Lecture 3: Indexing for Sequence Alignment + Estimating Sequencing Coverage

ECE 365 - Data Science and Genomics

Announcements

- Lab 2 (sequence alignment) released tomorrow

From last lecture: Best global alignment

- We use an algorithm based on *dynamic programming*

		G	C	A	T	T	C
	0	-1	-2	-3	-4	-5	-6
G	-1	1	0	-1	-2	-3	-4
A	-2	0	0	1	0	-1	-2
T	-3	-1	-1	0	2	1	0
T	-4	-2	-2	-1	1	3	2
A	-5	-3	-3	-1	0	2	2
C	-6	-4	-2	-2	-1	1	3

match:	$+1$
mismatch:	-1
gap:	-1

Needleman-Wunsch algorithm

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + m, \\ \quad (\text{if } x_i = y_j) \\ H_{i-1,j-1} - s, \\ \quad (\text{if } x_i \neq y_j) \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases}$$

GCATT-C
G-ATTAC

From last lecture: Best local alignment

- We can adapt the previous algorithm to find local alignments

match:	$+3$		
mismatch:	-3		
gap:	-2		

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Smith-Waterman algorithm

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + m, \\ \quad (\text{if } x_i = y_j) \\ H_{i-1,j-1} - s, \\ \quad (\text{if } x_i \neq y_j) \\ H_{i-1,j} - d \\ H_{i,j-1} - d \\ 0 \end{cases}$$

GTT-AC
GTTGAC

From last lecture: Alignment problem variations

Global alignment

GCATT-C
G-ATTAC

		G	C	A	T	T	C
	0	-1	-2	-3	-4	-5	-6
G	-1	1	0	-1	-2	-3	-4
A	-2	0	0	1	0	-1	-2
T	-3	-1	-1	0	2	1	0
T	-4	-2	-2	-1	1	3	2
A	-5	-3	-3	-1	0	2	2
C	-6	-4	-2	-2	-1	1	3

Local alignment

T **GTT-AC** GG
G **GTTGAC** TA

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Alignment to reference

GCACT-C
TCT **G-ACTAC** GCGT

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

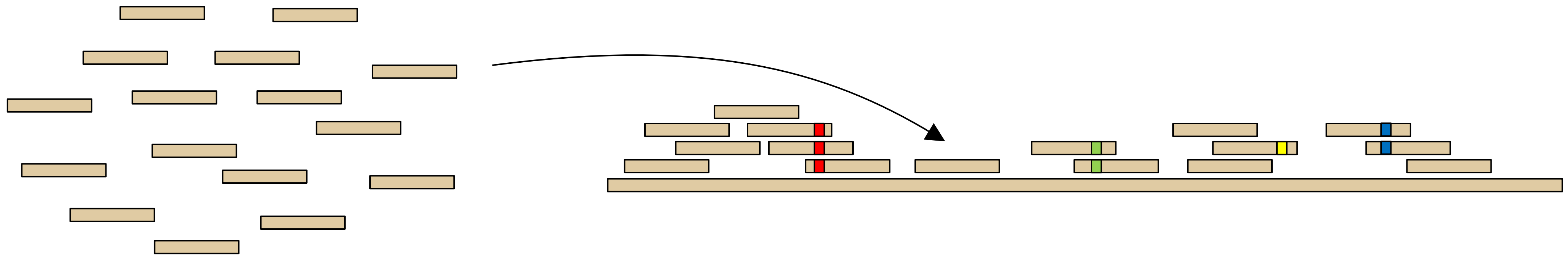
Overlap alignment

GCACT-C GCGTTCA
TTCT **G-ACTAC**

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

Are these methods computationally efficient?

- Needleman-Wunsch and Smith-Waterman algorithms are slow in practice
- Impractical to align many short reads to a reference genome:



- Is there a way to quickly figure out where a read belongs in the genome?

Indexing

- Does the sequence `cgtcagcggacagggc` appear in the genome below?

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaacccttttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

Indexing

- Does the sequence cgtcagcggacagggc appear in the genome below?

ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccag**cgtcagcggacagggc**gcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc

Indexing

- Does the sequence `cgtcagcggacagggc` appear in the genome below?

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgtttcccggtaccggttttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

- One idea: sort substrings like in a dictionary!

Indexing

$$\overbrace{\texttt{cgtcagcggacagggc}}^{16}$$

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaacccttttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

Indexing

$\overbrace{\texttt{cgtcagcggacagggc}}^{16}$

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggttttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaaggggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

all substrings of length 16

$\overbrace{\hphantom{aaaacttctgttctgt}}^{16}$	
aaaacttctgttctgt	457
aaaataacagtcacgc	546
aaaatgtcacgccgct	58
aaaatgtgttttgctc	475
aaaccaacacgccagg	591
aaaccctttttttcta	402
aaacttctgttctgtt	458
aaagccatcgttttcc	292
aaagccgtattccggt	529
aaagcgttgacgtaag	276
aaagttcacgcgtcgg	188
aaataacagtcacgct	547
aaatgtcacgccgctt	59
aaatgtgttttgctca	476
aacaacgaaaccaaca	584
aacacgccaggcttaa	596
aacagtcacgctttta	551
aaccaacacgccaggc	592
aacccttggatgcagg	430
aaccctttttttctaa	403
aacgaaaccaacacgc	587
aacgaaagccgtattc	525
aacttctgttctgttt	459
aagaaaatgtcacgcc	55
aagaaagccatcgttt	289
aagacccagcttcagg	122
aagagccccgtggtgg	33

Indexing

$$\overbrace{\texttt{cgtcagcggacagggc}}^{16}$$

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggttttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

- Another way to do indexing: Python dictionary

Indexing

$$\overbrace{\texttt{cgtcagcggacagggc}}^{16}$$

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggttttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

- Another way to do indexing: Python dictionary (hash function)

Indexing

$$\overbrace{\texttt{cgtcagcggacagggc}}^{16}$$

```
ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggttttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaaccctttttttctaagggggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

$\Longrightarrow$

all substrings of length 16

```
Dict()

ggtttaatgtggttct  -->  0
gtttaatgtggttctg  -->  1
tttaatgtggttctgc  -->  2
ttaatgtggttctgct  -->  3
taatgtggttctgctt  -->  4
aatgtggttctgcttg  -->  5
atgtggttctgcttgg  -->  6
tgtggttctgcttggc  -->  7
gtggttctgcttggcg  -->  8
tggttctgcttggcgg  -->  9
ggttctgcttggcggt  -->  10
gttctgcttggcggta  -->  11
ttctgcttggcggtag  -->  12
tctgcttggcggtagt  -->  13
ctgcttggcggtagtc  -->  14
tgcttggcggtagtca  -->  15
gcttggcggtagtcat  -->  16
cttggcggtagtcatt  -->  17
ttggcggtagtcatta  -->  18
```

- Another way to do indexing: Python dictionary (hash function)
- Let's look at this in a notebook

What if there are errors/mutations on read?

$X = \mathtt{cgt}\overset{\downarrow}{\mathtt{a}}\mathtt{agcggaca}\overset{\downarrow}{\mathtt{t}}\mathtt{ggc}$

$Y =$

ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccag**cgtcagcggacagggc**gcaagtg
ccgtgaatgggccgtacagttatgaaacccttttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc

What if there are errors/mutations on read?

$X =$ cgt**a**$\overbrace{\text{agcgga}}$ca**t**ggc

$Y =$

ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccag$\overbrace{\textbf{cgtcag}}$**cggacagggc**gcaagtg
ccgtgaatgggccgtacagttatgaaacccttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc

- One idea: Consider indexing substrings of length $k \approx 6$

What if there are errors/mutations on read?

$X =$ cgtaagcggacatggc

