CS 600.226: Data Structures

Michael Schatz

Nov 12, 2018

Lecture 31: Suffix Arrays

HW8

Assignment 8: Competitive Spelling Bee

Out on: November 2, 2018

Due by: November 9, 2018 before 10:00 pm

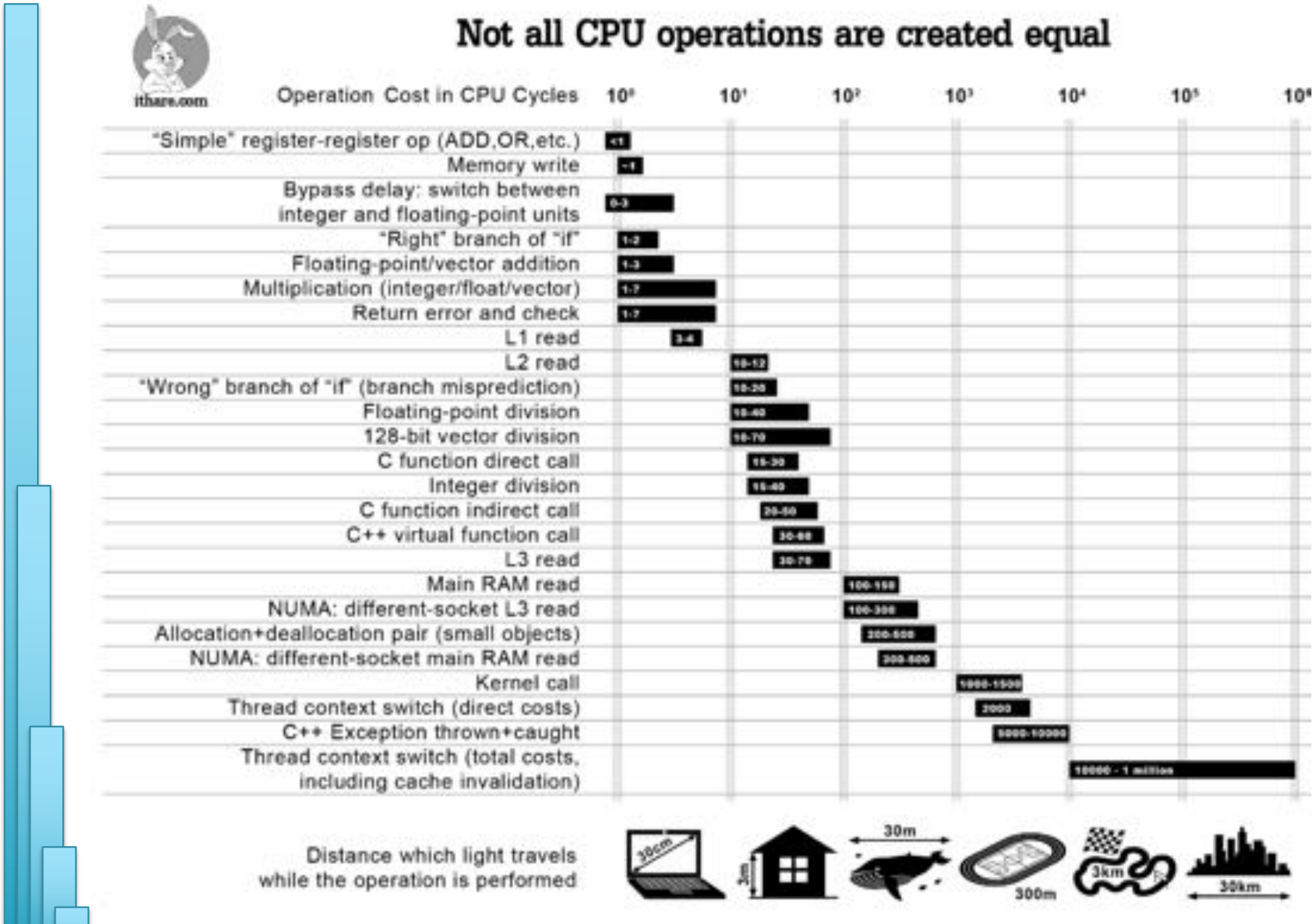
Collaboration: None

Grading:

- Packaging 10%,
- Style 10% (where applicable),
- Testing 10% (where applicable),
- Performance 30% (where applicable),
- Functionality 40% (where applicable)

Overview

Your "one" task for this assignment is to take the simple spell checker we give you and to turn it into the fastest, most memory-efficient spell checker in the course, subject to the constraints detailed below. You are expected to do this by (once again) implementing the Map interface, this time using one of several hash table techniques (your choice, see below).



http://ithare.com/infographics-operation-costs-in-cpu-clock-cycles/

Hashing

Array[13] = 10; Array[42] = 15 ***O(1)***

Array["Mike"] = 10; Array["Peter"] = 15 ***BST:O(lg n) -> Hash:O(1)***

Hash Function enables Map<K, V> as Map<Integer, V> as Array<V>

- h(): K -> Integer for any possible key K
- h() should distribute the keys uniformly over all integers
- if k_1 and k_2 are "close", $h(k_1)$ and $h(k_2)$ should be "far" apart

Typically want to return a small integer, so that we can use it as an index into an array

- An array with 4B cells in not very practical if we only expect a few thousand to a few million entries
- How do we restrict an arbitrary integer x into a value up to some maximum value n?

```
0 <= x % n < n
```

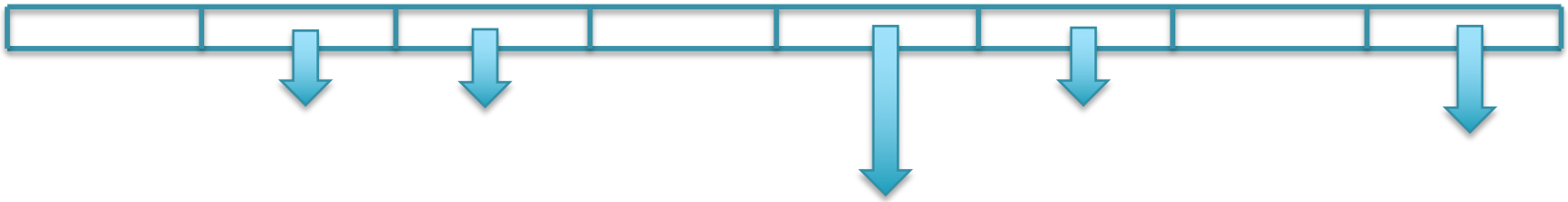
Compression function: c(i) = abs(i) % length(a)

Transforms from a large range of integers to a small range (to store in array a)

Collision Strategies

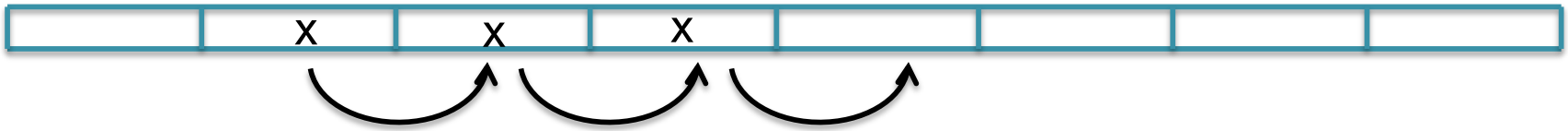
1. Separate chaining:

More object overhead, but degrades gracefully as load approaches 1



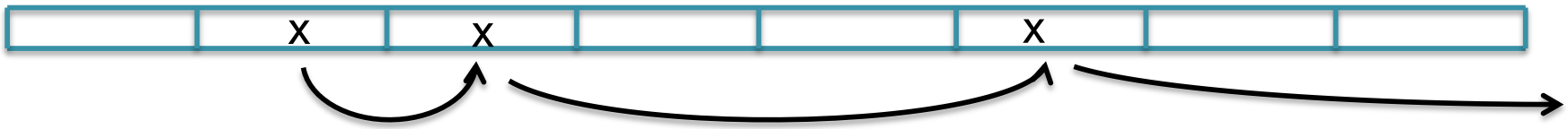
2. Linear Probing

Minimal overhead, easy to implement, but tends to form large clusters



3. Quadratic Probing

Slightly more complex, but skips over larger and larger amounts to find holes



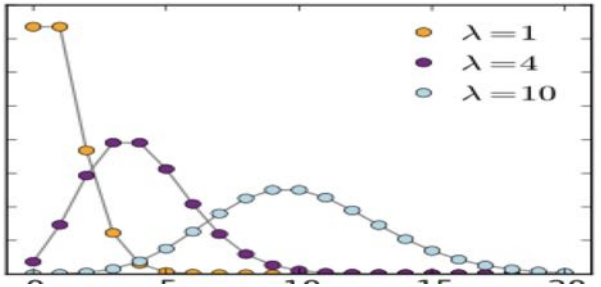
4. Double Hashing

Stride length depends on h_2(key)

Advanced Hashing Summary

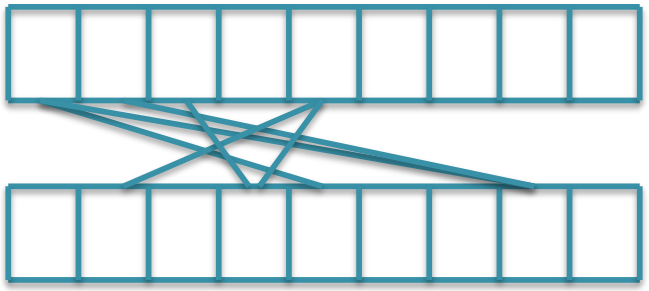
Collisions

- Be careful of collisions!
- Perfect hashing guarantees we avoid collisions, universal hashing lets us pick a new hash function from a family as needed



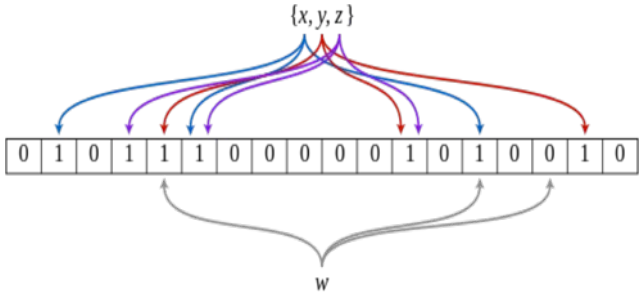
Cuckoo Hashing

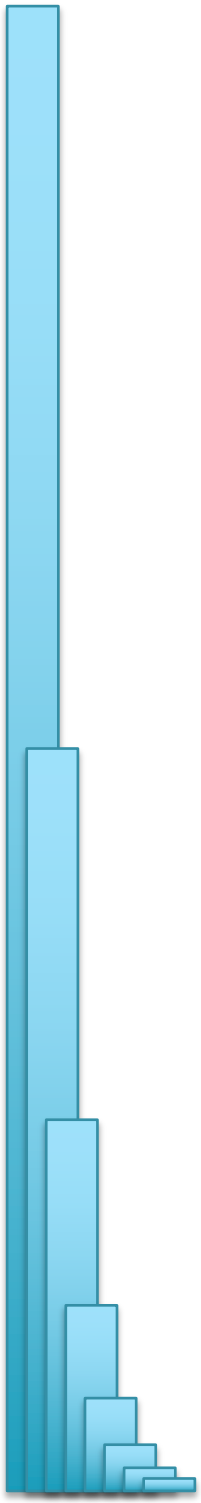
- Simple open addressing technique with O(1) lookup with expected O(1) insert
- Often outperforms other collision management schemes



