Bioinformatics

Extentions

- Longest Common Subsequence (match weight 1, otherwise 0)
- End-space free variant that encourages one string to align in the interior of the other, or the suffix of one to align with the prefix of the other (initial conditions 0, everything is countable in the last row and the last column) – shotgun sequence assembly
- Approximate occurrence of P in T (the optimal alignment of P to a substring of T has distance δ from the optimal alignment) (initial conditions 0). V(0,j)=0.

  -- locate a cell (m,j) with value greater than δ.

  -- traverse backpointers from (m,j) to (0,k).

  -- occurrence in T[k,j]

Algorithms on Strings, Trees and Sequences, D. Gusfield, Cambridge University Press, 10th edition 2007

1. **Map longest common subsequence to longest increasing subsequence**

Let us consider the two sequences S1 and S2 whose maximum common subsequence we want to find. Suppose the S2 is smaller in size than the S1. For each x character in the alphabet that appears at least once in S1, create a list of the locations where the x character appears in the S2 string. Write this list upside down.
The longest common subsequence problem (finding the maximum common subsequence of S1 and S2) is reduced to a problem of finding the longest ascending subsequence in the list.

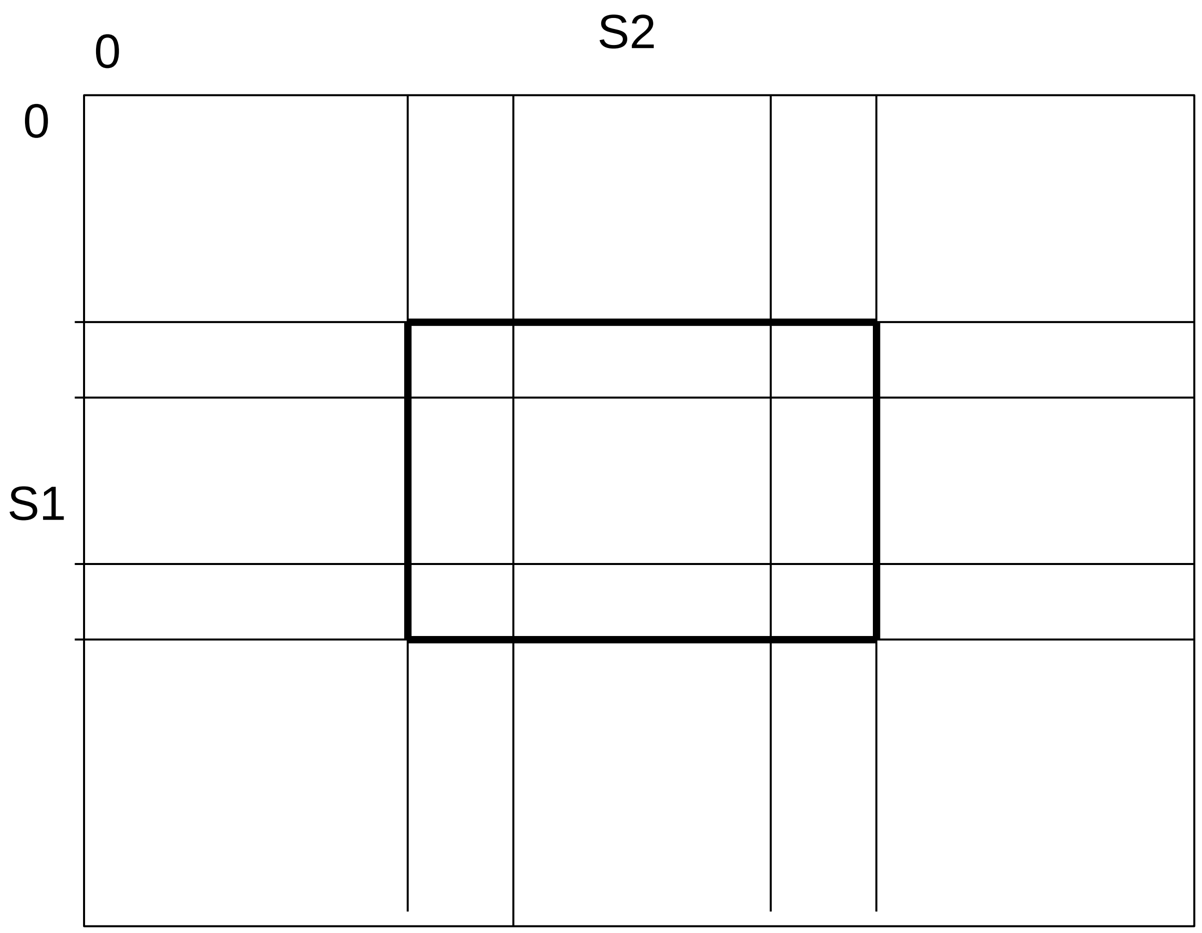
2. Compute the Longest increasing subsequence algorithm

