Dynamic Programming

these are the reasons.....

greedy ———— x
+
backtracking ———— x

D&C   lucky or unlucky to pick breaking points

DP   figures out breaking points, bottom-up

Fibonacci #'s

rec. $F(n) = F(n-1) + F(n-2)$ top → bottom

repeating a lot of work

iter.   fill an array   bottom → up

Same idea with DP.   Find recurrence, then find an
iterative version to fill an array.

⤷ usually a more
complex recurrence than in D&C

string reconstruction

problem $S[1 \cdots n]$

sub $S[i \cdots j]$ is a collection of words
problem

dictionary of words — constant time

2 cases $S[i \cdots j] \in$ Dict ← boolean
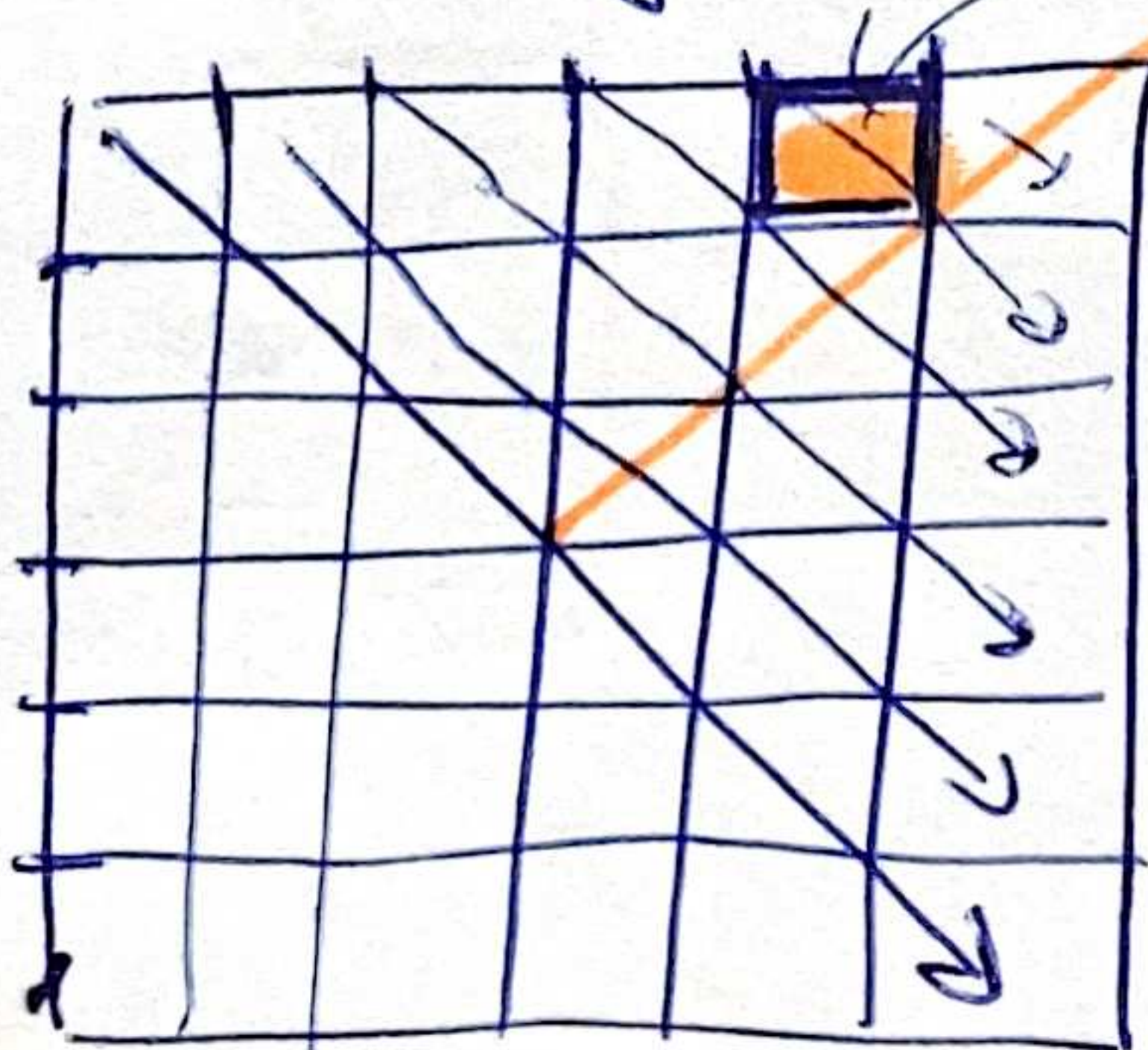
or $\exists k$, s.t. $S[i \cdots k]$, $S[k+1 \cdots j]$ are
$i \le k \le j-1$ both collection of words