Bloom Filters

- Store a huge set in a tiny amount of space allowing for a small rate of false positives.
- Used as a quick pre-filter to determine if the slow operation needs to take place





Part 1: Intro to Genomics aka Data Structures on Strings

DNA: The secret of life

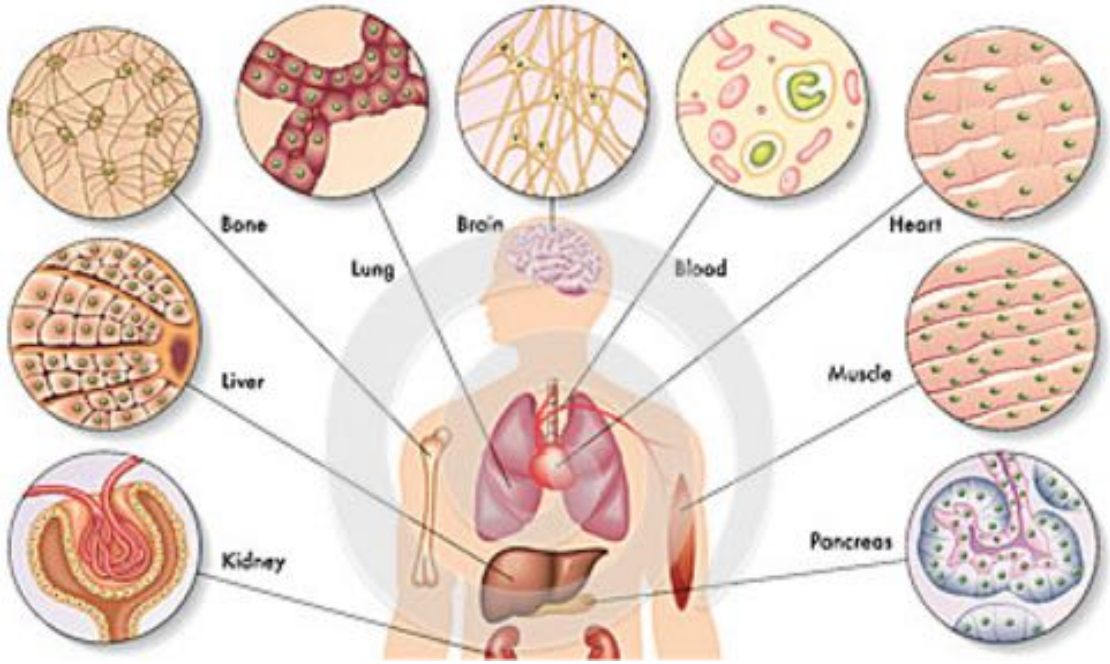
Your DNA, along with your environment and experiences, shapes who you are

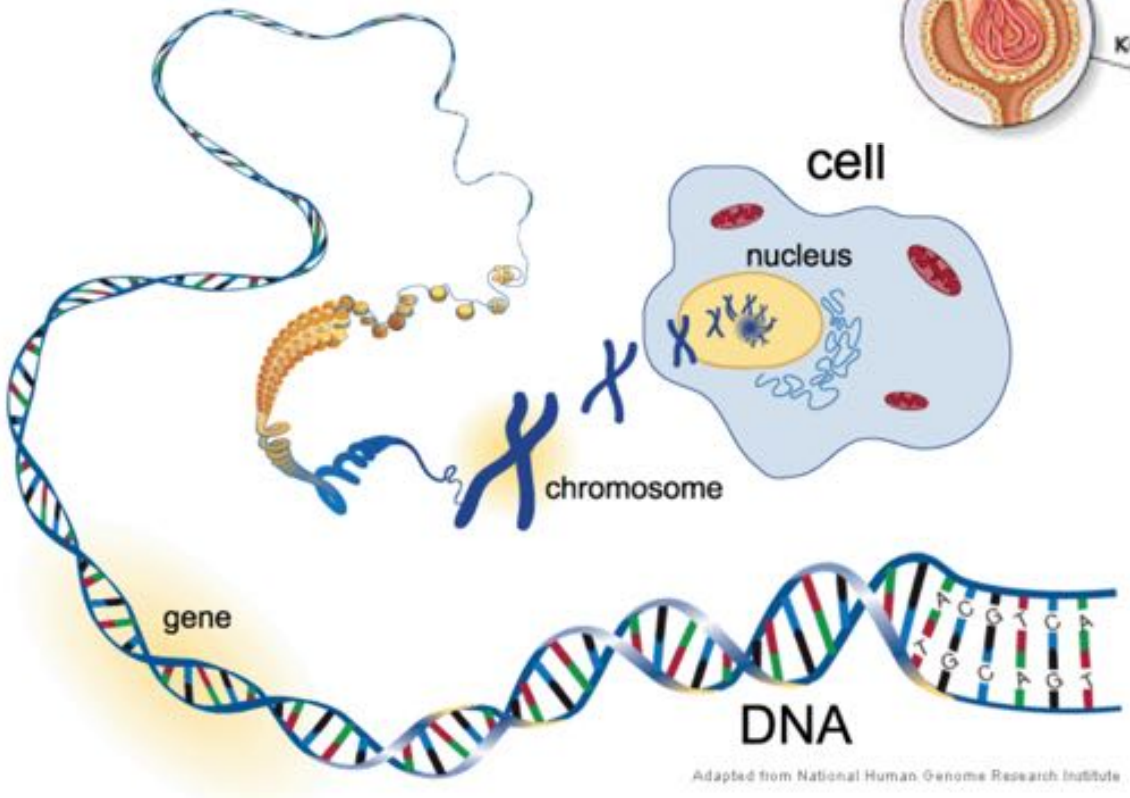
- Height
- Hair, eye, skin color
- Broad/narrow, small/large features
- Susceptibility to disease
- Response to drug treatments
- Longevity and cognition

Physical traits tend to be strongly genetic, social characteristics tend to be strongly environmental, and everything else is a combination

Cells & DNA

Each cell of your body contains an exact copy of your 3 billion base pair genome.

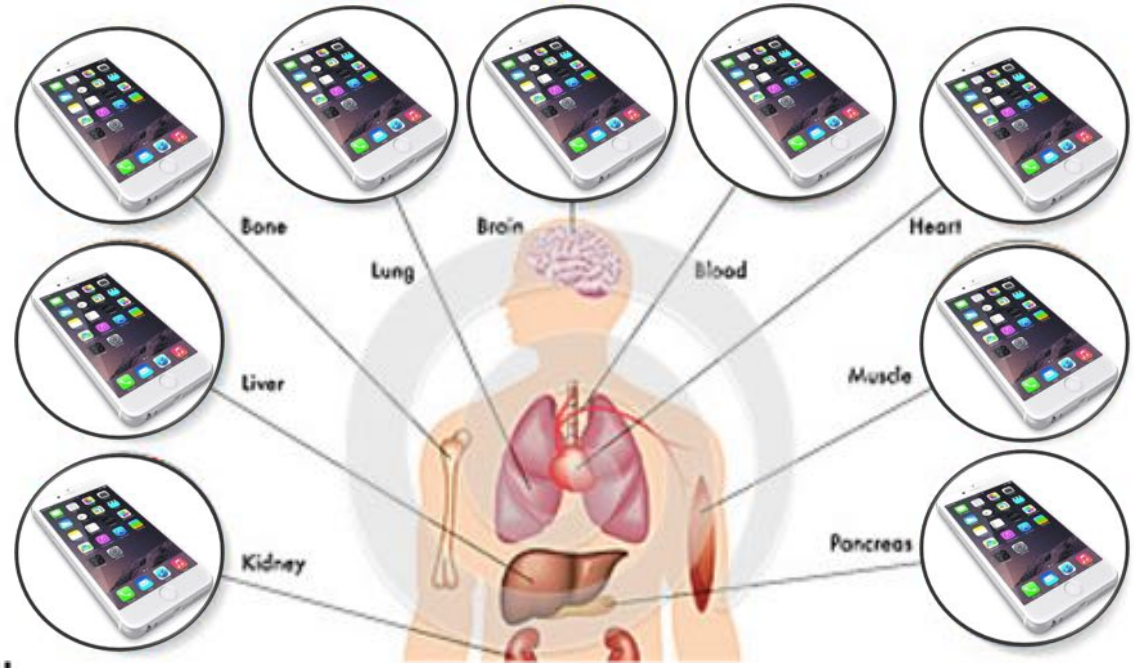


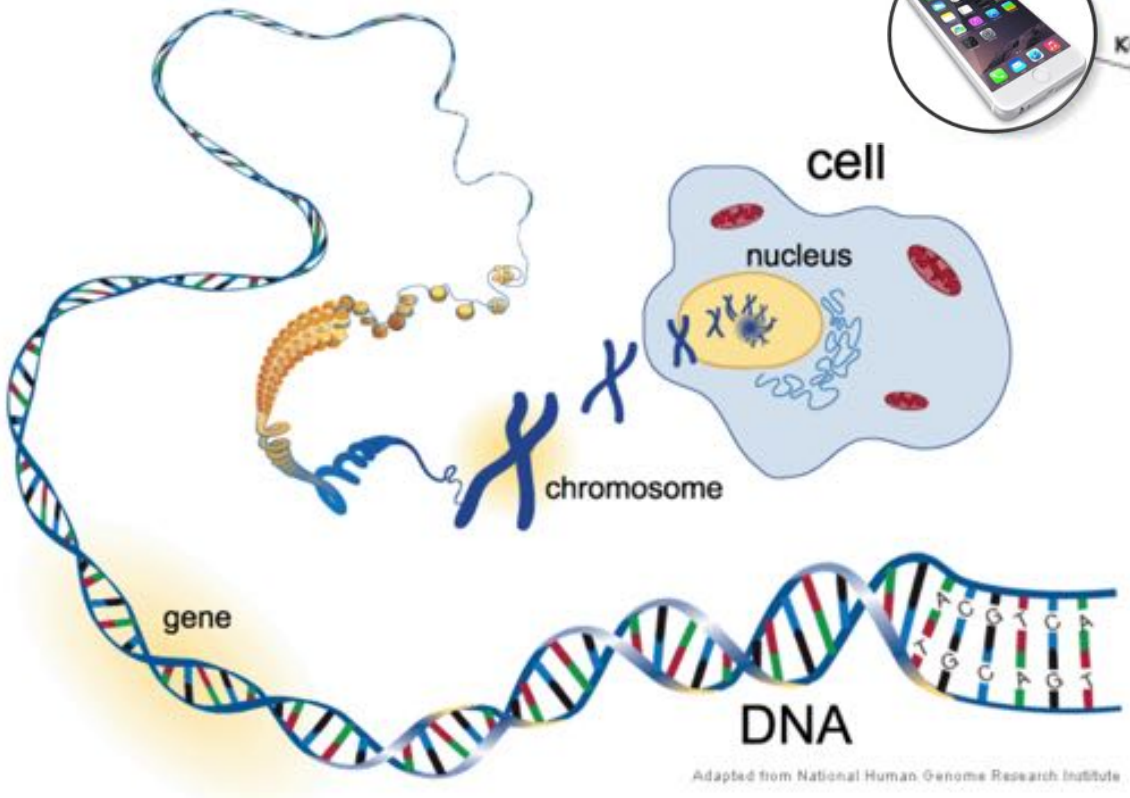


Your specific nucleotide sequence encodes the genetic program for your cells and ultimately your traits

