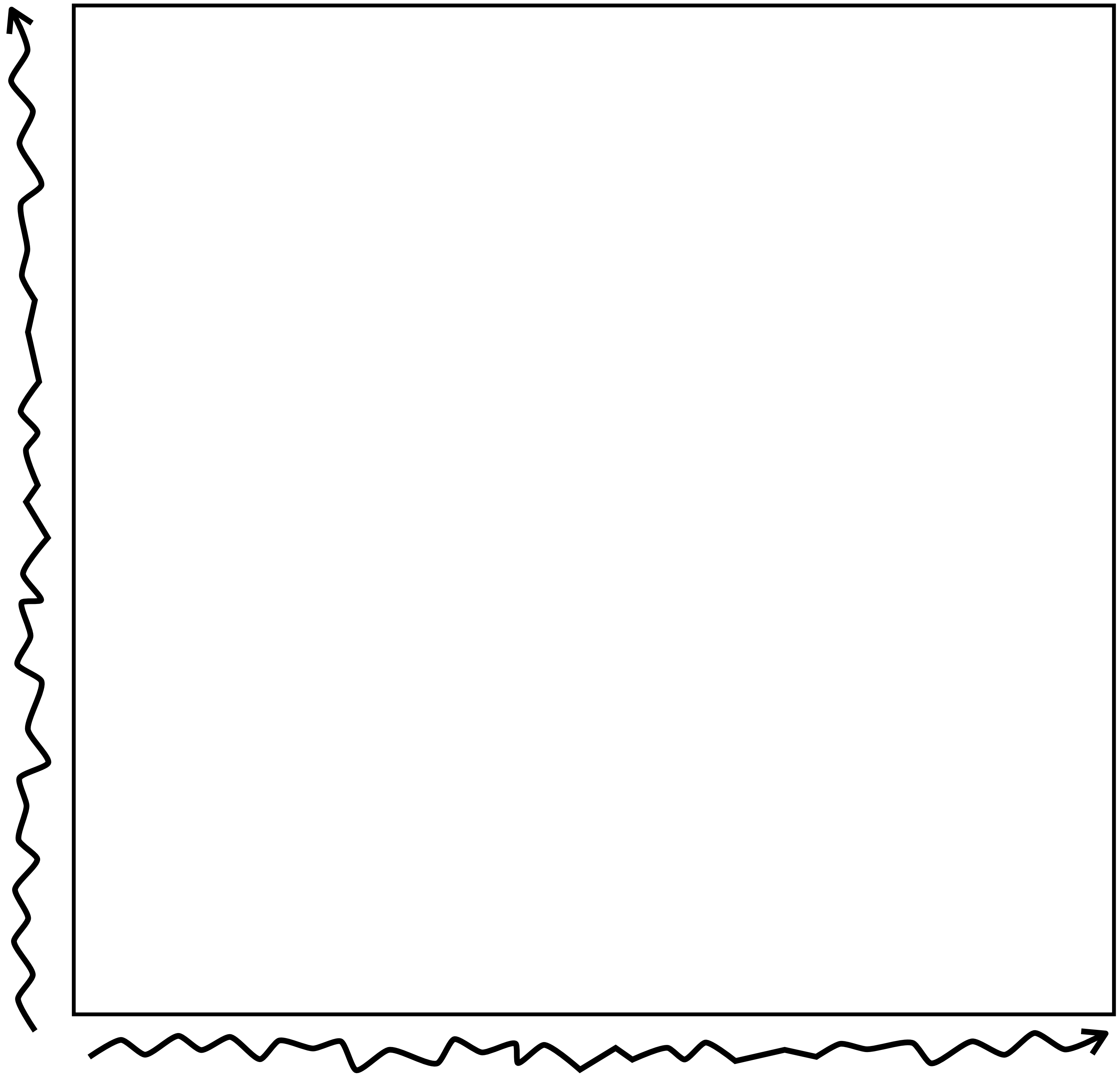
Four Russians' Speedup

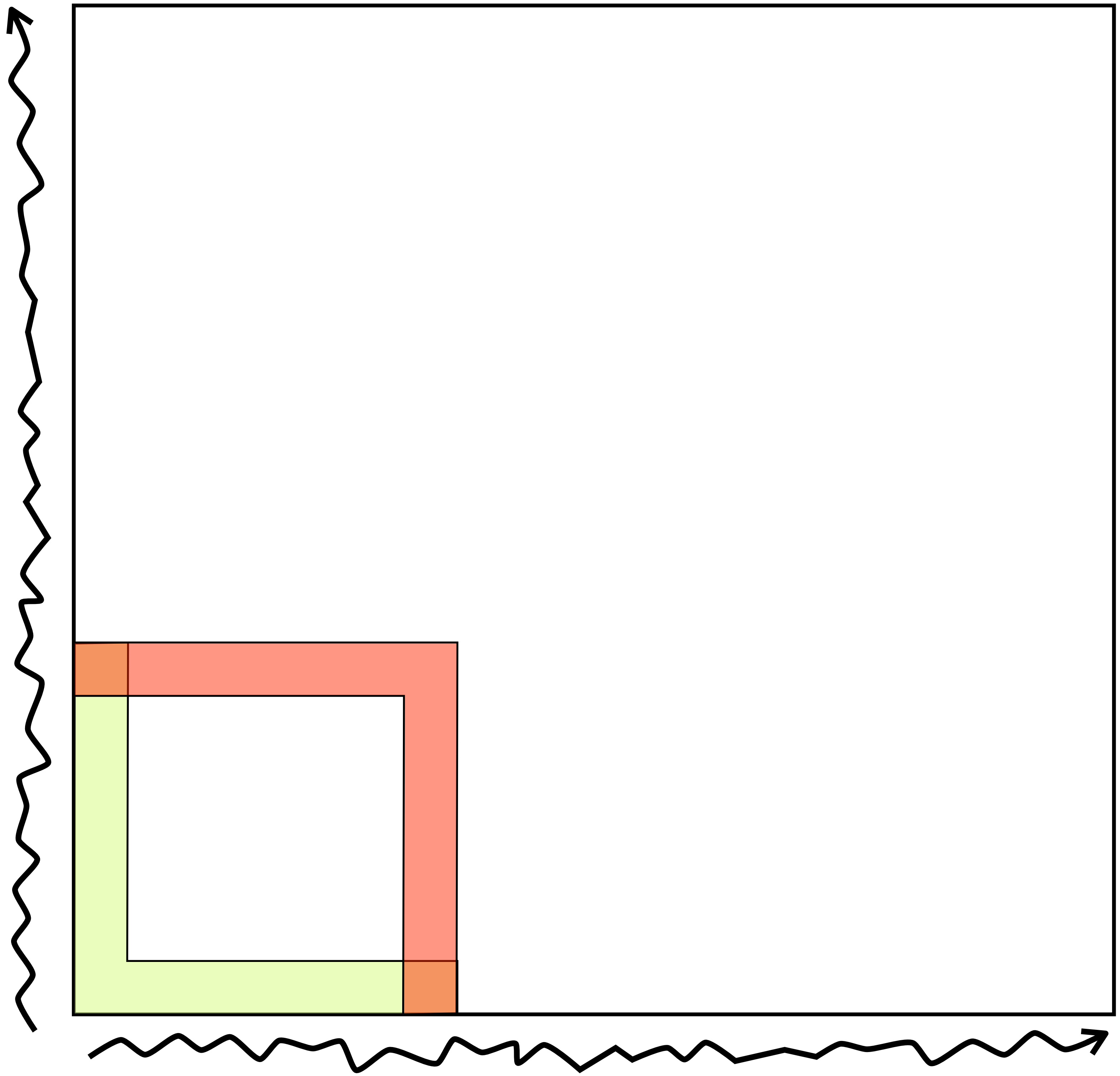
02-714

Slides by Carl Kingsford

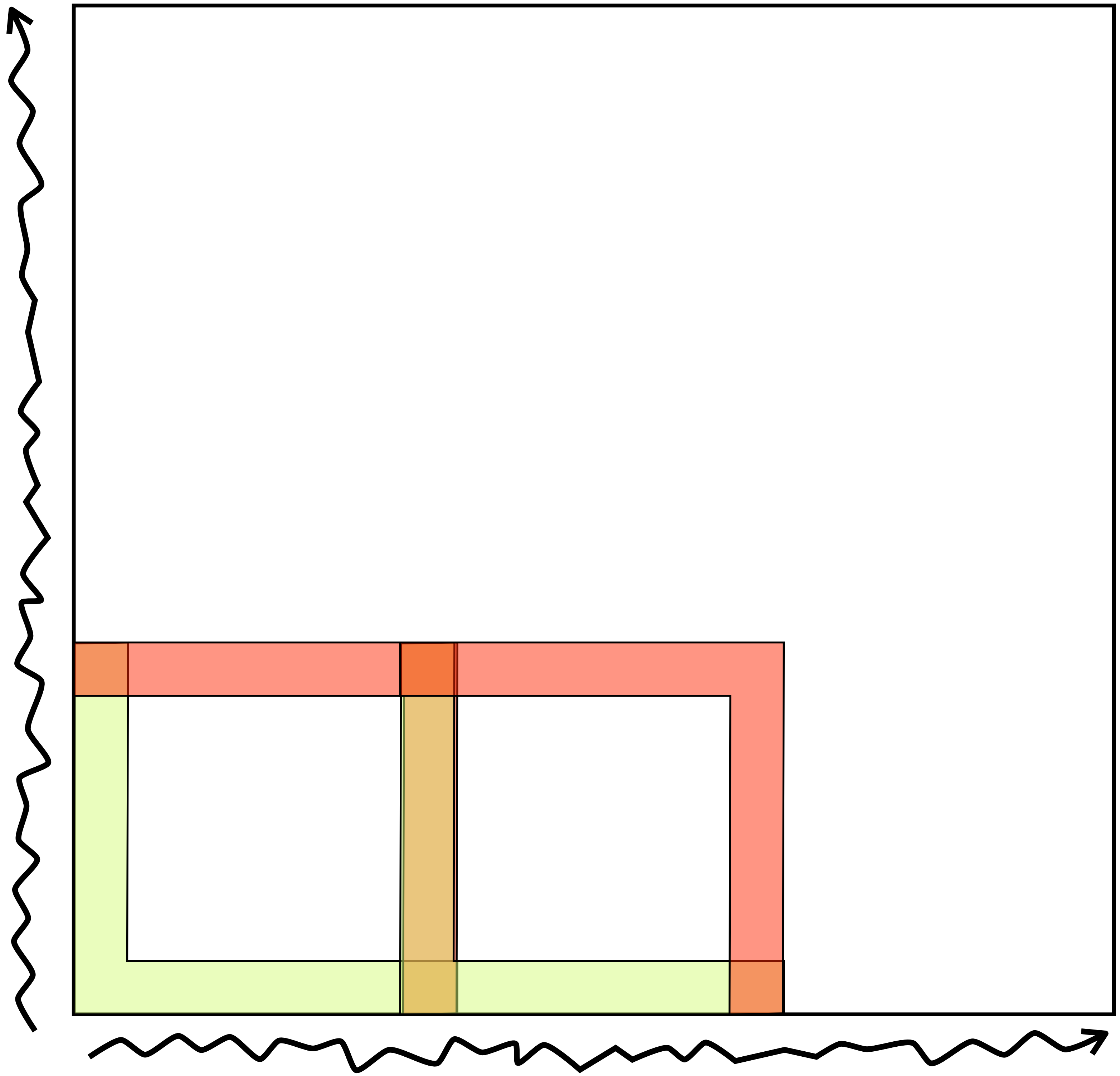
Block Edit Distance



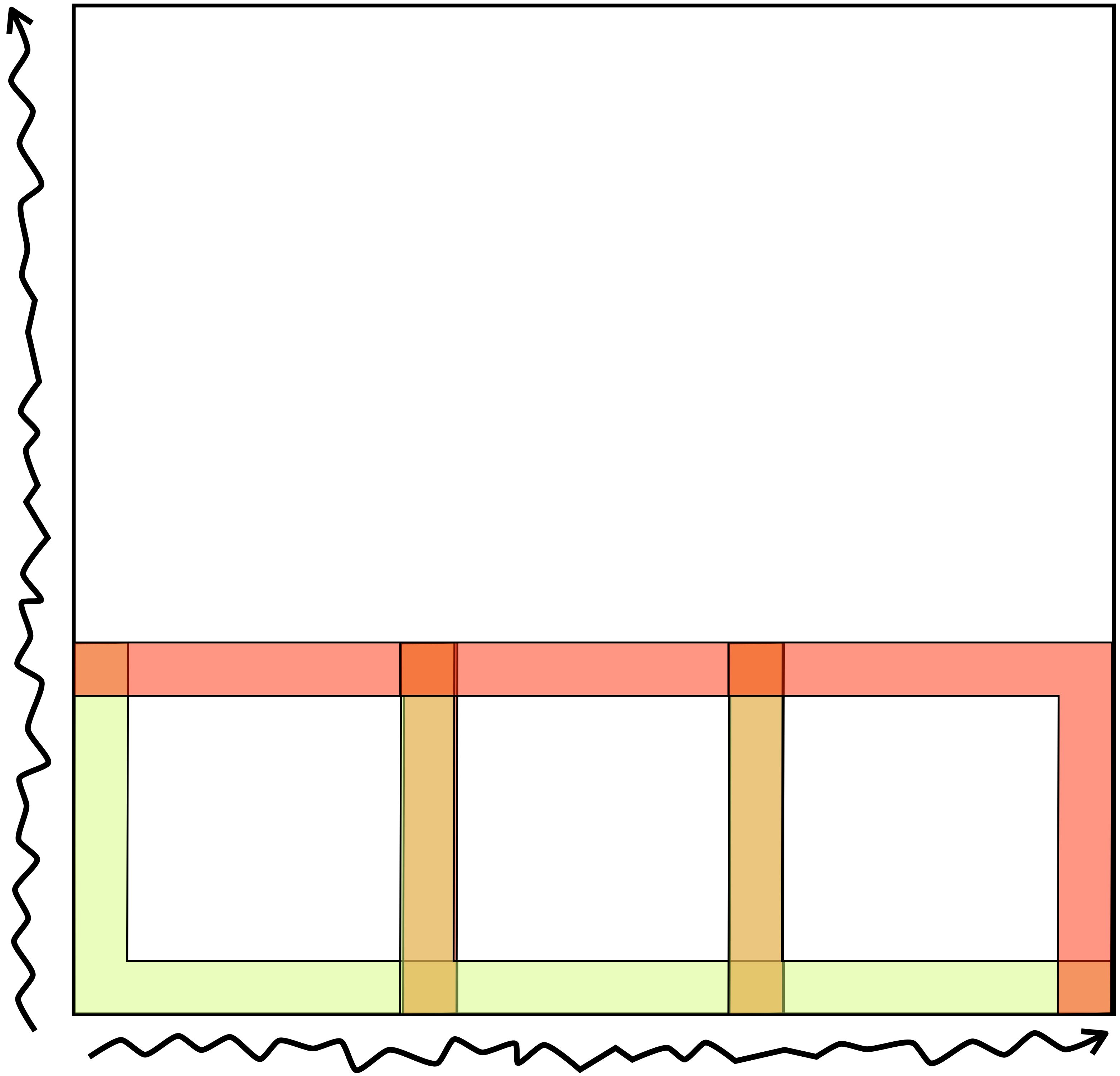
Block Edit Distance



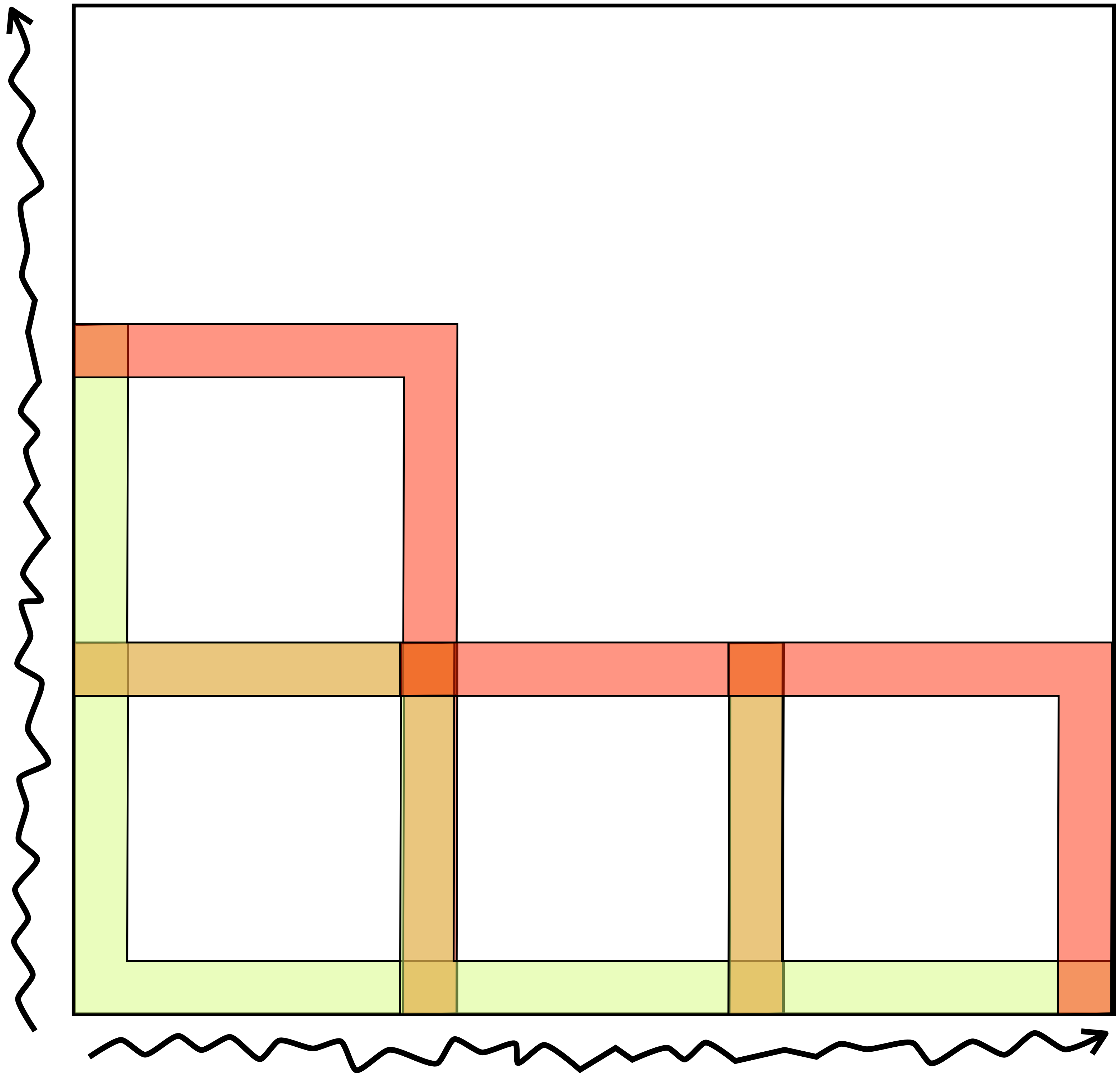
Block Edit Distance



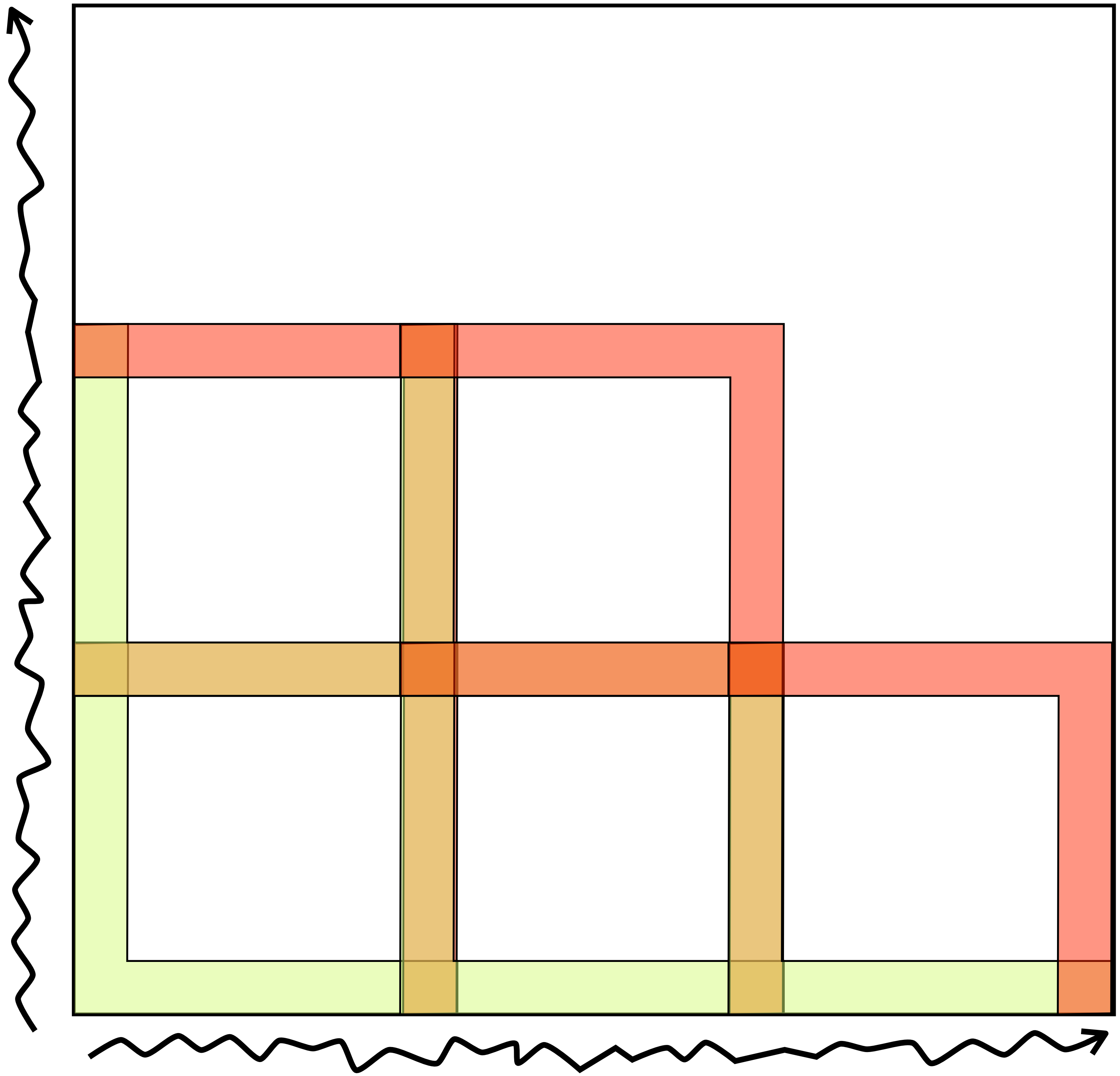
Block Edit Distance



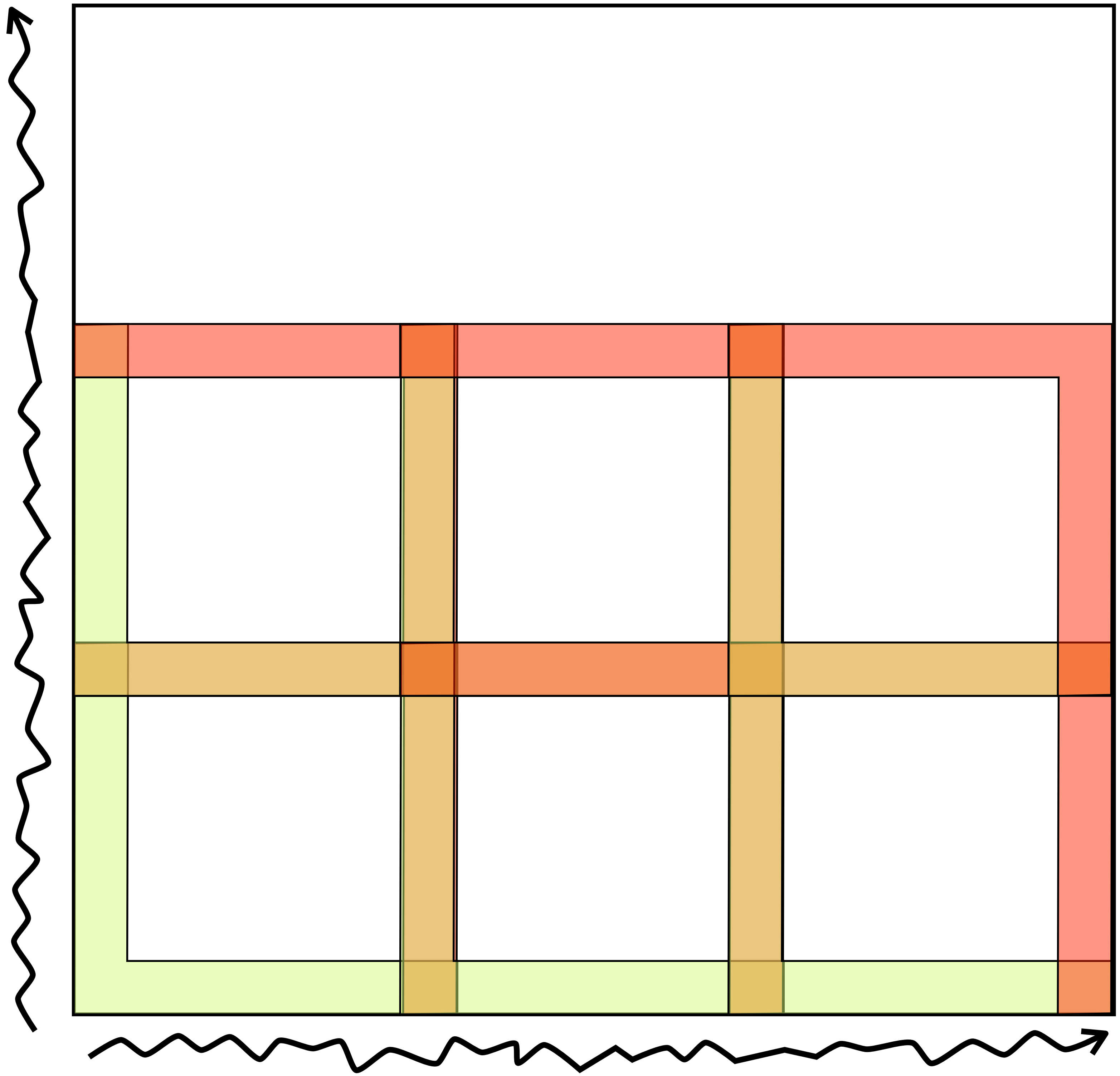
Block Edit Distance



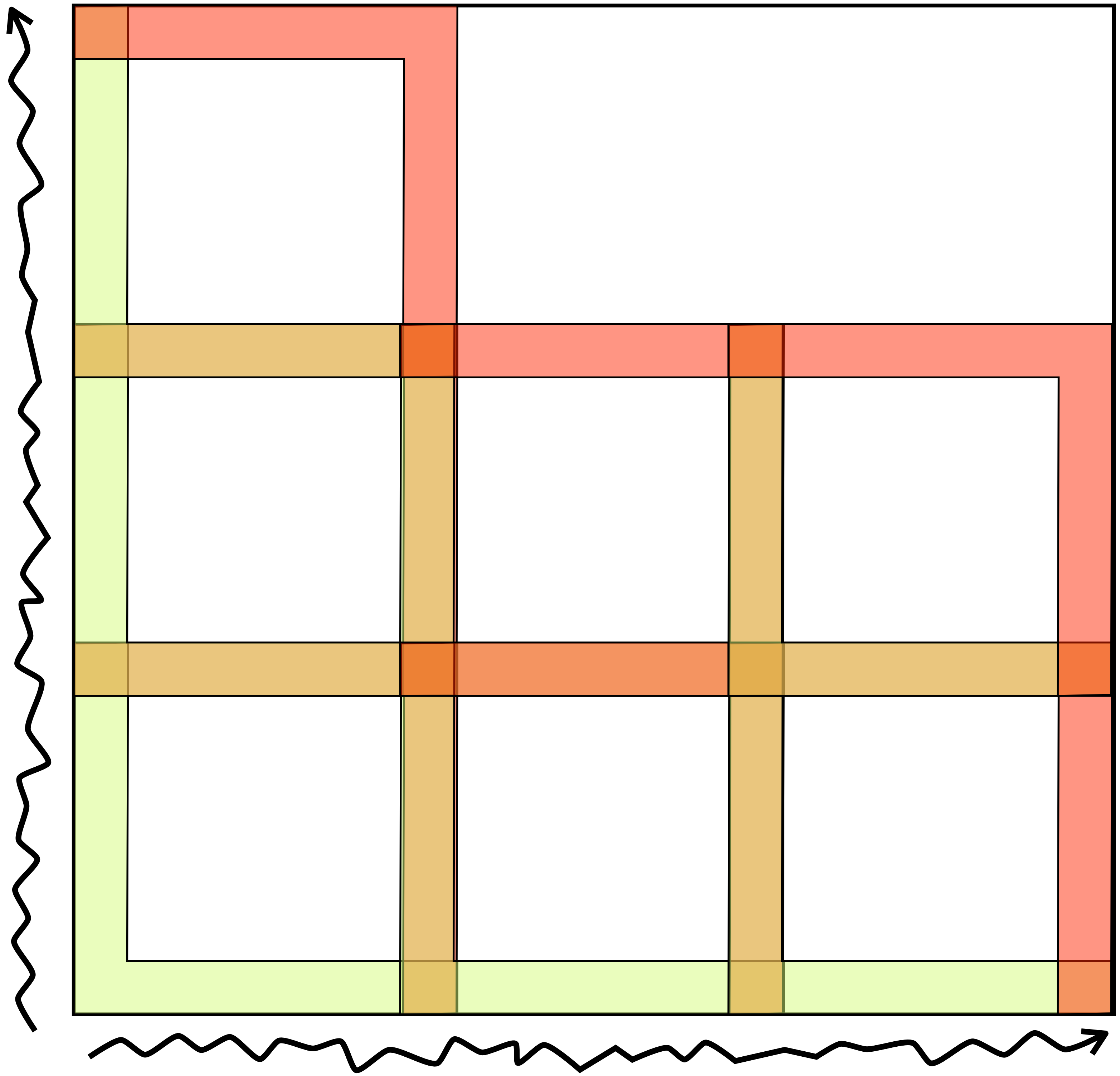
Block Edit Distance



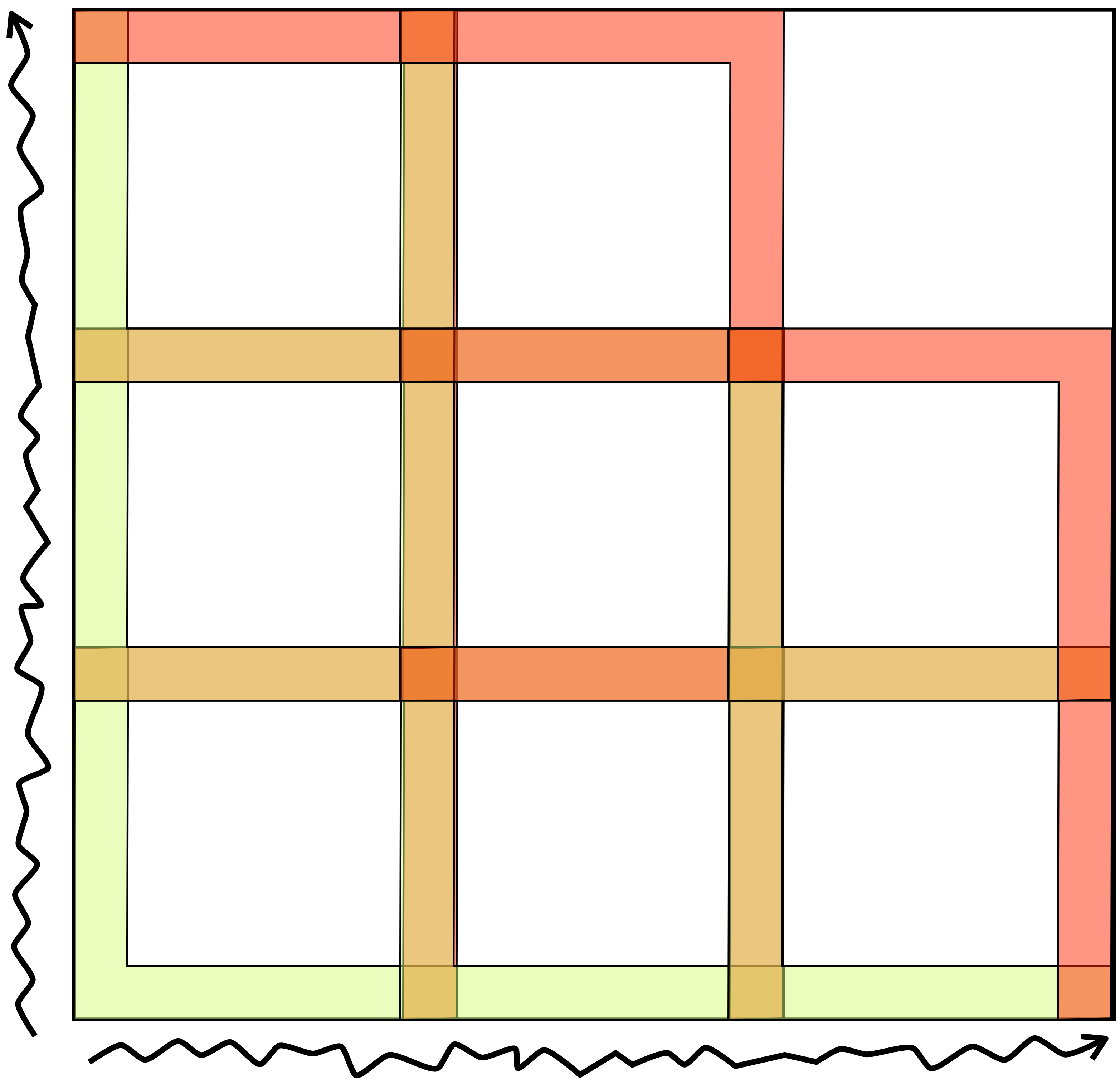
Block Edit Distance



