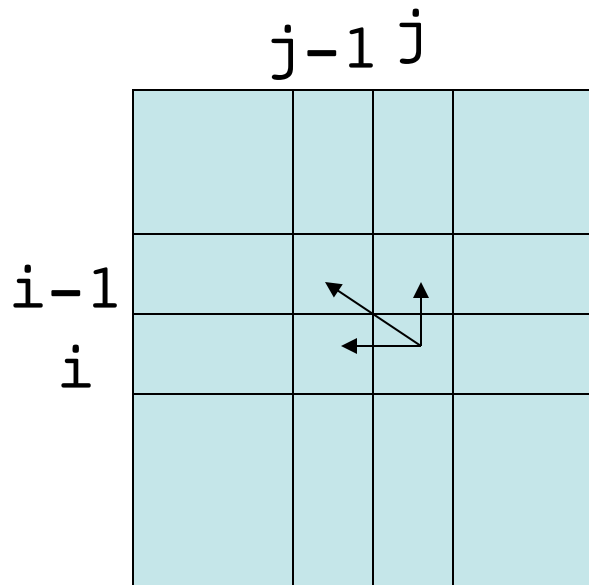


Back to Smith Waterman



An Example

T	C	A	T	-
T	G	C	A	A

A1

T		C	A	T
T	G	C	A	A

A2

- Align $s=TCAT$ with $t=TGCAA$
- Match Score = 1
- Mismatch score = -1, Indel Score = -1
- $C('T','T') = ?$; $C('T','A') = ?$; $C('-', 'A') = ?$
- Global Score A1?, Score A2?

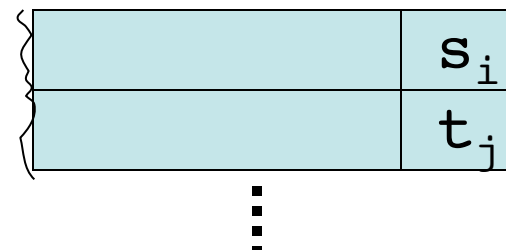
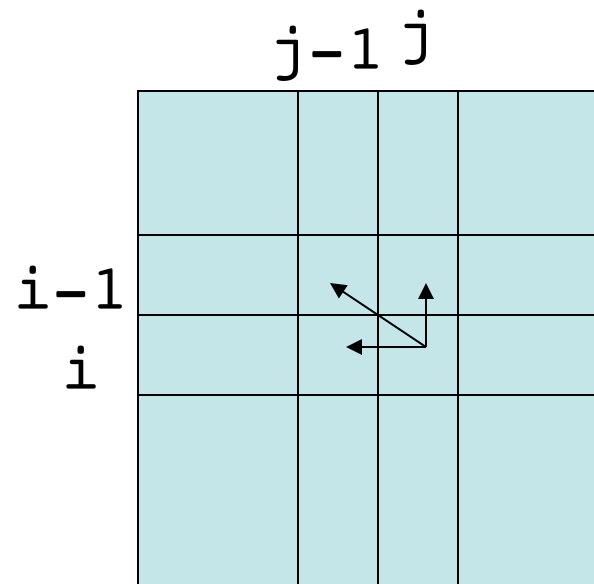
Example: Global alignment

$$S[i, j] = \max \begin{cases} S[i-1, j-1] + C(s_i, t_j) \\ S[i-1, j] + C(s_i, -) \\ S[i, j-1] + C(-, t_j) \end{cases}$$

	T	G	C	A	A	
	0	←-1	←-2	←-3	←-4	←-5
T	↑-1	↖1	←0	←-1	←-2	←-3
C	↑-2	↑0	↖0	↖1	←0	←-1
A	↑-3	↑-1	↖-1	↑0	↖2	↖1
T	↑-4	↑-2	↖-2	↑-1	↑1	↖1

Local Alignment

- The original recurrence still works, except when the optimum score $S[i,j]$ is negative
- When $S[i,j] < 0$, it means that the optimum local alignment cannot include the point (i,j) .
- So, we must reset the score to 0.