Cells & DNA

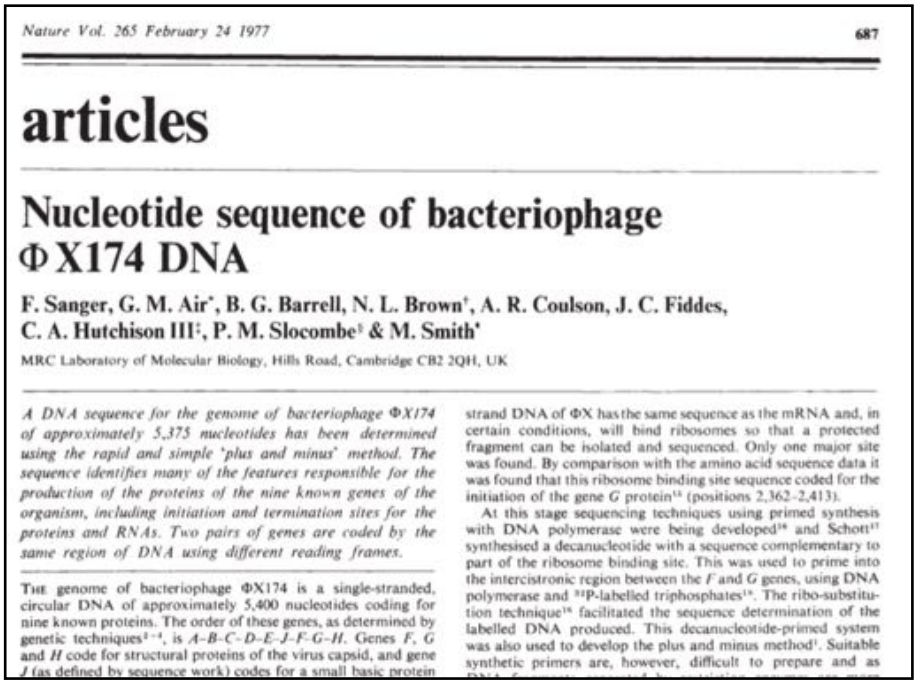
Each cell of your body contains an exact copy of your 3 billion base pair genome.





Your specific nucleotide sequence encodes the genetic program for your cells and ultimately your traits

The Origins of DNA Sequencing



Nature Vol. 265 February 24 1977 687

articles

Nucleotide sequence of bacteriophage ΦX174 DNA

F. Sanger, G. M. Air*, B. G. Barrell, N. L. Brown†, A. R. Coulson, J. C. Fiddes, C. A. Hutchison III‡, P. M. Slocombe§ & M. Smith*

MRC Laboratory of Molecular Biology, Hills Road, Cambridge CB2 2QH, UK

A DNA sequence for the genome of bacteriophage ΦX174 of approximately 5,375 nucleotides has been determined using the rapid and simple 'plus and minus' method. The sequence identifies many of the features responsible for the production of the proteins of the nine known genes of the organism, including initiation and termination sites for the proteins and RNAs. Two pairs of genes are coded by the same region of DNA using different reading frames.

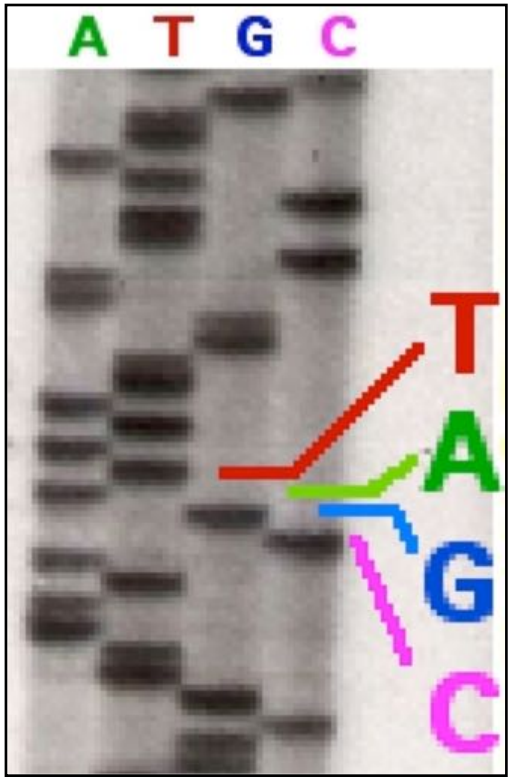
THE genome of bacteriophage ΦX174 is a single-stranded, circular DNA of approximately 5,400 nucleotides coding for nine known proteins. The order of these genes, as determined by genetic techniques[1-4], is *A–B–C–D–E–J–F–G–H*. Genes *F*, *G* and *H* code for structural proteins of the virus capsid, and gene *J* (as defined by sequence work) codes for a small basic protein

strand DNA of ΦX has the same sequence as the mRNA and, in certain conditions, will bind ribosomes so that a protected fragment can be isolated and sequenced. Only one major site was found. By comparison with the amino acid sequence data it was found that this ribosome binding site sequence coded for the initiation of the gene *G* protein[5,6] (positions 2,362–2,413).

At this stage sequencing techniques using primed synthesis with DNA polymerase were being developed[7,8] and Schott[11] synthesised a decanucleotide with a sequence complementary to part of the ribosome binding site. This was used to prime into the intercistronic region between the *F* and *G* genes, using DNA polymerase and ^{32}P-labelled triphosphates[12]. The ribo-substitution technique[13] facilitated the sequence determination of the labelled DNA produced. This decanucleotide-primed system was also used to develop the plus and minus method[1]. Suitable synthetic primers are, however, difficult to prepare and as

Sanger *et al. (1977) Nature*
1st Complete Organism
Bacteriophage ϕX174; 5375 bp

Awarded Nobel Prize in 1980



Radioactive Chain Termination
5000bp / week / person

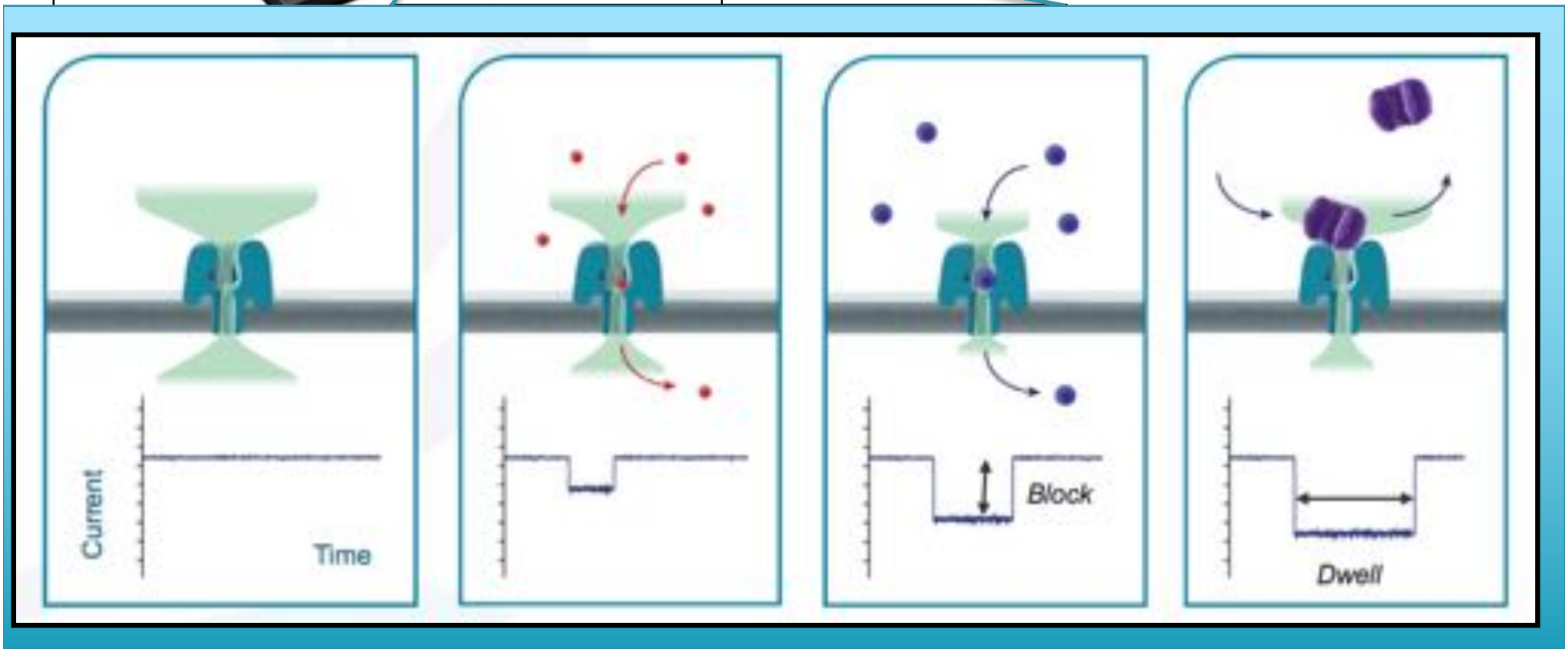
http://en.wikipedia.org/wiki/File:Sequencing.jpg
http://www.answers.com/topic/automated-sequencer

Milestones in DNA Sequencing

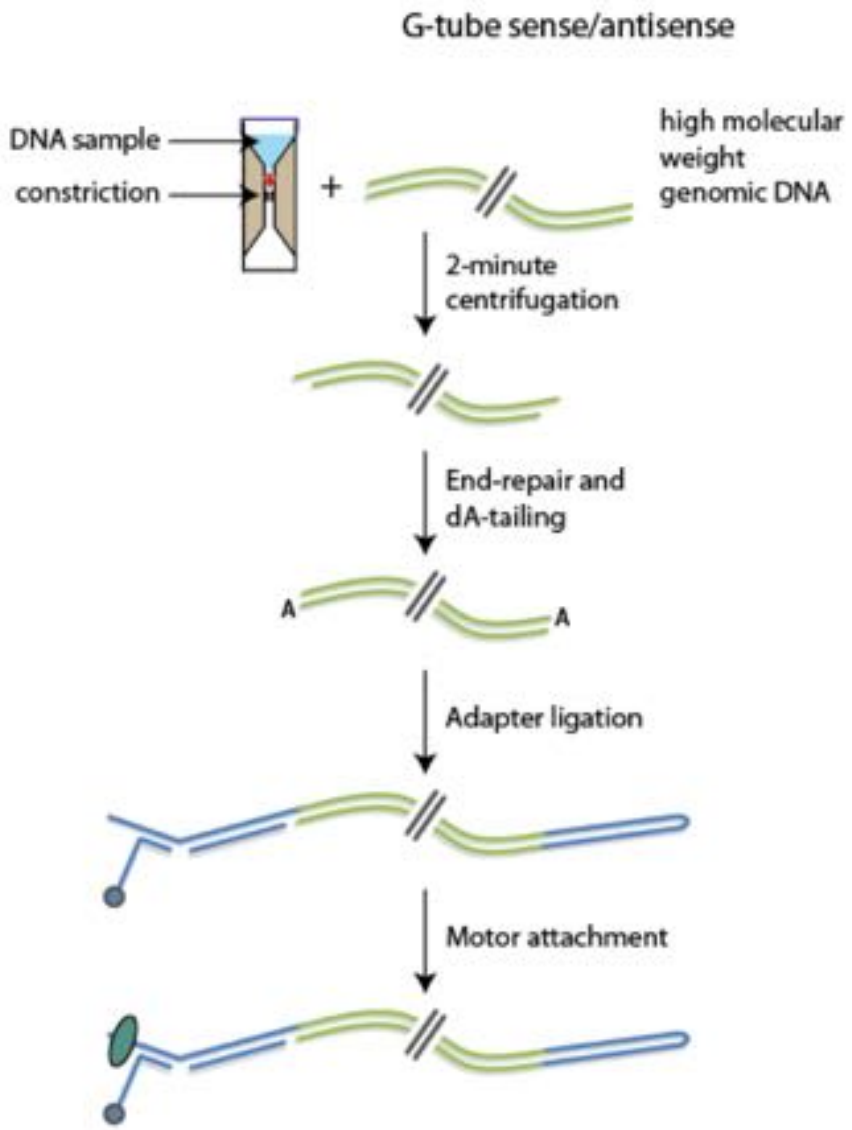
(TIGR/Celera, 1995-2001)

