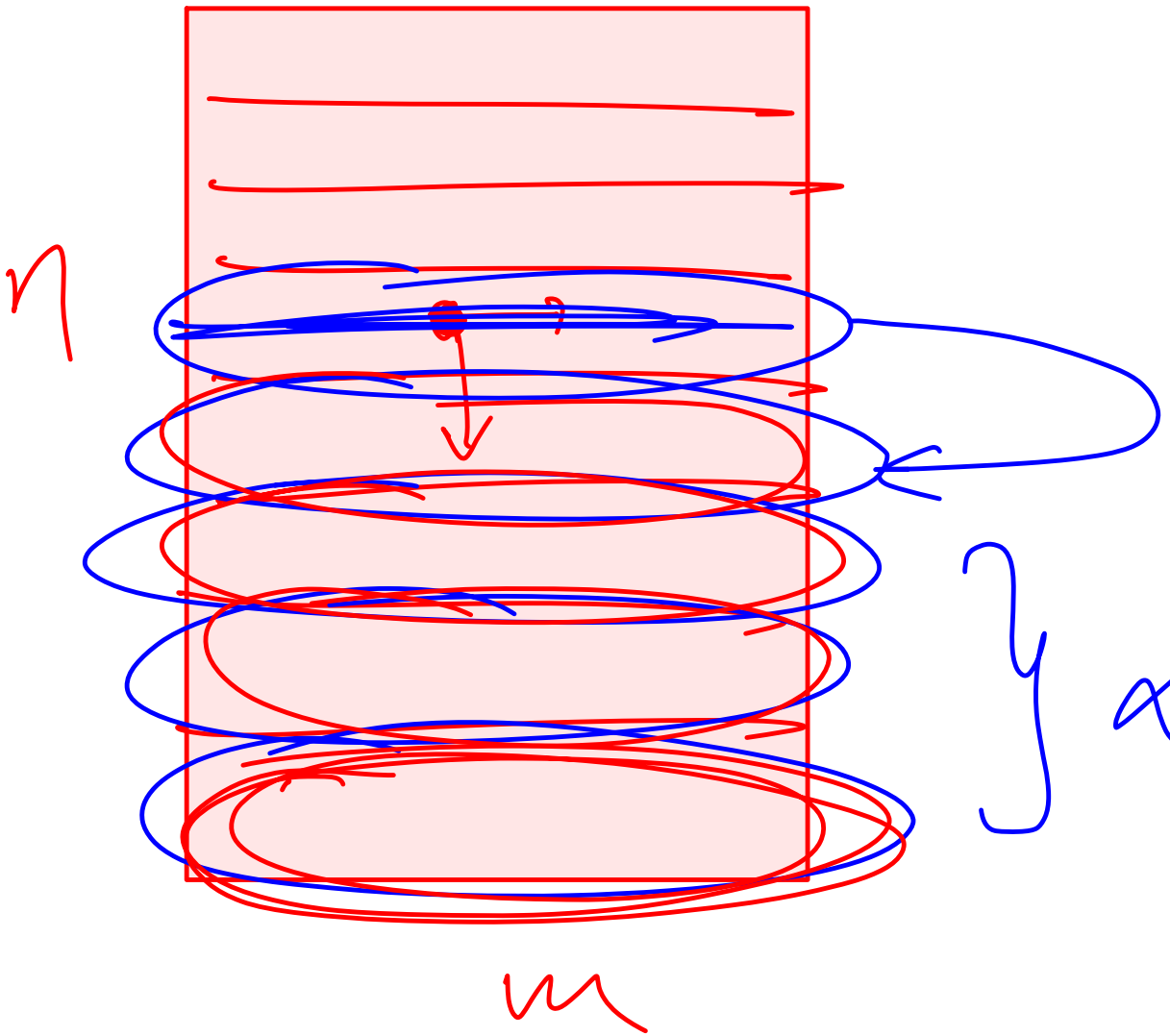
$dp(i+1)(anything) = next(anything)$



$$curr[b] = \max \begin{cases} next[b] \\ pay[i] + next[b - price[i]] \end{cases}$$