Array formulation

$S[i,j]$

$S[1,1]$, $S[2,j]$
$S[1,2]$, $S[3,j]$
$S[1,3]$, $S[4,j]$

Bottom-up

⟶ figure dependency

for d = 1 to n

for i = 1 to n − d + 1

$j = i + d − 1$

if $S[i, j] \in$ Dict

$A[i, j] = T$

for k = i to j − 1

if $A[i, k] = T$ and $A[k+1, j] = T$

$A[i, j] = T$

} build from small to large strings

$O(n^3)$

$O(n^2)$ lookups in dictionary

info to keep for reconstruction: keep k, the breaking point

Correctness   induction on d

Set up a better recursion, 2D to 1D array

$S[i, n]$ is a collection of words?

rec.

on string

$\exists k \quad S[i, k] \in$ Dict

$S[k+1, n]$ is a collection of words

$i \leq k \leq n−1$

iter

| | | | | | | v | v | x | x | |

or in the opposite direction

$O(n^2) \leftarrow$ not tight

$O(nL) \quad L = $ length of the longest word in Dict.

info to keep for reconstruction: keep breaking point k
or keep all k's for all possible ways

Edit Distance

how close are two sequences

Hamming distance
```
1 1 0 1 1 1 0
0 1 0 1 1 0 0
      ‖
      2
```
places where sequences do not match

bad for shifts/insertions
```
1 1 0 1 1 1 0      4
1 0 1 1 1 0 _
```

edit distance
# of edit operations to transform one string to another

op	cost
— insert	c_i
— delete	c_d
— replace	c_r

ex   activ ate
       ca veat
ops   D RD I   D

5 ops, if all costs are 1
then edit distance is 5
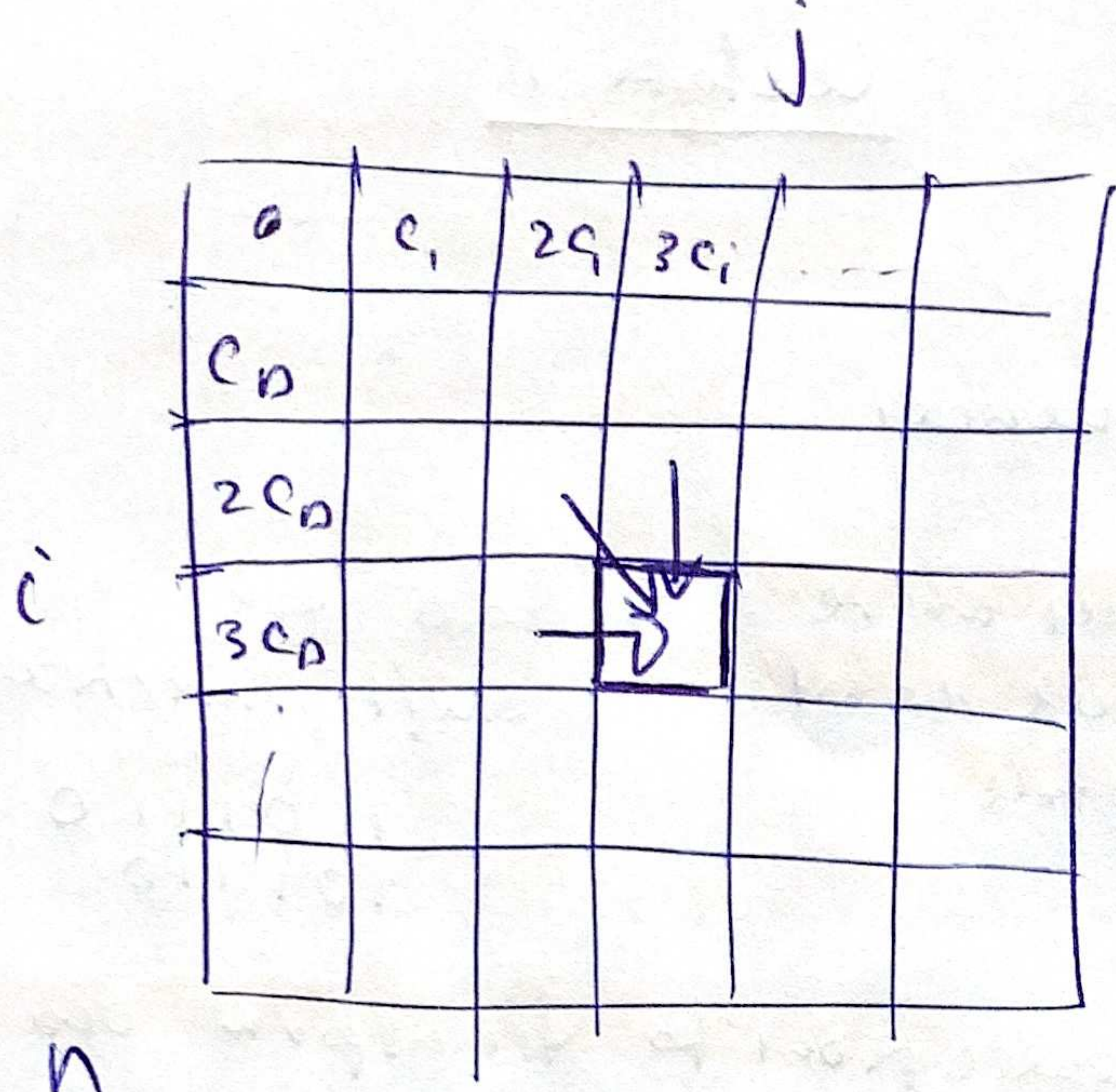
↓ $A[1....n]$
↓ $B[1.....m]$