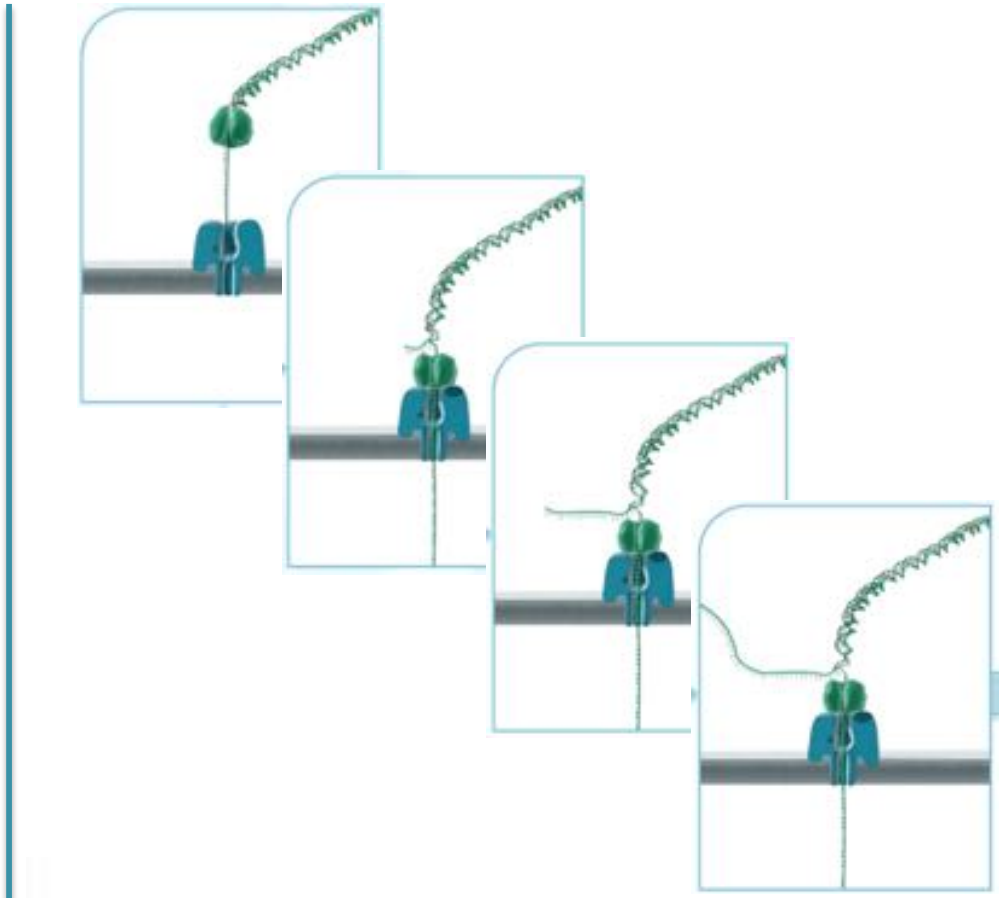
Oxford Nanopore MinION

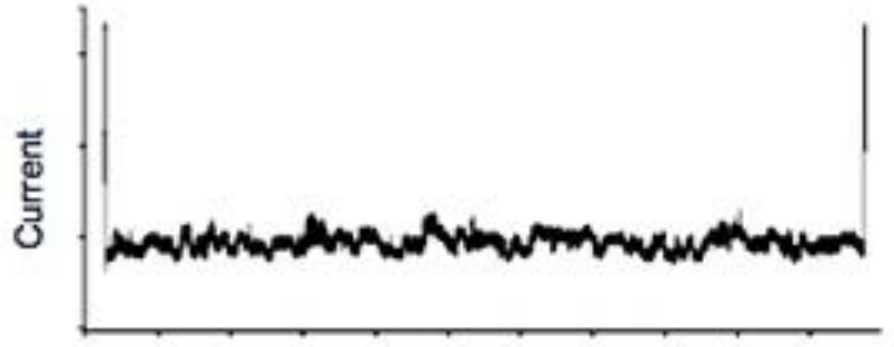
- Thumb drive sized sequencer powered over USB
- Capacity for 512 reads at once
- Senses DNA by measuring changes to ion flow