- Fibonacci Problem
  - dp[i] depends on dp[i - 1], dp[i - 2]
- Grid Problem
  - dp[i][j] depends on dp[i + 1][j], dp[i][j + 1]





$n \times m$

rows

columns

$O(m)$

$O(n)$

$n \times m = 10^7$

$n = 100$

$m = 10^5$

Transition Optimization

- Observe the transition equation.
- Can you do some pre-computation to evaluate the equation faster?
- Using clever observations.
- Using range query data structures
  - Segment Tree
  - Sparse Table

$dp[i]$ $\quad (0 \le i \le N)$

$dp[i] = dp[i-1] + dp[i-2] \cdots dp[0]$

$O(i)$

$$T.C = \sum_{i=0}^{N} T.T(i) = 0+1+2 \cdots N$$

$$= \frac{N \cdot (N+1)}{2} = N^2$$

$$= O(N^2)$$

estimate

$$= \#\text{ of states} \times \text{Avg TT}$$

$$= N \cdot O(N/2) = N^2$$

$dp[0]$ ⟶ calculate naively

$dp[1] = dp[0]$ ⟶ $sum[1] \rightarrow \sum_{i=0}^{1} dp[i]$

$dp[2] = dp[1] + dp[0]$ ⟶ $sum[2] \rightarrow \sum_{i=0}^{2} dp[i]$

$dp[3] = dp[2] + dp[1] + dp[0]$

↳ $sum[3] \rightarrow \sum_{i=0}^{3} dp[i]$

$dp[4] = dp[3] + dp[2] + dp[1] + dp[0]$

A =

Dp =

Sum =

$dp[i] = a[i] \times i + dp[i-1] + dp[i-2] \cdots dp[0]$

$dp[i+1] = a[i+1] \times (i+1) + dp[i] + dp[i-1] \cdots dp[0]$

$sum[i] = dp[i] + dp[i-1] + dp[i-2] \cdots dp[0]$

$dp[i+1] = a[i+1] \times (i+1) + sum[i]$

$sum[i+1] = sum[i] + dp[i+1]$

$dp[i+2] = a[i+1] \times (i+2) + sum[i+1]$

Example 1

$dp[n]$

$sum(i) = \sum_{k=0}^{i} dp[i]$

- Assume there is some hypothetical state definition
- Transition
  - dp[i] = dp[i - 1] + dp[i - 2] + dp[i - 3].... dp[0]
- Base Case
  - dp[0] = 1
- Final Subproblem
  - Find dp[n]

$dp[0] = 1$, $sum[0] = dp[0]$

```
for (int i = 1 ; i ≤ n ; i++) {
    dp[i] = sum[i-1] ;   ← O(1)
    sum[i] = sum[i-1] + dp[i]
                        ← O(1)
}
```

$O(N^2) \longrightarrow O(N)$

Example 2

sum[i] = dp[i] + dp[i-k] + dp[i-2k] ....

dp[i] = sum[i-k]

- Assume there is some hypothetical state definition
- Transition
  - dp[i] = dp[i - k] + dp[i - 2k] + dp[i - 3k]....
- Base Case
  - dp[0].... dp[k - 1] = some hypothetical values
- Final Subproblem
  - Find all dp[0]... dp[n]

sum[n];