subproblem $A[i...j] \to B[k...l]$ vs
              $A[1...i] \to B[1...j]$

ex: activ $A[1\ 5]$
    cave $B[1\ 4]$

$$C(i,j) = \min \begin{cases} C(i-1,j-1) & \text{if } A(i)=B[j] \quad \text{match} \\ C(i-1,j-1) + C_R & \text{if } A(i) \neq B[j] \\ C(i,j-1) + C_I \\ C(i-1,j) + C_D \end{cases}$$

rec

on cost function

iter

j m

	c_1	$2c_i$	$3c_i$	...
c_D				
$2c_D$				
$3c_D$				

i

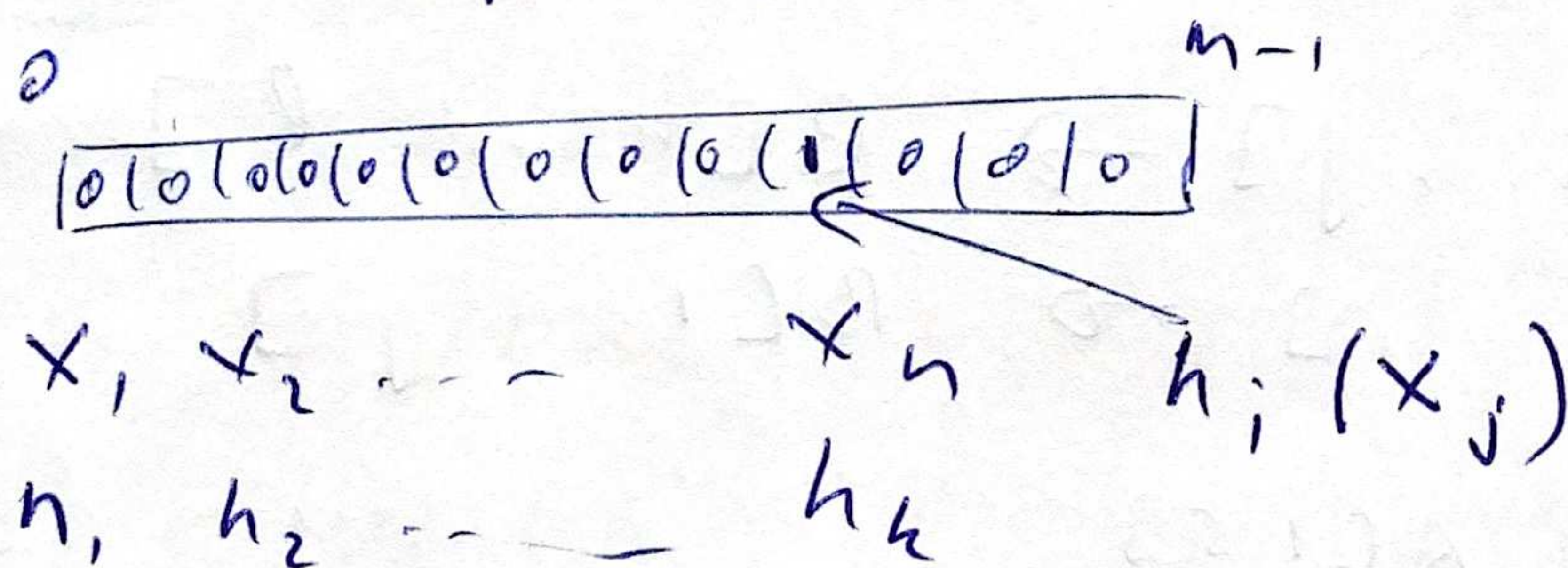n

array $O(nm)$

$O(1)$ to fill a cell

$\Rightarrow O(nm)$

to reconstruct
keep info of pointers
that provide min
in the recurrence

Review

Bloom filter data structure
  correctness vs. space
  allow false positives, in return get small space

0 $m-1$

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

$x_1, x_2 \cdots x_n$ $h_i(x_j)$

$n_1, h_2 \cdots h_k$

hash
into the same
array or
into separate
arrays

$\hookrightarrow$ asymptotically
the same

$$k = (\ln 2)\, \frac{m}{n}$$

$\uparrow$
optimum