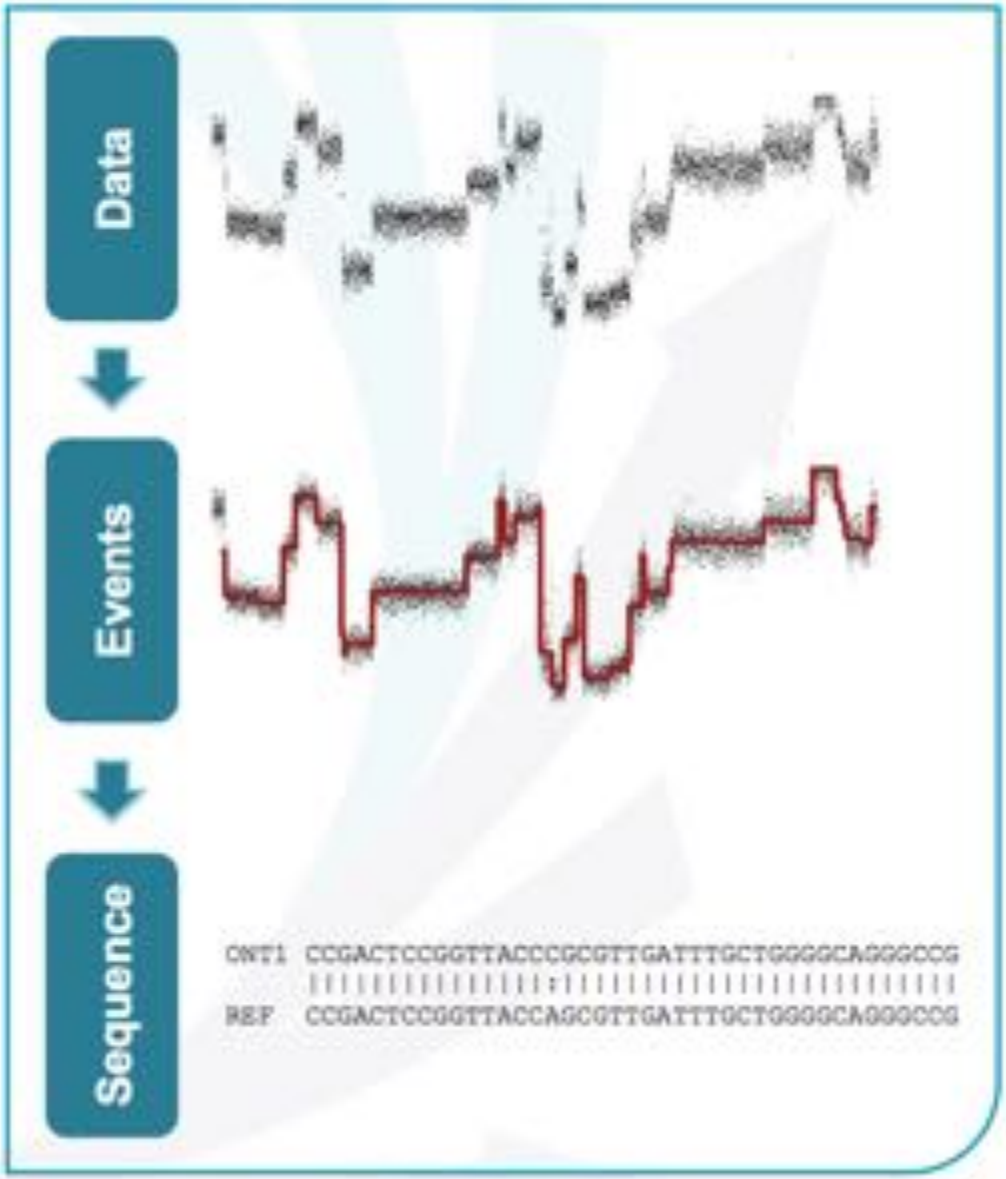
Nanopore Sequencing







Nanopore Basecalling

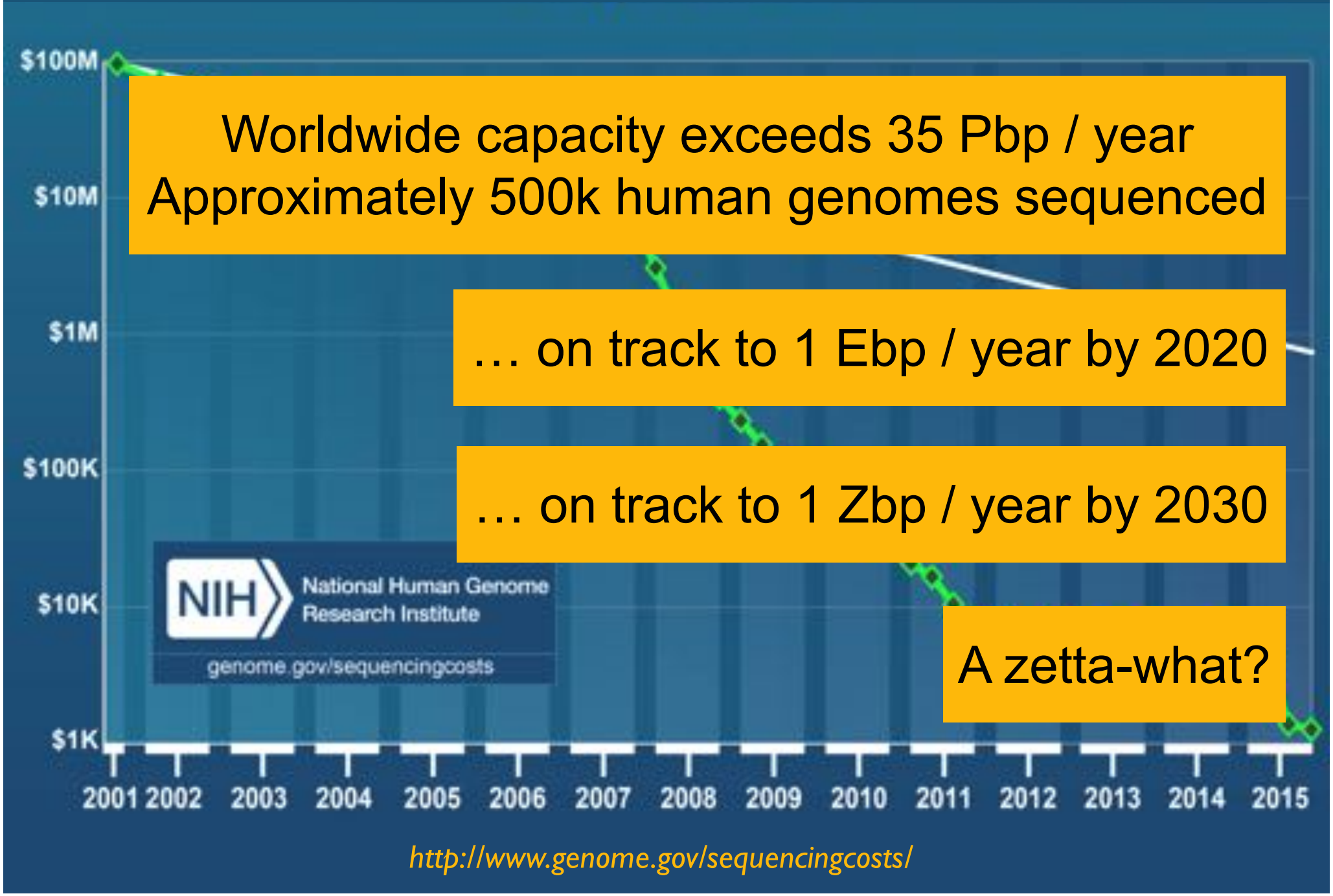


- Hidden Markov model
- Only four options per transition
- Pore type = distinct kmer length

- Form probabilistic path through measured states currents and transitions
  - e.g. Viterbi algorithm

Basecalling currently performed at Amazon with frequent updates to algorithm

Cost per Genome



Worldwide capacity exceeds 35 Pbp / year
Approximately 500k human genomes sequenced

… on track to 1 Ebp / year by 2020

… on track to 1 Zbp / year by 2030

A zetta-what?

http://www.genome.gov/sequencingcosts/

How much is a zettabase?

Unit	Size
Base	1
Kilobase	1,000
Megabase	1,000,000
Gigabase	1,000,000,000
Terabase	1,000,000,000,000
Petabase	1,000,000,000,000,000
Exabase	1,000,000,000,000,000,000
Zettabase	1,000,000,000,000,000,000,000

How much is a zettabase?

100 GB / Genome
4.7GB / DVD
~20 DVDs / Genome

X

10,000,000,000 Genomes

=

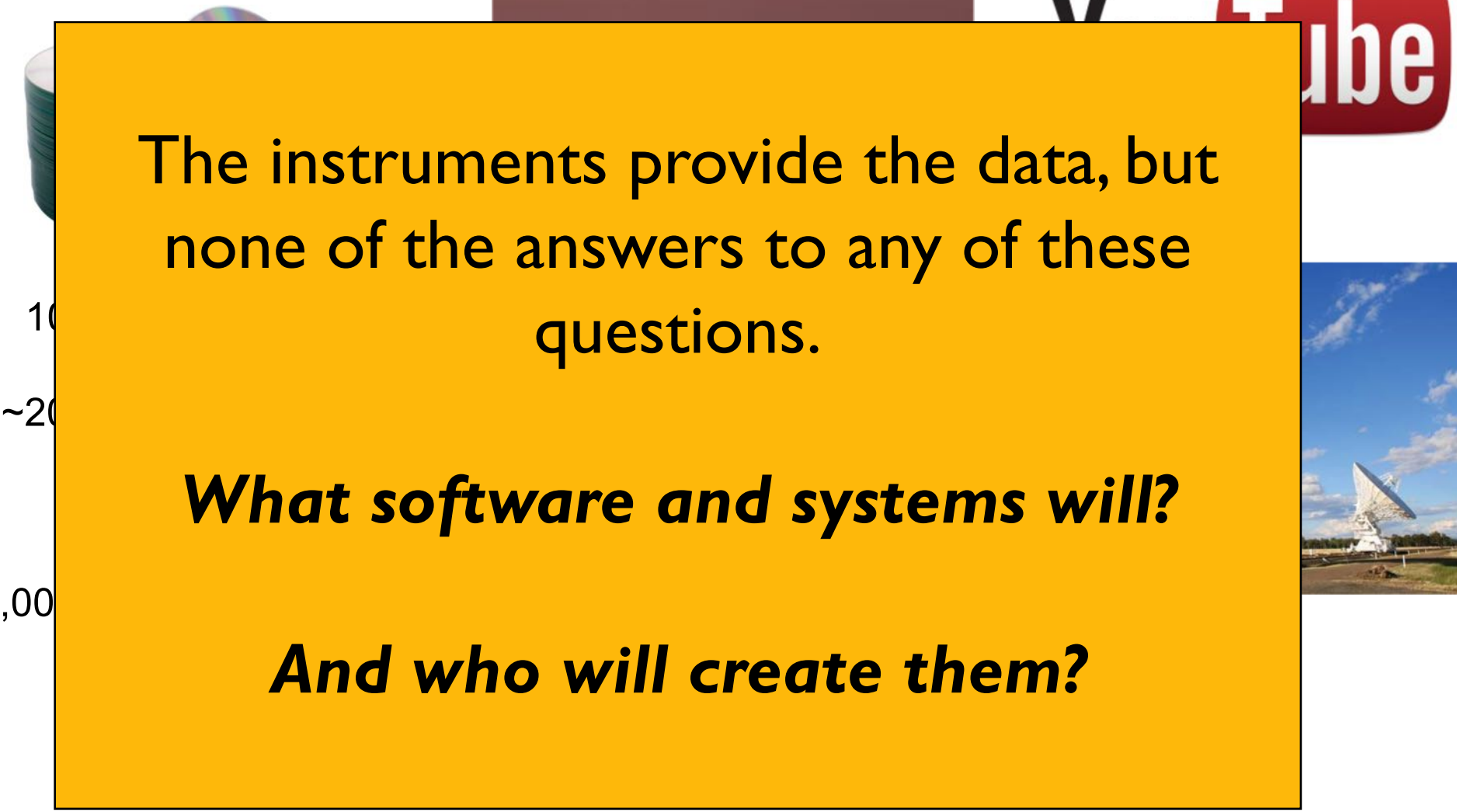
1ZB Data
200,000,000,000 DVDs

150,000 miles of DVDs
~ ½ distance to moon

YouTube

Both currently ~100PB
And growing exponentially

How much is a zettabase?



The instruments provide the data, but none of the answers to any of these questions.

What software and systems will?

And who will create them?

1ZB Data
200,000,000,000 DVDs

150,000 miles of DVDs
~ ½ distance to moon

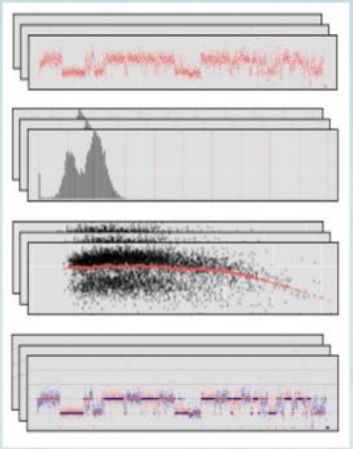
Both currently ~100PB
And growing exponentially

Schatzlab Overview

Human Genetics

Genetics of Autism Spectrum Disorders

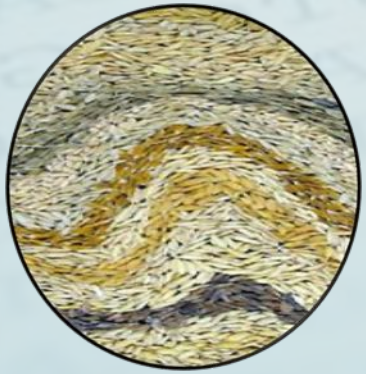
Narzisi *et al.* (2015)
Iossifov *et al.* (2014)



Cancer Biology

Indels, CNVs, SVs, & Cell Phylogenetics

Nattestad *et al.* (2018)
Goodwin *et al.* (2015)



Agricultural Genomics

Rice, Corn, Wheat and many others

Chin *et al.* (2016)
Schatz *et al.* (2014)

Evolutionary Biology

Inflorescence diversity

Lemmon *et al.* (2016)
Park *et al.* (2011)

Computational Biology @ CS

Alexis Battle

Ben Langmead

James Taylor

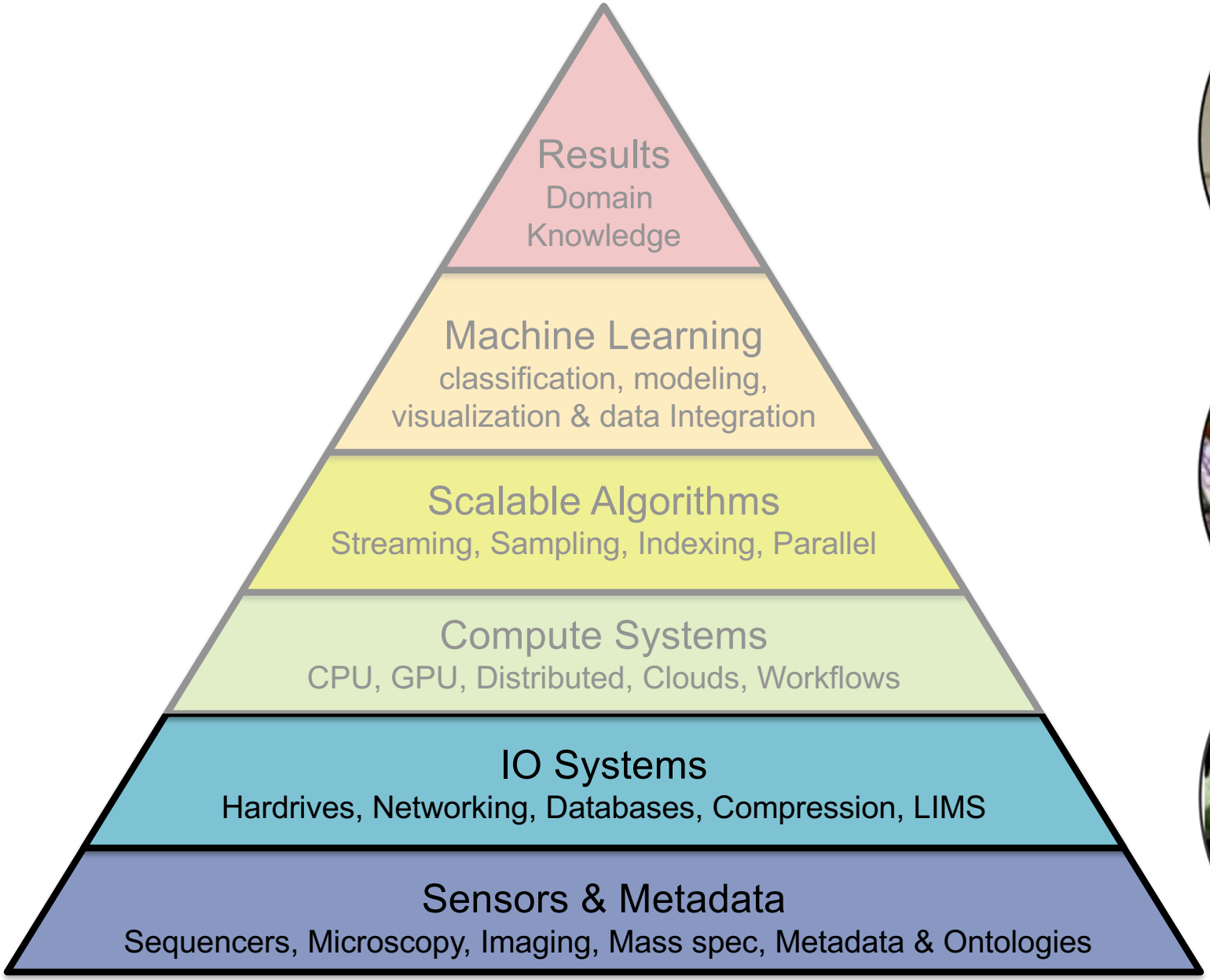
Liliana Florea

Mihaela Pertea

Steven Salzberg

Computational Biology @ JHU

Computational Biology



Computational Biology

Nanopore sequencing meets epigenetics
Schatz, MC (2017) *Nature Methods* 14, 347–348 (2017) doi:10.1038/nmeth.4240