Starting from the left, examine each consecutive number and place it at the end of the first (leftmost) descending subsequence that can expand. If there are no descending subsequences that can be expanded, then start a new (descending) subsequence to the right of all existing descending subsequences.

Interesting paper: Maxime Crochemore, Ely Porat: Fast computation of a longest increasing subsequence and application. Inf. Comput. 208(9): 1054-1059 (2010)

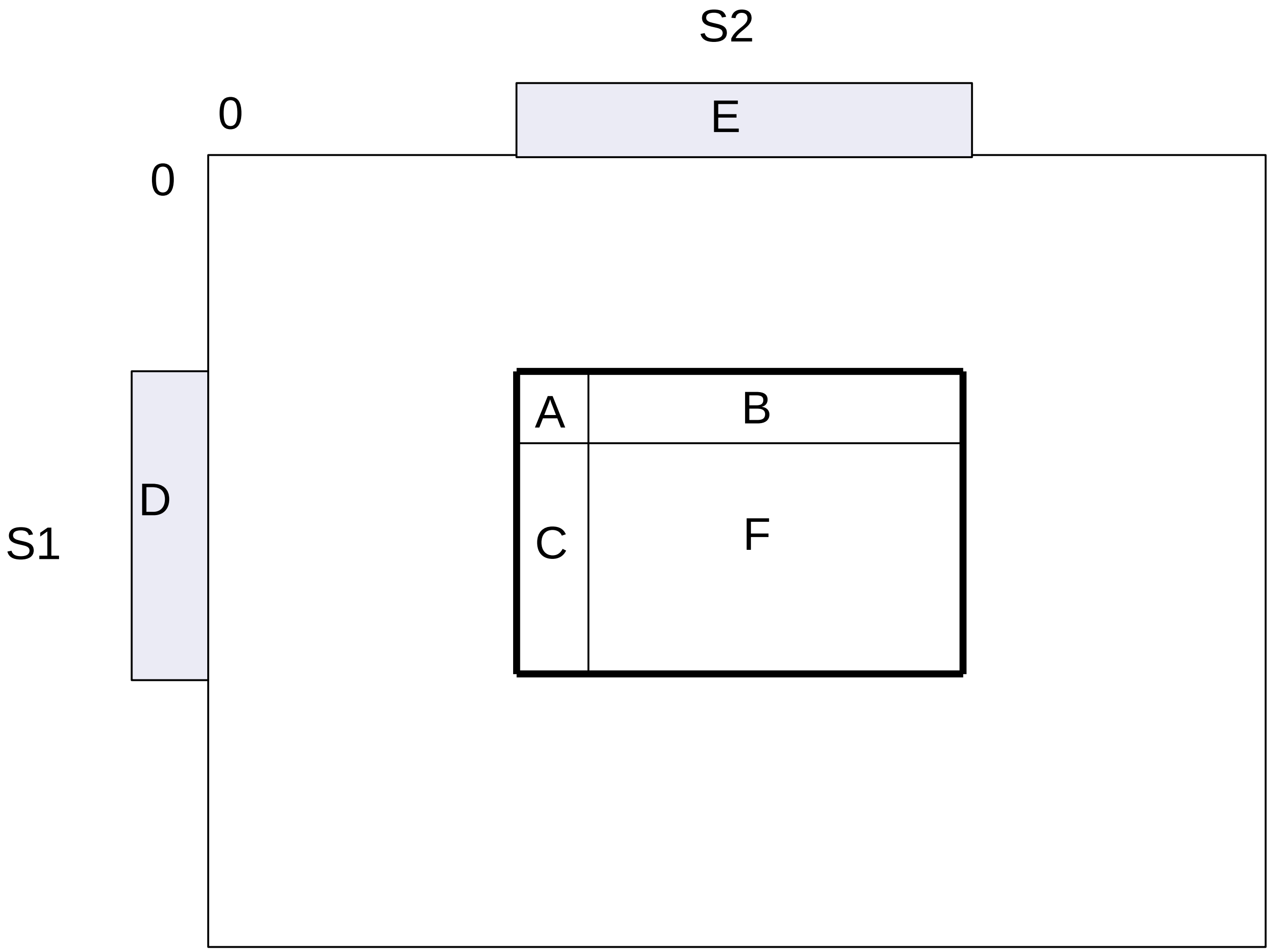
4-Russians speedup
(Alizarin, Dinic, Kronrod, and Faradzev)

