Biostatistics-Lecture 18 High-throughput sequencing and sequence alignment

Ruibin Xi

Peking University

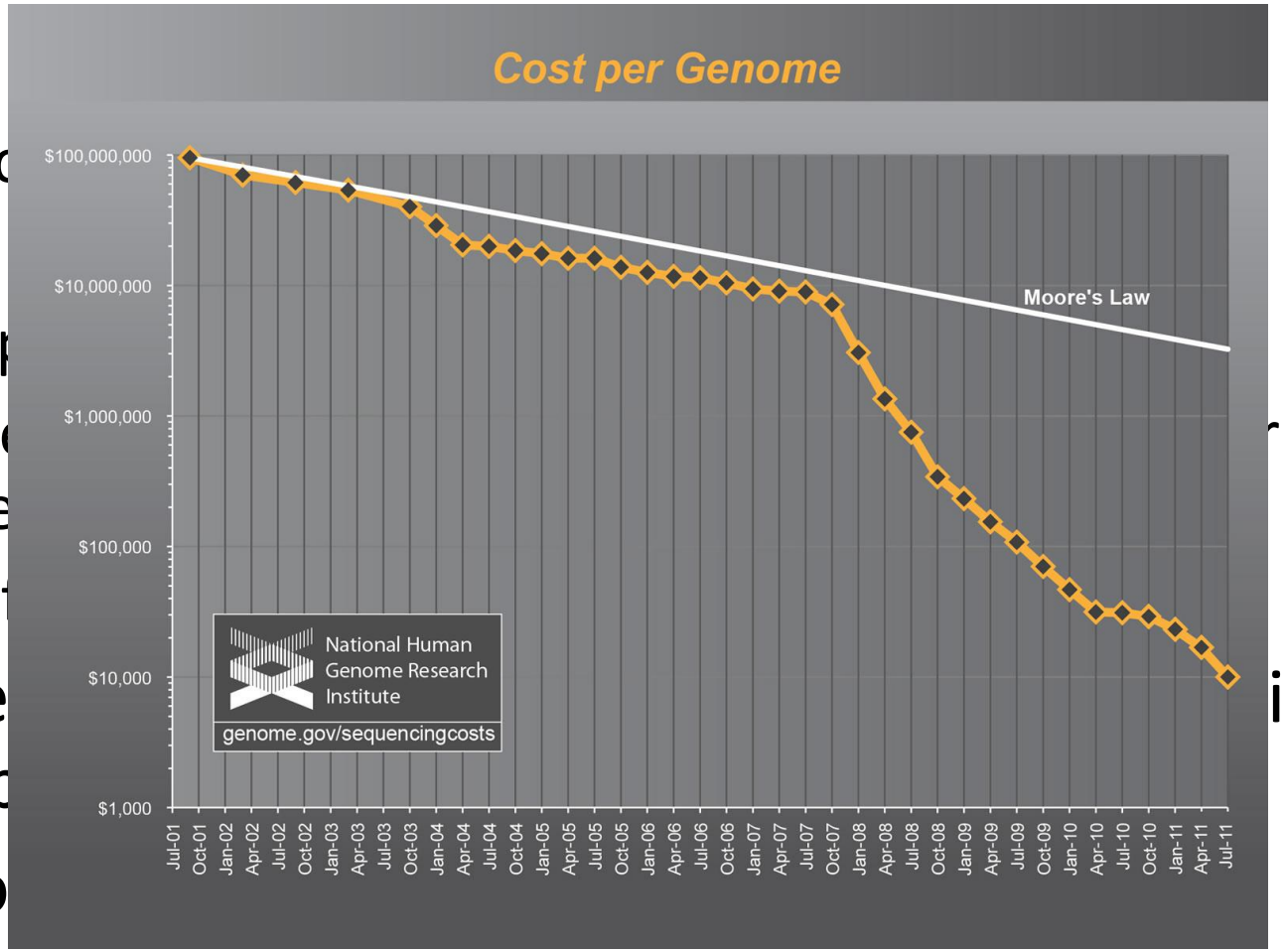
School of Mathematical Sciences

High-throughput sequencing

- HTS platforms
  - Roche 454 platforms
  - Illumina/Solexa platforms (most widely used)
  - Applied Biosystem (ABI) SOLiD
  - Helicos HeliSope™ sequencer(single molecular sequencing)
  - Life Technologies platforms
- The throughput is increasing and the price is dropping
- Short read but high throughput