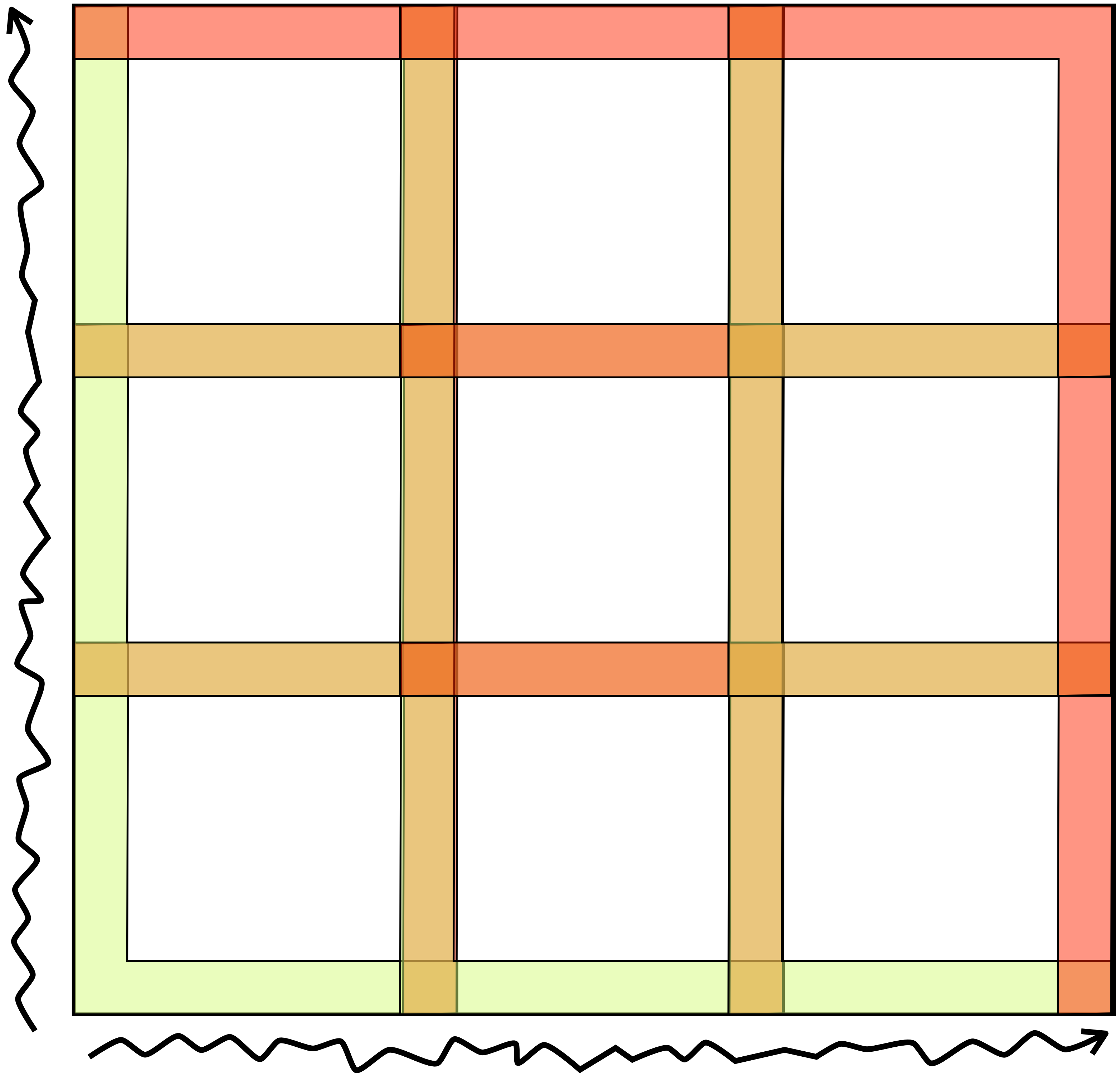
Block Edit Distance



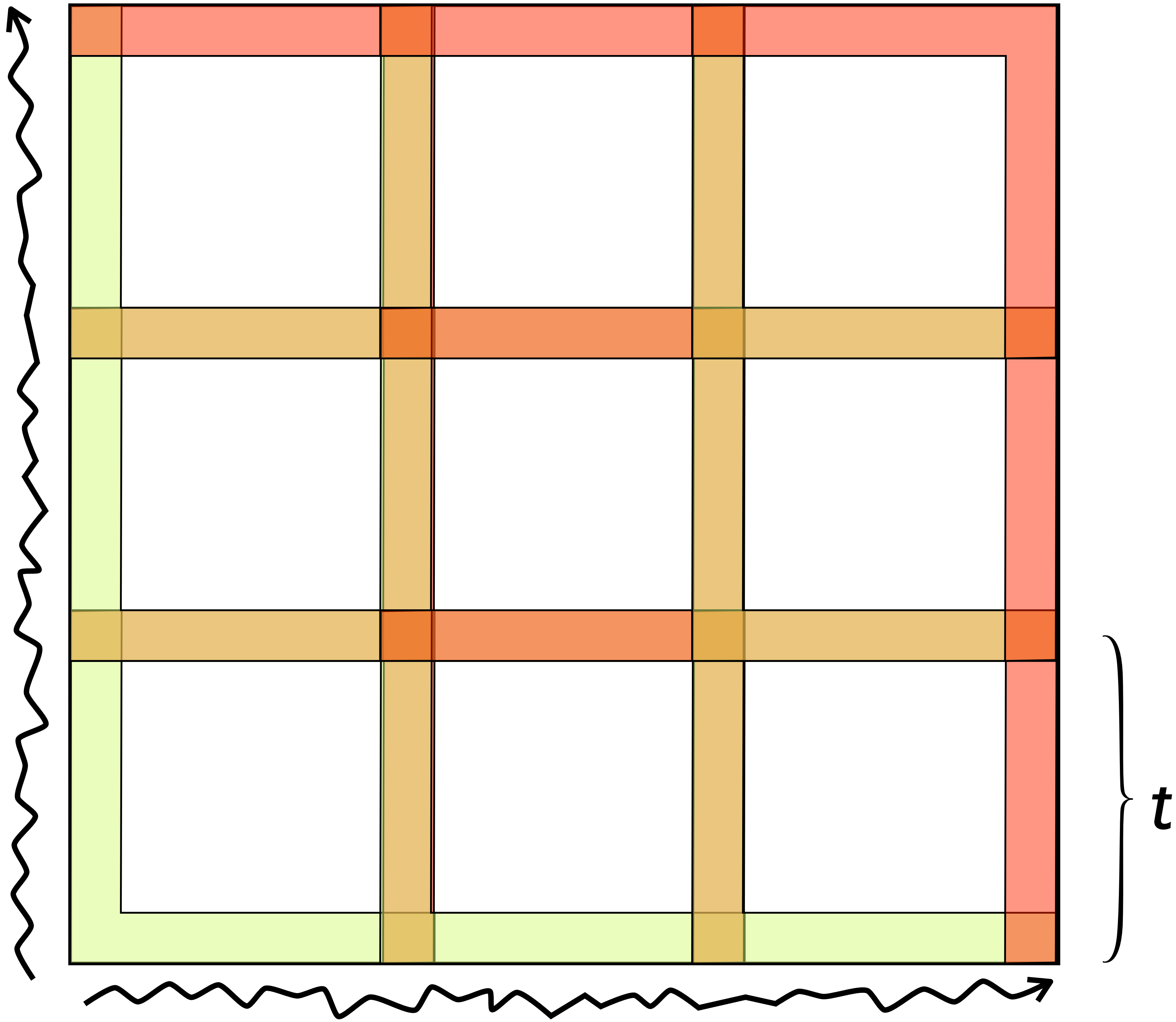
Block Edit Distance



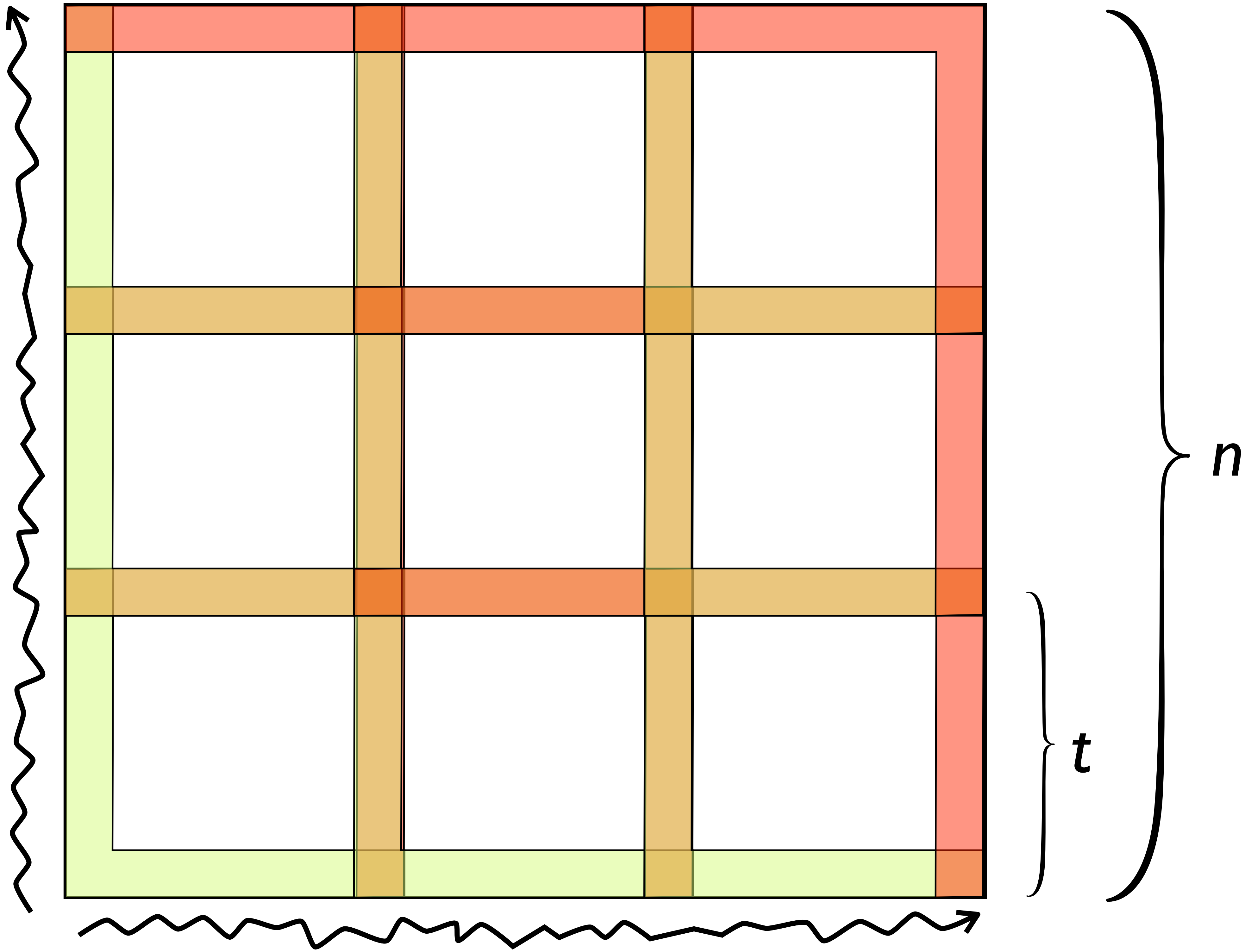
Block Edit Distance



Block Edit Distance

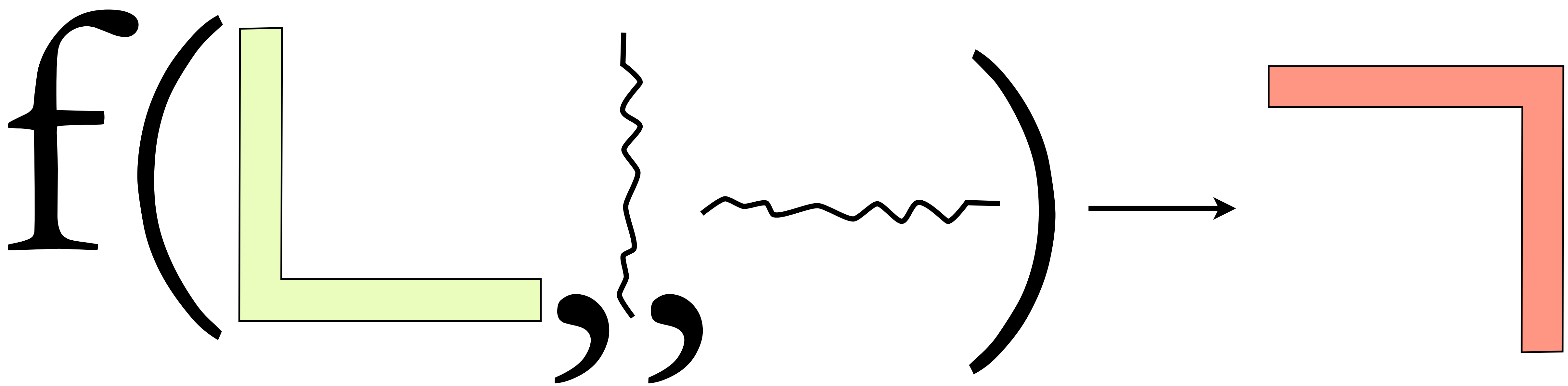


Block Edit Distance



Block Function

Assume we have a function of the following form:



If we can compute *f* faster than O(t^2), we win.

We will see how to compute it in O(t) time.

Assumptions

We’re computing the plain edit distance: gaps and mismatches cost 1 and matches cost 0.

The alphabet Σ is a constant size.

