Algorithms in Bioinformatics Spring 2023 Lecture 2

Eleazar Eskin

University of California, Los Angeles

Administrative updates and reminders

- Check course website for latest announcements: https://bruinlearn.ucla.edu/courses/160773

will also post lecture slides before class

- Discussion sections will be primarily covering chapter 1 this week
  - 1A – Fri 12-1:50pm BROAD 2100A
  - 1B – Fri 12-1:50pm KAPLAN A65
  - 1C – Fri 2-3:50pm DODD 175
  - 1D – Fri 2-3:50pm KAPLAN 169
- Everyone on waitlist at the end of class Tue should have a PTE in MyUCLA
- If still need a PTE, email a TA or instructor with UID and if not on waitlist preferred discussion section(s)

Textbook

- Accounts have been created for receipts uploaded by yesterday
- Use same email with online book site as given in the receipt upload form
- Additional receipts for print book (https://www.bioinformaticsalgorithms.org/) uploaded to https://forms.gle/iA7JLc7LK2a3Qiuv8 by Tue April 11th will be processed for access to online site by Wed. Apr 12th
- Link for online only purchase https://stepik.org/a/170824
- Form to transfer previous purchase of textbook to our course instance for free: https://forms.gle/ms6BRrC6o5eevBGf7

Upcoming due dates

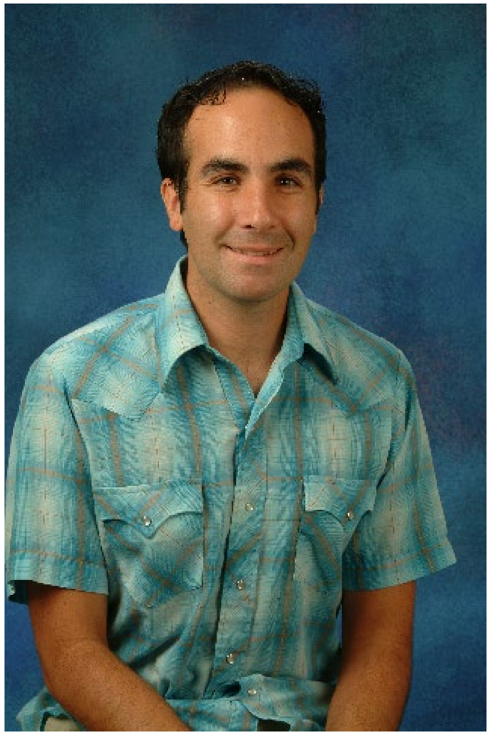
- HW1 due Thur. 4/13 (recommend to finish earlier)
- Post Question on Paper 1 due Thur. 4/13

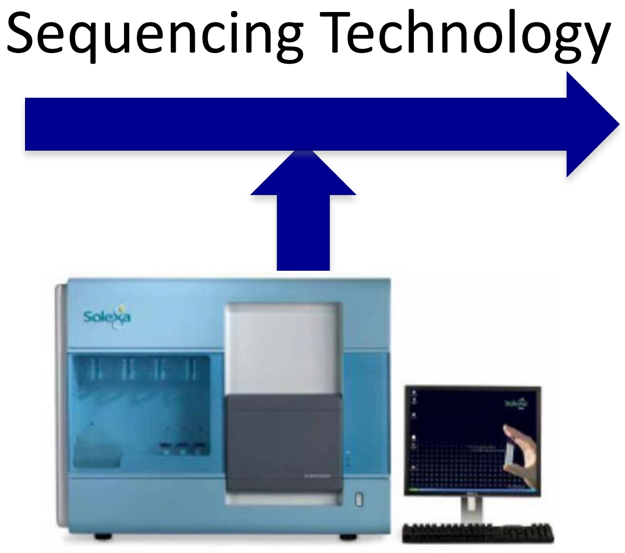
Sequencing + Read Mapping

Lecture 2.

April 6th, 2023

Active Research Problem: Short Read Re-seqeuncing
Where are my mutations?





Illumina / Solexa
Genetic Analyzer 1G
1000 Mb/run, 35bp reads

AGAGC**A**GTCGACA**G**GTA
TAG**T**CTACATGAGATC**G**
ACATGAGATC**G**GTAGAG
C**C**GTGAGATC**G**ACATGA
TAG**C**CAGAGC**A**GTCGAC
A**G**GTATAG**T**CTACATGA
GATC**G**ACATGAGATC**G**G
TAGAGC**C**GTGAGATC**G**A
CATGATAG**C**CAGAGC**A**G
TCGACA**G**GTATAG**T**CTA
CATGAGATC**G**ACATGAG
ATC**G**GTAGAGC**C**GTGAG
ATC**G**ACATGATAG**C**CAG
AGC**A**GTCGACA**G**GTATA
G**T**CTACATGAGATC**G**AC
ATGAGATC**G**GTAGAGC**C**
GTGAGATC**G**ACATGATA
G**C**CAGAGC**A**GTCGACA**G**
GTATAG**T**CTACATGAGA
TC**G**ACATGAGATC**G**GTA
GAGC**C**GTGAGATC**G**ACA
TGATAG**C**CAGAGC**A**GTC
GACA**G**GTATAG**T**CTACA
TGAGATC**G**ACATGAGAT
C**G**GTAGAGC**C**GTGAGAT
C**G**ACATGATAG**C**CAGAG
C**A**GTCGACA**G**GTATAG**T**
CTACATGAGATC**G**ACAT
GAGATC**G**GTAGAGC**C**GT
GAGATC**G**ACATGATAG**C**

- Next generation sequencing.
  - **Cheap sequencing.**
  - **“Short Reads”**

Short Read Sequencing Problem (A Computer Science Problem)

Full DNA Sequence

```
AGAGCAGTCGACAGGTA
TAGTCTACATGAGATCG
ACATGAGATCGGTAGAG
CCGTGAGATCGACATGA
TAGCCAGAGCAGTCGAC
AGGTATAGTCTACATGA
GATCGACATGAGATCGG
TAGAGCCGTGAGATCGA
CATGATAGCCAGAGCAG
TCGACAGGTATAGTCTA
CATGAGATCGACATGAG
ATCGGTAGAGCCGTGAG
ATCGACATGATAGCCAG
AGCAGTCGACAGGTATA
GTCTACATGAGATCGAC
ATGAGATCGGTAGAGCC
GTGAGATCGACATGATA
GCCAGAGCAGTCGACAG
GTATAGTCTACATGAGA
TCGACATGAGATCGGTA
GAGCCGTGAGATCGACA
TGATAGCCAGAGCAGTC
GACAGGTATAGTCTACA
TGAGATCGACATGAGAT
CGGTAGAGCCGTGAGAT
CGACATGATAGCCAGAG
CAGTCGACAGGTATAGT
CTACATGAGATCGACAT
GAGATCGGTAGAGCCGT
GAGATCGACATGATAGC
```

- Short read sequencers generate random short substrings from the DNA sequence of a certain length.

```
ATGAGATCGGTAGAGCCGTGAGAT
GAGCAGTCGACAGGTATAGTCTAC
AGAGCAGTCGACAGGTATAGTCTA
TGAGATCGACATGATAGCCAGAGC
TAGCCAGAGCAGTCGACAGGTATA
GATAGCCAGAGCAGTCGACAGGTA
GAGATCGACATGATAGCCAGAGCA
GCAGTCGACAGGTATAGTCTACAT
AGCAGTCGACAGGTATAGTCTACA
TCGACATGAGATCGGTAGAGCCGT
CAGTCGACAGGTATAGTCTACATG
GAGATCGACATGATAGCCAGAGCA
GTAGAGCCGTGAGATCGACATGAT
```

How do we recover the original sequence?

Short Reads Difficulties

```
ATGAGATCGGTAGAGCCGTGAGAT
GAGCAGTCGACAGGTATAGTCTAC
AGAGCAGTCGACAGGTATAGTCTA
TGAGATCGACATGATAGCCAGAGC
TAGCCAGAGCAGTCGACAGGTATA
GATAGCCAGAGCAGTCGACAGGTA
GAGATCGACATGATAGCCAGAGCA
GCAGTCGACAGGTATAGTCTACAT
AGCAGTCGACAGGTATAGTCTACA
TCGACATGAGATCGGTAGAGCCGT
CAGTCGACAGGTATAGTCTACATG
GAGATCGACATGATAGCCAGAGCA
GTAGAGCCGTGAGATCGACATGAT
```

- We don’t know where each read comes from!
- Can’t identify where the mutations are!
- What do we do?

Key Idea: “Re”-Sequencing

We know that my genome is very close to the Human genome.

My Genome:
TACATGAGATC**G**ACATGAGATC**G**GTAGAGC**C**GTGAGATC

A Sequence Read:
TCGACATGAGATCGGTAGAGCCGT

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          TCGACATGAGATCGGTAGAGCCGT
```

Recovered Sequence:
TACATGAGATC**G**ACATGAGATC**G**GTAGAGC**C**GTGAGATC

"Re"-Sequencing Challenges (Why do we need Computer Science?)

- Sequences are long!
  - **Human Genome is 3,000,000,000 long.**
- Sequencers generate many reads!
  - **A single run generates over 300,000,000 reads.**
- We need efficient algorithms to "map" each read to its location in the genome.
  - **A trivial mapping algorithm will take thousands of years to compute for a genome.**

There are other challenges which we are not mentioning.

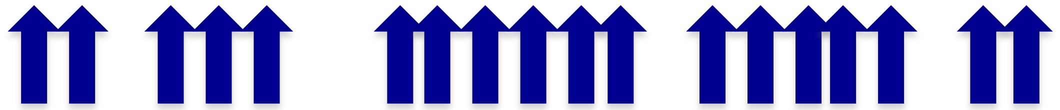
Trivial Mapping Algorithm

The Human Genome:
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:
```
TCGACATGAGATCGGTAGAGCCGT
```

- We can slide our read along the genome and count the total mismatches between the read and the genome.
- If the mismatches are below a threshold, we say that it is a match.

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
TCGACATGAGATCGGTAGAGCCGT
```



Total of 18 mismatches. Not below threshold. Not a match.

