XOR Queries	Topic-related tasks
Count on Tree	
Sign on Fence	
Forbidden Sum	
Let There be Rainbows	
Grid City	“Stolen” from a contest, sorted by (expected) hardest to easiest
Goofy Golf	
Collecting Apples	
Door of the Ancient	
Presidential Game	
Odd GCD Matching	

H	XOR Queries	Topic-related tasks
K	Count on Tree	
G	Sign on Fence	
F	Forbidden Sum	
J	Let There be Rainbows	
A	Grid City	“Stolen” from a contest, sorted by (expected) hardest to easiest
B	Goofy Golf	
C	Collecting Apples	
D	Door of the Ancient	
I	Presidential Game	
E	Odd GCD Matching	

Dynamic Programming Optimisation

Jonathan Irvin Gunawan

μ

A B

Σ ☺ τ

θ γ δ λ ψ ω Ε Δ α

♙ ♙ ♙ ♙ ♙ ♙ ♖ ♘ ♗ ♕ ♔ ♗ ♘ ♖ ♙ ♙ ♙ ♙ ♙ ♙

prerequisite

dynamic programming (dp)

convex hull

divide and conquer

let’s start simple

dp with reversed state

useful when you have a dp where the possible state are large, but the possible values are small

example:
given 0/1 knapsack problem

$1 \leq N \leq 100$
$1 \leq Wi, W \leq 1e9$
$1 \leq Vi \leq 100$

usual knapsack solution is O(N*W), does not work for this problem notice that the constraint for the values is small instead of dp[total_weight] = max_value, we can reverse the state and the value

dp2[total_value] = min_weight
in order get a total value of total_value, what is the minimum total weight of the items

```
reset(dp, INT_MIN), dp[0] = 0
for i in 1..N
for j in W..0
dp[x] = max(dp[x], dp[x - w[i]] + v[i])
```

```
reset(dp2, INT_MAX), dp[0] = 0
for i in 1..N
for j in sum(Vi)..0
dp2[x] = min(dp2[x], dp2[x - v[i]] + w[i])
```

the answer is the maximum v that still satisfies dp2[v] ≤ W this is now

O(N * sum(Vi))