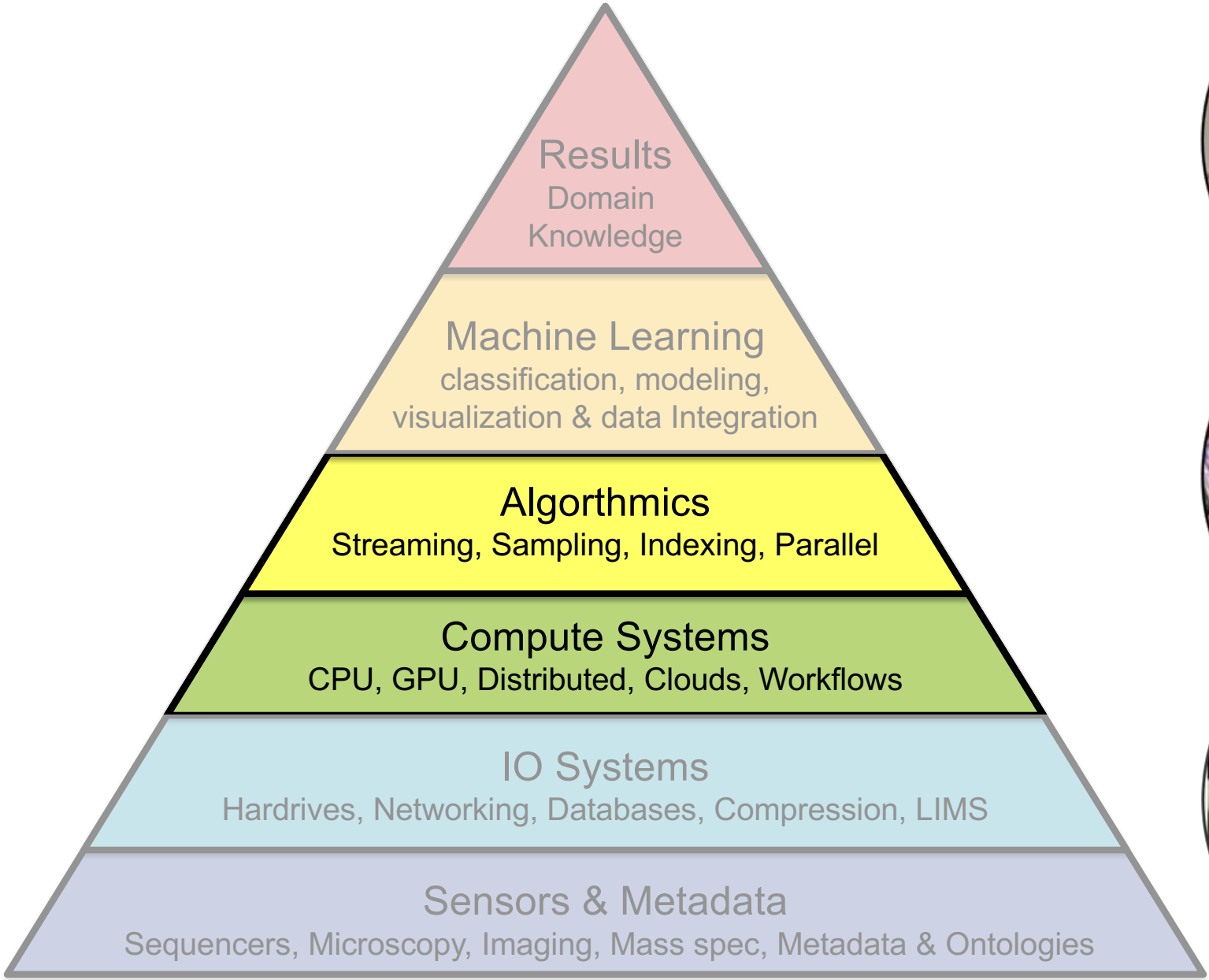
ISSN 1088-9051

August 2018

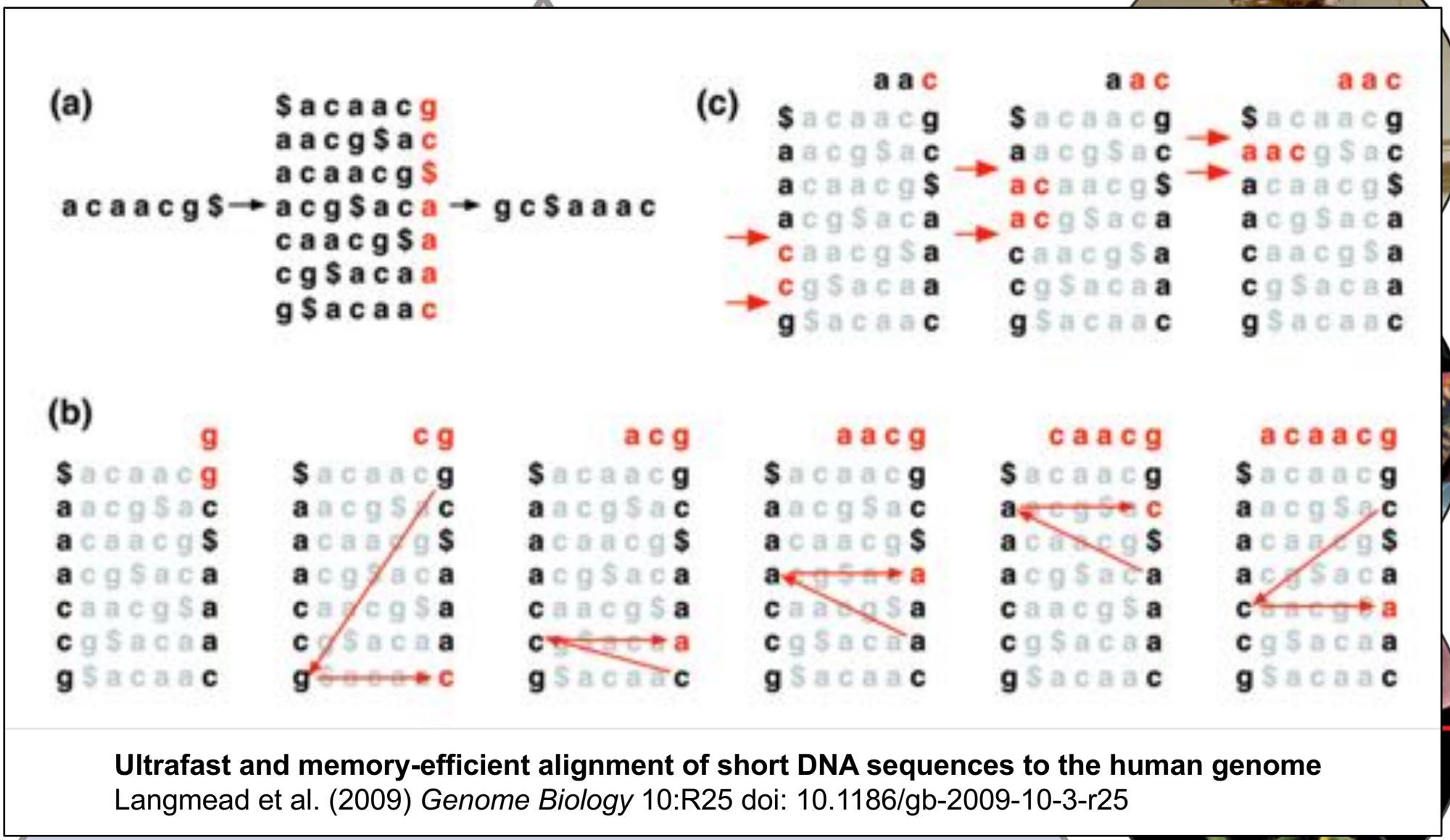
GENOME RESEARCH

Volume 28 Number 8

Computational Biology



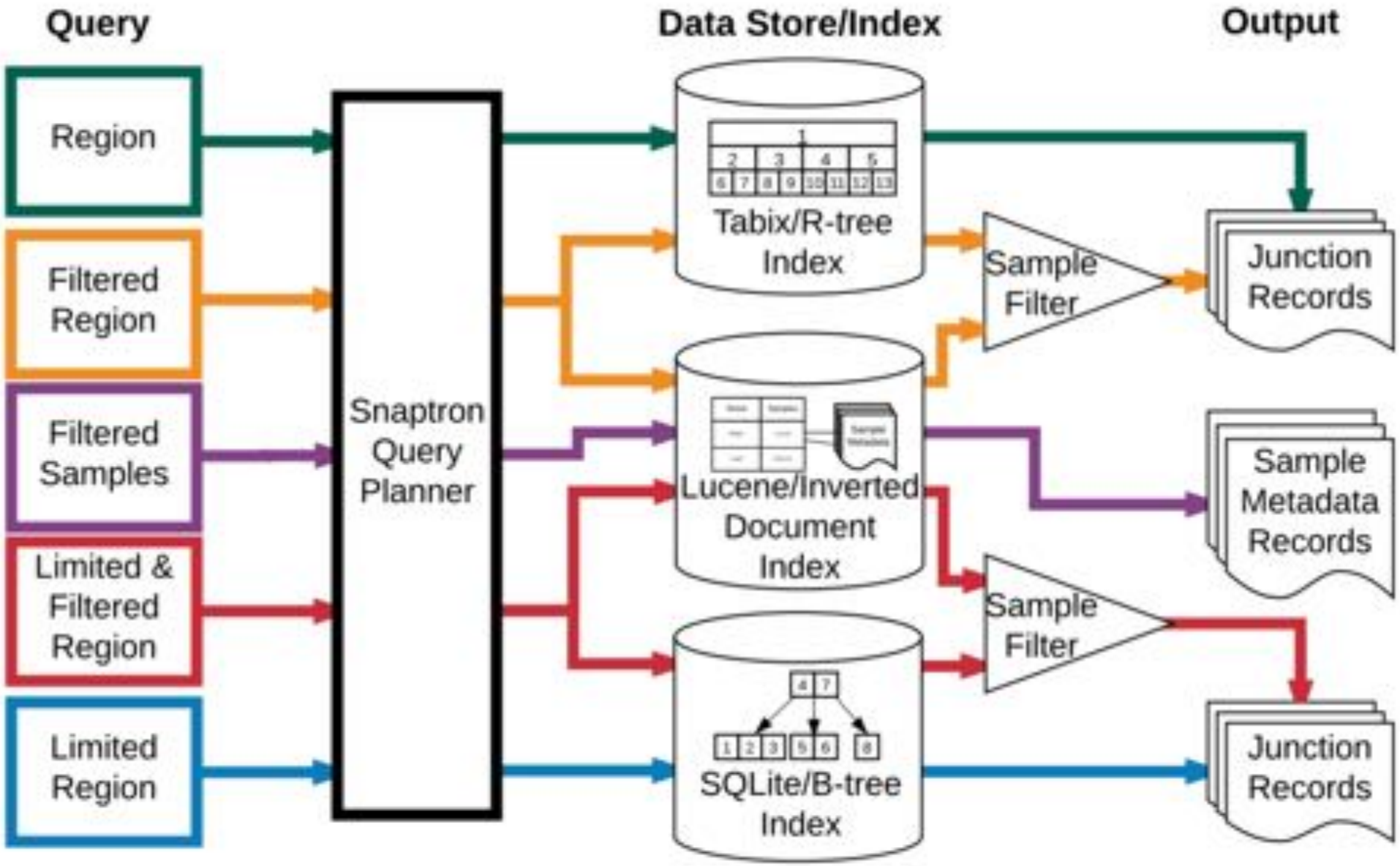
Computational Biology



Ultrafast and memory-efficient alignment of short DNA sequences to the human genome
Langmead et al. (2009) *Genome Biology* 10:R25 doi: 10.1186/gb-2009-10-3-r25

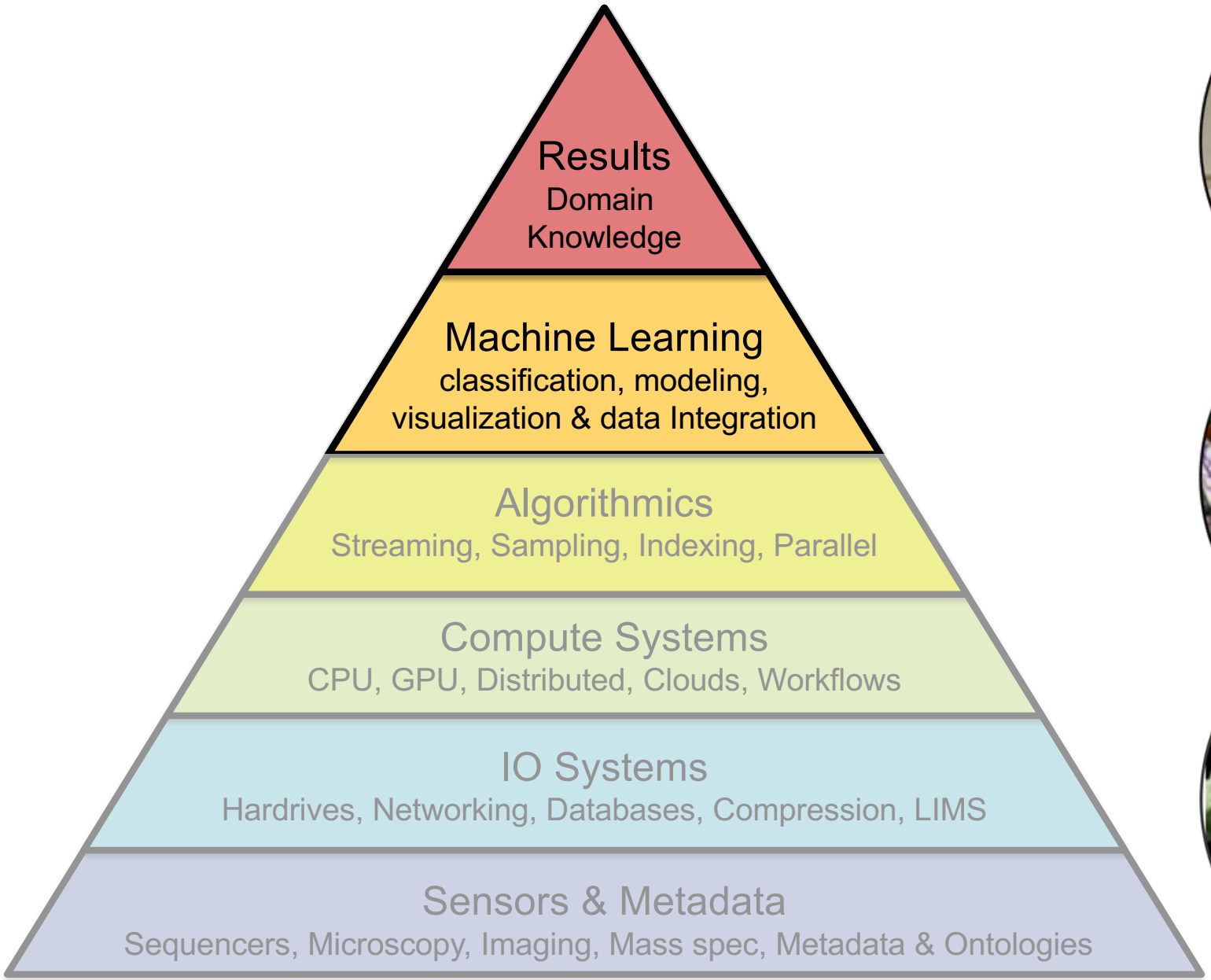
Sequencers, Microscopy, Imaging, Mass spec, Metadata & Ontologies

Computational Biology

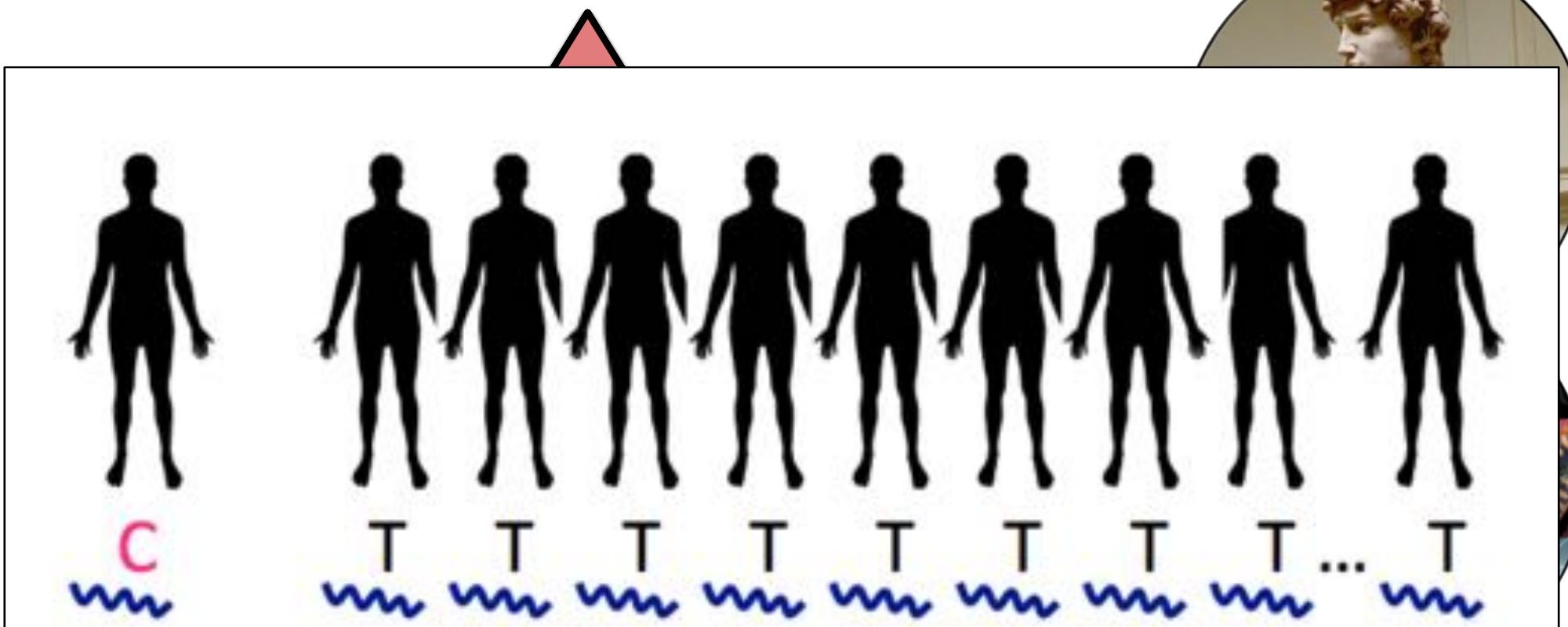


Reproducible RNA-seq analysis using recount2
Collado-Torres et al (2017) Nature Biotechnology. 35, 319–321