Trivial Mapping Algorithm

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:

```
TCGACATGAGATCGGTAGAGCCGT
```

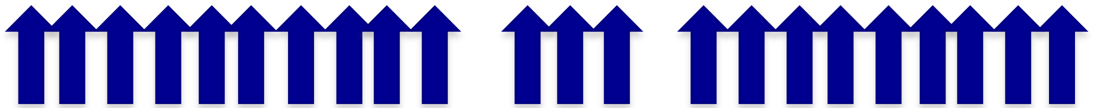
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
 TCGACATGAGATCGGTAGAGCCGT
 ↑ ↑↑↑ ↑↑↑↑↑↑ ↑↑↑   ↑ ↑
```

Total of 15 mismatches. Not below threshold. Not a match.

Trivial Mapping Algorithm

The Human Genome:
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:
```
TCGACATGAGATCGGTAGAGCCGT
```

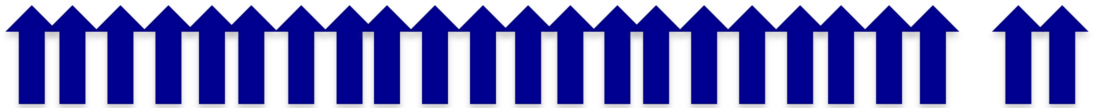
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
  TCGACATGAGATCGGTAGAGCCGT
```


Total of 23 mismatches. Not below threshold. Not a match.

Trivial Mapping Algorithm

The Human Genome:
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:
```
TCGACATGAGATCGGTAGAGCCGT
```

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
        TCGACATGAGATCGGTAGAGCCGT
```


Total of 23 mismatches. Not below threshold. Not a match.

Trivial Mapping Algorithm

The Human Genome:
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC

A Sequence Read:
TCGACATGAGATCGGTAGAGCCGT

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         TCGACATGAGATCGGTAGAGCCGT
           ↑          ↑       ↑
```

Total of 3 mismatches. Below threshold. A match!

Complexity of Trivial Algorithm

- 3,000,000,000 length genome (N)
- 300,000,000 reads to map (M)
- Reads are of length 30 (L)
- Number of mismatches allowed is 2 (D).
- Each comparison of match vs. mismatch takes 1/1,000,000 seconds (t).

Total Time = N*M*L*t = 27,000,000,000,000 seconds or 854,164 years!

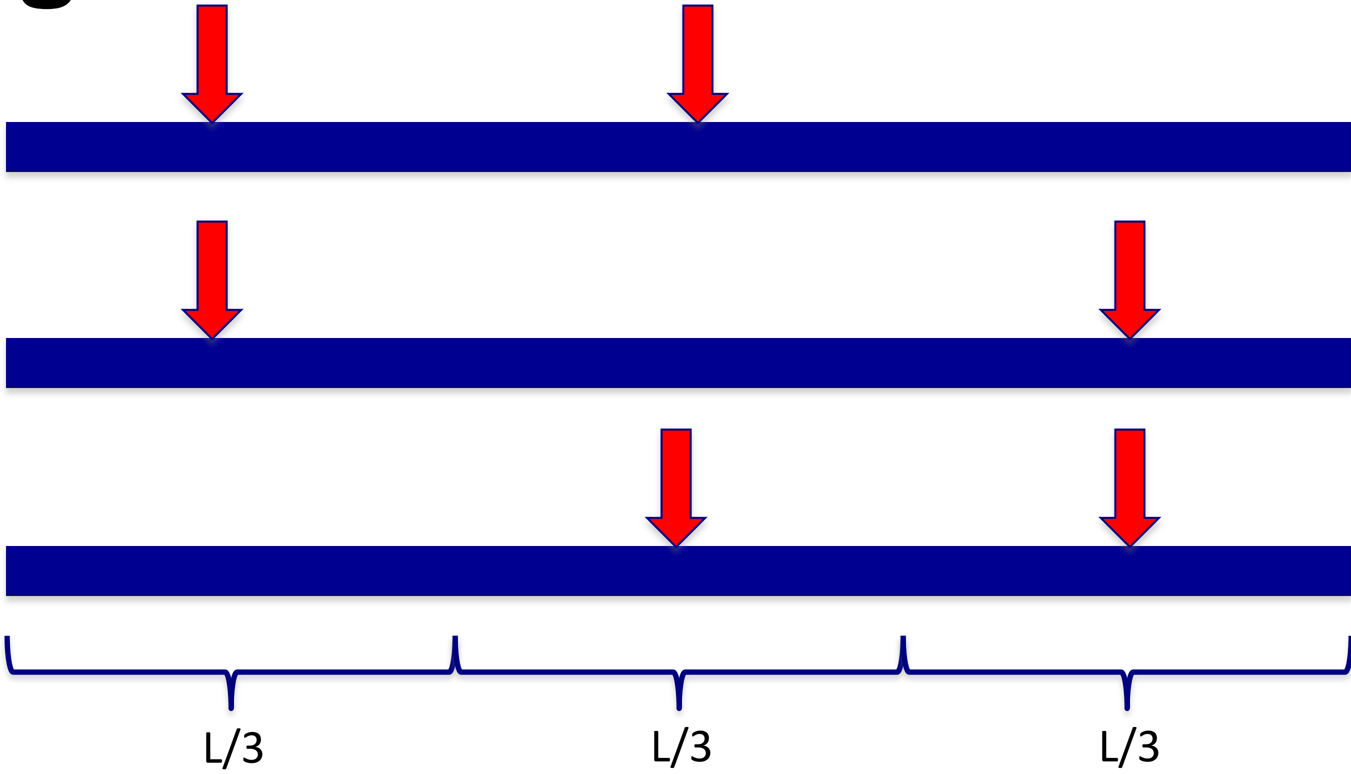
Some observations

- Most positions in the genome match very poorly.
- We are looking for only a few mismatches.

  (D is small)
- A substring of our read will match perfectly.

Perfect Matching Read Substrings

Three "worst" possible cases for placement of mutations.



- In each case, there is a perfect match of L/3.

Finding a perfect match of length L/3

- Intuition: Create an index (or phone book) for the genome.
- We can look up an entry quickly.

Sequence	Positions
AAAAAAAAAA	32453, 64543, 76335
AAAAAAAAAC	64534, 84323, 96536
AAAAAAAAAG	12352, 32534, 56346
AAAAAAAAAT	23245, 54333, 75464
AAAAAAAACA	
AAAAAAAACC	43523, 67543
…	
CAAAAAAAAA	32345, 65442
CAAAAAAAAC	34653, 67323, 76354
…	
TCGACATGAG	54234, 67344, 75423
TCGACATGAT	11213, 22323
…	
TTTTTTTTTG	64252
TTTTTTTTTT	64246, 77355, 78453

If L=30, each entry will have a key of length 10. Each entry will contain on average $N/4^{10}$ positions. (Approximately 3,000).

If L=45, each entry will have a key of length 15. Each entry will contain on average 3 positions.

Complexity of Indexing Algorithm

- We need to look up each third of the read in the index.
- For L=30, our index will contain entries of length 10. Each entry will contain on average $N/(4^{L/3})$ or 3,000 positions.
- For each position, we need to compute the number of mismatches.
- Our running time is $L*M*3*N/(4^{L/3})*T$=81,000,000 seconds or 937 days.
- If L=45, then the time is 81,000 seconds or 22.5 hours.

More problems: Sequencing Errors

- Each sequence read can have some random errors.

My Genome:

```
TACATGAGATCGACATGAGATCGGTAGAGCCGTGAGATC
```

A Sequence Read:

```
TCGACATGAGATCGGTAGAACCGT
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         TCGACATGAGATCGGTAGAACCGT
```

Recovered Sequence:

```
TACATGAGATCGACATGAGATCGGTAGAACCGTGAGATC
```

Sequencing Errors: Solution

- Collect redundant data.

My Genome:

```
TACATGAGATCGACATGAGATCGGTAGAGCCGTGAGATC
```

Sequence Reads:

```
TCGACATGAGATCGGTAGAACCGT
GACAAGAGATCGGTAGAGCCGTGA
TGAGATCGGGAGAGCCGTGAGATC
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         TCGACATGAGATCGGTAGAACCGT
           GACAAGAGATCGGTAGAGCCGTGA
               TGAGATCGGGAGAGCCGTGAGATC
```

Recovered Sequence:

```
TACATGAGATCGACATGAGATCGGTAGAGCCGTGAGATC
```

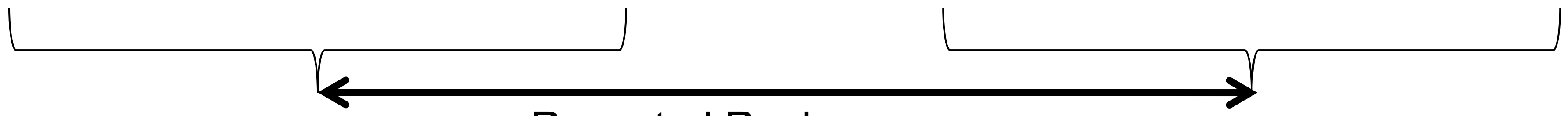
How much coverage do we need?

- If error rate is *e*, and we are going to predict the consensus sequence, what is the error rate if the coverage is 3.
- We will make a prediction with an error if two out of our three reads have an error in the same place.
- This is approximately $3e^2$.

“Re”-Sequencing Problems

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTACATGAGATCCACAT
```



My Genome:

```
TACATGAGATCGACATGAGATCGGTACATGAGATCCACAT
```

A Sequence Read:

```
ACATGAGATCGACAT
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTACATGAGATCCACAT
 ACATGAGATCGACAT          ACATGAGATCGACAT
```

Error!

Recovered Sequence:

```
TACATGAGATCGACATGAGATCGGTACATGAGATCGACAT
```

“Re”-Sequencing Problems

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTACATGAGATCCACAT
```

My Genome:

```
TACATGAGGGGGGGGGGAGATCGGTACATGAGATCCACAT
```

A Sequence Read:

```
GAGGGGGGGGG
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTACATGAGATCCACAT
        GAGGGGGGGGG
```

Too many mismatches to match the read to the reference.
Since we don’t know where it came from, we can’t identify the difference in the target seqeunce.