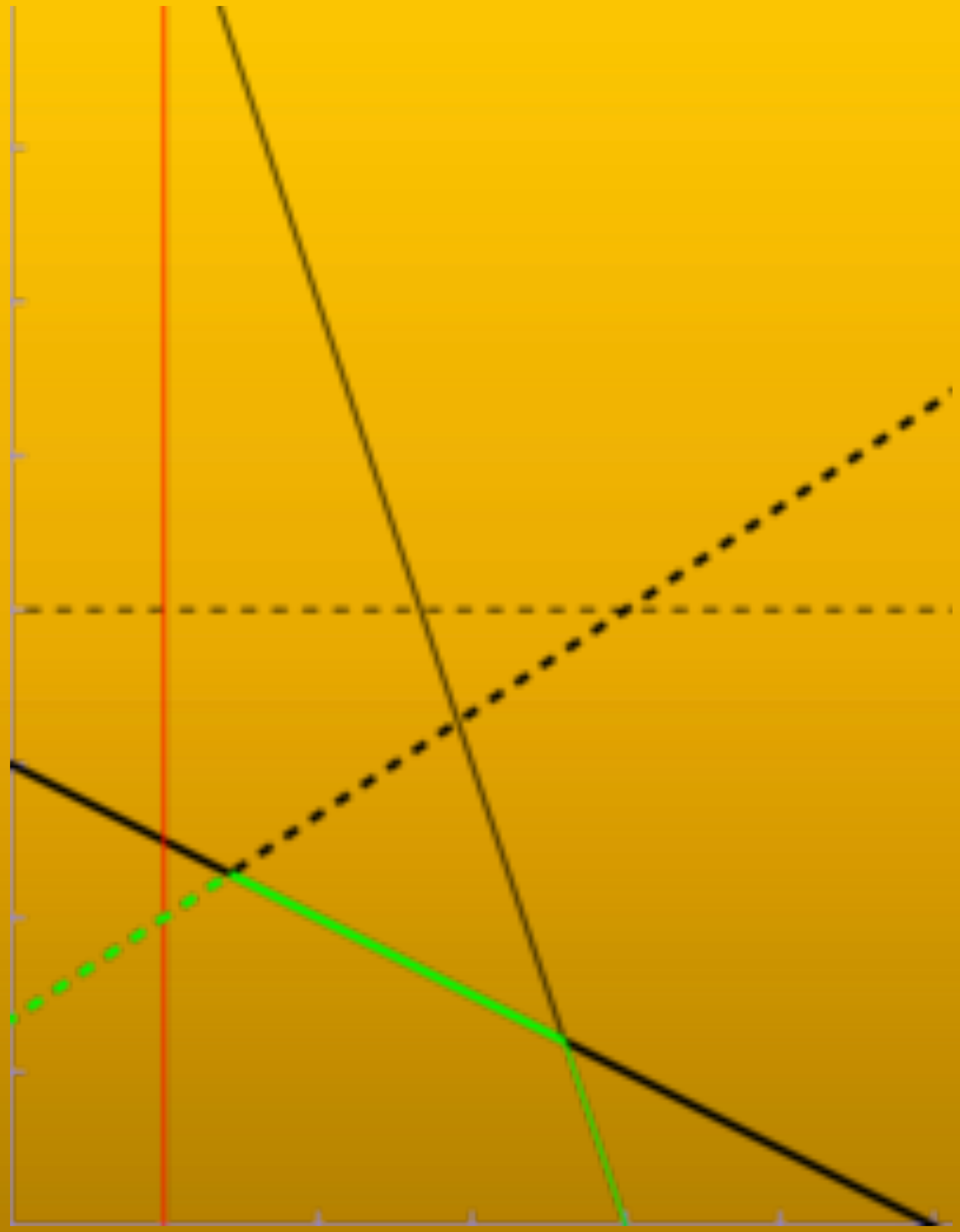
next

convex hull optimization

not specific for dp, but quite often used as dp optimisation

basic formulation:
given N lines $y = m*x + c$.
there are Q queries. at $x=xi$, which line produces the minimum $m*xi + c$

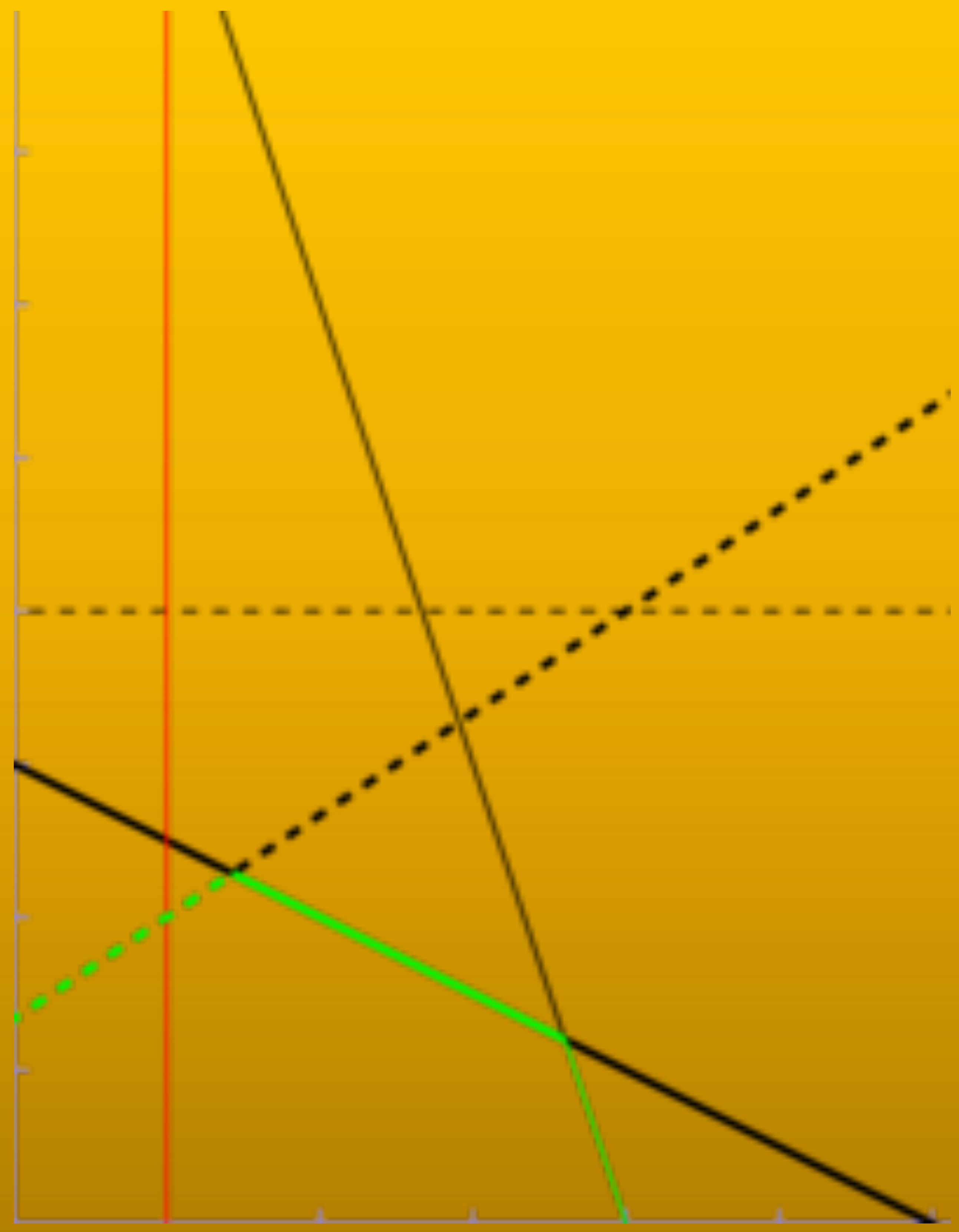
idea: each line can be the minimum for a contiguous values of x (can be unbounded)