High-throughput sequencing

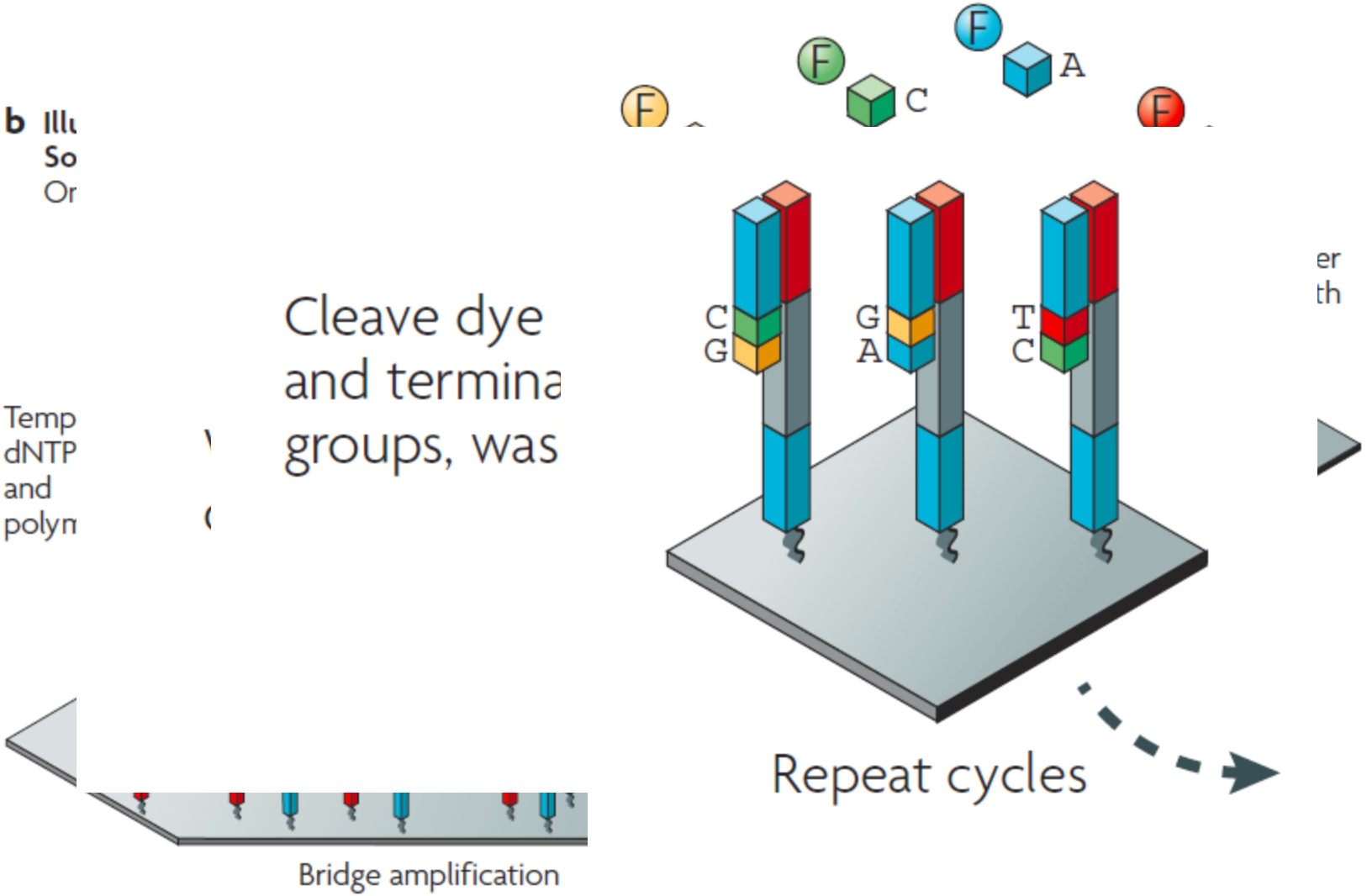
- HTS
  - Ro
  - Ill
  - Ap
  - He se
  - Li
- The is drop
- Sho



What sequencing data can do

- Detection of genomic variations (Whole genome sequencing or targeted sequencing)
  - Single nucleotide polymorphisms (SNP)
  - Copy number variations (CNV)
  - Structural variations (SV)
- Analyze protein interactions with DNA (ChIP-seq)
- Whole Transcriptome study (RNA-seq)
- DNA methylation study
- and many more …..

Sequencing (Illunima)



Mardis Nature Reivew Genetics (2010)

What the data look like?

```
@SRR062640.324 HWUSI-EAS709_103325707:6:1:1112:3886/1
TCAAGGTTTCTCTGTGGTCCTGTCAGCCAAGAGGGAGTCCGCAGTTGACTGTAGGGACTAGGGCTTATTTTCATTTCTCAATCCTGACAAGGTAAGGCTG
+
GFGGGEBEEEEEEAEE?EEDGGGGGDFF=FFEDFBEDEEEE5BDE=?>:CE:BEEA=CA>EEE:A5=CCCB=@EE5?ACCEEE@E?A=>>CA@ABE@:::
@SRR062640.326 HWUSI-EAS709_103325707:6:1:1112:3443/1
CATATTGGGCATTGGTATAATTTAGTTCTGCTATAATGTACAAAAATGACACAGAGGTAGTAAGAGTGATGAGAGACTTAAGCTCTTTCGGCATCTGCTT
+
FDFFFEAEEEFEDFAEDEEEFFEFFFFDFDEFFFFFFFFEFF5DD@@?EEFDFDBEEBCEB5-AADBD:?DCDC=-BCD=DF:FDFFAF??AB-EC=A##
@SRR062640.327 HWUSI-EAS709_103325707:6:1:1112:12981/1
AAACAATGCTAGAAAGAATATAACCTCATCATATAAAATTAGGCTTTACTTTCTAAATAAAAGCTTTATTTGAAGGGAAAACTAGATAAAATTCAAAAGA
+
GGGFGGGGGGGGGGGDGGEGGFDFEGGGEGFGGGGEGGGGGDGGFFFGGEGEGEGGDGAAD:5>DEFE@FAAA?AEFE?CBDE:BFFF?EDCEDF5BAAC7
@SRR062640.328 HWUSI-EAS709_103325707:6:1:1113:5059/1
CCCAGGTTCAAGCGATCCTCCTGCCTCAGCCTTGCAAACATCTGGGATTAGCCGAGATTGCACTGCTGCACTCCCGTCTGTGCACCTGGAAACCCTGTCT
+
E:EAEEE=DBEEB=DEEABEEEEDEE=BED=ADDD6>>B=E?A5AD-B:DEECEBA5:A:EBE5AEDCB5A?:>+A????BCB:-C=A=@?:@?=5A:<A
```