"Re"-Sequencing: Insertions

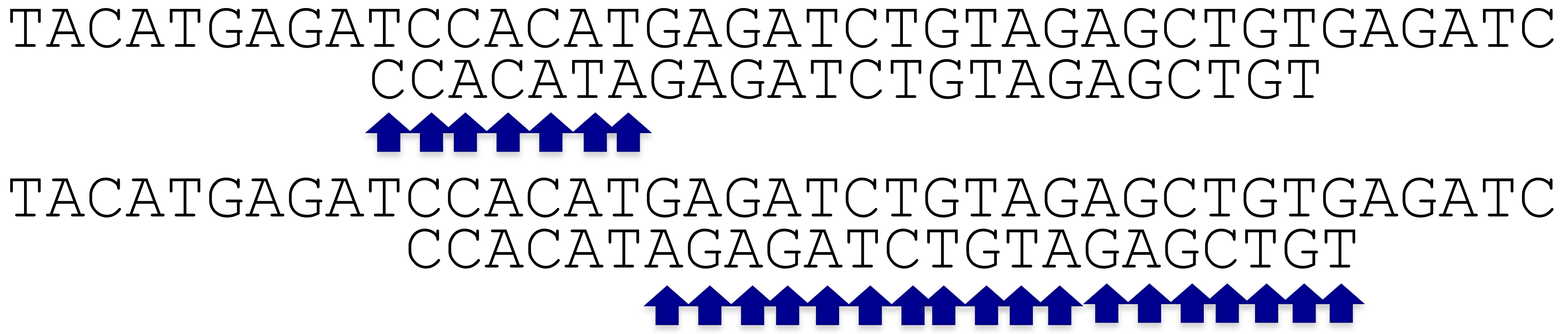
My Genome:

```
TACATGAGATCCACATAGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:

```
CCACATAGAGATCTGTAGAGCTGT
```

The Human Genome:



```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         CCACATAGAGATCTGTAGAGCTGT

TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

How do we deal with this case?

“Re”-Sequencing: Insertions

My Genome:

```
TACATGAGATCCACATAGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:

```
CCACATAGAGATCTGTAGAGCTGT
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         CCACATAGAGATCTGTAGAGCTGT
```

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

Solution: Add Insertion to the Human Genome

```
TACATGAGATCCACAT-GAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

Many other challenges

- Coverage of sequence reads is not uniform
  - **Some places we have many reads, while some we have fewer. How do we design an approach so we can always recover the sequence.**
- Large memory requirements
  - **We need to fit our index into RAM. Often tens of Gigabytes or greater.**

Sequencing Coverage

Lecture 2.

April 6th, 2023

(Slides from Jae-Hoon Sul)

Sequence Mapping Coverage

- If a genome is length N (human is 3,000,000,000), and the total length of all sequence reads collected is M, the coverage ratio is defined at M/N.
- Often written with an “x”. For example, 10x or 20x coverage.

Sequencing Coverage Statistics

- If length of the genome is N the probability of the event that a single read position starts at a single position in the genome is 1/N (very small).
- If the number of reads is K, the total number of read positions that start at a single genome position is the number of times that an event with probability 1/N happens out of K trials.
- Poisson distribution.

Sequencing Coverage Statistics

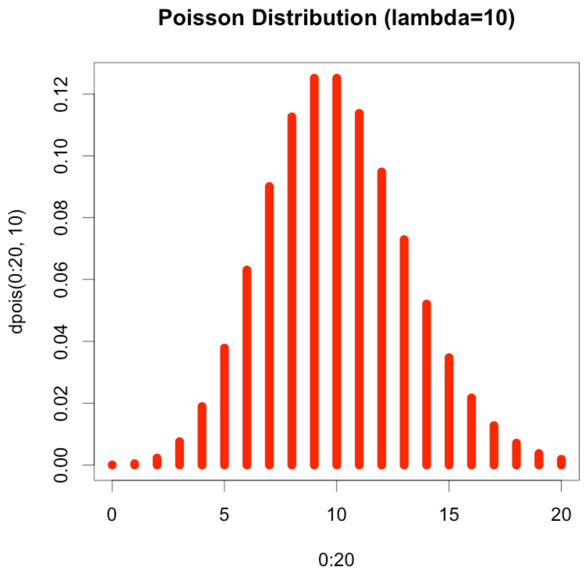
- If length of the genome is N the probability of the event that a single read of length L position spanning a single position in the genome approximately L/N (also very small).
- If the sum of the length of all K reads of length L is M=K*L, the total number of read positions that span a single genome position is approximately the number of times that an event with probability 1/N happens out of M trials.
- Approximately Poisson distribution.

Poisson Distribution

- Discrete probability distribution to compute probability of (rare) events given known mean
- Only one parameter: λ, mean of distribution
- Probability Mass Function

$$\Pr(N_t = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

- Mean = λ
- Variance = λ



Poisson Distribution to Sequencing Coverage

- λ=M/N.
- Probability that exactly X reads span a certain position.
  - **dpois(X, λ)**
- Probability that X or fewer reads span a certain position.
  - **ppois(X, λ)**
- At least Y% of the genome is covered with this much or fewer reads
  - **qpois(Y, λ)**

Poisson and Sequencing Coverage

- Probability that X or fewer reads span a certain position.
  - **ppois(X,λ) =** $\sum_{i=0}^{X} \mathrm{dpois}(i,\lambda)$

Coverage examples

- For human genome (L=3,000,000,000) sequenced at 30x coverage, what is the probability that a specific location has exactly 30 coverage?
- **λ=30 dpois(30,λ)=dpois(30,30)=0.072**
- What is the probability that a specific location has at least 30 coverage?
- **1-ppois(29,λ)=1-ppois(29,30)=0.524**
- What is the probability that a specific location has at least 10 coverage?
- **1-ppois(9,30)=0.9999929**

Coverage examples

- For human genome (L=3,000,000,000) sequenced at 30x coverage, what is the probability that a specific location has exactly one read spanning it?
- **λ=30 dpois(1,λ)=2.9x10^{-12}**
- What is the probability that a specific location has at least 6 coverage?
- **λ=30 1-ppois(5,λ)=.99999**
- How many positions in the genome have less then 6 coverage ?
- **3,000,000,000*ppois(5,λ)=67.7**

Diploid Coverage

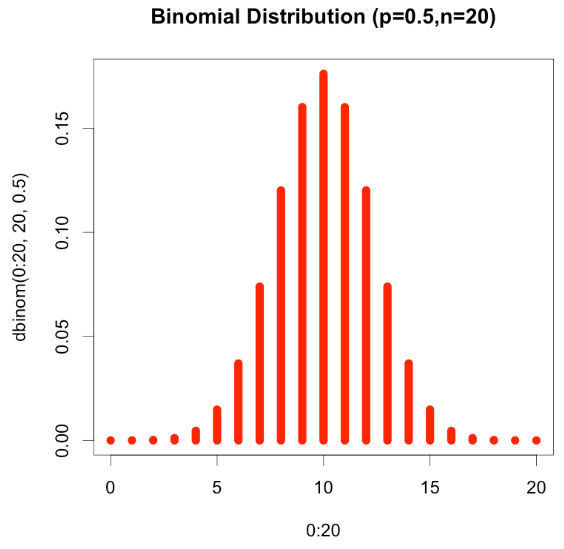
- Since humans have 2 chromosomes each read comes from one chromosome at random. If a position in the reference is covered by Y reads, the probability that X of the reads come from the first chromosome follows the binomial distribution with parameter .5.
  - **dbinom(X,Y,0.5)**
- At least X coverage for each chromosome out of Y reads $$\sum_{i=X}^{Y-X} \mathrm{dbinom}(i, Y, 0.5)$$

Binomial Distribution

- Discrete probability distribution to compute probability of having X successes in Y trials
- Example: What's the probability of having k heads in n tosses with fair coin (p = 0.5)?
- Probability Mass Function

$$\binom{n}{k} p^k (1-p)^{n-k}$$

- Mean = n*p
- Variance = n*p*(1–p)



Diploid Coverage Examples

- If a position is covered by 10 reads, what is the probability that exactly 3 reads come from the first chromosome?
- **dbinom(3,10,.5)=.117**
- If a position is covered by 10 reads, what is the probability that at least 4 reads come from the first chromosome?
- **1-pbinom(3,10,.5)=.828**
- If a position is covered by 10 reads, what is the probability that at least 4 reads come from each chromosome?
- **dbinom(4,10,.5)+dbinom(5,10,.5)+dbinom(6,10,.5)=.656**

Minimum Diploid Coverage

- If we want the sequence coverage is λ=M/N, the portion of the genome that has at least X coverage of each chromosome is

$$\sum_{i=2X}^{\infty} \mathrm{dpois}(i,\lambda) \sum_{j=X}^{i-X} \mathrm{dbinom}(j,i,0.5)$$

Diploid Coverage Examples

- If genome is covered with coverage 30, what is the probability that a position will have at least 10 reads from each chromosome?

$$\sum_{i=20}^{\infty} \mathrm{dpois}(i, 30) \sum_{j=10}^{i-10} \mathrm{dbinom}(j, i, 0.5)$$

SNP Calling

- Inferring single base differences from sequencing.
- Several challenges:
  - **Sequencing errors**
  - **Alignment “mapping” problems**
  - **Statistical Uncertainty**

SNP Calling Standard Approaches

- Consensus Algorithm
  - **Map reads to genome**
  - **Place read in best mapping position (randomly break ties)**
  - **SNP call is based on majority vote.**
- Probabilistic Algorithm
  - **Map reads to genome**
  - **Place read in best mapping position (randomly break ties)**
  - **Compute "posterior probablility"**
- Mapping uncertainty methods
  - **Map reads to genome**
  - **Record mapping uncertainty**
  - **Compute "posterior probability" incorporating mapping uncertainty**

Sequencing Errors

- Each sequence read can have some random errors.