green is
minimum line
(upper hull)

once we know the interval endpoints, we can answer each query using binary search how to find interval endpoints?

note that the upper hull will have decreasing slope similar to graham scan: sort the lines by slope and maintain a stack keep popping lines from stack if they are obsoleted



```
struct Line {
  int m, c;
  int calc(int x) {
    return m * x + c;
  }
};
```

```
// a.m > b.m > c.m
bool obsolete(Line a, Line b, Line c) {
  // a and c intersect at
  // x_ac = (c.c - a.c) / (a.m - c.m)
  // a and b intersect at
  // x_ab = (b.c - a.c) / (a.m - b.m)

  // b is obsolete if x_ac < x_ab
  return (c.c - a.c) * (a.m - b.m)
       < (a.m - c.m) * (b.c - a.c)
}
```

```
vector<Line> lines;
void insert(Line l) {
  while (lines.size() > 1) {
    int sz = lines.size();
    if (obsolete(lines[sz - 2], lines[sz - 1],
                 l)) {
      lines.pop_back();
    } else break;
  }
  lines.push_back(l);
}
```

example problem

APIO 2010
Commando

Given array X of N integers . You want to partition them contiguously such that the sum for a * sum(X_i)^2 + b * sum(X_i) + c among all partitions is maximized.

$N \leq 1e6$

$-5 \leq a \leq -1$

$|b|, |c| \leq 1e7$

$1 \leq X[i] \leq 100$

simple dp

dp[i] = maximum sum only considering X[1..i]

dp[0] = 0

dp[i] = max(1≤j≤i)

a * (pre[i] - pre[j-1])^2

+ b * (pre[i] - pre[j-1]) + c + dp[j-1]

let pj = pre[j-1], pi = pre[i]

dp[i] = max(1≤i≤j)

a * (pi-pj)^2 + b * (pi - pj) + c + dp[j-1]

a*pi^2 - a*2*pi*pj + **a*pj^2** + **b*pi - b*pj** + **c** + **dp[j-1]**

a*pi^2 + b*pi + c

+ pi * (-a*2*pj)

+ a*pj^2 - b*pj + dp[j-1]

dp[i] = **a*pi^2 + b*pi + c** +
max(1≤j≤i)
+ pi * (-a*2*pj)
+ a*pj^2 - b*pj + dp[j-1]

insert line m = (-a*2*pj), c = (a*pj^2 - b*pj + dp[j+1])

(-2*a*pj) increases with larger j (-2*a is positive)
gradient is increasing

dp[i] = **a*pi^2 + b*pi + c** +
max(1≤j≤i)
+ pi * (-a*2*pj)
+ a*pj^2 - b*pj + dp[j-1]

insert line m = (-a*2*pj), c = (a*pj^2 - b*pj + dp[j+1])

query is pi, also increases with larger i
binary search is not needed

$O(N)$

O 

what if gradient might not be monotonic?