Computational Biology



Computational Biology



Genetic effects on gene expression across human tissues.
The GTEx Consortium (2017) Nature 550, 204–213

Hardrives, Networking, Databases, Compression, LIMS

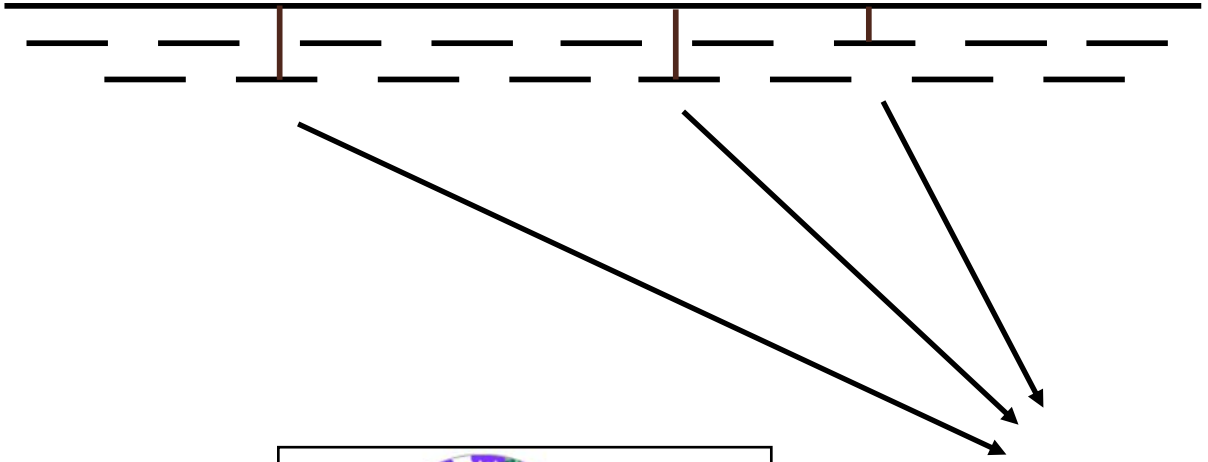
Sensors & Metadata

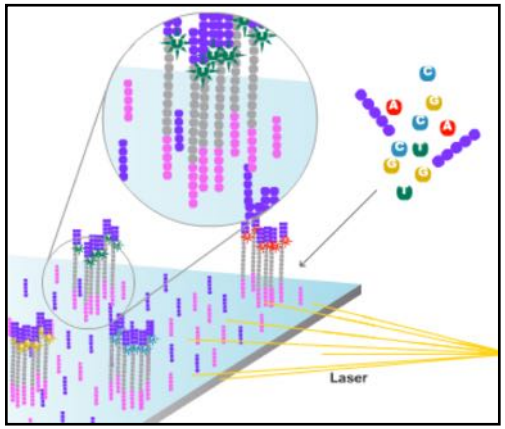
Sequencers, Microscopy, Imaging, Mass spec, Metadata & Ontologies


change
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
BEING HUMAN
Cataloguing the effects of genetic variation on gene expression in healthy tissues
PAGES 190, 204, 239, 244 & 249

Personal Genomics

How does your genome compare to the reference?