My Genome:

TACATGAGATC**G**ACATGAGATC**G**GTAGAGC**C**GTGAGATC

A Sequence Read:

TCGACATGAGATCGGTAGA**A**CCGT

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         TCGACATGAGATCGGTAGAACCGT
```

Recovered Sequence:

TACATGAGATC**G**ACATGAGATC**G**GTAGA**A**C**C**GTGAGATC

Consensus Algorithm

- Take majority vote.

My Genome:
TACATGAGATCGACATGAGATCGGTAGAGCCGTGAGATC

Sequence Reads:

```
TCGACATGAGATCGGTAGAACCGT
GACAAGAGATCGGTAGAGCCGTGA
TGAGATCGGGAGAGCCGTGAGATC
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         TCGACATGAGATCGGTAGAACCGT
           GACAAGAGATCGGTAGAGCCGTGA
               TGAGATCGGGAGAGCCGTGAGATC
```

Recovered Sequence:
TACATGAGATCGACATGAGATCGGTAGAGCCGTGAGATC

How much coverage do we need?

- If error rate is *e*, and we are going to predict the consensus sequence, what is the error rate if the coverage is X.
- We will make a prediction with an error more than X/2-1 out of the X reads have an error in the same place.
- **1-pbinom(X/2,X,e)**

How much coverage do we need?

- If error rate is *e*, and we are going to predict the consensus sequence, what is the error rate if the coverage is 3.
- We will make a prediction with an error if two out of our three reads have an error in the same place.

$$\text{pbinom}(2,3,e) = e^3 + \begin{pmatrix} 3 \\ 2 \end{pmatrix}(1-e)e^2$$

- This is approximately $3e^2$.

Diploid Sequencing

- Humans have 2 chromosomes.
- Each chromosome may have a different SNP.
- Some reads come from 1 chromosome, some come from other chromsome.
- Why does consensus method not work?
- How do we address this problem?

Insertions and Deletions

Lecture 2.

April 6th, 2023

"Re"-Sequencing: Insertions

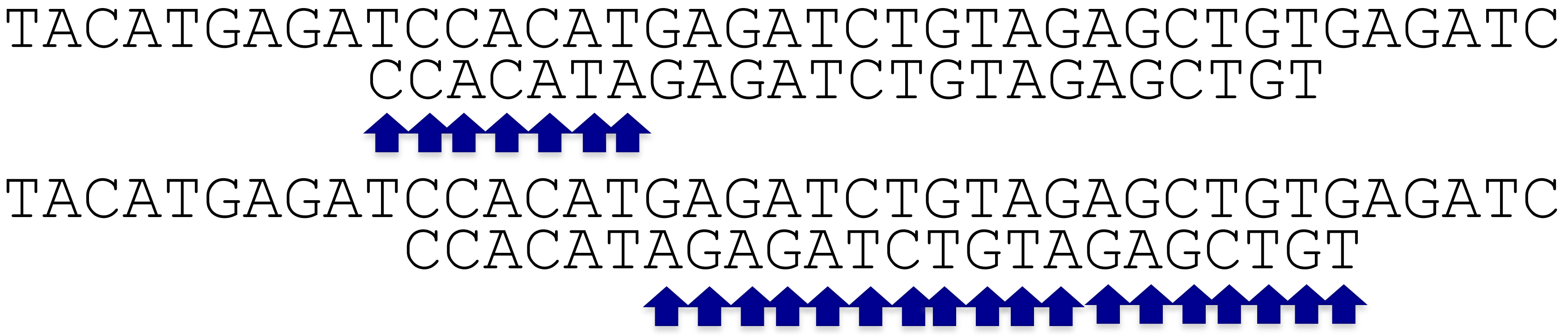
My Genome:

TACATGAGATCCACATAGAGATCTGTAGAGCTGTGAGATC

A Sequence Read:

CCACATAGAGATCTGTAGAGCTGT

The Human Genome:



```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         CCACATAGAGATCTGTAGAGCTGT

TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

How do we deal with this case?

“Re”-Sequencing: Insertions

My Genome:

```
TACATGAGATCCACATAGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:

```
CCACATAGAGATCTGTAGAGCTGT
```

The Human Genome:

```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
         CCACATAGAGATCTGTAGAGCTGT
```

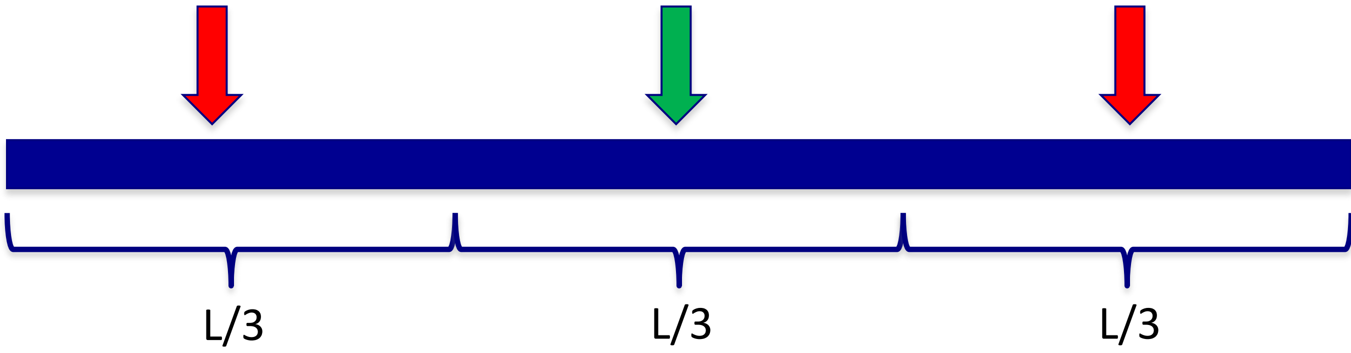
```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

Solution: Add Insertion to the Human Genome

```
TACATGAGATCCACAT-GAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT
```

Indel in Middle of Read

If indel is in the middle of read.



- Both outside regions of size L/3 will match perfectly.
- Because of coverage, indel will be in middle at least for one read.
- Important: Middle distance will be L/3+1 or L/3-1

"Re"-Sequencing: Insertions

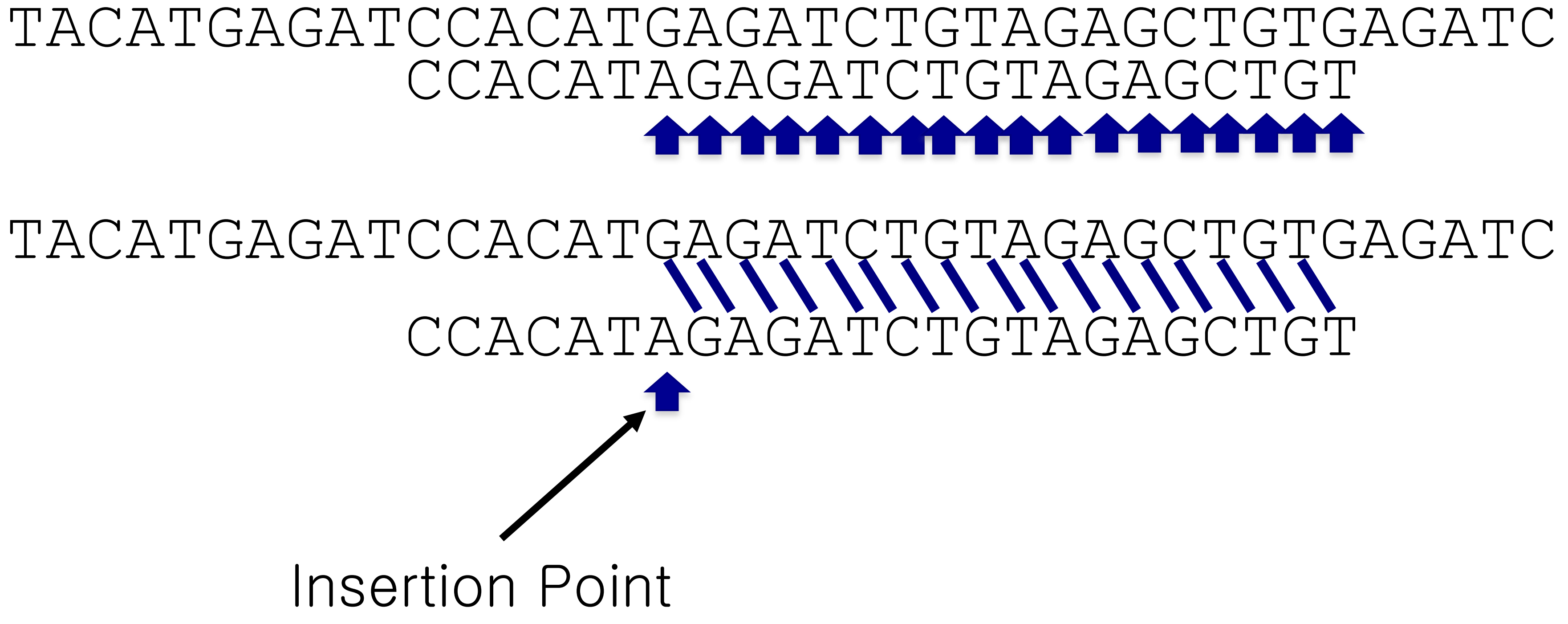
My Genome:

```
TACATGAGATCCACATAGAGATCTGTAGAGCTGTGAGATC
```

A Sequence Read:

```
CCACATAGAGATCTGTAGAGCTGT
```

The Human Genome:



```
TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC
          CCACATAGAGATCTGTAGAGCTGT

TACATGAGATCCACATGAGATCTGTAGAGCTGTGAGATC

          CCACATAGAGATCTGTAGAGCTGT
```

Indel Algorithms

- Trivial Algorithm
  - **Try all inerstion points for a read**
  - **If read matches (with insertion) below number of mismatches, then we desclare a match and identify and indel**
- More Efficient Algorithm
  - **Look for perfect match in first part of read**
  - **Try insertion point at point of first mismatch**
  - **More complicated but faster**
- More accurate Algorithm
  - **Perform alignment between read and reference**
- Extremely Accurate Algorithm
  - **Align all reads with indel together.**
  - **Multiple Sequence Alignment!**

“Re”-Sequencing + Burroughs Wheeler Transform

Lecture 2.