find where the lines should be (based on gradient)
remove obsoleted lines to the left and to the right
use std::set for removal in the middle of data structure

amortized logarithmic time

next

dp dnc

let’s say the common dp

$$dp[i][j] = \min_{(1 \le k \le N)} dp[i-1][k] + cost(i,j,k)$$

let's say the common dp

$$dp[i][j] = \min_{1 \le k \le N} dp[i-1][k] + cost(i,j,k)$$

and $OPT(i, j) \le OPT(i, j + 1)$

find dp[i][1..N] can be done in O(N lg N)

find dp[N/2] first
then we can find opt of dp[1..N/2]
only in 1..opt(N/2)
and dp[N/2..N]
only in opt(N/2)..N

```
void dnc(int L, int R, int optL, int optR) {
  if (L > R) {
    return;
  }
  int M = (L + R) >> 1;
  int opt = optL;
  for (int i = optL; i <= optR; ++i) {
    if (cost(M, opt) < cost(M, i)) {
      opt = i;
    }
  }
  dp[M] = cost(M, opt);
  dnc(L, M - 1, optL, opt);
  dnc(M + 1, R, opt, optR);
}
```

each layer takes at most 2N iterations

there are O(lg N) layers

total O(N lg N)

example

https://www.hackerrank.com/contests/world-codesprint-5/challenges/mining

given N mines.
mine i is located X[i] from the left and contains W[i] gold
we need to gather the gold to only K “pick-up” mines
moving gold from mine i to mine j takes
$|X[i] - X[j]|$ cost
determine minimum cost

$1 \leq N, K \leq 5000$
X is increasing

dp[rem][i] = minimum cost of gathering gold[i..N] to rem pick-up mines

dp[rem][i] = min(j≥i)
dp[rem-1][j+1] + gather cost(i,j)

O(KN^2)

we can find dp[rem][1..N] in O(N lg N)

$OPT(i + 1) \geq OPT(i)$
proof by contradiction

suppose OPT(i) = k, OPT(i+1) = j, j < k

dp[i] ≤ dp[i+1]

cost(i,k) + dp'[k+1] ≤ **cost(i+1,j) + dp'[j+1]**

OPT(i+1) = j

cost(i+1,j) + dp'[j+1] ≤ cost(i+1,k) + dp'[k+1]

therefore

cost(i,k) + dp'[k+1] ≤ cost(i+1,k) + dp'[k+1]

$$\text{cost}(i,k) + \cancel{dp'[k+1]} \leq \text{cost}(i+1,k) + \cancel{dp'[k+1]}$$

$$\text{cost}(i,k) \leq \text{cost}(i+1,k)$$

contradiction

$O(K*N*\lg N)$

another dnc task

Codeforces Round #406 (Div 1) problem C

Codeforces Round #406 (Div 1) problem C

Given N people in a line, each having a color. For each $1 \le k \le N$, we want to partition the people so that each group is a contiguous interval and has at most k distinct colours. Determine the minimum number of groups

$$1 \le N \le 1e5$$

```
int naive(int k) // do naively in O(N)

void solve(int l, int r) {
  if (l + 1 >= r) return;
  int mid = l + r >> 1;
  ans[mid] = ans[l] == ans[r]
             ? ans[l] : naive(mid);
  solve(l, mid);
  solve(mid, r);
}

ans[1] = naive(1);
ans[n] = 1;
solve(1, n);
```