$n = k(t-1)$ for some k (that is the blocks perfectly tile the matrix, with a single overlapping row and column between each adjacent pair)

Precomputing f

The way we compute f fast is to precompute f(x) for all possible x = (∟ , 〰, ⌇).

How may different x values are there?

$$\underbrace{(n+1)^{2t}}\underbrace{|\Sigma|^{2t}}$$

Every cell contains a number between 0 and n.

This many pairs of strings, each of length t.

Computing each would take $O(t^2)$ time, taking in total $O((n+1)^{2t}|\Sigma|^{2t}t^2) = O(n^2)$ time. **Bad!**

Offset Encoding

The trick to making it work is realizing that in fact there are fewer possible functionally different inputs to x.

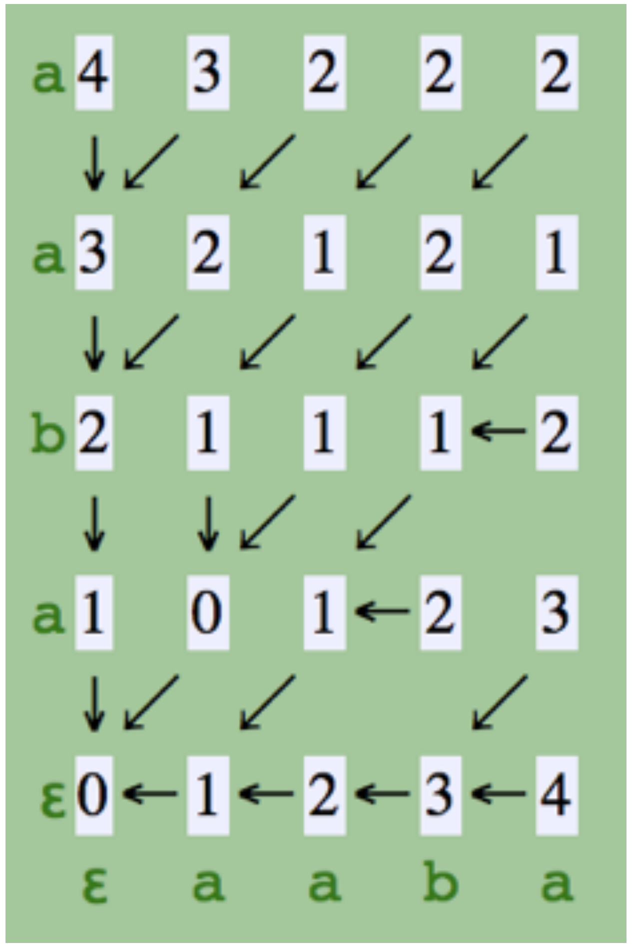
The elements of the rows and columns in the input are not independent.

Notation. D is the matrix and D(i,j) is the value at position i,j.

Lemma. Adjacent values of D in a row, column, or diagonal differ by at most 1.

Consider element q of row i:

- D(i,q) ≤ D(i,q-1)+1 because we can always insert a gap if we wanted to.
- Suppose we throw away character q to consider D(i, q-1):
  - If character q is matched, the edit distance increases by ≤ 1 (we can align what is was matched to against a gap): D(i,q-1) ≤ D(i,q)+1
  - If character q is not matched, the edit distance goes down (by 1 since we eliminate a gap): D(i,q-1) ≤ D(i,q)
  - Therefore: D(i,q-1) -1 ≤ D(i,q)



Offset Encoding, II

Can encode a row of the matrix as an initial value plus a sequence of -1,0,1:

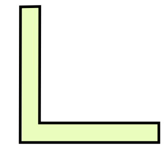
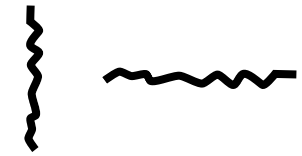
Example. 567767 → 5 1 1 0 -1 1

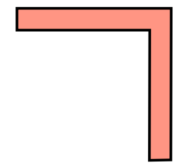
Definition. An *offset vector* is the encoding of a row or column as above, except that the first entry is set to 0.