Fastq Format detail:

1st line: the name of a short read

2nd line: the read itself (a short sequence of A,C,G,T)

3rd line: the name of the short read or plus (+) sign

4th line: the quality score

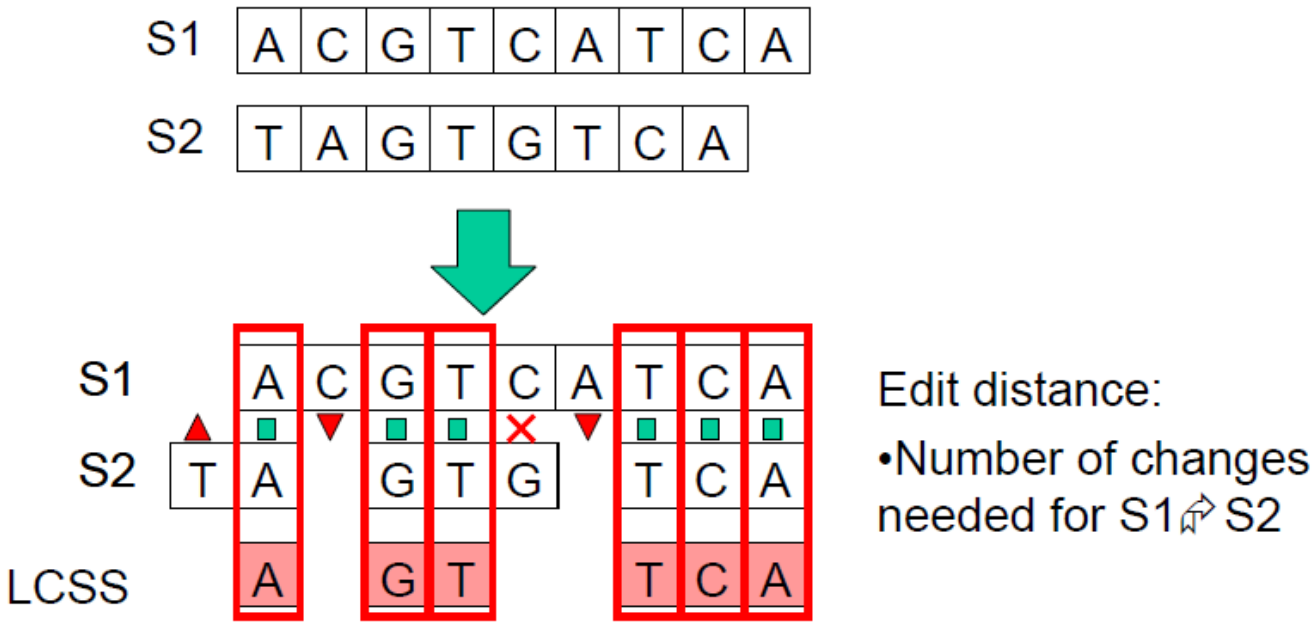
General strategy for analyzing HTS data

- Alignment-based

- Assembly-based

Comparing two DNA sequences

- Given two possibly related strings S1 and S2
  - What is the longest common subsequence?



Edit distance:

•Number of changes needed for S1 ⇨ S2

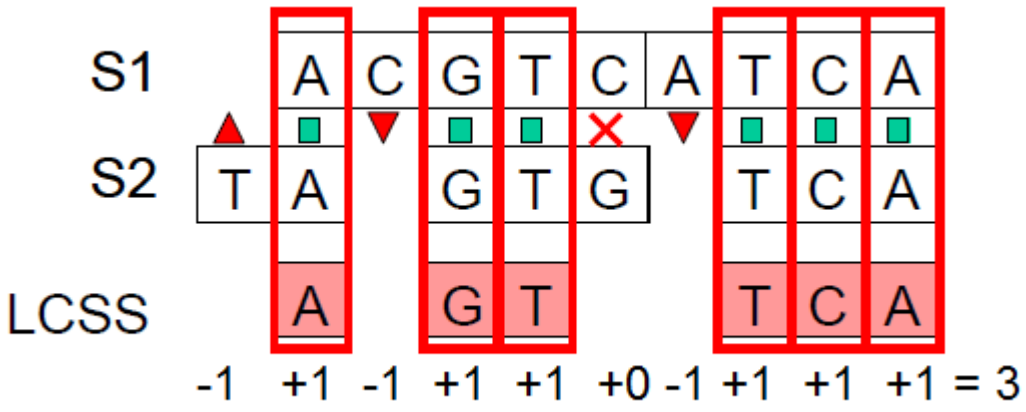
How can we evaluate an alignment

S1	A	C	G	T	C	A	T	C	A
S2	T	A	G	T	G	T	C	A	

- Need scoring function:
  - Score(alignment) = Total cost of editing S1 into S2
  - Cost of mutation
  - Cost of insertion / deletion
  - Reward of match
- Need algorithm for inferring best alignment
  - Enumeration?
  - How would you do it?
  - How many alignments are there?

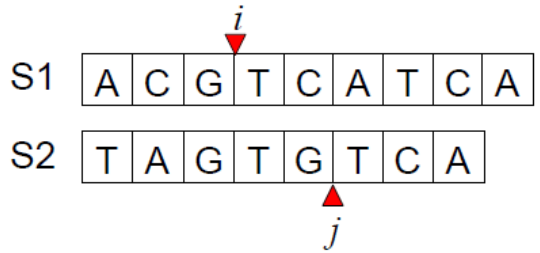
How can we evaluate an alignment

- Scores:
  - Mutation (mismatch): 0
  - Match: 1
  - Gap: -1

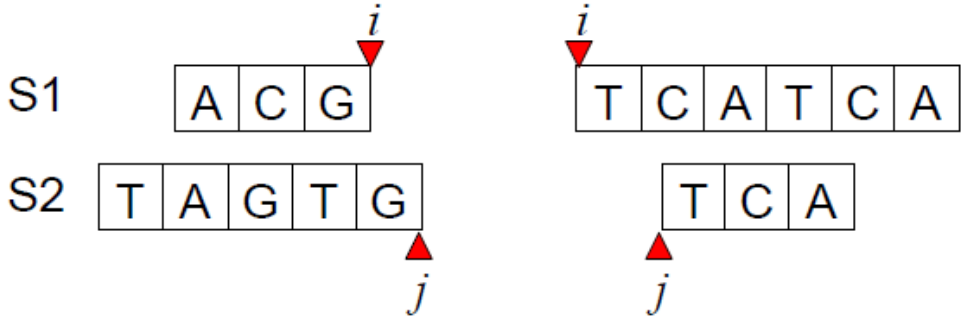


Find a best alignment

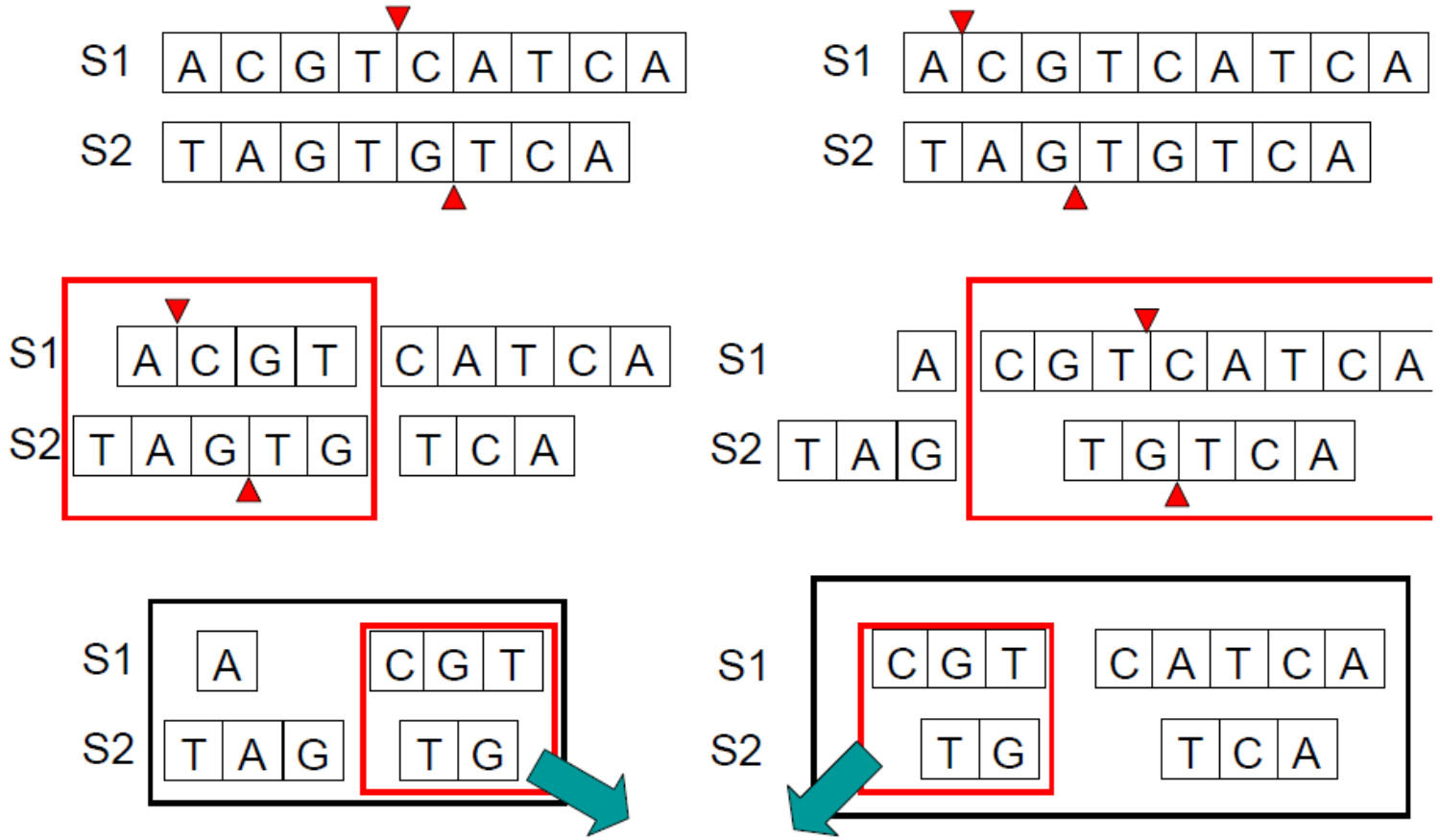
- Exhaustively search all possible alignments
  - Computationally too expensive!!!
- Observation: for a pair (i,j)



