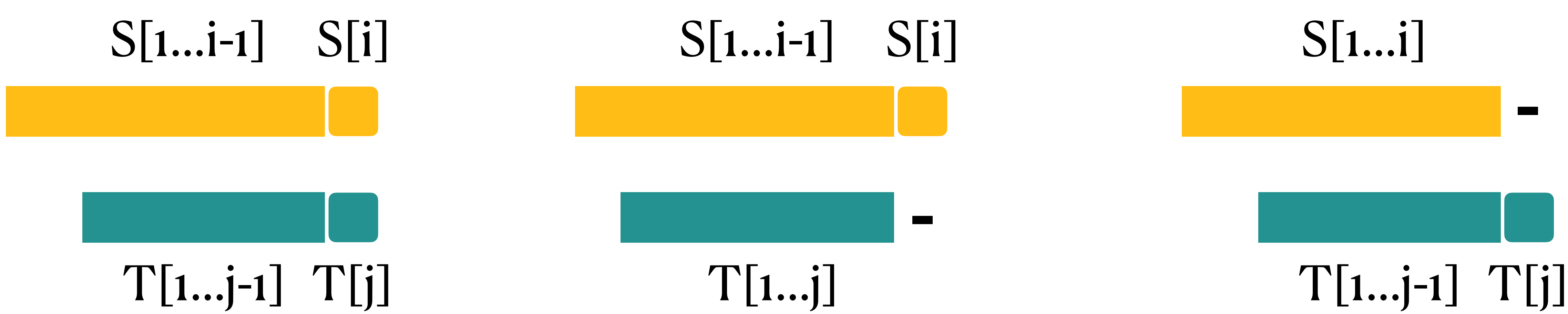
CSE 566 Spring 2023

Global & Local Alignment

Instructor: Mingfu Shao

Alignment with Affine Gap

- Affine gap cost = gap-open + k * gap-extension
- The DP for unit gap cost does not work: the cost of (S[i], -) depends on the previous column, and the choice in OPT(i-1, j) may not be optimal for OPT(i, j).
- Solution: let the subproblem include the last column

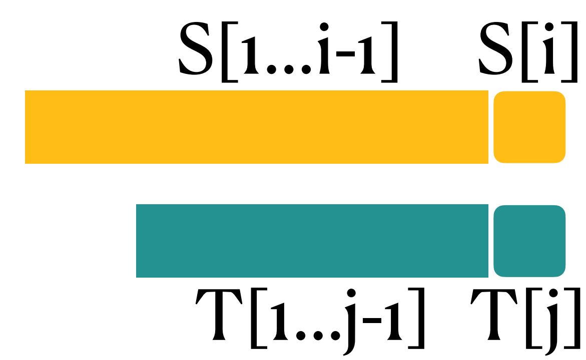


OPT(i-1, j)

Subproblems for Affine Gap

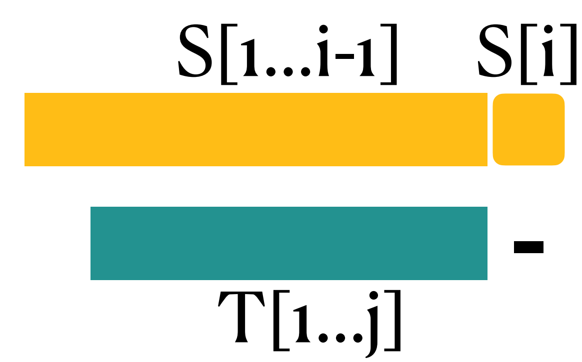
3mn

- M(i, j): minimized cost of S[1…i] and T[1…j] such that S[i] and T[j] is aligned (i.e., either a match or mismatch).

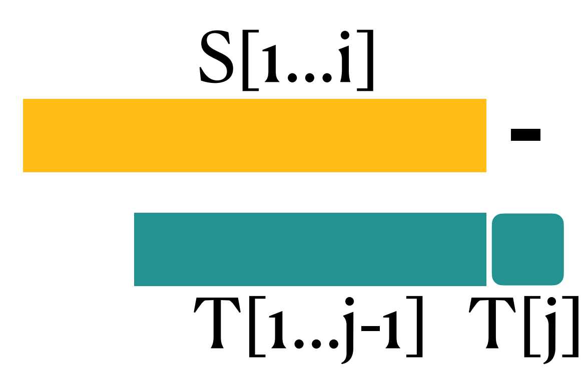


A | A   C | G

- X(i, j): minimized cost of S[1…i] and T[1…j] such that S[i] is aligned to "-".