April 6th, 2023

(Some slides from Ben Langmead)

Index for L/3 (is BIG!)

- Intuition: Create an index (or phone book) for the genome.
- We can look up an entry quickly.

Sequence	Positions
AAAAAAAAAA	32453, 64543, 76335
AAAAAAAAAC	64534, 84323, 96536
AAAAAAAAAG	12352, 32534, 56346
AAAAAAAAAT	23245, 54333, 75464
AAAAAAAACA	
AAAAAAAACC	43523, 67543
…	
CAAAAAAAAA	32345, 65442
CAAAAAAAAC	34653, 67323, 76354
…	
TCGACATGAG	54234, 67344, 75423
TCGACATGAT	11213, 22323
…	
TTTTTTTTTG	64252
TTTTTTTTTT	64246, 77355, 78453

If L=30, each entry will have a key of length 10. Each entry will contain on average $N/4^{10}$ positions. (Approximately 3,000).

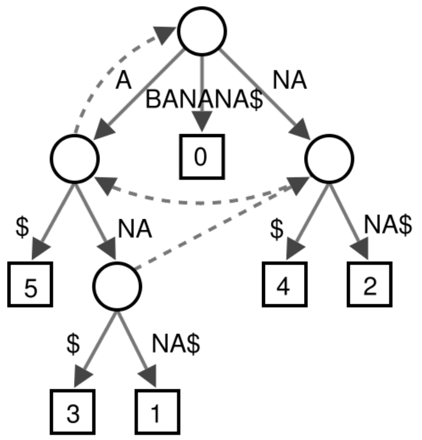
If L=45, each entry will have a key of length 15. Each entry will contain on average 3 positions.

Indexing a genome

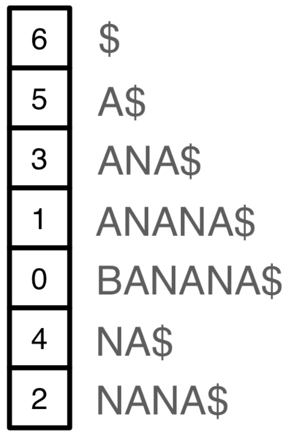
- To find exactly matching substrings, we need to build an index for the whole genome.
- Problem: The genome is BIG!

Indexing

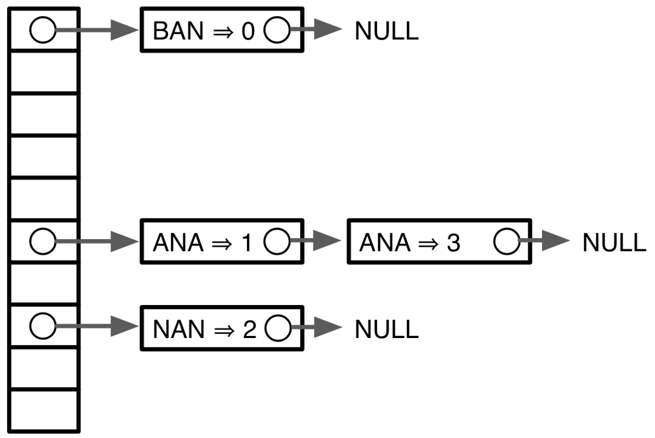
- Genome indices can be big. For human:



> 35 GBs



> 12 GBs



> 12 GBs

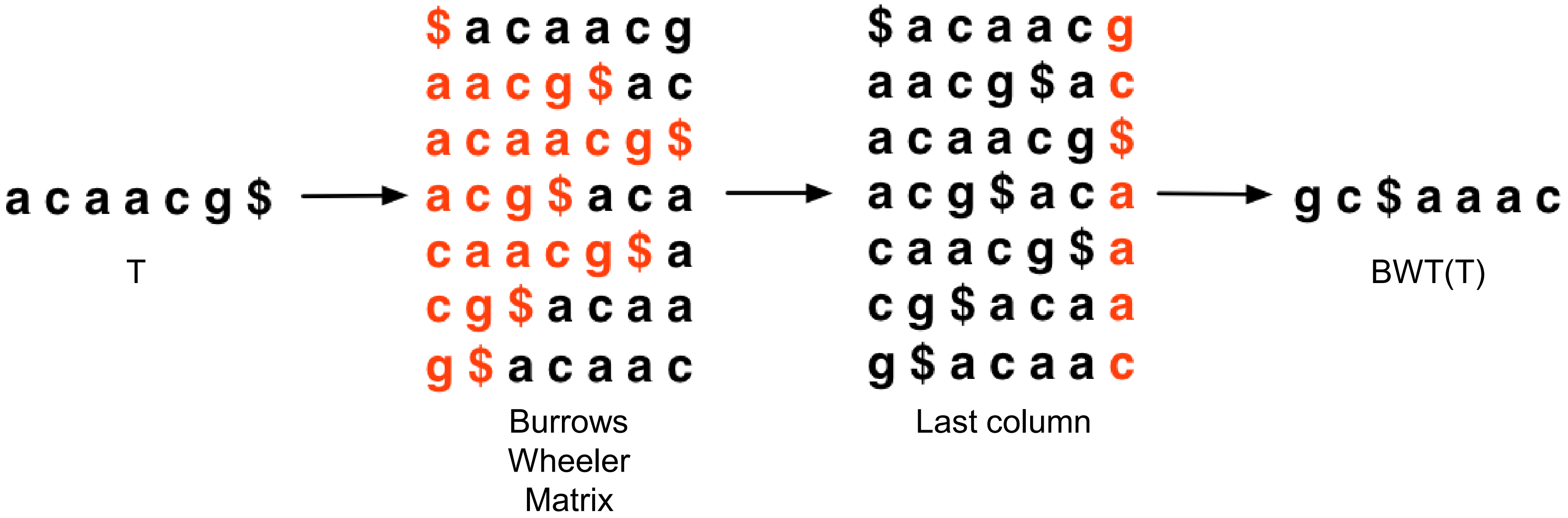
- Large memory requirement implications
  - **Requires large memory machine (expensive)**
  - **Partition genome and index each part (slow)**

Memory Efficient but Slow Algorithm

- Store just the sequence
  - **4 DNA bases per byte (2 bits each)**
  - **3,000,000,000 / 4 ~ 750 MB**
- When looking up string, just loop through the sequence.
- Very slow, but very memory efficient!

Burrows-Wheeler Transform

- http://en.wikipedia.org/wiki/Burrows-Wheeler_transform
- Reversible permutation used originally in compression



- Once BWT(T) is built, *all else shown here is discarded*
  - **Matrix will be shown for illustration only**

Burrows M, Wheeler DJ: A block sorting lossless data compression algorithm. *Digital Equipment Corporation, Palo Alto, CA* 1994, Technical Report 124; 1994

Burrows-Wheeler Transform

- Store only last column
- First column can be recovered by counting symbols in last column because it is sorted

```
a c a a c g $
```

T

```
$ a c a a c g
a a c g $ a c
a c a a c g $
a c g $ a c a
c a a c g $ a
c g $ a c a a
g $ a c a a c
```

Burrows Wheeler Matrix

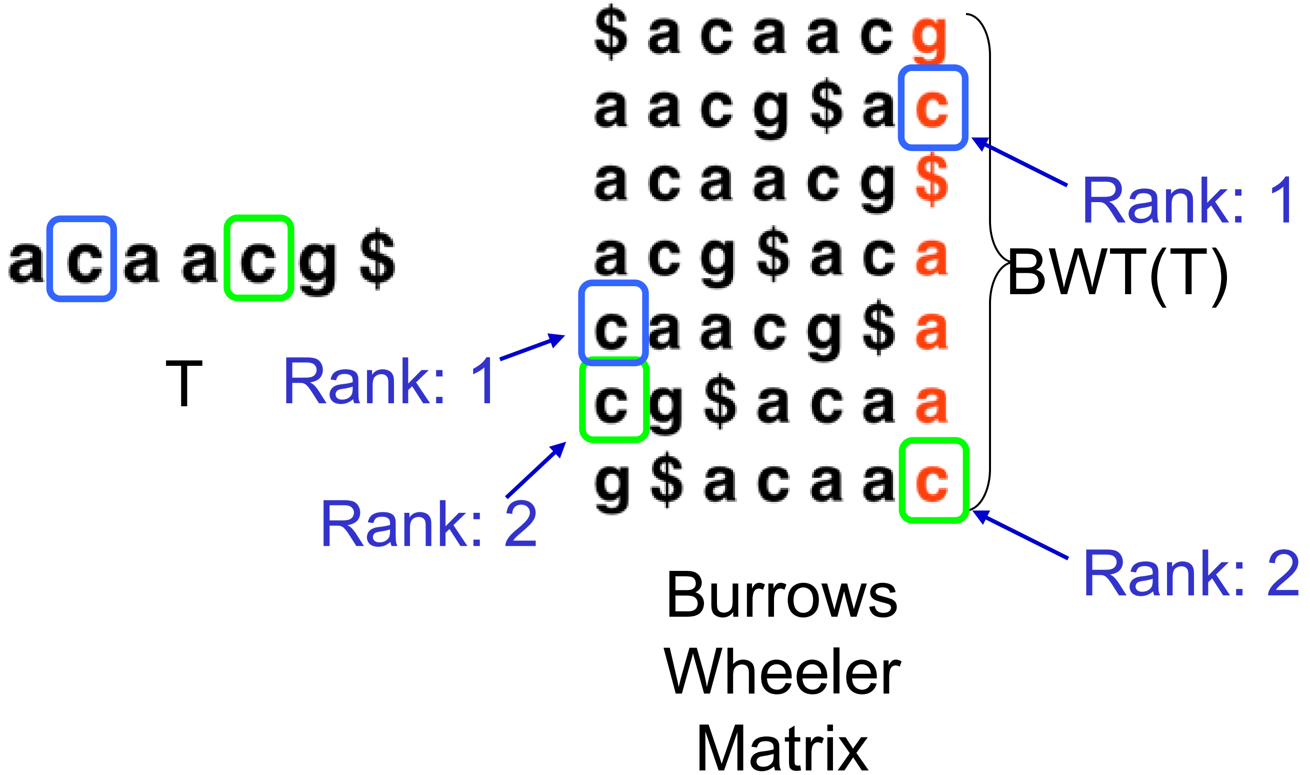
BWT(T)