Local Alignment Trick (Smith-Waterman algorithm)

$$S[i, j] = \max \begin{cases} 0 & \blacksquare \\ S[i-1, j-1] + C(s_i, t_j) & \swarrow \\ S[i-1, j] + C(s_i, -) & \uparrow \\ S[i, j-1] + C(-, t_j) & \leftarrow \end{cases}$$

How can we compute the local alignment itself?

Ex: Local Alignment

$$S[i, j] = \max \begin{cases} 0 & \blacksquare \\ S[i-1, j-1] + C(s_i, t_j) & \swarrow \\ S[i-1, j] + C(s_i, -) & \uparrow \\ S[i, j-1] + C(-, t_j) & \leftarrow \end{cases}$$

		T	G	C	A	A
	0	0	0	0	0	0
T	0	1	0	0	0	0
C	0	0	0	1	0	0
A	0	0	0	0	2	1
T	0	0	0	0	1	1

Generalizing Gap Cost

- It is more likely for gaps to be contiguous
- The penalty for a gap of length l should be

$$g_0 + g_e * l$$

```

>gi|39930593|ref|NP_345197.1| hypocretin (orexin) receptor 1; orexin receptor
type 1 [Mus
      musculus] gi|37704009|gb|AA001326.1| orexin receptor type-1 [Mus
musculus]
      Length = 416

      Score = 526 bits (1354), Expect = e-148
      Identities = 264/399 (66%), Positives = 307/399 (76%), Gaps = 14/399 (3%)

Query: 24  EPFLNPTDYDDEEFLRYLWREYLNHPKEYENVLIAQYTIIVFVVALIGVLFQVAVWENHHM 85
      EPP P DY+DE FLRYLWR+YL+PK+YENVLIA Y+ VF++AL+GN LVC+AVW+HHHM
Sbjct: 19  EPFHLPPDYEDE-FLRYLWRDYLYPQYENVLIAATVAVFLIALVQNTLVCLAWRNHHM 77

Query: 84  RTVINYFIVNLSLADVLVTITCLPATELVVDITETWFFSQSLCKVIFYLQXXXXXXXXXXXX 145
      RTVINYFIVNLSLADVLVT CLPA+L+VDITE+W FSQ+LCKVIFYLQ
Sbjct: 78  RTVINYFIVNLSLADVLVTAICLPAQLVDITETSNLFSQALCKVIFYLQAVNSVAVLTL 137

Query: 146 XCIALDRNYAICHPLPKSTARRARNSXXXXXXXXXXXXXXXXXQAIWMECSNLPGLANHTT 205
      IALDRNYAICHPL+PKSTA+RAR S FQA VMECSN+LP LAN+T
Sbjct: 130 SPIALDRNYAICHPLPKSTARRARNSILGIMAVSLAVMVFQAAMVECSNLPGLANRTR 197

Query: 206 LFTVCDEHNQGEVYPMYHICFPLVTYMAPLCLNILAYLQIFRELMCRQIPQTSVQGR 265
      LF+VUDEHN E+YPR+YH CFF+VTY+APL LM +AY QIFRELM RQIPQT+S + R
Sbjct: 190 LFSVCDEHNADELYPKIYHSCFFIVTYLAPLSLMSHAYFQIFRELMCRQIPQTSALVHN 257

Query: 246 WEQ---QQFVSGPRSSQQSEKRTSAAVAEIQIRARRETARMLNVVLLVFAICYLPIISIL 323
      WE+ +Q +Q +S + + R A AE+EQ+RARRETA+MLNVVLLVFA+CYLPIIS+L
Sbjct: 250 WKRPSEQLEAQHQGLCTEFQPRARAPLAEVQRARRETARMLNVVLLVFAICYLPIISVL 317

Query: 324 NVLKNVFGMFTHTEDRETVMVMPFQSHMLVYANSAANPIIYNFLSCEFREPEAFPSOCL 383
      NVLKNVFGMF DRE VIA FTFQSHMLVYANSAANPIIYNFLSCEFRE+PEAFPSOCL
Sbjct: 310 NVLKNVFGMFRQA$DRENVIACFTFQSHMLVYANSAANPIIYNFLSCEFREQPEAFPSOCL 377

Query: 384 GVHHRQGBELARGRTSTESRKSLLTQIGENFDNVSEKLSH 422
      G S+ KSL+ Q + +VSE+SEH
Sbjct: 370 P-----SLRQSSARHKSLSLQ--SRCSVSEKLSH 405
    
```

Using affine gap penalties

$$S[i, j] = \max_l \begin{cases} 0 \\ S[i-1, j-1] + C(s_i, t_j) \\ S[i-l, j] + go + ge * l \\ S[i, j-l] + go + ge * l \end{cases}$$

- What is the time taken for this?
- What are the values that l can take?
- Can we get rid of the extra Dimension?