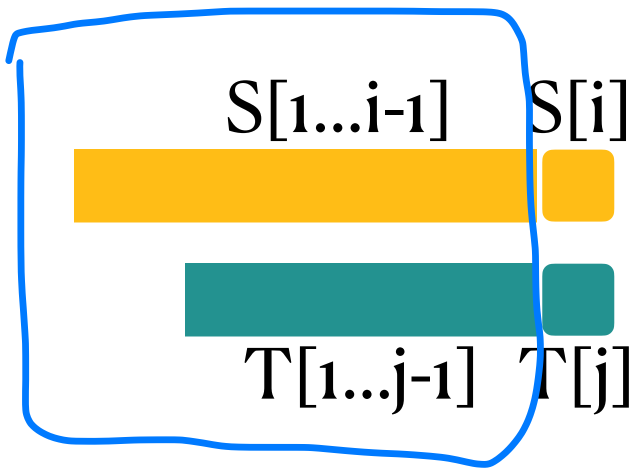


- Y(i, j): minimized cost of S[1…i] and T[1…j] such that T[j] is aligned to "-".

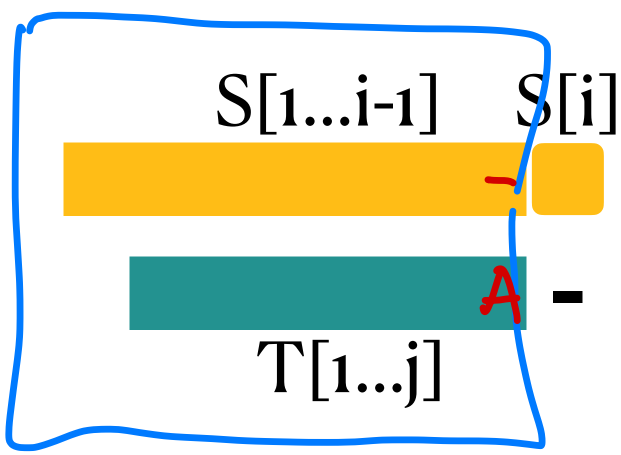


	A	C	G	T
A	0			
C		0		
G			0	
T				0

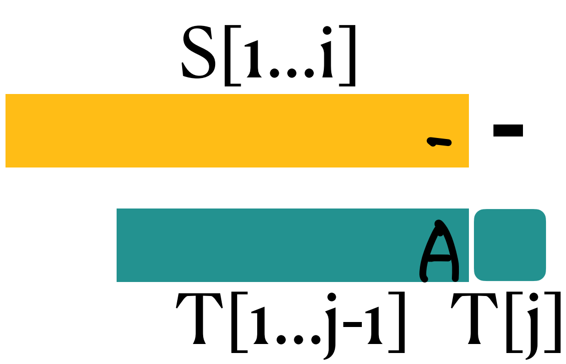
Recurrences



$$\underline{M(i,j)} = \underline{cost(S[i], T[j])} + \min \begin{cases} M(i-1, j-1) \\ X(i-1, j-1) \\ Y(i-1, j-1) \end{cases}$$



$$\underline{X(i,j)} = \min \begin{cases} M(i-1, j) + GO + GE \\ X(i-1, j) + GE \\ \underline{Y(i-1, j)} + GO + GE \end{cases}$$



$$\underline{Y(i,j)} = \min \begin{cases} M(i, j-1) + GO + GE \\ X(i, j-1) + GO + GE \\ Y(i, j-1) + GE \end{cases}$$

Details of the Algorithm

M

S= \ T=		A	C	G	T
	0	∞	∞	∞	∞
A	∞				
G	∞				
G	∞				.

M(m,n)

A
-

X

S=		A	C	G	T
	∞	∞	∞	∞	∞
A	GO +GE				
G	GO +2GE				
G	GO +3GE				,

Y

		A	C	G	T
A					
G					
G					.

- Initialization
- Minimized cost = $\min\{M(m,n), X(m,n), Y(m,n)\}$.
- Tracing back with pointers linking 3 tables.
- Running time: O(mn)

Local Alignment

```
S=ATGAACTTCGCATCCGATGGCATCT
T=GTCGTTCGGATGCCGATCGT
```

global alignment

```
AT-GAACTTCGCAT-CCGATGGCATCT
|| |   ||||||| |||||  |  ||
GTCG---TTCGGATGCCGAT--C--GT
```

local alignment

```
             S'
ATGAACTTCGCAT-CCGATGGCATCT
      ||||||| |||||
  GTCGTTCGGATGCCGATCGT
                T'
```

- To identify *conserved* regions (functional elements, such as promoters, enhancers, exons, protein domains, etc)

Try to Formulate

- Problem: given two strings S and T, to find a *substring* S' of S and a *substring* T' of T and an alignment A between S' and T' such that the cost of A is minimized.
- But, the optimal solution will always be S'=T'=empty (as the cost is positive).

