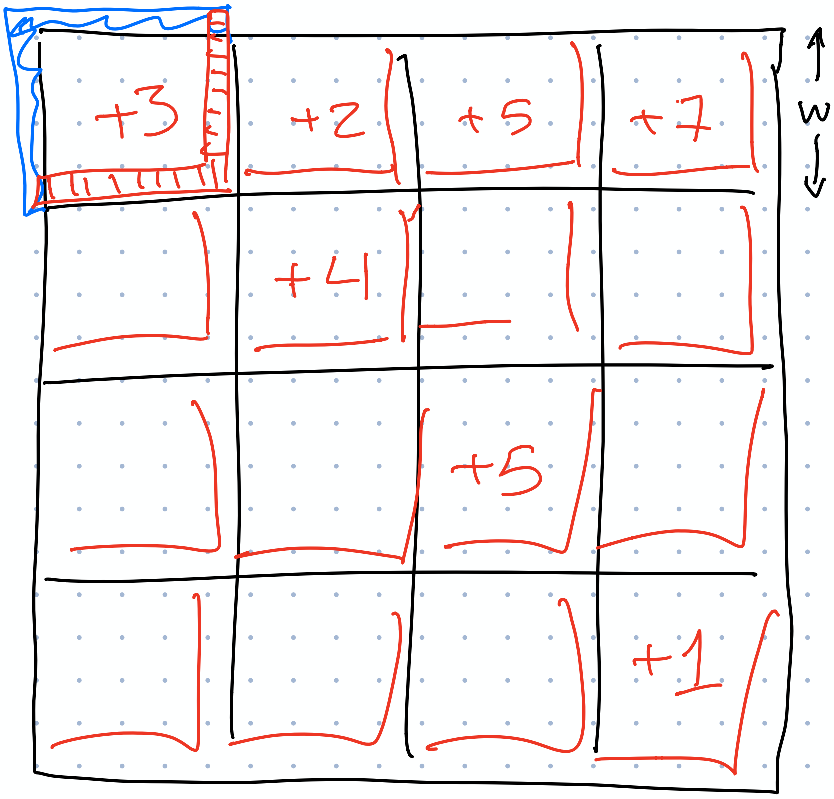
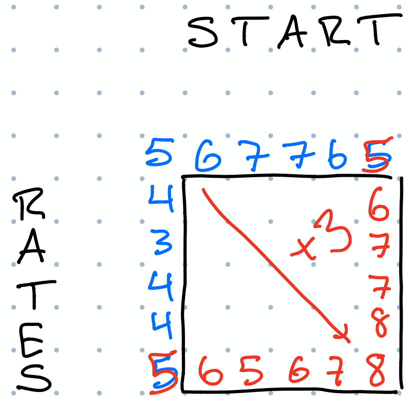
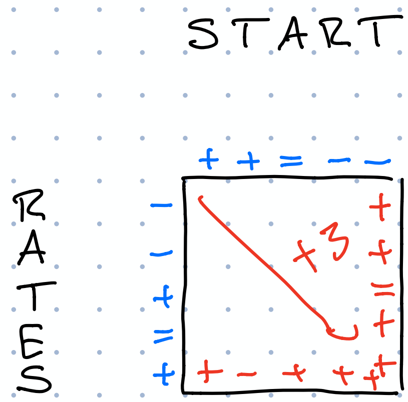
"Four Russians"



Lemma: $|\mathrm{Edit}[i,j] - \mathrm{Edit}[i-1,j]| \le 1$

Assume input strings are in Σ^* where $|\Sigma| = O(1)$





Encode block:

- Top left $\mathrm{Edit}[i,j]$ — side data

input:
- Offset vectors $T \in \{+1,0,-1\}^w$, $L \in \{+1,0,-1\}$
- Substrings $A[i..i+w]$, $B[j..j+w]$

output:
- Offset vectors $R, B \in \{-1,0,1\}^w$
- Global shift

Naively: Solve any block in $O(w^2)$ time

How many distinct blocks? $3^w \cdot 3^w \cdot \Sigma^w \cdot \Sigma^w = (3\Sigma)^{2w}$

Solve all of them in $O((3\Sigma)^{2w} \cdot w^2)$ time

$\Longrightarrow$ Main "DP" takes $O\left(\frac{n^2}{w^2}\right)$ time

Let $w = \frac{\log n}{2 \log(3\Sigma)}$

$O(n \log^2 n)$

$O(n^2/\log^2 n)$ ✓

Combine with C+R: $O(n^2/\log n)$ time

$O(n/\log n)$ space

OPT path thru blocks

$O(n)$ time + space $\longrightarrow$ path

Order LIS – longest incr. subsequence

naive: $O(n^2)$ time

1 2 3 5 8 13 21 9

Given $A[1..n]$

$LowIS(i, \ell)$ = smallest possible last number in an increasing subseq of $A[1..i]$ of length ℓ.

$$LowIS(i,\ell) = \begin{cases} -\infty & \text{if } \ell = 0 \\ +\infty & \text{if } \ell > i \\ A[i] & \text{if } A[i] > LowIS(i-1, \ell-1) \text{ and } A[i] < LowIS(i-1, \ell) \\ LowIS(i-1,\ell) & \text{o/w} \end{cases}$$