  - For a given split ***(i, j)***, the best alignment is:
    - Best alignment of S1[1..i] and S2[1..j]
    - \+ Best alignment of S1[i..n] and S2[j..m]



Find a best alignment



Identical sub-problems! We can reuse our work!

Find a best alignment—Solution I

- Create a big dictionary, indexed by aligned seqs
  - When you encounter a new pair of sequences
  - If it is in the dictionary:
    - Look up the solution
  - If it is not in the dictionary
    - Compute the solution
    - Insert the solution in the dictionary
- Ensures that there is no duplicated work
  - Only need to compute each sub-alignment once!

Find a best alignment—Solution II

- Create a big table, indexed by (i,j)
  - Fill it in from the beginning all the way till the end
  - You know that you'll need every subpart
  - Guaranteed to explore entire search space
- Ensures that there is no duplicated work
  - Only need to compute each sub-alignment once!
- Very simple computationally!

Dynamical programming

- For example, we want to align the following 2 sequences:

A C T C

A T C

- There are 3 possibilities for the first position in the alignment
  - 1. Add gap in the 1st sequence
  - 2. Add gap in the 2nd sequence
  - 3. not place gap in both sequences

1st position	Score	Remaining Sequence
-- A	-1	ACTC TC
A --	-1	CTC ATC
A A	1	CTC TC

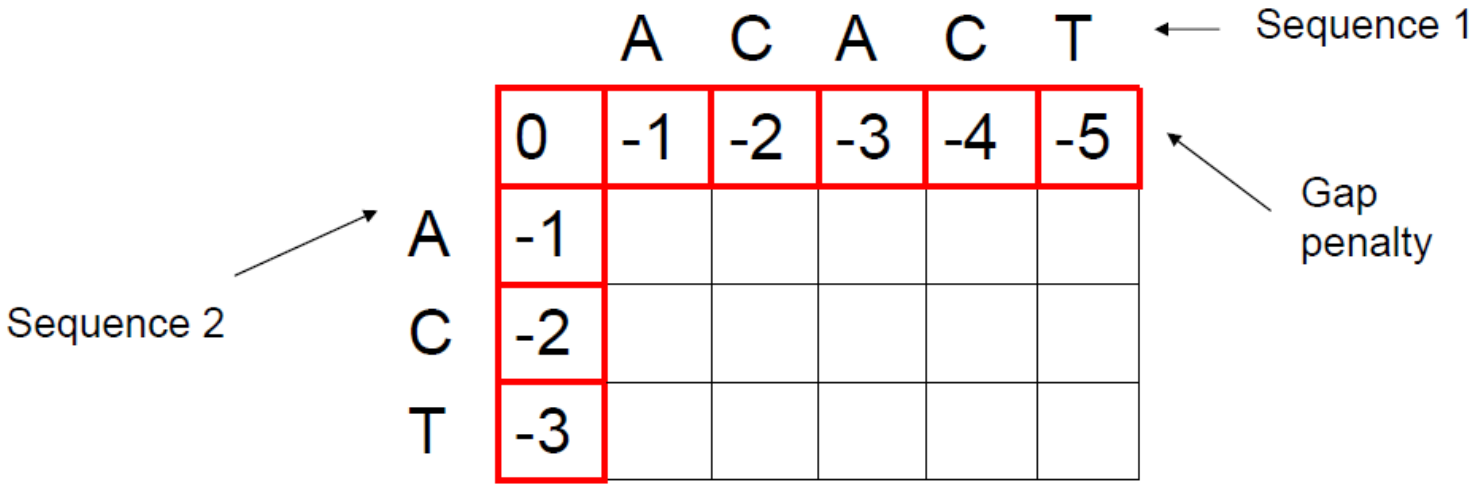
Dynamical programming

- Example: Compute the best alignment for the following two sequences: (match score = 1, mismatch score = 0, gap penalty = -1)