Better Formulation

- Problem: given two strings S and T, to find a *substring* S’ of S and a *substring* T’ of T and an alignment A between S’ and T’ such that the **score of A is maximized**.
- Scoring an alignment:
  - matches -> positive score
  - Mismatches -> negative score
  - Gaps -> negative score

An Example

Match: 5; mismatch: -4; gap-open: -10; gap-extension: -1

```
AT-GAACTTCGCAT-CCGATGGCATCT
|| |   ||||||| |||||  |  ||
GTCG---TTCGGATGCCGAT--C--GT
```

Score = 15 * 5 + 3 * (-4) + 5 * (-10) + 9 * (-1) = 4

```
                   i
 ATGAACTTCGCAT-CCGATGGCATCT
       ||||||| |||||j
   GTCGTTCGGATGCCGATCGT
```

Score = 11 * 5 + 1 * (-4) + 1 * (-10) + 1 * (-1) = 40

Algorithm for Unit Gap Cost

- Problem: given S and T, to find a *substring* S' of S and a *substring* T' of T and an alignment A between S' and T' such that the score of A (with unit gap cost) is maximized.

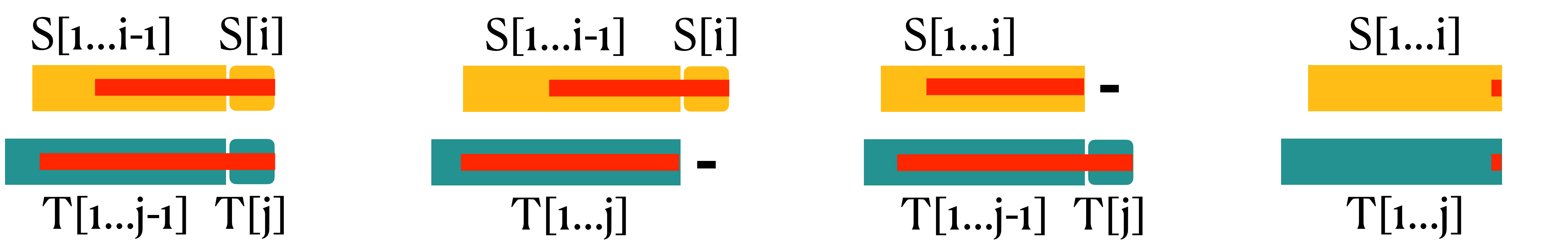
- Define OPT(i, j) as the maximized score between some *suffix* of S[1...i] and some *suffix* T[1...j].
- Note that the the length of the suffix can be 0.
- Empty suffixes to allow for "starting over"!

Recurrence

OPT(i,j)

- All possibilities in optimal local alignment of S[1...i] and T[1...j]

suffix of S[1..i] >= 1	suffix of S[1..i] >= 1	suffix of S[1..i] >= 0	suffix of S[1..i] = 0