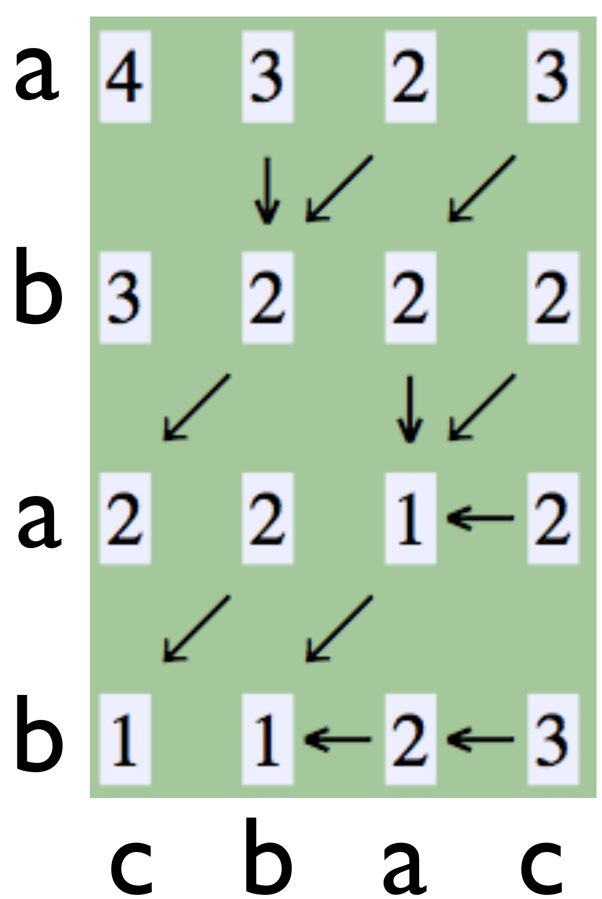
Example. 567767 → 0 1 1 0 0 -1 1

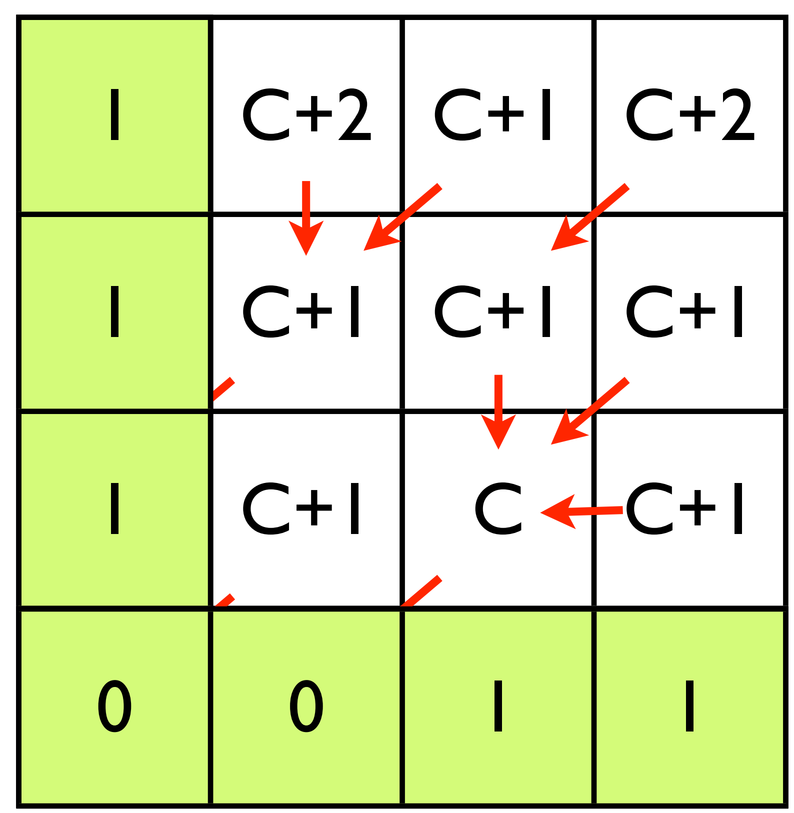
So: given the first value C and the offset vector, you can reconstruct the row or column.

Offset Encoding, III

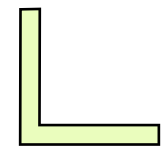
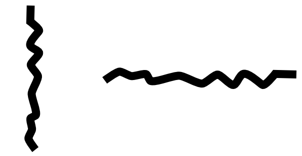
Thm. Given only the offset vectors of  and 

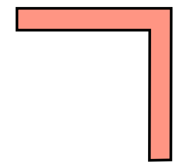
one can compute the offset vectors of 

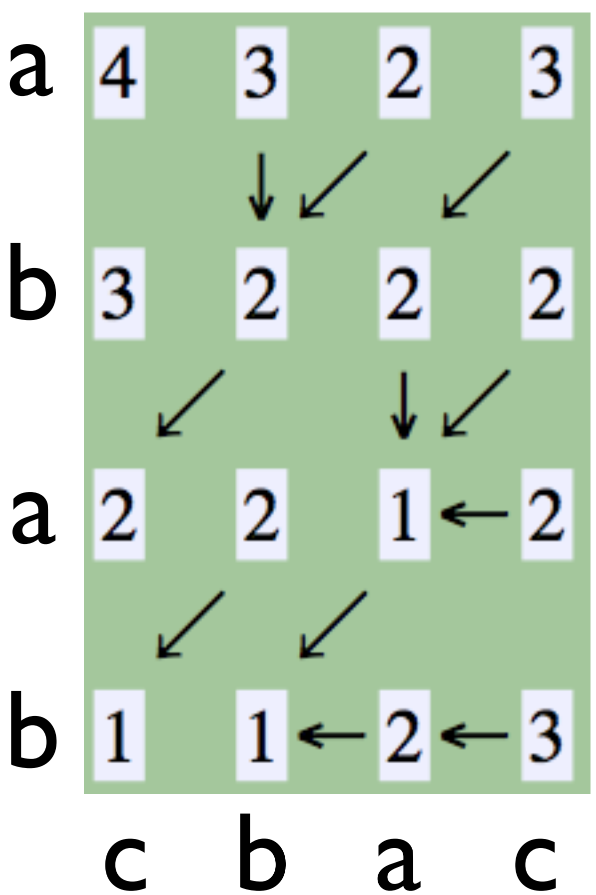


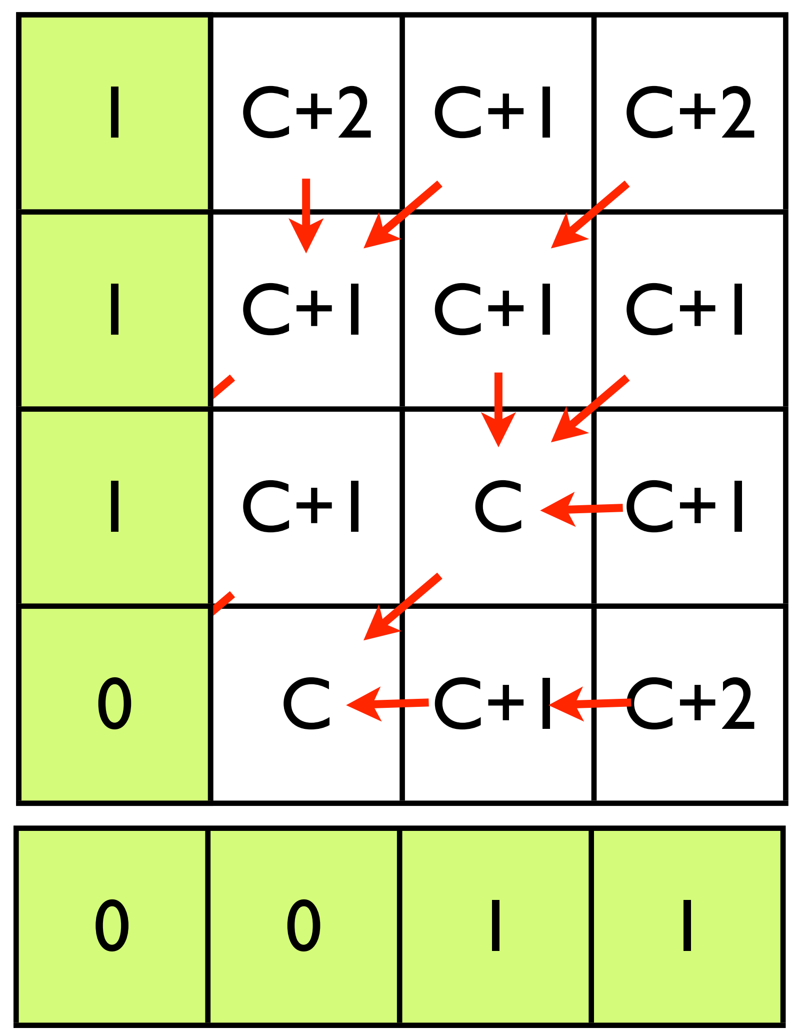


Offset Encoding, III

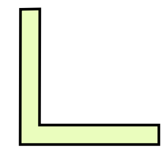
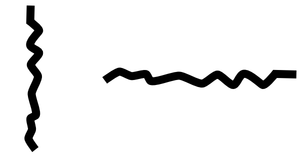
Thm. Given only the offset vectors of  and 