- *Arlazarov, V.; Dinic, E.; Kronrod, M.; Faradžev, I. (1970), "On economical construction of the transitive closure of a directed graph", Dokl. Akad. Nauk SSSR, **194** (11)*. Original title: "Об экономном построении транзитивного замыкания ориентированного графа", published in *Доклады Академии Наук СССР* **134** (3), 1970.


0
S2
0
S1

The Four-Russians speedup-t-blocks

- Definition: A t-block is a t by t square in the dynamic programming table.
- The rough idea of the Four-Russians method is to partition the dynamic programming table into t-blocks and compute the essential values in the table one t-block at a time, rather than one cell at a time.
- Lemma: The distance values in a t-block starting in position (i,j) are a function of the values in its first row and column and the substrings S_1[i…i+t-1] and S_2[j..j+t-1].
- Definition: Given above lemma and the following figure, we define the block function as the function from the five inputs (A,B,C,D,E) to the output F
- The values in the last row and column of a r-block are also a function of the inputs (A, B , C, D, E). We call the function from those inputs to the values in the last row and column of a r-block, the restricted block function.


S2
0
E
0
A
B
D
S1
C
F

Accounting detail

- block edit distance algorithm
  - the sizes of the input and the output of the restricted block function are both $O(t)$. There are $\Theta(n^2/t^2)$ blocks, hence the total time used by the block edit distance algorithm is $o(n^2/t)$. Setting t to $\Theta(\log n)$, the time is $O(n^2/\log n)$. However in the unit-cost RAM model of computation, each output value can be retrieved in constant time since $t=O(\log n)$. =>the method is reduced to $O(n^2 /(\log n)^2)$.
- precomputation time
  - The key issue involves the number of input choices to the restricted block function.
  - Every cell has an integer from zero to n=>$(n+1)^t$ possible values for any t-length row or column.
  - If the alphabet has size σ, then there are σ^t, possible substrings of length t=>#distinct input combinations to the restricted block function: is $(n + 1)^{2t} \sigma^{2t}$
  - For each input, it takes $\Theta(t^2)$ time to evaluate the last row and column of the resulting r-block (by running the standard dynamic program). Thus the overall time used in this way to precompute the function outputs to all possible input choices is $\Theta((n + 1)^{2t} \sigma^{2t} t^2)$.

The trick: offset encoding

- Lemma: In any row, column, or diagonal of the dynamic programming table for edit distance, two adjacent cells can have a value that differs by at most one.
- Definition: The offset vector is a t-length vector of values from {-1,0,1}, where the first entry must be zero.
- Theorem: Consider a t-block with upper left corner in position (i,j). The two offset vectors for the last row and last column of the block can be determined from the two offset vectors for the first row and column of the block and from substrings S1[1..i] and S2[1..j]. That is, no D value is needed in the input in order to determine the offset vectors in the last row and column of the block.
- Definition: The function that determines the two offset vectors for the last row and last column from the two offset vectors for the first row and column of a block together with substrings S1[1..i] and S2[1..j] is called the offset function
- We have $3\sigma^{2t} * t^2$ precomputation time, that for t=$(log_{3\sigma}n)/2$ is O(n$(logn)^2$)

4-Russians algorithm

1. Cover the dynamic programming table n by n with t-blocks, where the last column of each t-block is shared with the first column of the t-block to its right (if any) and the last row of each t-block is shared with the first row of the t-block below it (if any).
2. Initialize the values in the first row and column of the complete table according to the basic conditions of the recursion. Calculate the offset values in the first row and column.
3. In order, use the offset block function to sequentially determine the offset vectors of the last row and column of each block. Due to the overlapping nature of blocks, the offset vector in the last column (or row) of a block provides the next offset vector in the first column (or row) of the block to its right (or below it).
4. Let Q be the set of offset values calculated for cells in row n.D(n, n) = D(n, O)+ Q = n + Q .