last

dp knuth-yao optimisation

let's say the common dp

$$dp[i][j] = cost(i,j) + \min(i \le k < j)$$

$$dp[i][k] + dp[k+1][j]$$

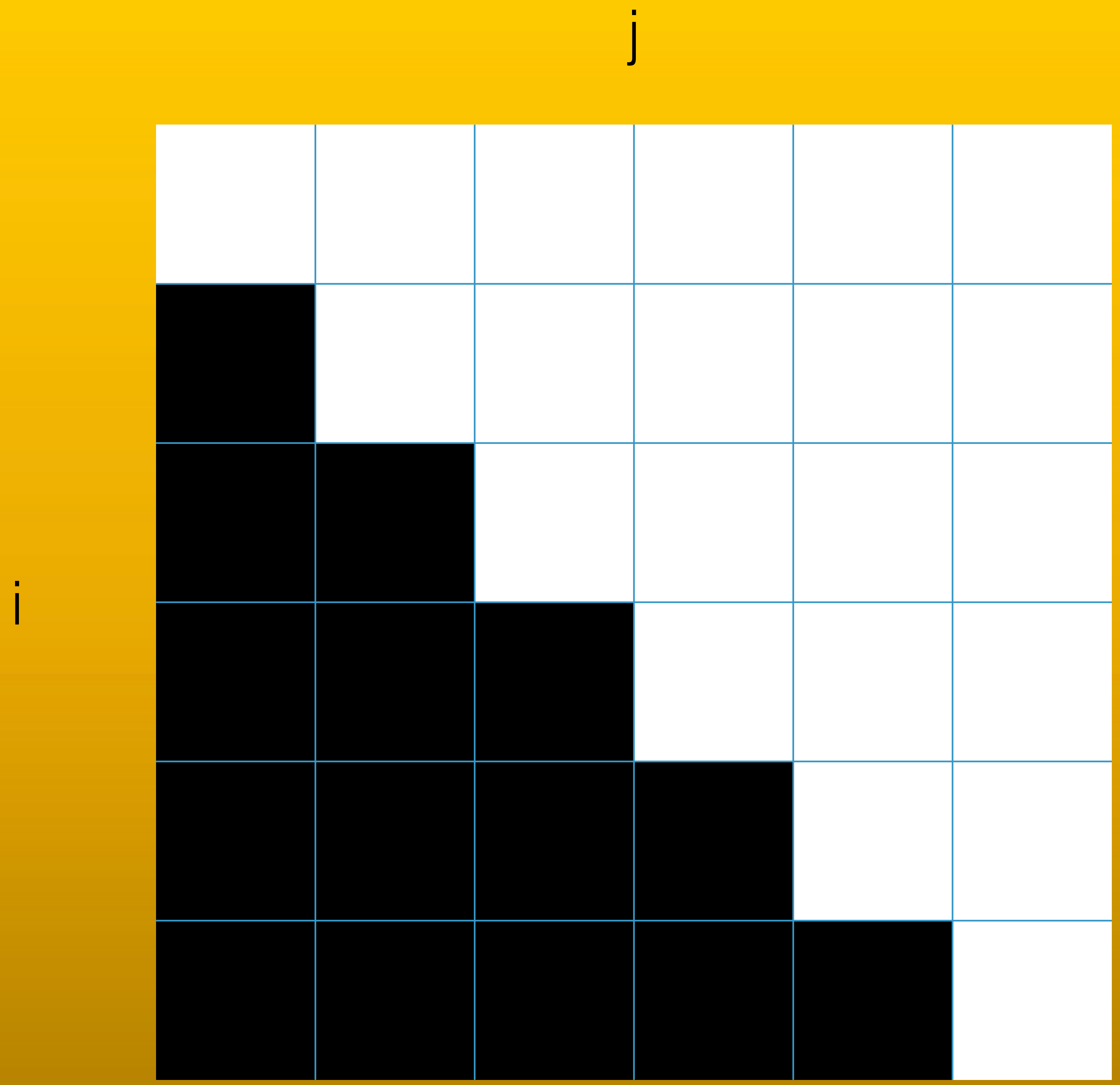