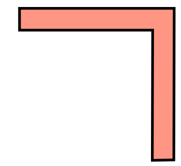
one can compute the offset vectors of 

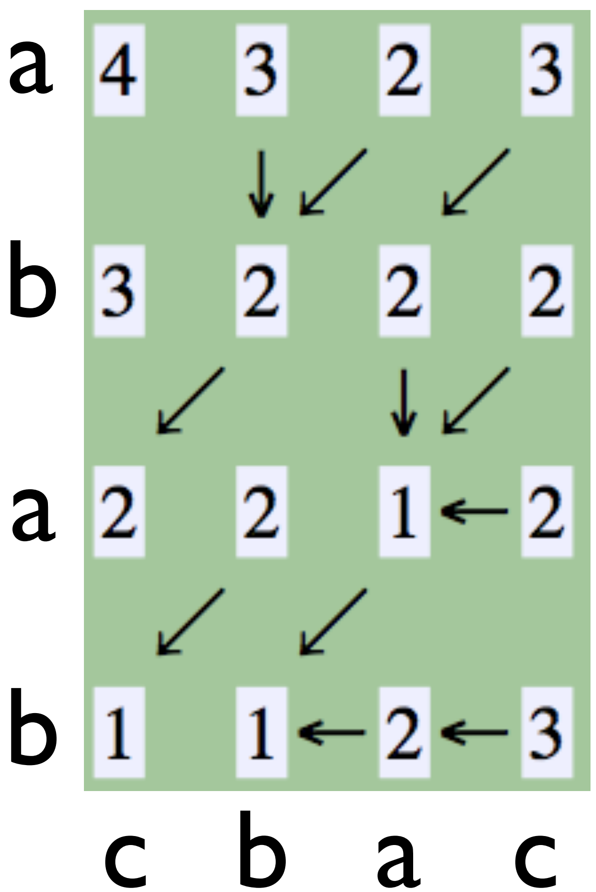


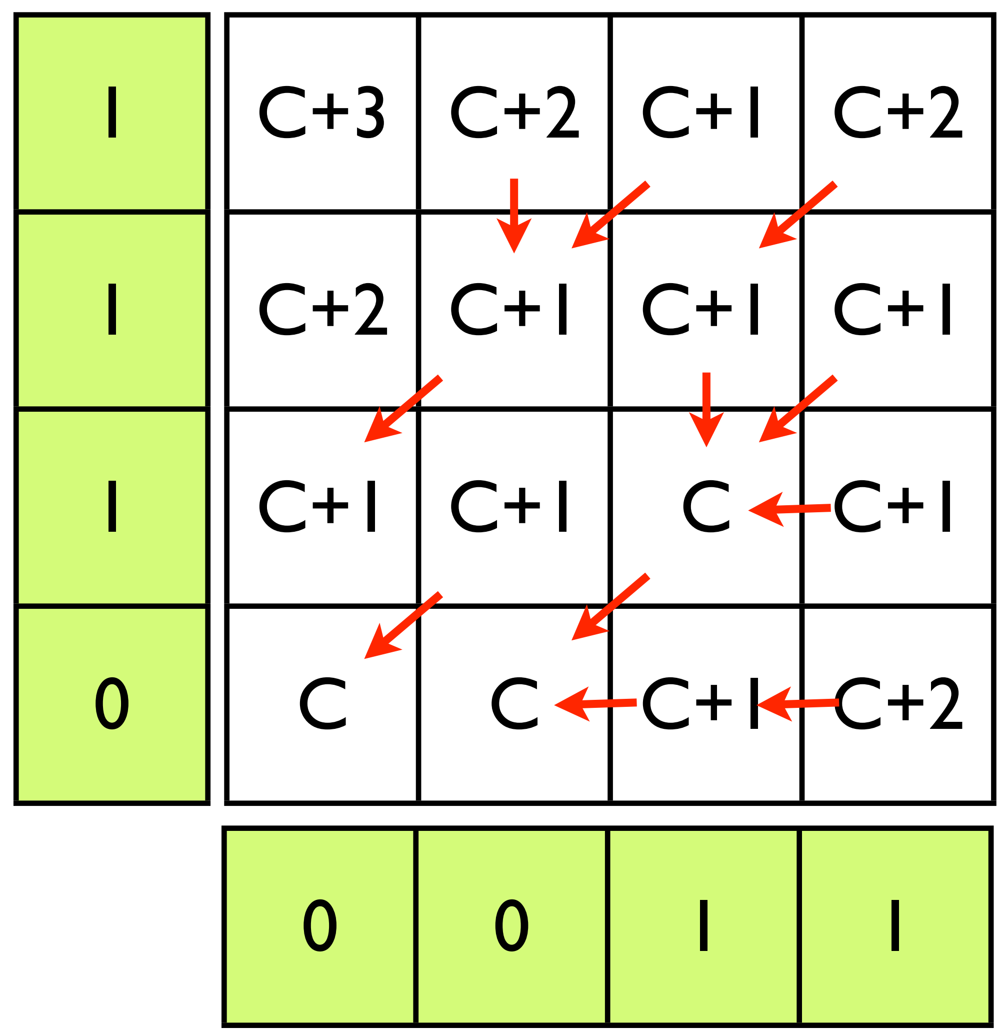


Offset Encoding, III

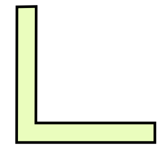
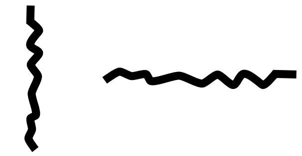
Thm. Given only the offset vectors of  and 

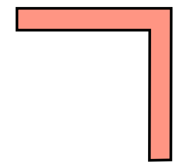
one can compute the offset vectors of 

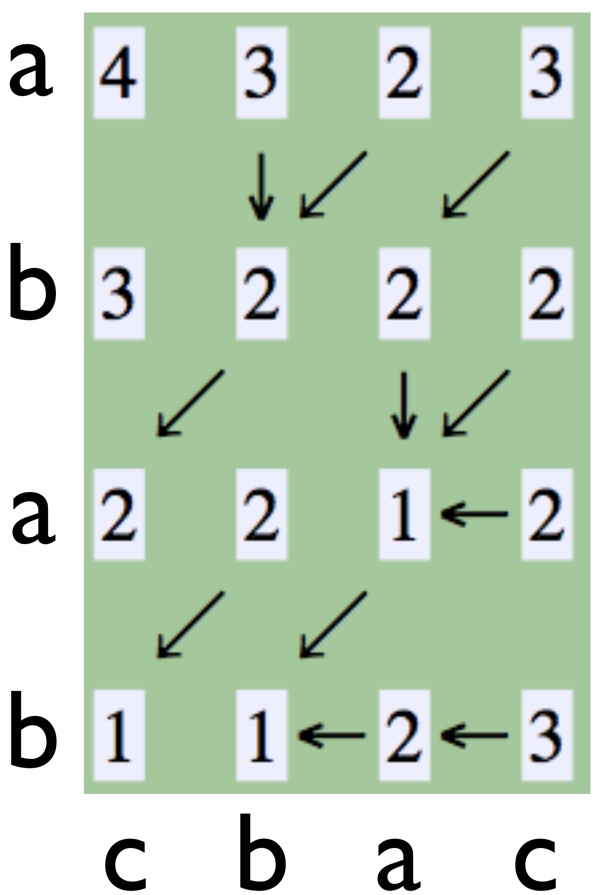


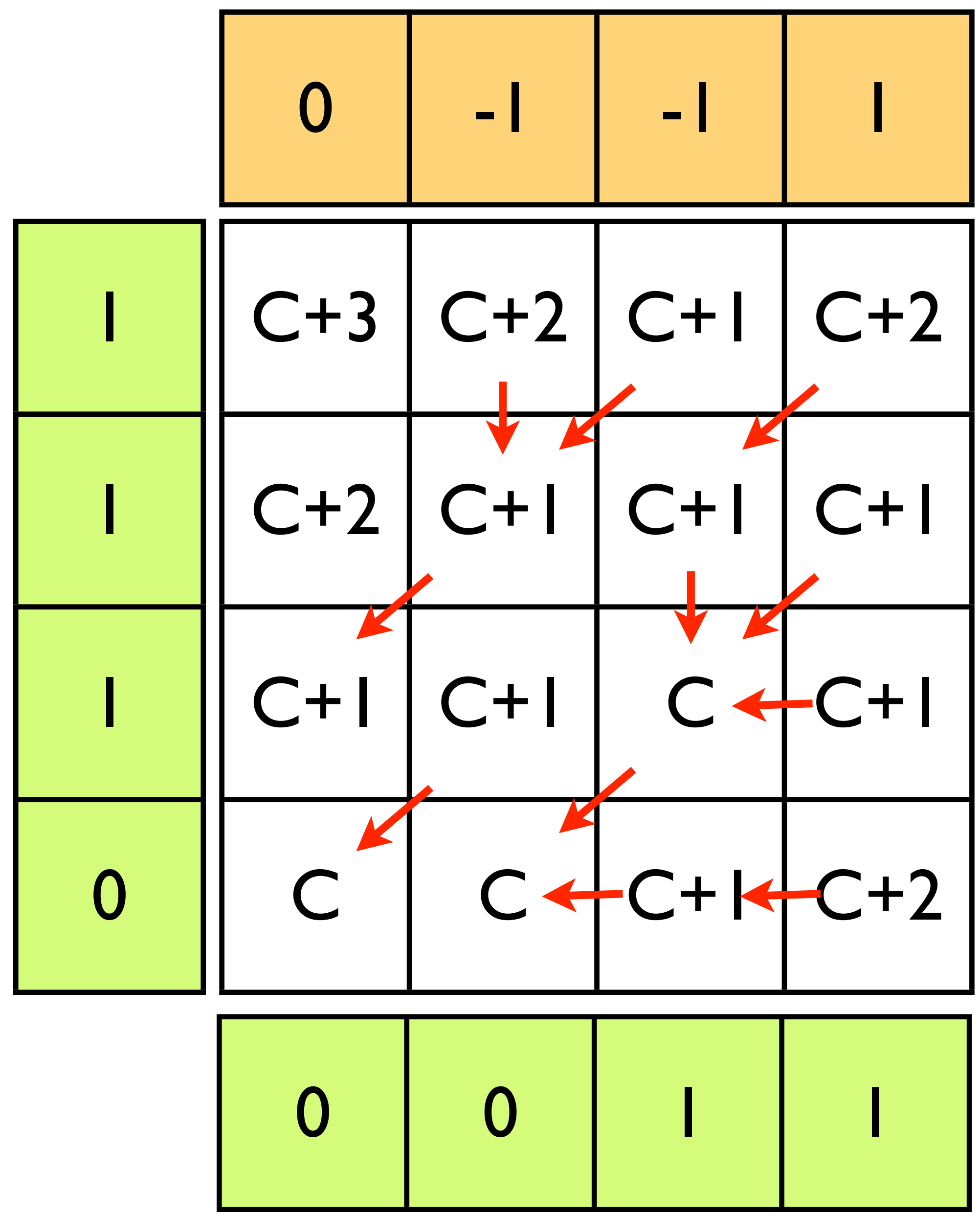


Offset Encoding, III

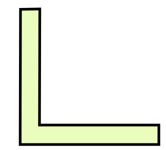
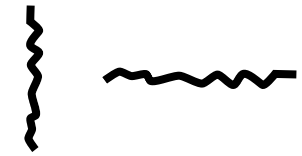
Thm. Given only the offset vectors of  and 