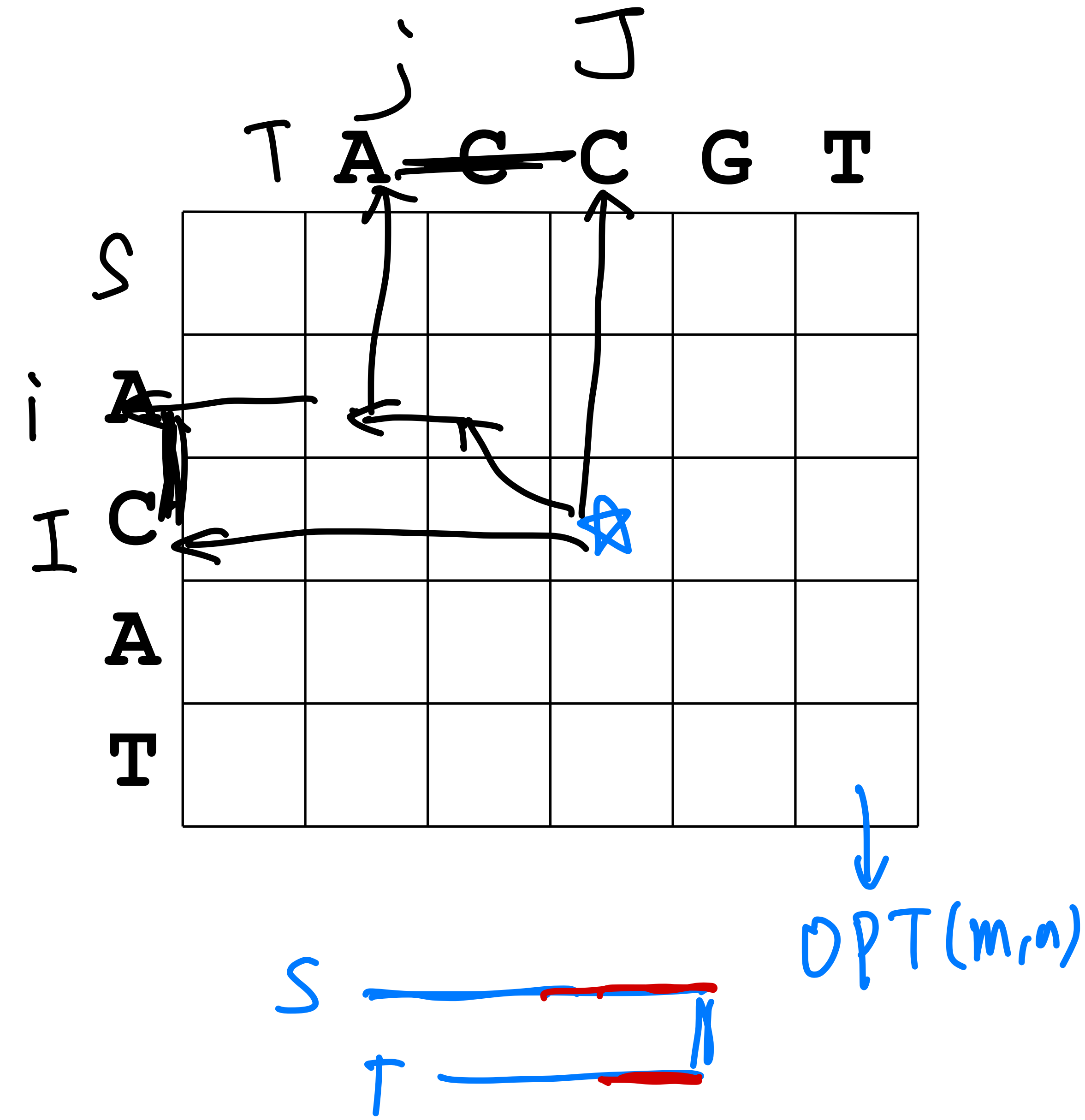


suffix of T[1..j] >= 1	suffix of S[1..j] >= 0	suffix of S[1..j] >= 1	suffix of S[1..j] = 0

$$OPT(i,j) = \max \begin{cases} OPT(i-1, j-1) + score(S[i], T[j]) \\ OPT(i-1, j) \cancel{+} - 1 \\ OPT(i, j-1) \cancel{+} - 1 \\ 0 \end{cases}$$

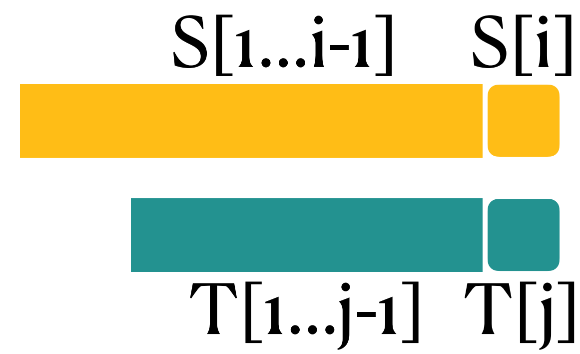
The Algorithm

- Step 1: Initialization
- Step 2: fill up the table following the recurrence
- Step 3: $\max_{1 \le i \le m, 1 \le j \le n} OPT(i, j)$ gives the optimal score.
- Step 4: backtrace to find the optimal alignment
- Running time: O(mn)