PACKMYBOXWITHFIVEDOZENLIQWORJUGS

i

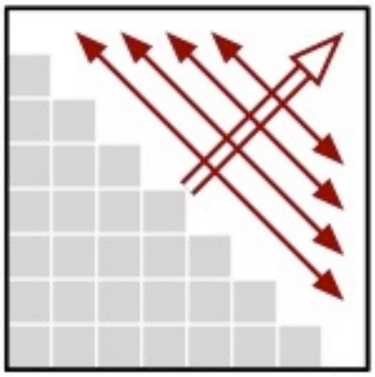
$-\infty$ | ∞ ∞ ∞ ∞ - - -
$-\infty$ | P
$-\infty$ | A
AC
ACK
ACKM
A~~C~~KMY
B → ABKMY
ABKMO
ABKMOX
ABKMOW

ABIMOW

Every row LowIS[i, ...] is sorted

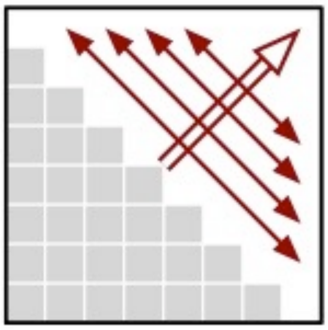
Replace linear search with binary search

$\boxed{O(n \log n) \text{ time}}$



```
OptimalBST(f[1..n]):
    InitF(f[1..n])
    for i ← 1 to n+1
        OptCost[i, i−1] ← 0
    for d ← 0 to n−1
        for i ← 1 to n−d        ⟨⟨…or whatever⟩⟩
            ComputeOptCost(i, i+d)
    return OptCost[1, n]
```

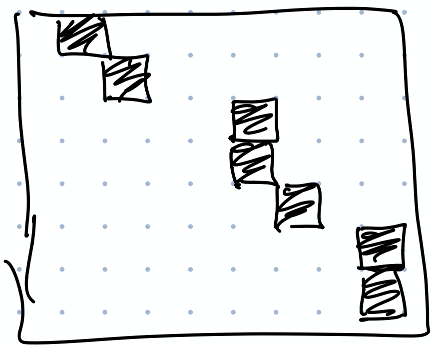
```
ComputeOptCost(i, k):
    OptCost[i, k] ← ∞
    for r ← i to k
        tmp ← OptCost[i, r−1] + OptCost[r+1, k]
        if OptCost[i, k] > tmp
            OptCost[i, k] ← tmp
    OptCost[i, k] ← OptCost[i, k] + F[i, k]
```



```
FasterOptimalSearchTree(f[1..n]):
    InitF(f[1..n])
    for i ← 1 downto n
        OptCost[i, i−1] ← 0
        OptRoot[i, i−1] ← i
    for d ← 0 to n
        for i ← 1 to n
            ComputeCostAndRoot(i, i+d)
    return OptCost[1, n]
```

```
ComputeCostAndRoot(i, j):
    OptCost[i, j] ← ∞
    for r ← OptRoot[i, j−1] to OptRoot[i+1, j]
        tmp ← OptCost[i, r−1] + OptCost[r+1, j]
        if OptCost[i, j] > tmp
            OptCost[i, j] ← tmp
            OptRoot[i, j] ← r
    OptCost[i, j] ← OptCost[i, j] + F[i, j]
```

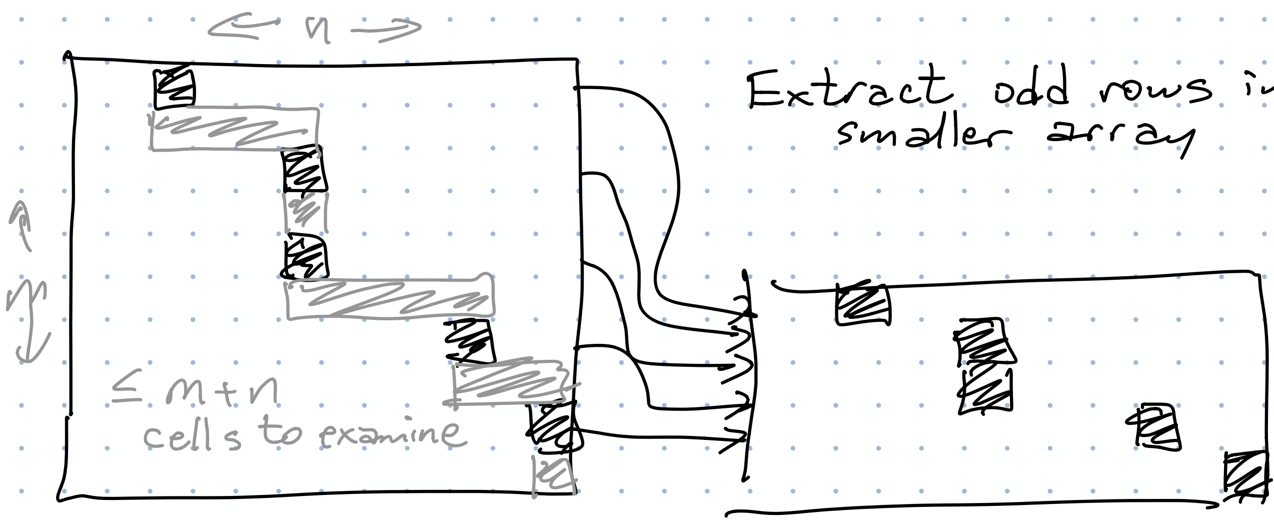
Monotone Array = leftmost smallest elements in each row move to right in later rows



$LM[i]$ = smallest index of smallest value in row i

$LM[i] \le LM[i+1]$ for all i

Find smallest element in every row



$$T(m,n) = T\left(\frac{m}{2}, n\right) + O(m+n)$$

$$= O(m + n\cancel{\log m})$$