Affine gap penalties

- Define $D[i,j]$: Score of the best alignment, given that the final column is a 'deletion' (s_i is aligned to a gap)
- Define $I[i,j]$: Score of the best alignment, given that the final column is an insertion (t_j is aligned to a gap)

Optimum alignment of $s[1..i-1]$, and $t[1..j]$

	$s[i]$
	—

Optimum alignment of $s[1..i]$, and $t[1..j-1]$

	-
	$t[j]$

$$S[i,j] = \max \begin{cases} S[i-1, j-1] + C(s_i, t_j) \\ D[i,j] \\ I[i,j] \end{cases}$$

$O(nm)$ solution for affine gap costs

$$D[i, j] = \max \begin{cases} D[i-1, j] + ge \\ S[i-1, j] + go + ge \end{cases}$$

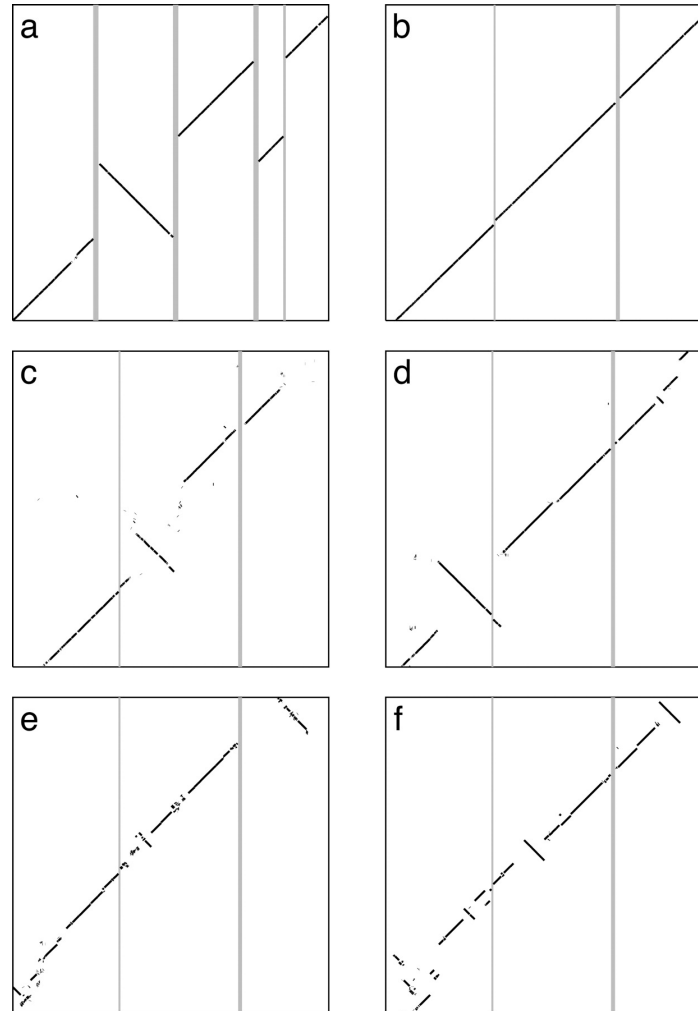
$$I[i, j] = \max \begin{cases} I[i, j-1] + ge \\ S[i, j-1] + go + ge \end{cases}$$

Alignment Space?

$$S[i, j] = \max \begin{cases} S[i-1, j-1] + C(s_i, t_j) \\ S[i-1, j] + C(s_i, -) \\ S[i, j-1] + C(-, t_j) \end{cases}$$

- How much space do we need?
- Is the space requirement too much?