  A C A C T

  A C T

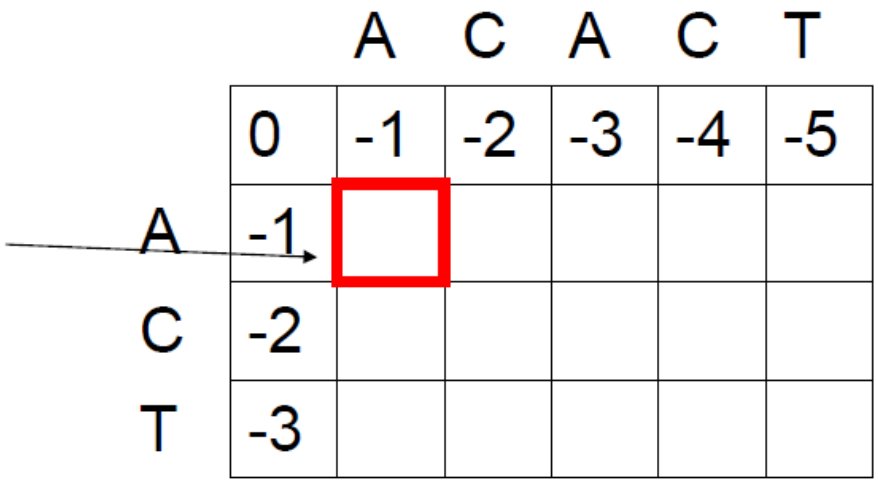
- Step 1: Create a table similar to the following



Dynamical programming

- Step 2:

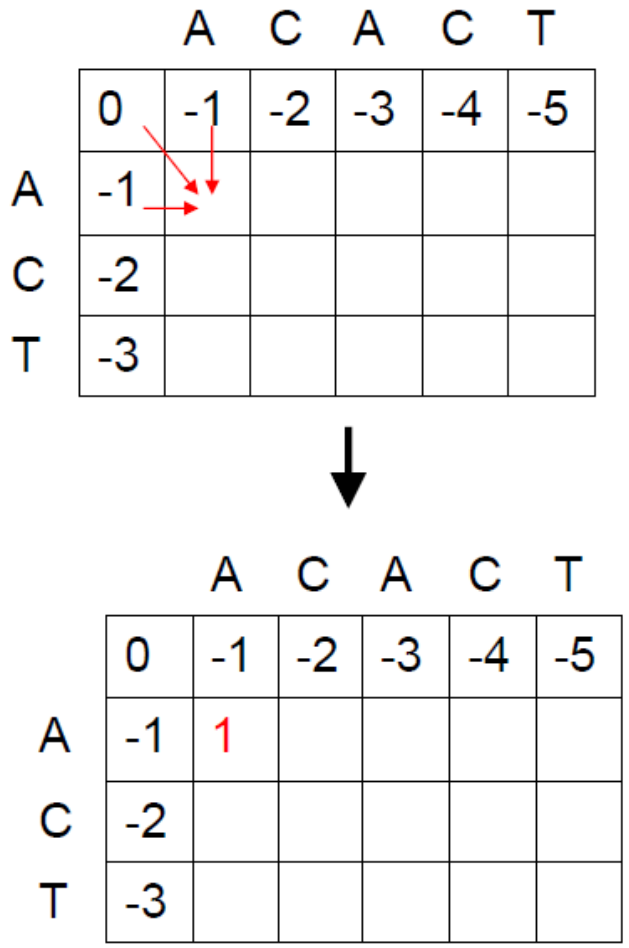
Start from this position



- There are 3 kinds of movement:
  1. Vertical move: add gap in top sequence
  2. Horizontal move: add gap in left sequence
  3. Diagonal move: align nucleotides from each sequence

Dynamical programming

- Vertical move:
  - ***Upper value + gap penalty***
  - i.e. (-1) -1 = -2
- Horizontal move:
  - ***Left value + gap penalty***
  - i.e. (-1) -1 = -2
- Diagonal move:
  - ***Top left value + match / mismatch*** **score** for nucleotides at the axes
  - i.e. the first nucleotides of both sequences are identical, so match score =1
  - ➔ 0 + 1 = 1
- Fill in the maximum value of the three in the cell
  - i.e. Max = 1 (diagonal move)



Dynamical programming

- Step 3:
  - Repeat step 2 along row 2

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C						
T						

Dynamical programming

- Step 4: Fill in the table row by row

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1	1	2	1	1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1				
C	-2					
T	-3					