let's say the common dp

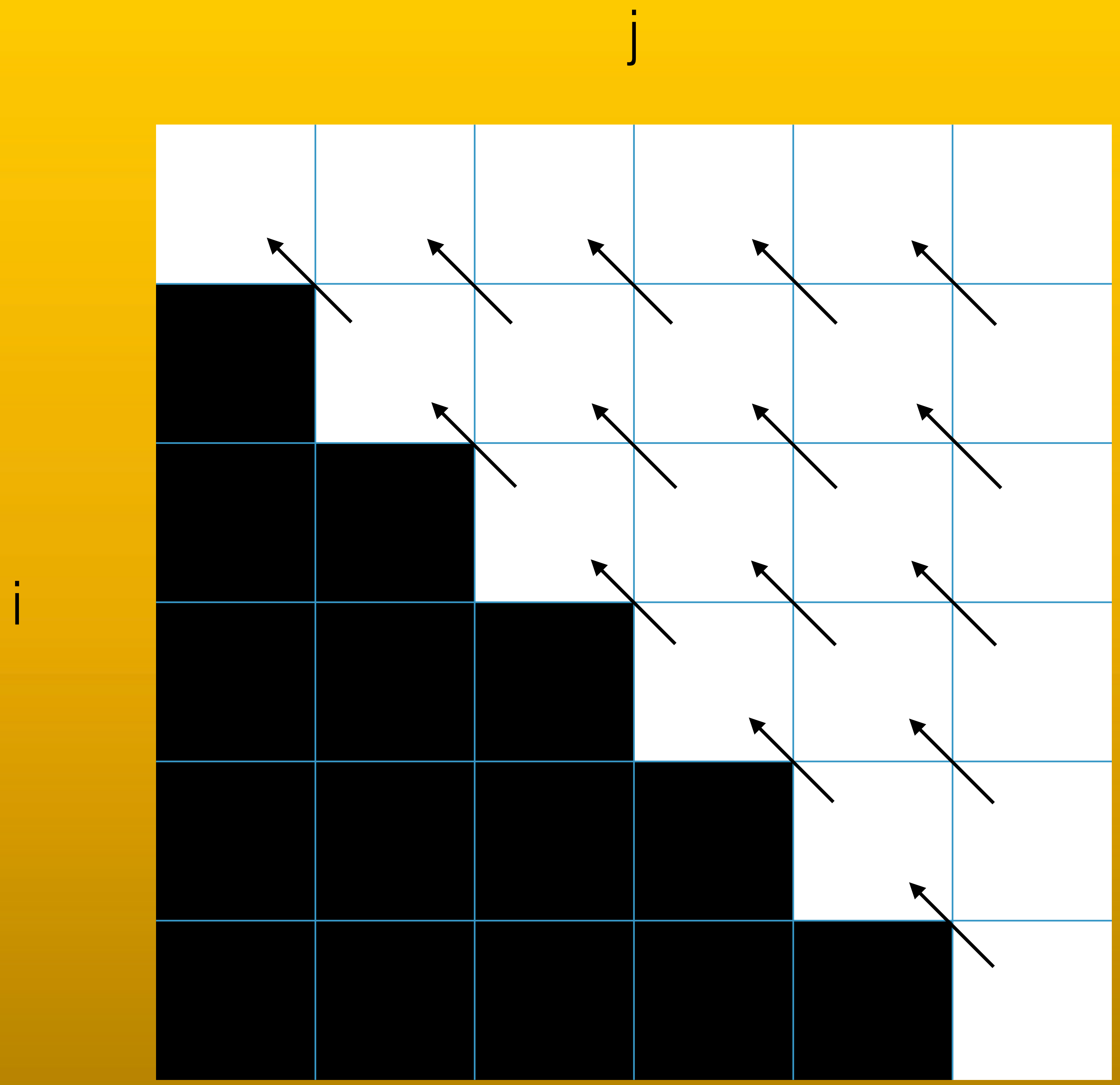
$$dp[i][j] = cost(i,j) + \min_{i \le k < j} \left(dp[i][k] + dp[k+1][j] \right)$$