$\boxed{\text{A:3 C:2 G:1 T:0}}$

```
$
a
a
a
c
c
g
```

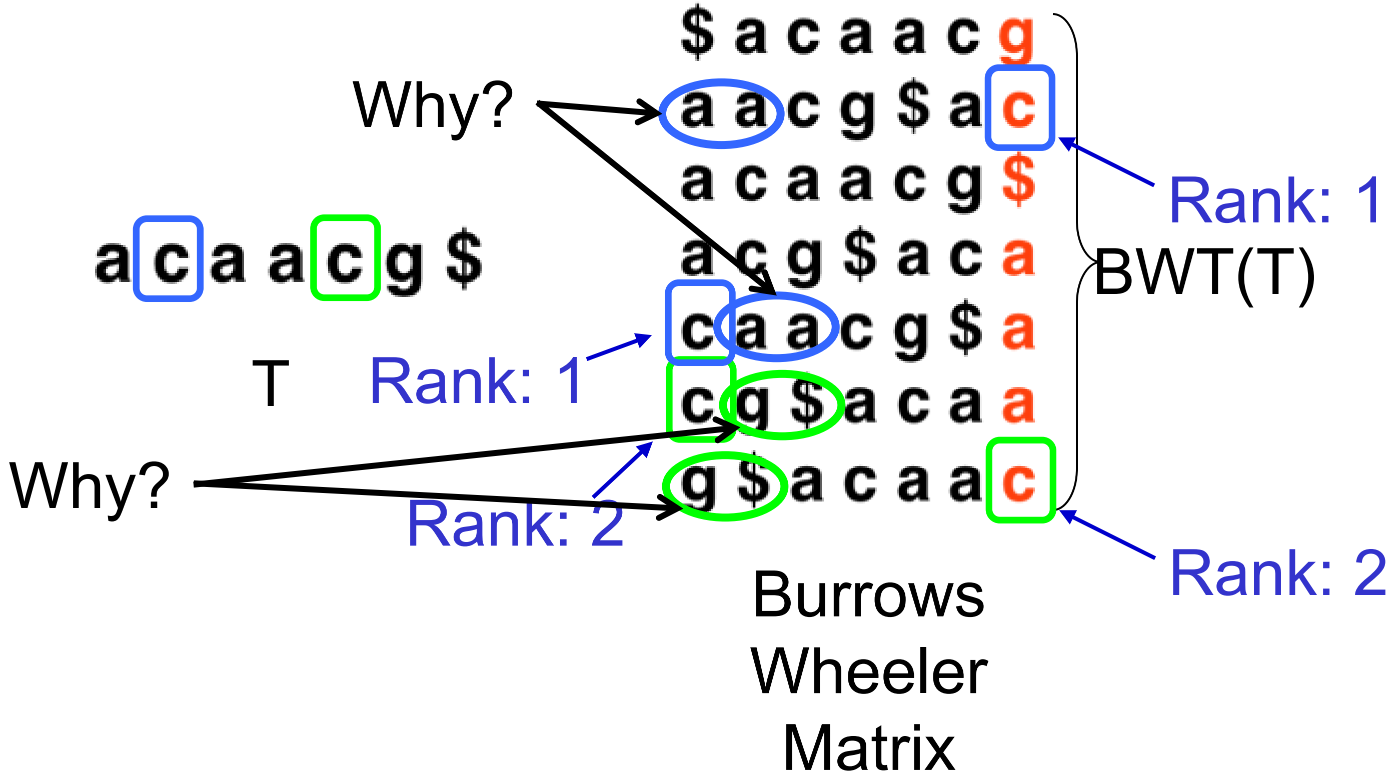
Burrows-Wheeler Transform

- Property that makes BWT(T) reversible is "LF Mapping"
  - i^th occurrence of a character in Last column is same *text* occurrence as the i^th occurrence in First column



Burrows-Wheeler Transform

- Property that makes BWT(T) reversible is "LF Mapping"
  - ith occurrence of a character in Last column is same *text* occurrence as the ith occurrence in First column

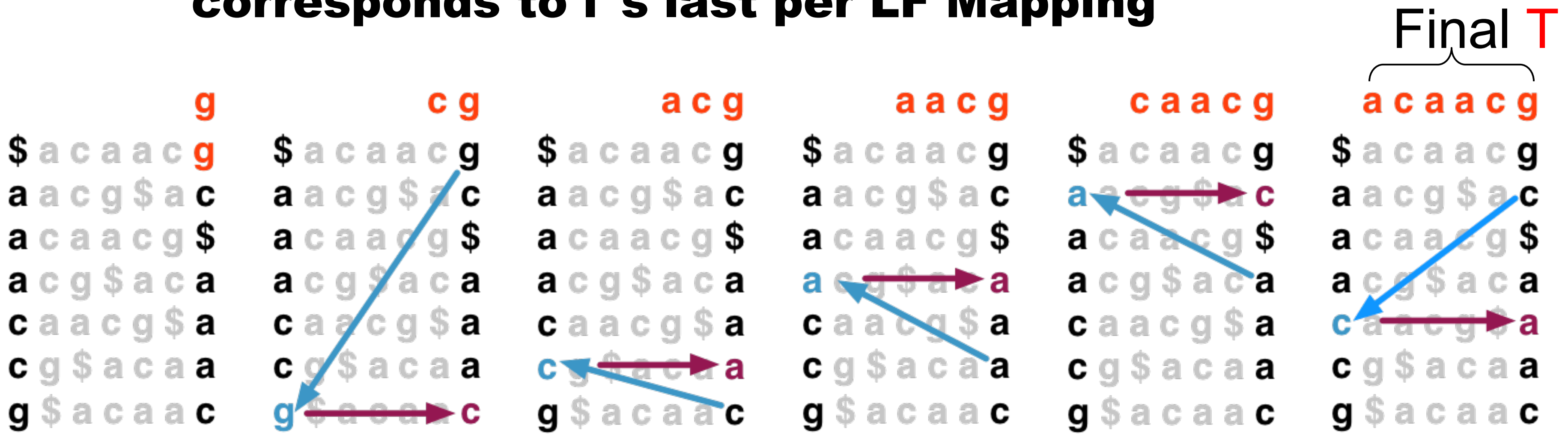


Burrows-Wheeler Transform

- To recreate T from BWT(T), repeatedly apply rule:

  T = BWT[LF(i)] + T; i = LF(i)

  - **Where LF(i) maps row i to row whose first character corresponds to i's last per LF Mapping**



- Could be called "unpermute" or "walk-left" algorithm

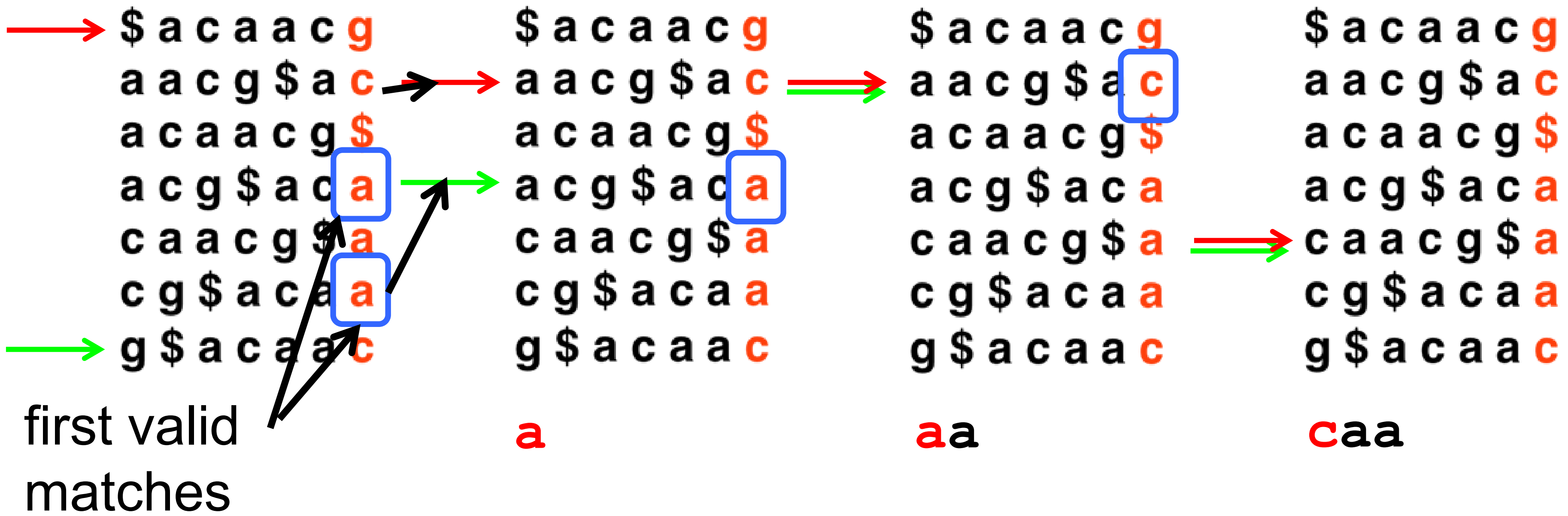
FM Index

- Ferragina & Manzini propose “FM Index” based on BWT
- Observed:
  - **LF Mapping also allows *exact matching* within T**
  - **LF(i) can be made fast with *checkpointing***
  - **...and more (see FOCS paper)**

- Ferragina P, Manzini G: Opportunistic data structures with applications. *FOCS. IEEE Computer Society; 2000.*
- Ferragina P, Manzini G: An experimental study of an opportunistic index. *SIAM symposium on Discrete algorithms*. Washington, D.C.; 2001.