Fig. 1. Dot-plot representation of sample assembly comparison results



Ex: Genomic assembly comparisons (6Mb)

Istrail, Sorin et al. (2004) Proc. Natl. Acad. Sci. USA 101, 1916-1921

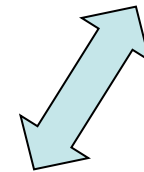
Alignment (Linear Space)

- Score computation

$$S[i, j] = \max \begin{cases} S[i-1, j-1] + C(s_i, t_j) \\ S[i-1, j] + C(s_i, -) \\ S[i, j-1] + C(-, t_j) \end{cases}$$

For $i = 1$ to n

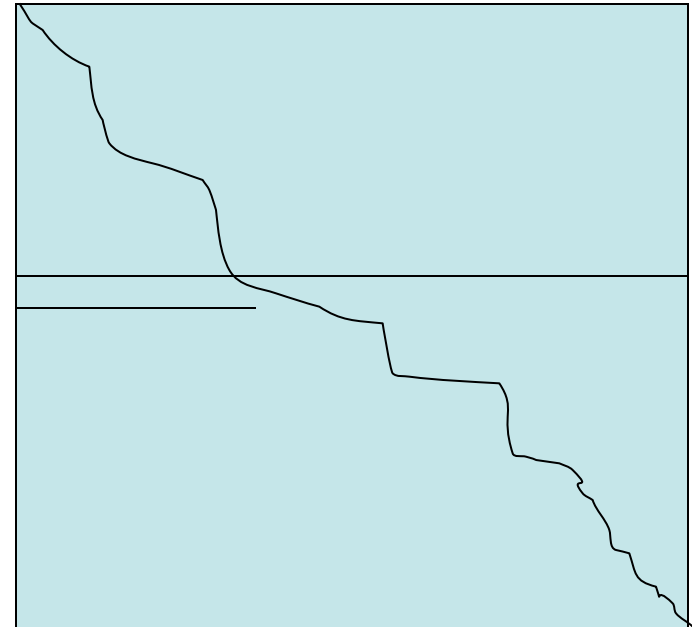
For $j = 1$ to m



$$\begin{cases} i_2 = i \% 2; & i_1 = (i - 1) \% 2; \\ S[i_2, j] = \max \begin{cases} S[i_1, j-1] + C(s_i, t_j) \\ S[i_1, j] + C(s_i, -) \\ S[i_2, j-1] + C(-, t_j) \end{cases} \end{cases}$$

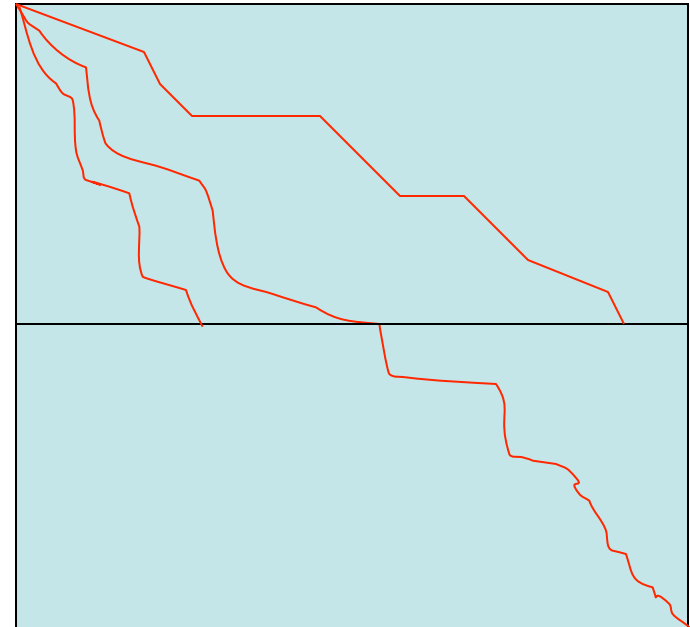
Linear Space Alignment

- In Linear Space, we can do each row of the D.P.
- We need to compute the optimum path from the origin $(0,0)$ to (m,n)



Linear Space (cont'd)

- At $i=n/2$, we know scores of all the optimal paths ending at that row.
- Define $F[j] = S[n/2, j]$
- One of these j is on the true path. Which one?