for(int i=0; i<k; i++){
  sum[i] = dp[i]
}

for(int i=k; i<=n; i++){
  dp[i] = sum[i-k]
  sum[i] = dp[i] + sum[i-k]
}

$dp[i] = dp[i-k] + dp[i-2k] + dp[i-3k] \cdots\cdots$

$sum[i] = dp[i] + dp[i-k] + dp[i-2k] \cdots\cdots$

$dp[i+k] = dp[i] + dp[i-k] \cdots\cdots$

$\rightarrow sum[i]$

$sum[i+k] = dp[i+k] + sum[i]$

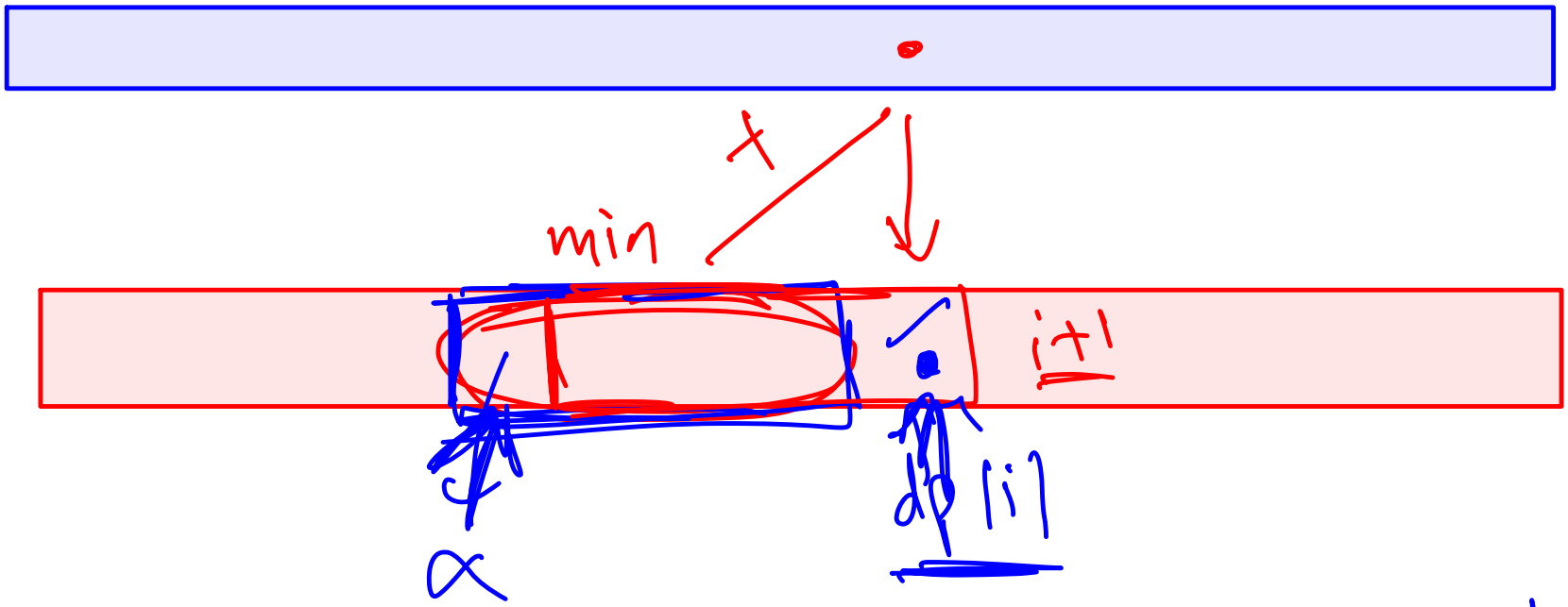
Example 3

k = 10

- Assume there is some hypothetical state definition
- Transition
  - dp[i] = arr[i] + min(dp[i - 1], dp[i - 2].... dp[i - k])
- Base Case
  - dp[0] = some hypothetical value
- Final Subproblem
  - Find all dp[0]... dp[n]

multiset

↳ O(N·K) → O(N log k)



$dp[i] = arr[i] + \min(dp[i-1], dp[i-2] \ldots dp[i-k])$

multiset → { all integers in a sorted way }

multiset<int> m;

m.insert(x);

m.erase(x) → all instances of x

m.erase(m.find(x)) → del only 1 x

Example 4

- Assume there is some hypothetical state definition
- Transition
  - dp[i] = arr[i] + min(dp[i - 1], dp[i - 2].... dp[i - arr[i]])
- Base Case
  - dp[0] = some hypothetical value
- Final Subproblem
  - Find all dp[0]... dp[n]

Segment tree

$dp[0] = 1$

$dp[i] = dp[i-1] + dp[i-2] + dp[i-3] \cdots\cdots$

$dp[i-1] = dp[i-2] + dp[i-3] \cdots\cdots$

$dp[i-2] = dp[i-3] + dp[i-4] \cdots\cdots$

$dp[i] = dp[i-1] + dp[i-1]$

$= 2 \cdot dp[i-1]$