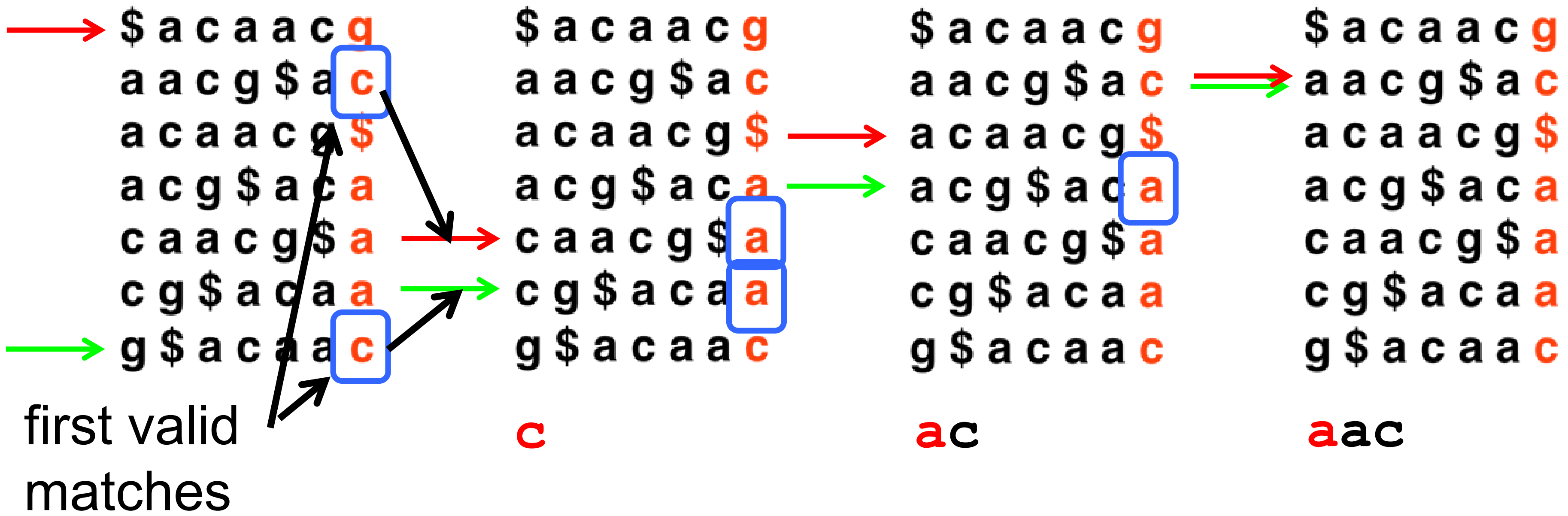
Exact Matching with FM Index

- Look up pattern in reverse.
- Use 2 pointers to represent range of matches.
- Find first valid match for next symbol in range.
  - **Example: searching for "caa"**



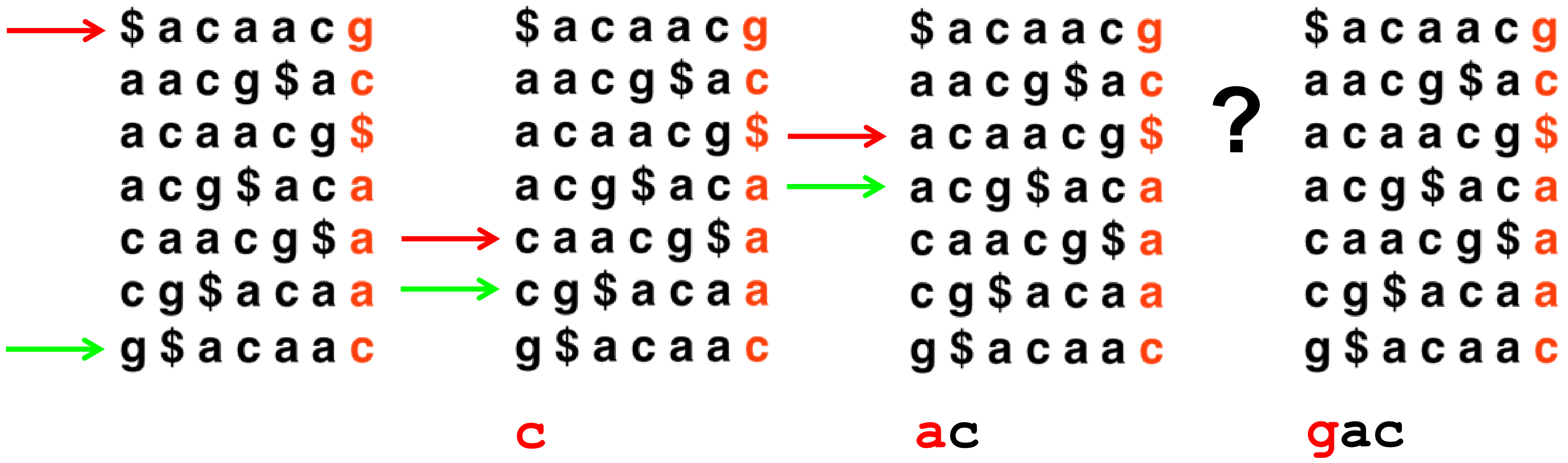
Exact Matching with FM Index

- Look up pattern in reverse.
- Use 2 pointers to represent range of matches.
- Find first valid match for next symbol in range.
  - **Example: searching for "aac"**



Exact Matching with FM Index

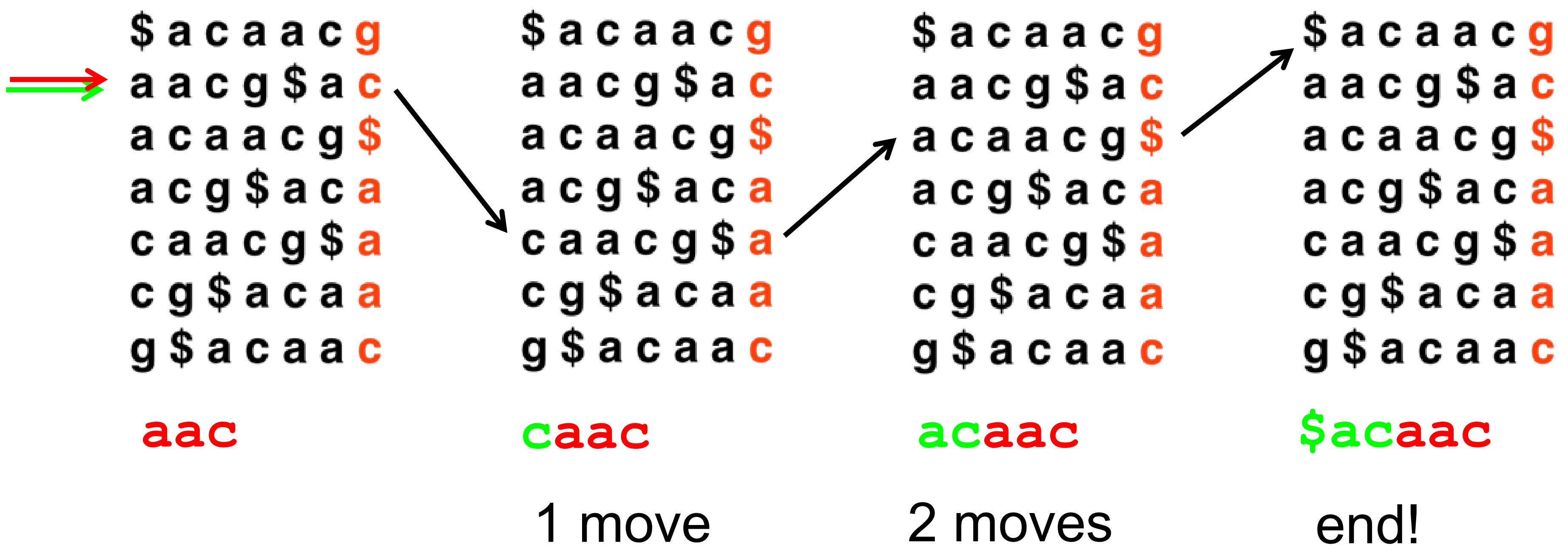
- If no match…
  - **Example: searching for "gac"**



- Pointers will get lost.
- FM index can quickly check for a match.

Where in sequence is the match?

- Use "walk-left" to build sequence to start
- Count number of sequences
  - **Example: searching for "aac"**

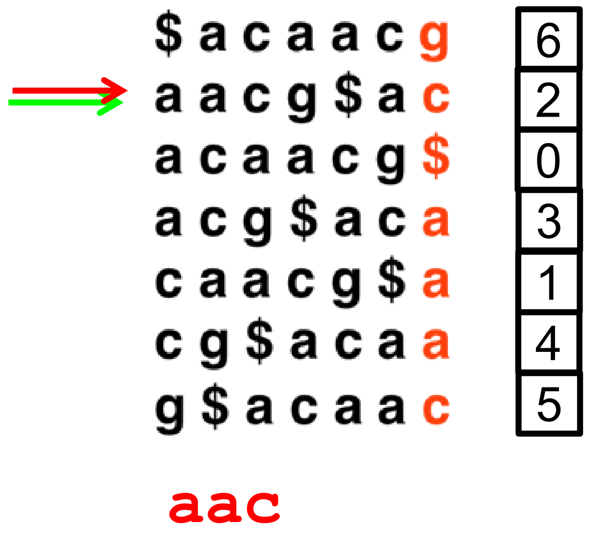


```
$acaacg    $acaacg    $acaacg    $acaacg
aacg$ac    aacg$ac    aacg$ac    aacg$ac
acaacg$    acaacg$    acaacg$    acaacg$
acg$aca    acg$aca    acg$aca    acg$aca
caacg$a    caacg$a    caacg$a    caacg$a
cg$acaa    cg$acaa    cg$acaa    cg$acaa
g$acaac    g$acaac    g$acaac    g$acaac

aac        caac       acaac      $acaac

           1 move     2 moves    end!
```

- Number of moves back is start position of match
  - **Example: "aac" is in position 2.**

Where in sequence is match?