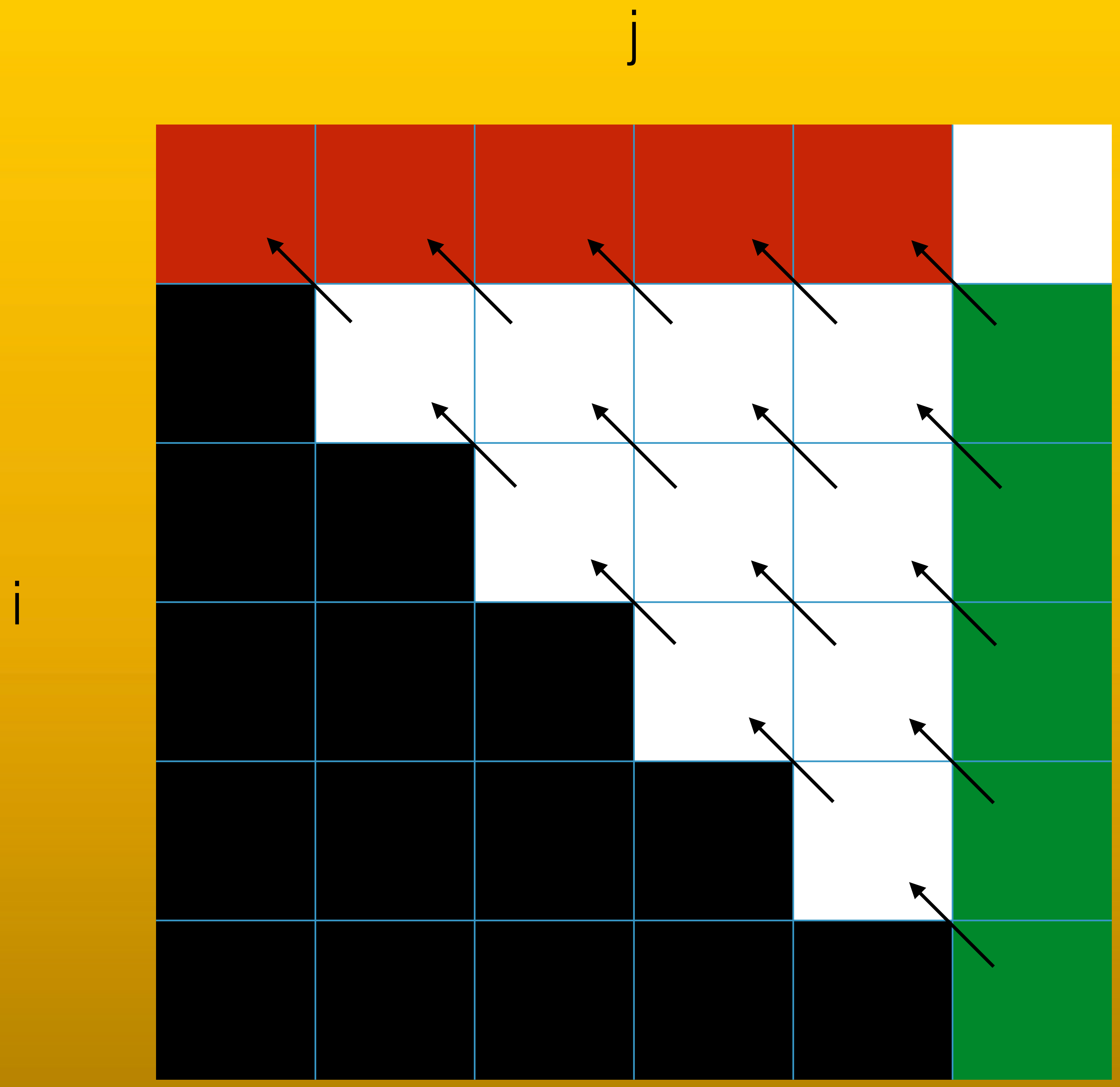
and $OPT(i,j-1) \le OPT(i,j) \le OPT(i+1,j)$

it’s obvious that the loop can be optimized

but what’s the total running time now?


j
i


j
i


j
i

sum of at most N opt(i,j) = O(N^2)

basically each dp(i,j) is amortized O(1)

$$\text{cost}(i,j) \leq \text{cost}(i,j+1) \text{ and}$$
$$\text{cost}(i,i+1) + \text{cost}(i+1,i+2) \leq \text{cost}(i,i+2) + \text{cost}(i+1,i+1)$$

implies

$$OPT(i,j-1) \leq OPT(i,j) \leq OPT(i+1,j)$$

proof is just messy math work

left for exercise

$$\mathrm{cost}(i,i+1) + \mathrm{cost}(i+1,i+2) \leq \mathrm{cost}(i,i+2) + \mathrm{cost}(i+1,i+1)$$

means broader range have more cost (e.g., quadratic function)

classic usage:
optimal binary search
tree problem

given N elements. i-th element is going to be queried E[i] times. construct the optimal binary search tree to minimize total query time.

ok let’s do some tasks

SPOJ ACQUIRE

Given N rectangular plots with width and height. You can buy one land to cover a group for rectangular plots where the cost is maximum width * maximum height

Determine minimum total cost to cover all rectangular plots

$1 \leq N \leq 50k$

https://www.hackerrank.com/contests/worldcupsemifinals/challenges/find-number/problem