Backward alignment

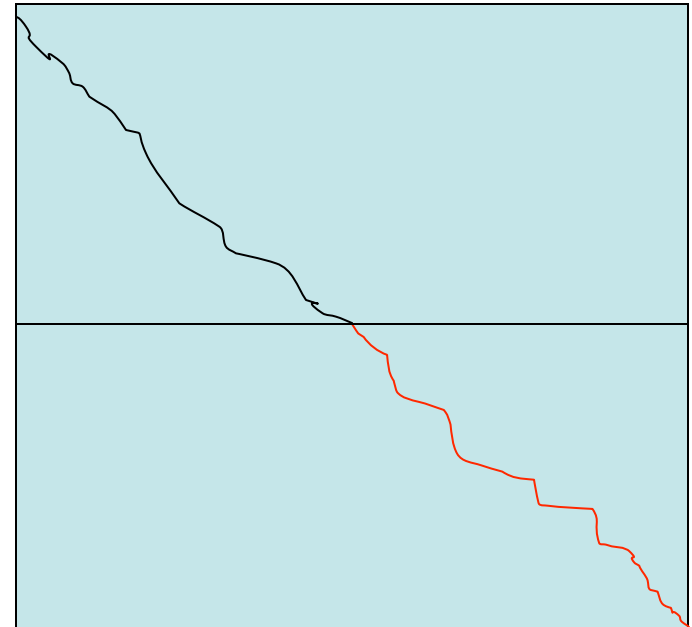
- Let $S_b[i,j]$ be the optimal score of aligning $s[i+1..n]$ with $t[j+1..m]$

$$S_b[i,j] = \max \begin{cases} S_b[i+1,j+1] + C(s_i, t_j) \\ S_b[i+1,j] + C(s_i, -) \\ S_b[i,j+1] + C(-, t_j) \end{cases}$$

- Boundary cases?
- $S_b[n,j]$? $S_b[m,j]$?

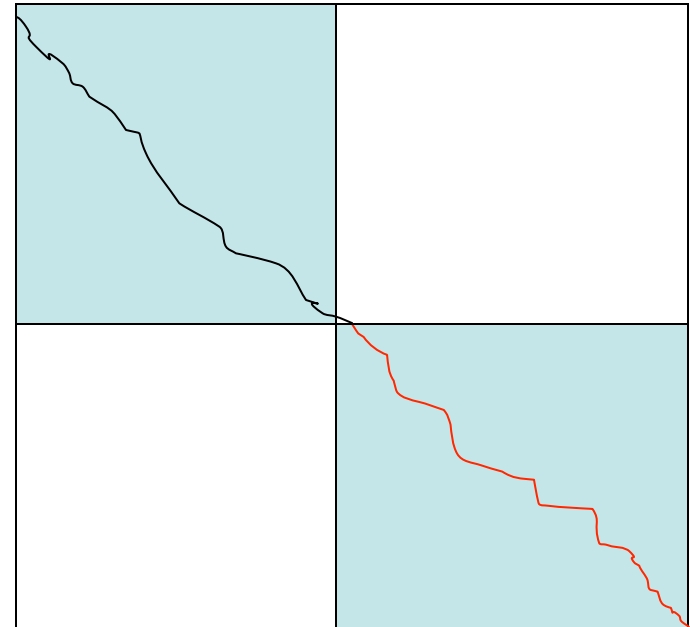
Forward, Backward computation

- At the optimal coordinate, j
 - $F[j]+B[j]=S[n,m]$
- In $O(nm)$ time, and $O(m)$ space, we can compute one of the coordinates on the optimum path.



Linear Space Alignment

- $\text{Align}(1..n, 1..m)$
 - For all $1 \leq j \leq m$
 - Compute $F[j] = S(n/2, j)$
 - For all $1 \leq j \leq m$
 - Compute $B[j] = S_b(n/2, j)$
 - $j^* = \max_j \{F[j] + B[j]\}$
 - $X = \text{Align}(1..n/2, 1..j^*)$
 - $Y = \text{Align}(n/2+1..n, j^*+1..m)$
 - Return X, j^*, Y



Linear Space complexity

- $T(nm) = c.nm + T(nm/2) = O(nm)$
- $\text{Space} = O(m)$