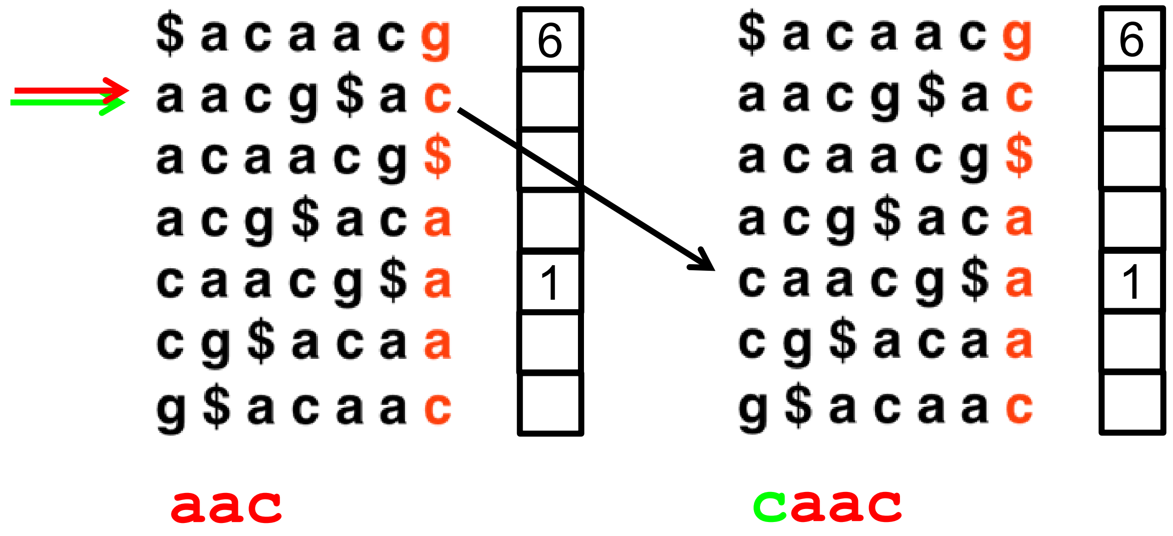
- "walk-left" to start of sequence is slow
- Requires on average N/2 steps to reach start.
- Alternate strategy: keep index of positions.
  - **Example: searching for "aac"**



- Problem: requires as much storage as hashtable!

Where in sequence is match?

- Key Idea: Store fraction of array (sampling)
- Only store some positions and "walk-left"
- Combines two previous strategies
  - **Example: searching for "aac"**



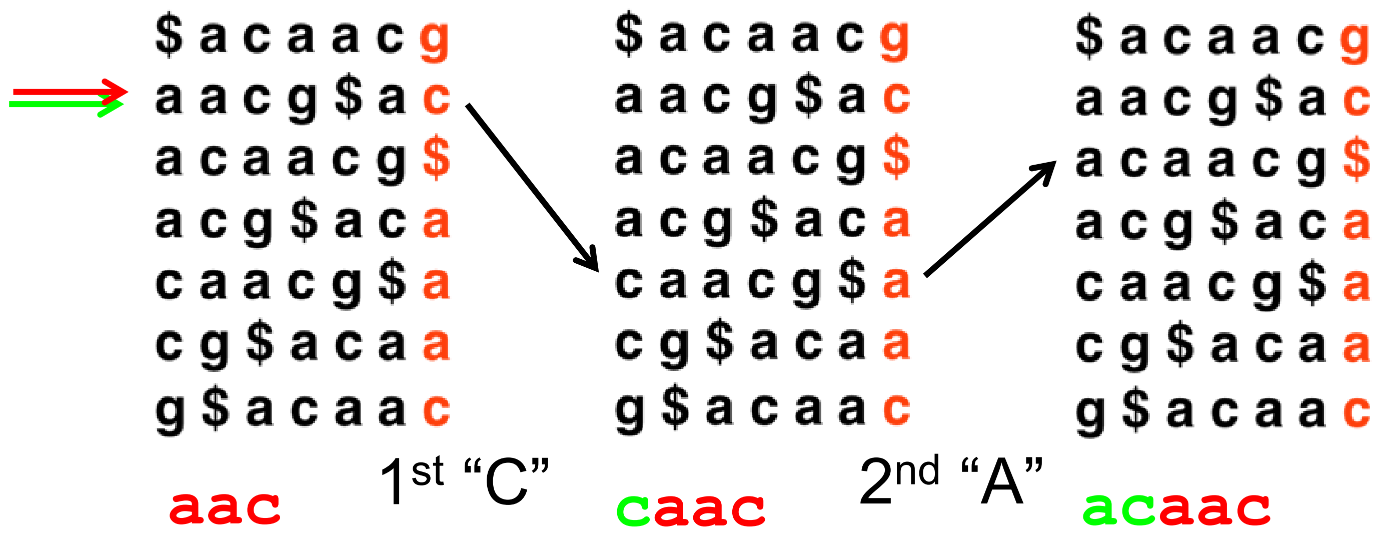
Position = number of moves + position in array

For "aac" = 1 + 1 = 2

- How many values to store provides defines time/space tradeoff.

"walk-left" optimization

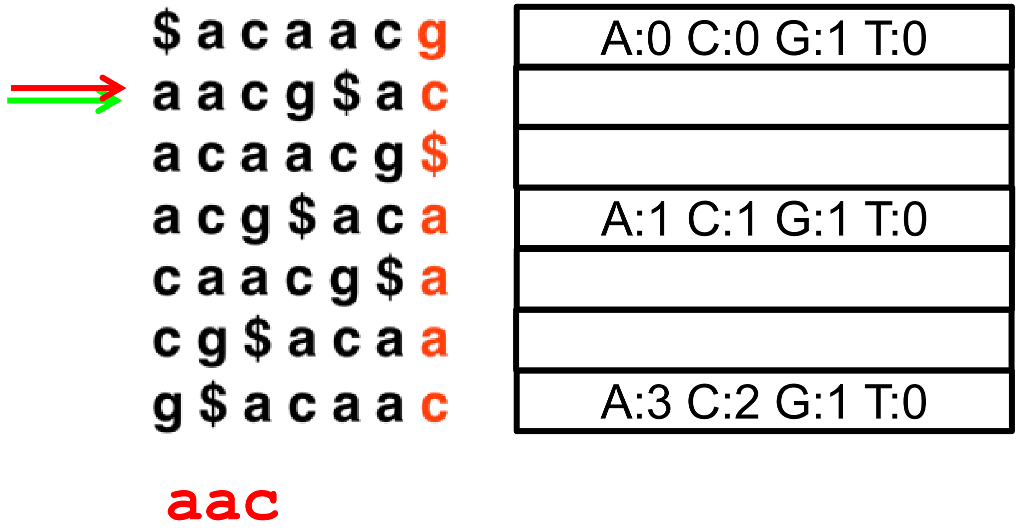
- Each "walk-left" requires counting previous occurrences of symbol in BWT
  - **Example: searching for "aac"**



- Requires counting occurrences in N/2 length string
- Really slow!

"walk-left" optimization

- Idea: use checkpoints to store previous counts
  - **Example: searching for "aac"**



- Requires counting occurrences only until checkpoint.
- Really fast!

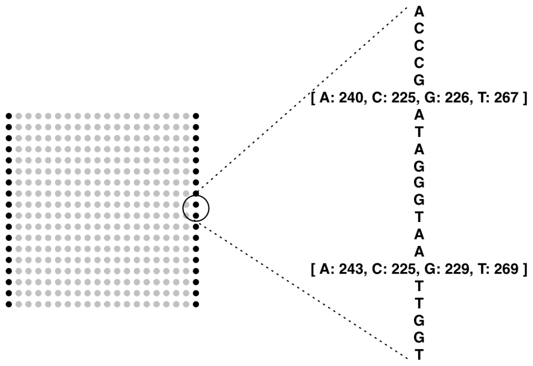
FM Index is Small (Bowtie)

- Entire FM Index on DNA reference consists of:
  - **BWT (same size as T)**
  - **Checkpoints (~15% size of T)**
  - **SA sample (~50% size of T)**
- Total: ~1.65x the size of T

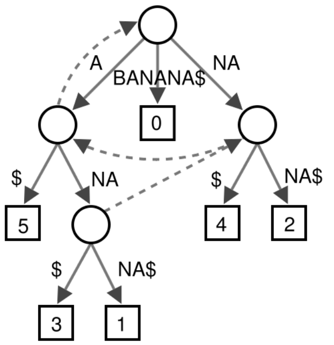
Assuming 2-bit-per-base encoding and no compression, as in Bowtie

Assuming a 16-byte checkpoint every 448 characters, as in Bowtie

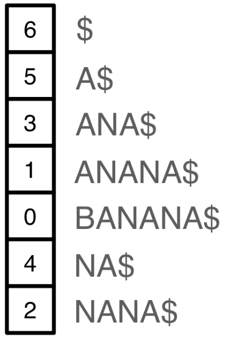
Assuming Bowtie defaults for suffix-array sampling rate, etc


A
C
C
C
G
[A: 240, C: 225, G: 226, T: 267]
A
T
A
G
G
G
T
A
A
[A: 243, C: 225, G: 229, T: 269]
T
T
G
G
T

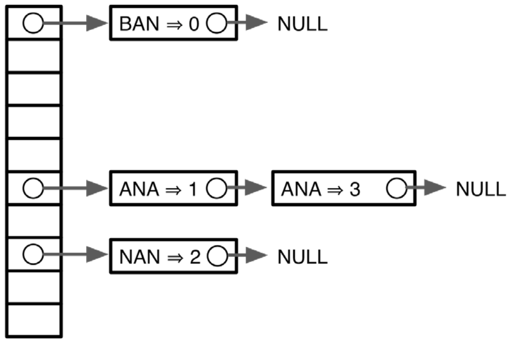
~1.65x



>45x



>15x



>15x

Reference Paper

- Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10:R25.
  - (Some slides from paper)