-2

-2

1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0			
C	-2					
T	-3					

0

-3

-1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1		
C	-2					
T	-3					

-1

-4

-1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	
C	-2					
T	-3					

-2

-5

-3

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2					
T	-3					

-3

-6

-4

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0				
T	-3					

-3

0

-1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2			
T	-3					

-1

-1

2

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1		
T	-3					

1

-2

0

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	
T	-3					

0

-3

0

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3					

-1

-4

-2

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1				

-4

-1

-2

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1	1			

-2

1

0

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1	1	2		

0

1

2

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1	1	2	1	

1

-1

1

Dynamical programming

		A	C	A	C	T
	0	-1	-2	-3	-4	-5
A	-1	1	0	-1	-2	-3
C	-2	0	2	1	0	-1
T	-3	-1	1	2	1	1

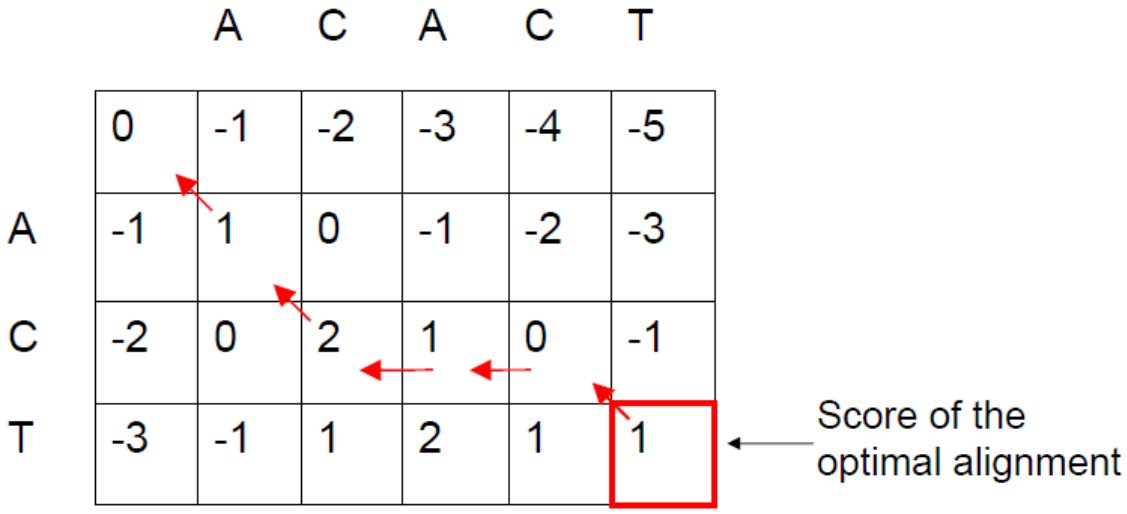
0

-2

1

Dynamical programming

- Step 5: Reconstruct the optimal alignment from the lower rightmost entry. Create the path by moving to position that could legally produce the score
- Value of the lower rightmost entry is the score of the optimal alignment
- In this example, there are two possible paths. One of them is reconstructed as following



Dynamical programming

- Step 6: Convert the path to an alignment
- Rule:
  - Vertical move➔ gap in the top sequence
  - Horizontal move➔ gap in the left sequence
  - Diagonal move ➔ alignment of nucleotides from each sequence
- In the example, the reconstructed path is:

- The converted alignment is

A	C	A	C	T
A	C	--	--	T

Semiglobal alignment

- Searching the best alignment between a short sequence and a long sequence
- Internal and terminal gap are scored the differently, because terminal gaps are probably caused by difference in sequence lengths
- To find the semiglobal alignment, make 2 modifications to the basic dynamic programming algorithm
  1. Avoiding penalties of initial gaps - Initial first row and column of table with zeros
  2. Avoiding penalties of end gaps - vertical moves in last column and horizontal moves in last row have no gap penalty

Semiglobal alignment

- Find the optimal semiglobal alignment of the following sequences: (match score =1, mismatch score=0, gap penalty=-1)

AACAGTCT

AGT

1. Initialize first row and first column of table with zeros

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

- Each horizontal move in first row means adding one initial gap in the left sequence

  → no gap penalty for initial gap in the left sequence

- Each vertical move in first column means adding one initial gap in the top sequence

  → no gap penalty for initial gap in the top sequence

Semiglobal alignment

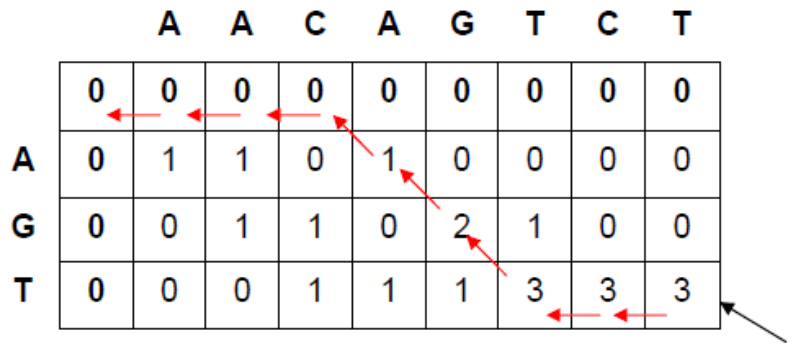
2. Vertical move in last column and horizontal move in last row has no gap penalty
   - Consider the last cell in last column(gap penalty = -1):
     - ***Vertical move: 0
     - ***Horizontal move: 3 ← **No gap penalty**
     - Diagonal move: 0 + match score = 1