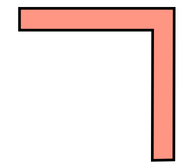
one can compute the offset vectors of 

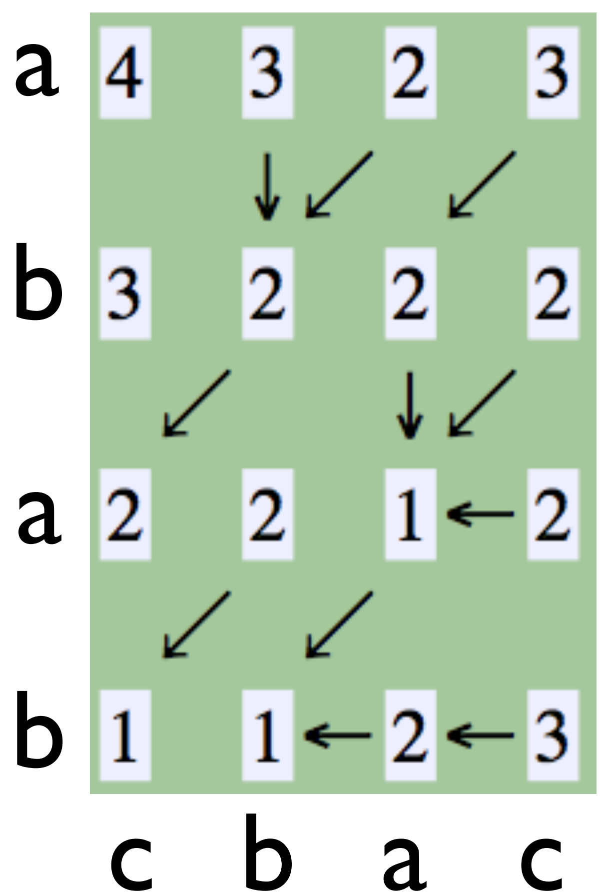


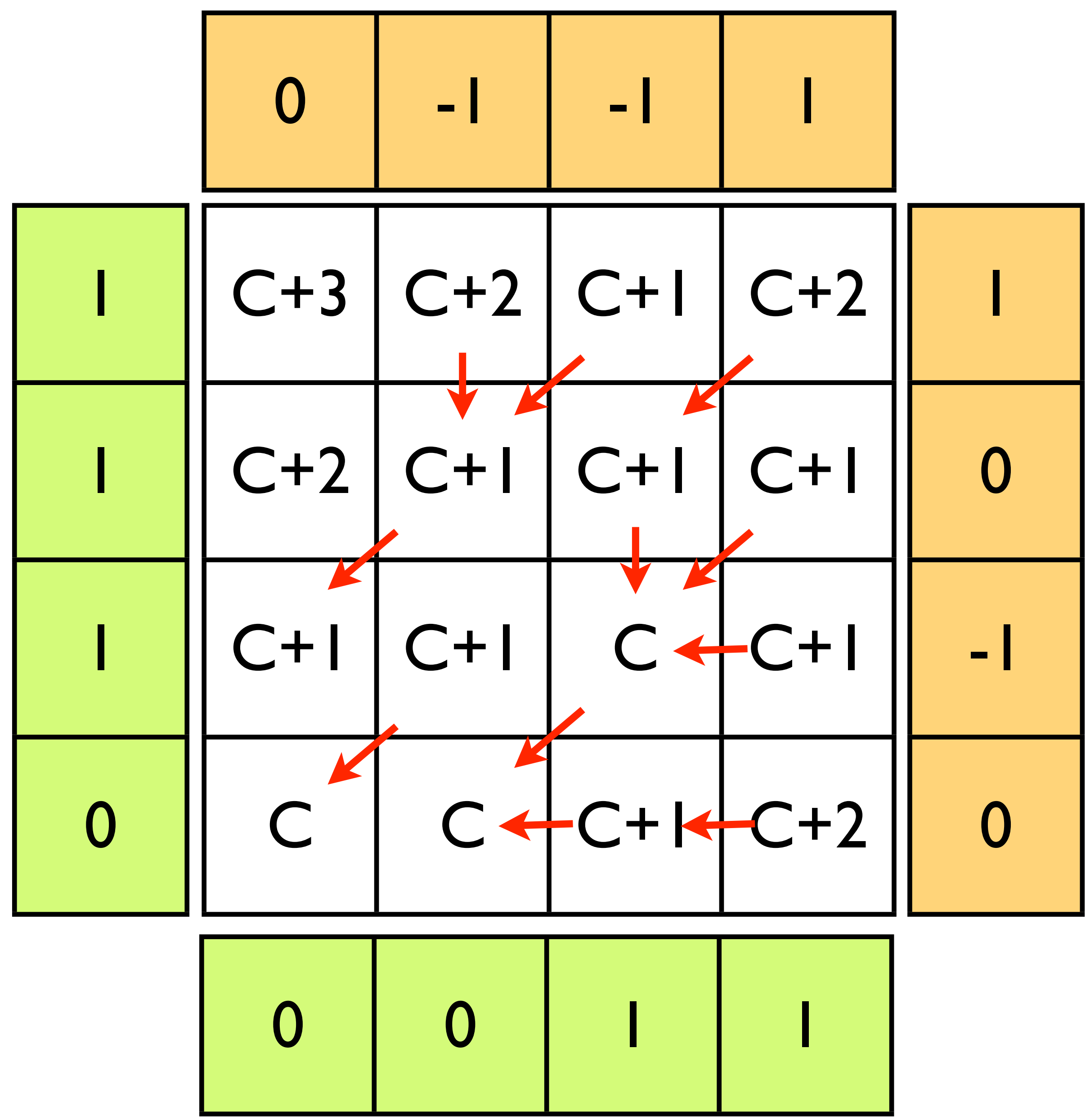


Offset Encoding, III

Thm. Given only the offset vectors of  and 

one can compute the offset vectors of 





Preprocessing Time

There are $2^{2(t-1)}$ offset vectors.

There are $2^{2(t-1)}|\Sigma|^{2t}$ possible inputs x to f.

Computing all values of f(x) takes now time $O((2|\Sigma|)^{2t}\, t^2)$.

Setting $t = \log_{2|\Sigma|} n$, this becomes $O(n(\log n)^2)$

Storing f for quick access

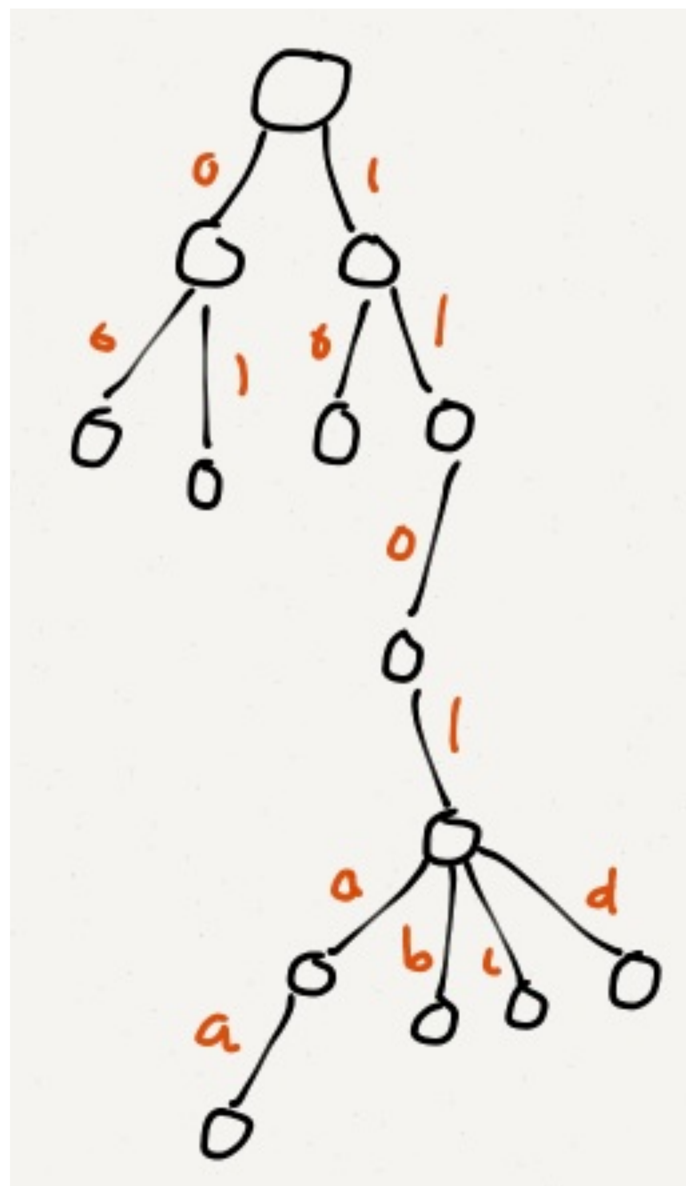
We have $2^{2(t-1)}|\Sigma|^{2t}$ possible inputs x to f.

How do we store the values f(x) so we can access f(x) in time O(t)?

Storing f for quick access

We have $2^{2(t-1)}|\Sigma|^{2t}$ possible inputs x to f.

How do we store the values f(x) so we can access f(x) in time O(t)?



Depth $\approx 3t = O(t)$

Total Running time

We have $O(n^2 / t^2)$ blocks to compute.

Accessing f(x) for each takes time O(t), so our time to "fill in" the matrix is $O(tn^2/t^2) = O(n^2/t)$

With t = O(log n) the total time is:

$$O(n^2 / \log n + n(\log n)^2) = O(n^2 / \log n)$$ **FTW!**

(In the RAM model, where we can access things of size log n in constant time, we get the even better time of $O(n^2 / \log^2 n)$)

In Practice

Often useful to take t = some constant instead of log n.

Doesn't give you an asymptotic speed up, but now runs in time $O(n^2 / t)$ so the constant factor is better.