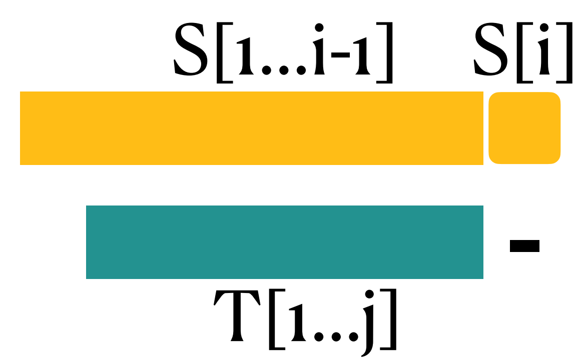


Subproblems for Affine Gap

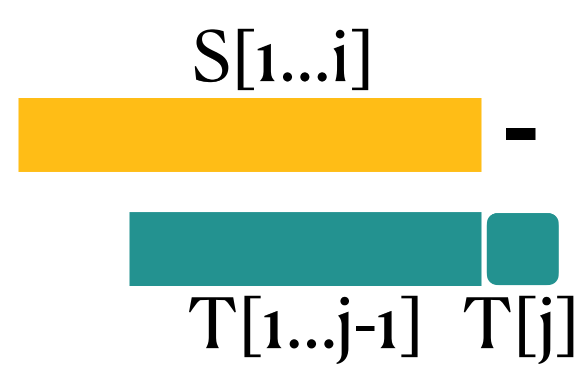
- M(i, j): maximized score of suffixes of S[1…i] and T[1…j] s.t. S[i] and T[j] is aligned.



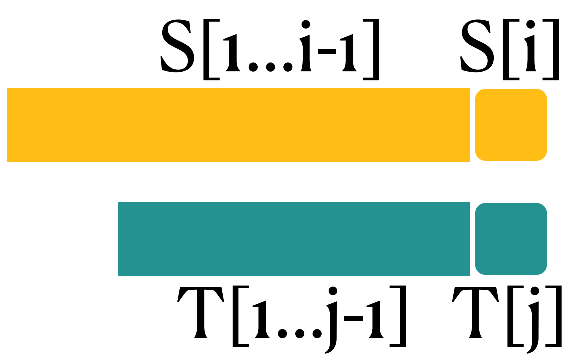
- X(i, j): maximized score of suffixes of S[1…i] and T[1…j] s.t. S[i] is aligned to "-".



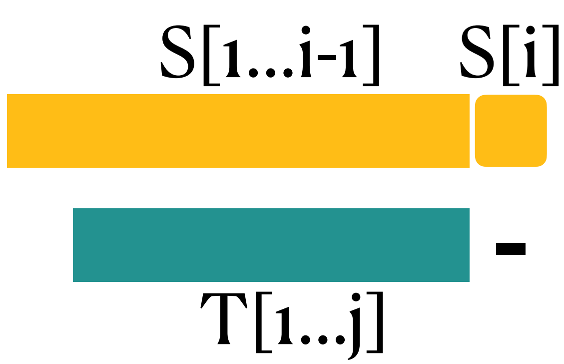
- Y(i, j): maximized score of suffixes of S[1…i] and T[1…j] s.t. T[j] is aligned to "-".



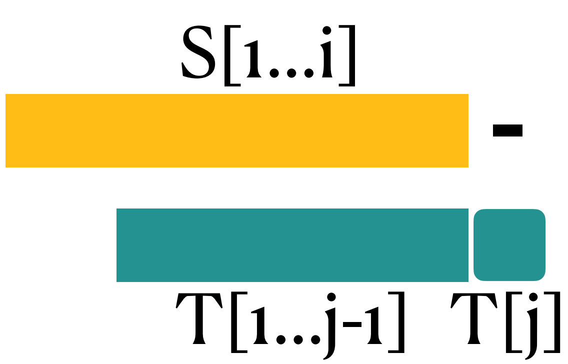
Recurrences



$$M(i,j) = score(S[i], T[j]) + \max \begin{cases} M(i-1, j-1) \\ X(i-1, j-1) \\ Y(i-1, j-1) \\ 0 \end{cases}$$



$$X(i,j) = \max \begin{cases} M(i-1, j) + GO + GE \\ X(i-1, j) + GE \\ Y(i-1, j) + GO + GE \\ 0 \end{cases}$$



$$Y(i,j) = \max \begin{cases} M(i, j-1) + GO + GE \\ X(i, j-1) + GO + GE \\ Y(i, j-1) + GE \\ 0 \end{cases}$$

The Algorithm

		A	C	G	T
A					
G					
G					

		A	C	G	T
A					
G					
G					

		A	C	G	T
A					
G					
G					

- Optimal score = $\underline{\max_{1 \le i \le m, 1 \le j \le n} \{M(i,j), X(i,j), Y(i,j), 0\}}$.
- Tracing back with pointers linking 3 tables.
- Running time: O(mn)