     → Maximum score of the 3 movements = 3

		A	A	C	A	G	T	C	T
	0	0	0	0	0	0	0	0	0
A	0	1	1	0	1	0	0	0	0
G	0	0	1	1	0	2	1	0	0
T	0	0	0	1	1	1	3	3	3

Semiglobal alignment

- Similarly, no gap penalty for vertical move in last column



		A	A	C	A	G	T	C	T
	0	0	0	0	0	0	0	0	0
A	0	1	1	0	1	0	0	0	0
G	0	0	1	1	0	2	1	0	0
T	0	0	0	1	1	1	3	3	3

AACAGTCT

_ _ _AGT_ _

alignment score

- The 2 horizontal movements at the bottom of the table mean adding 2 end gaps to the left sequence
- The score along the horizontal path remains unchanged
  → No gap penalty is involved at end gaps

Local Alignment

- Searching for the best matching subsequences within two sequences
- Semiglobal alignment is insufficient for searching best match subsequences, because mismatching positions and gaps outside target subsequences produce non-favorable score
- Smith-Waterman algorithm:
  - Besides the three types of movement (vertical, horizontal and diagonal) in basic dynamic programming algorithm, there has the forth option – place zero if all other movements have negative score
  - Initial the first row and column of table with zeros
  - After filling in all partial scores, find the maximum partial score (instead of lower rightmost entry) and work backward until a zero is reached

Local Alignment

- Example:
  - Find the local alignment for the following sequences: (match score =1, mismatch score=0, gap penalty=-1)

ATTCGATCC

ACGAT

  - Initial the partial score table as the following

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

Local Alignment

- Fill in each of partial scores with one of the four options (Score of vertical move, horizontal move, diagonal move or zero)

		A	T	T	C	G	A	T	C	C
	0	0	0	0	0	0	0	0	0	0
A	0	1	0	0	0	0	1	0	0	0
C	0	0	1	0	1	0	0	1	1	1
G	0	0	0	1	0	2	1	0	1	1
A	0	1	0	0	1	1	3	2	1	0
T	0	0	2	1	0	1	2	4	3	2

Local Alignment

- Find the maximum partial score in the table
- Work backward until reach a zero

		A	T	T	C	G	A	T	C	C
	0	0	0	0	0	0	0	0	0	0
A	0	1	0	0	0	0	1	0	0	0
C	0	0	1	0	1	0	0	1	1	1
G	0	0	0	1	0	2	1	0	1	1
A	0	1	0	0	1	1	3	2	1	0
T	0	0	2	1	0	1	2	4	3	2

maximum partial score

- Best local alignment:

CGAT

CGAT

BLAST and BLAT

- BLAST: Basic Local Alignment Search Tool

- BLAT: BLAST Like Alignment Tool

BLAST

- BLAST:
  - A search algorithm for finding local alignments of two sequences S and T
  - An associated theory for evaluating the statistical significant
- Terminology and notation
  - S(a,b): scoring system
  - High-scoring Segment Pair (HSP):
    - Cannot be extended or shortened without dropping the score

BLAST

- The number of HSPs with a score ≥ S approximately follows a Poisson Distribution (under the null hypothesis) with parameter

$$v = Kmne^{-\lambda S}$$

  - Assumptions

$$E = \sum_{a,b \in \Sigma} p_a p_b S(a,b) < 0$$

    Some probability to take positive score
  - Based on extreme value theory
  - E-value $E(S) = Kmne^{-\lambda S}$
  - Bit score $S' = \frac{\lambda S - \ln K}{\ln 2}.$

BLAST

- Algorithm: seed and extend
  - Build an index for k-mers of the query sequence
  - Find the hits of the k-mers in the database sequence in the query sequence
  - Extend the seeds with a score ≥ a threshold and find the HSPs with a score ≥ S
  - Evaluate the statistical significance

BLAT

- Strategy: seed and extend
  - In the seed stage, detects regions of two sequences that are likely to be homologous
  - In the extend stage, those regions are examined in detail and alignments are produced.

- Index the non-overlapping K-mers of the database sequences instead of the query sequence

BLAT

- Strateg
  - In the
  - In the
- Index th

Table 1. Timing of **BLAT** vs. **WU-TBLASTX** on a Data Set of 1000 Mouse Reads and a **RepeatMasked** Human Chromosome 22

Method	K	N	Matrix	Time
WU-TBLASTX	5	1	+15/−12	2736 s
WU-TBLASTX	5	1	BLOSUM62	2714 s
BLAT	5	1	+2/−1	61 s
BLAT	4	2	+2/−1	37 s

The first WU-TBLASTX run was performed using the settings used in Exofish. The second WU-TBLASTX run was performed using the settings B = 9000 V = 9000 hspmax = 4 topcomboN = 1 W = 5 E = 0.01 Z = 3000000000 nogaps filter = xnu + seg. The K column indicates the size of the perfectly matching hit that serves as a seed for an alignment. The N column indicates how many hits in a gapless 100-amino acid window were required to trigger a detailed alignment. The Matrix column describes the match/mismatch scores or the substitution score matrix used.

BLAT

- Seeding strategy
  - Single perfect K-mer matches
  - Single near perfect K-mer matches
  - Multiple perfect K-mer matches