```
Y =

ggtttaatgtggttctgcttggcggtagtcattaagagccccgtggtggccaat
caagaaaatgtcacgccgcttcccagcactttcagctgttttgtcgtagcccat
caccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcggtt
ctgcgtcggcatggattctgcacggcaaagttcacgcgtcggtttgccataatt
aaggacgcgcctggattcaccttgcgatcggcaatcgcaggaatgagagagcag
ataatgaaagcgttgacgtaagaaagccatcgttttcccggtaccggtttttgc
gcctgcccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtg
ccgtgaatgggccgtacagttatgaaacccttttttctaagggcttctacaa
cccttggatgcagggcgaagtcgggaaaacttctgttctgtttaaaatgtgttt
tgctcatagtgtggtagatctcagcttactattggctttaacgaaagccgtatt
ccggtgaaaataacagtcacgcttttagttgttaatgttacaccaacaacgaaa
ccaacacgccaggcttaattcctgtggagttatatatgagcgtaaatcggatcc
```

all substrings
of length $k = 6$

Python Dict ()

```
ggttta --> [0]
gtttaa --> [1, 471]
tttaat --> [2]
ttaatg --> [3, 571]
taatgt --> [4, 572]
aatgtg --> [5, 477]
atgtgg --> [6]
tgtggt --> [7, 495]
gtggtt --> [8]
tggttc --> [9]
ggttct --> [10, 158]
gttctg --> [11, 159, 466]
ttctgc --> [12, 160, 177]
tctgct --> [13]
ctgctt --> [14]
tgcttg --> [15]
gcttgg --> [16]
cttggc --> [17]
ttggcg --> [18]
tggcgg --> [19]
ggcggt --> [20]
```

- One idea: Consider indexing substrings of length k

What if there are errors/mutations on read?

$X =$ `cgtaagcggacatggc`

take substrings
of length $k = 6$

What if there are errors/mutations on read?

$X =$ cgt**a**agcggaca**t**ggc

take substrings of length $k = 6$

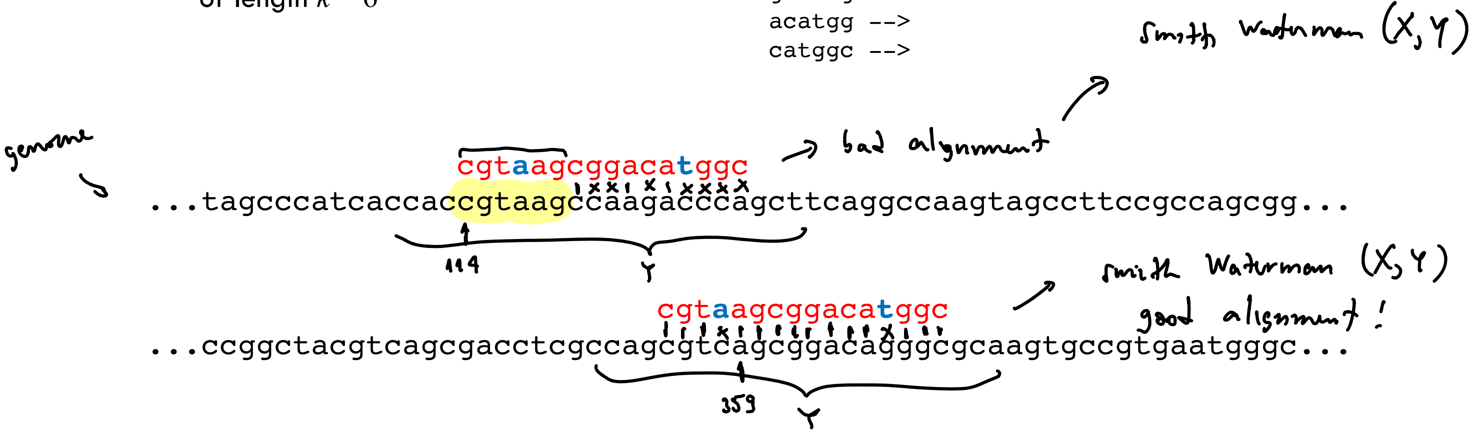
```
cgtaag --> [114, 286]
gtaagc --> [115]
taagcg -->
aagcgg -->
agcgga --> [359]
gcggac --> [360]
cggaca --> [361]
ggacat -->
gacatg -->
acatgg -->
catggc -->
```

What if there are errors/mutations on read?

$X =$ cgtaagcggacatggc

take substrings
of length $k = 6$