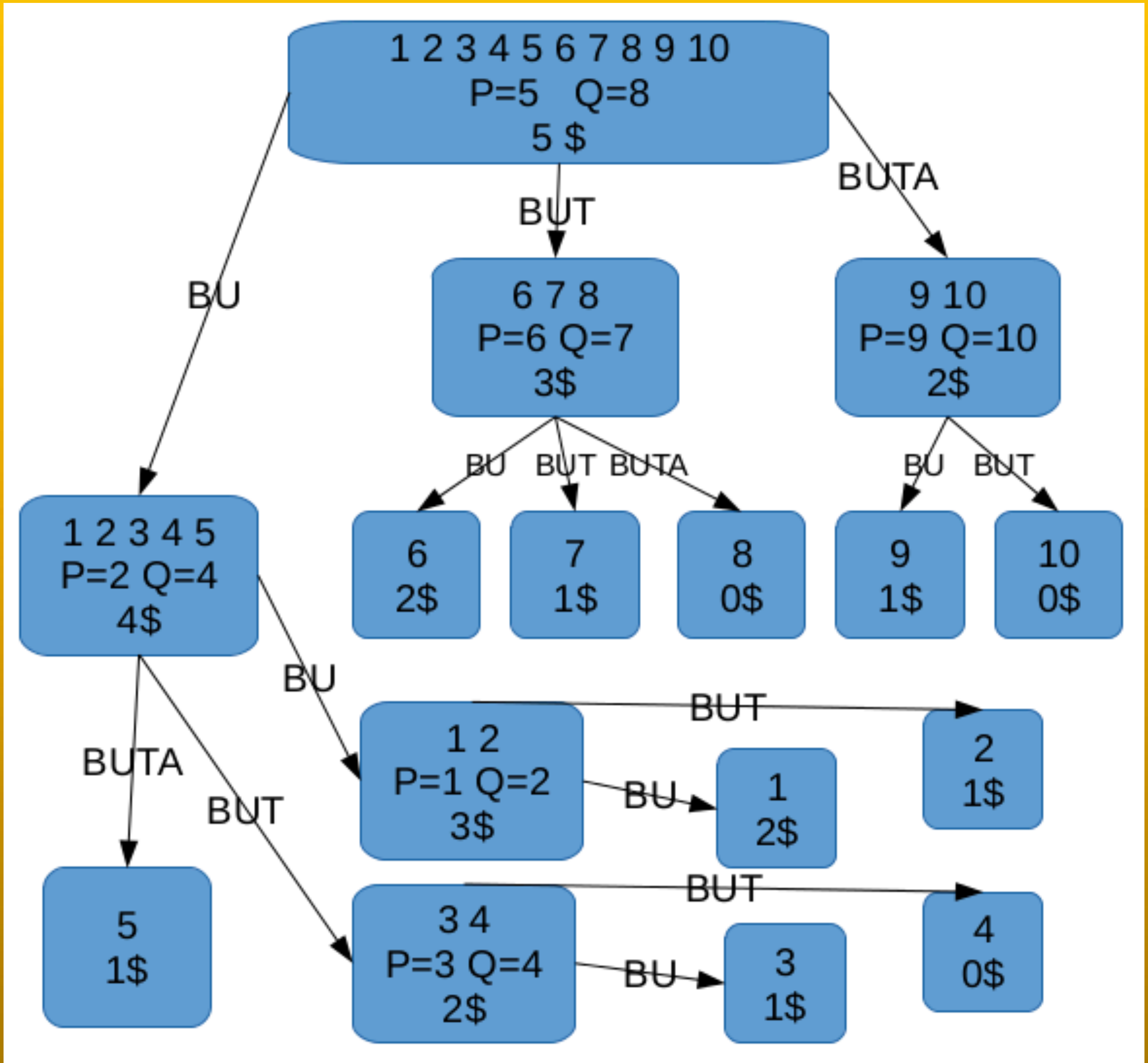
You are guessing a number X between 1 to N. Each guess you can give P and Q ($0 \le P \le Q \le N$). You are told whether $X \le P$, $P < X \le Q$, or $Q < X$ and need to pay \$A, \$B, or \$C respectively

Find minimum cost to get X

$1 \le N \le 1e15$

$1 \le A,B,C \le 100$

example, N = 10, A = 1, B = 2, C = 3. answer=5



APIO 2014
Split the Sequence

You have array A of N elements. You want to do this K times:

1. Choose any array that has more than one element
2. Split the array into two
3. Point increased by multiplication of sums of elements of the two splitted arrays

Find maximum total number of points

$2 \leq N \leq 1e5$

$1 \leq K \leq \min(N-1,200)$

Topcoder SRM 708 PalindromicSubseq

There is a string of N characters.
For each i, calculate the number of palindromic subsequences containing i-th character.
The same character on different indices are considered different.

$1 \leq N \leq 3000$

EOF

Q&A?