```
cgtaag --> [114, 286]
gtaagc --> [115]
taagcg -->
aagcgg -->
agcgga --> [359]
gcggac --> [360]
cggaca --> [361]
ggacat -->
gacatg -->
acatgg -->
catggc -->
```



Smith Waterman (X, Y)
genome
cgtaagcggacatggc
bad alignment
...tagcccatcaccaccgtaagccaagacccagcttcaggccaagtagccttccgccagcgg...
114
Y
Smith Waterman (X, Y)
good alignment!
cgtaagcggacatggc
...ccggctacgtcagcgacctcgccagcgtcagcggacagggcgcaagtgccgtgaatgggc...
359
Y

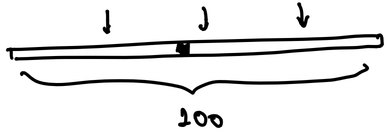
How do we choose k?

$\lceil 3.2 \rceil = 4$

$\lceil 5 \rceil = 5$

- Based on the sequencing technology (e.g., Illumina: $L = 100$, error rate $= 0.1\%$)

. Suppose at most 1 error per read



Claim : there is a segment of length $\frac{100-1}{2} = 49.5$ with no errors

. Suppose $\leq t$ errors per read

there is a segment of length $\underbrace{\left\lceil \frac{L-t}{t+1} \right\rceil}$ with no errors

$\hookrightarrow$ set $= k$

How many reads have $> t$ errors ?

# of errors in a read : $X \sim \text{Binomial}(100, 0.001)$

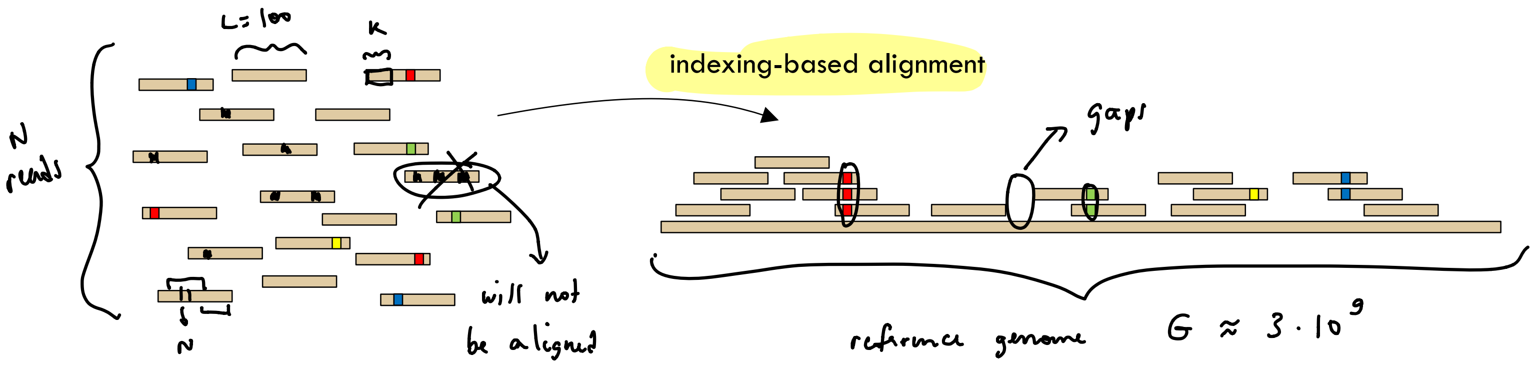
fraction of reads with $> t$ errors : $P(X > t)$

e.g. for $t = 2$.

$P(X > 2) = 0.00015$

$k = \left\lceil \frac{100-2}{3} \right\rceil = 33$

Big picture:



How to pick N?