Heart Disease

Cancer

Presidential smile

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
G	A	T	T	A	C	A									

No match at offset 1

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
	G	A	T	T	A	C	A								

Match at offset 2

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
		G	A	T	T	A	C	A	…						

No match at offset 3…

Searching for GATTACA

- Where is GATTACA in the human genome?
- Strategy 1: Brute Force

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

No match at offset 9 <- Checking each possible position takes time

Brute Force Analysis

- Brute Force:
  - At every possible offset in the genome:
    - Do all of the characters of the query match?

- Analysis
  - Simple, easy to understand
  - Genome length = n [3B]
  - Query length = m [7]
  - Comparisons: (n-m+1) * m [21B]

- Overall runtime: O(nm)

[How long would it take if we double the genome size, read length?]

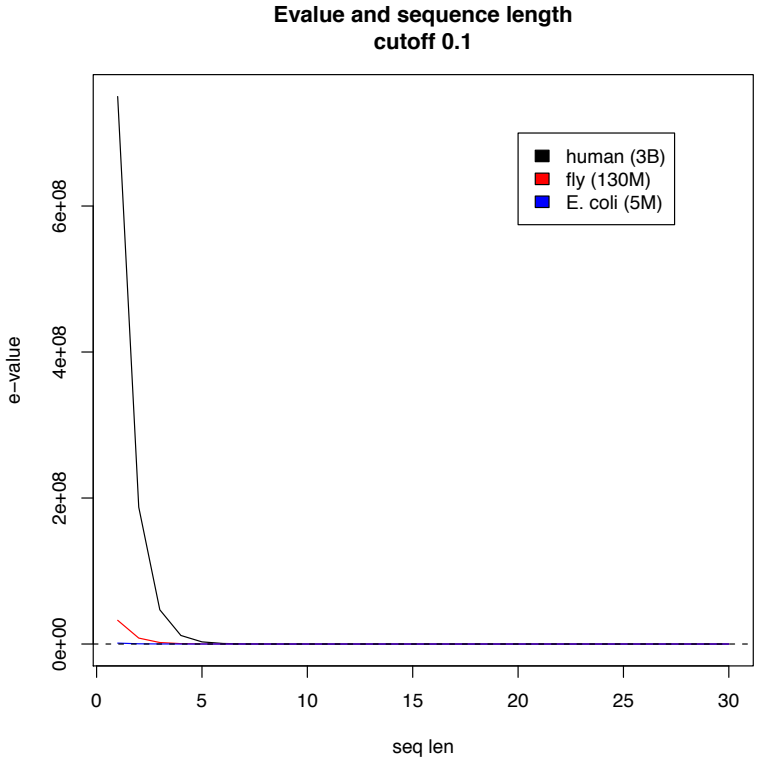
[How long would it take if we double both?]

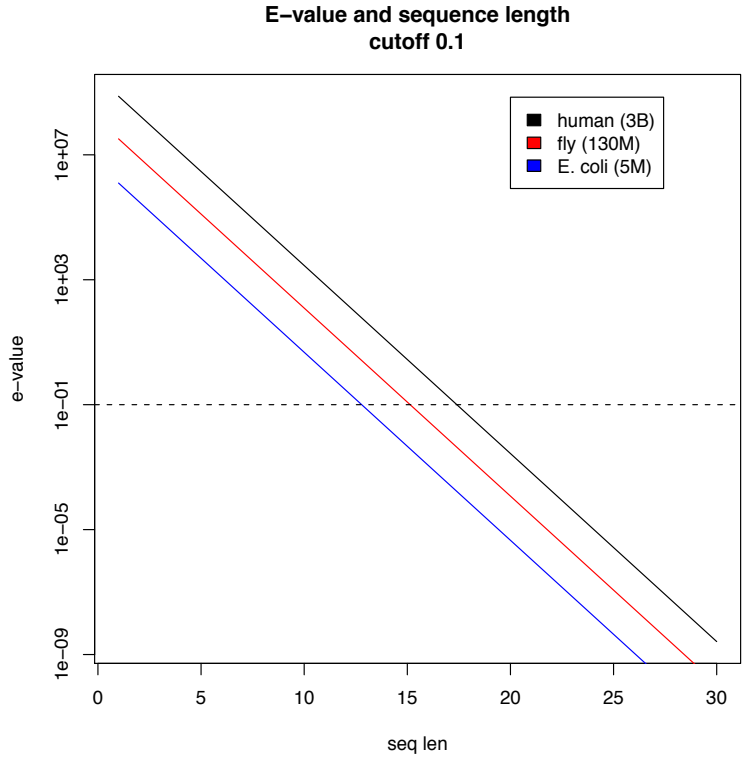
Expected Occurrences

The expected number of occurrences (e-value) of a given sequence in a genome depends on the length of the genome and inversely on the length of the sequence

- 1 in 4 bases are G, 1 in 16 positions are GA, 1 in 64 positions are GAT, …
- 1 in 16,384 should be GATTACA
- $E=n/(4^m)$ [183,105 expected occurrences]

[How long do the reads need to be for a significant match?]





Brute Force Reflections

Why check every position?

– GATTACA can't possibly start at position 15 [WHY?]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
T	G	A	T	T	A	C	A	G	A	T	T	A	C	C	…
								G	A	T	T	A	C	A	

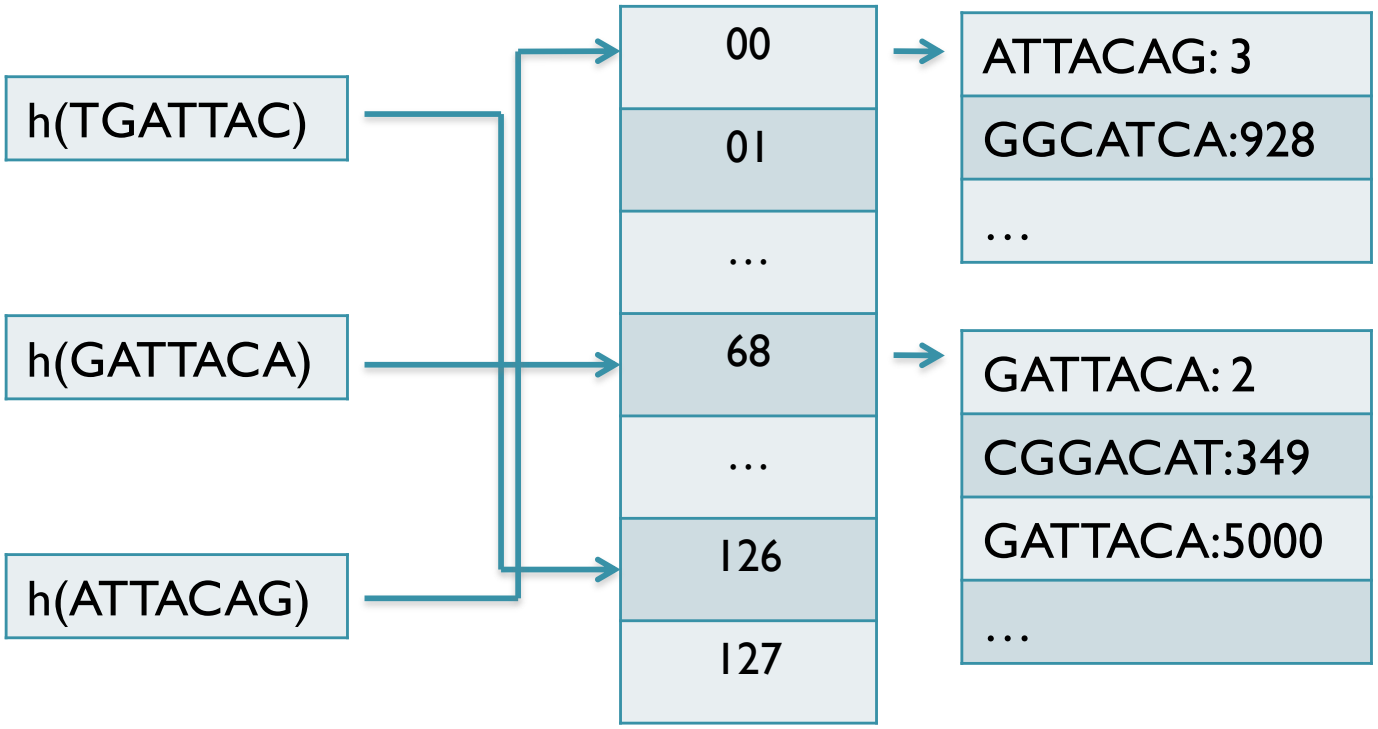
– Improve runtime to O(n + m) [3B + 7]
  - If we double both, it just takes twice as long
  - Knuth-Morris-Pratt, 1977
  - Boyer-Moyer, 1977, 1991

– For one-off scans, this is the best we can do (optimal performance)
  - We have to read every character of the genome, and every character of the query
  - For short queries, runtime is dominated by the length of the genome

How can we make this go faster?

Hash Table Lookup

- By construction, multiple keys have the same hash value
  - Store elements with the same key in a bucket chained together
    - A good hash evenly distributes the values: R/H have the same hash value
  - Looking up a value scans the entire bucket
    - Slows down the search as a function of the hash table load



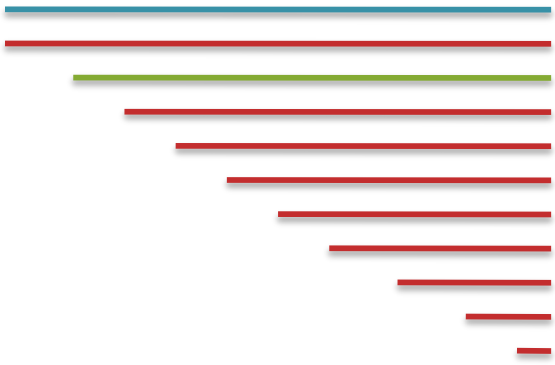
How many elements do we expect per bucket?

What if we don't know how long the queries will be?

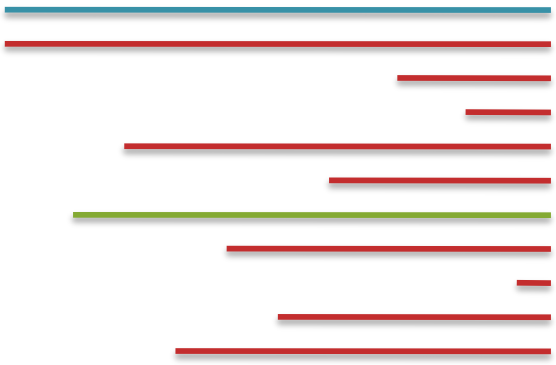
Full-Text Indexing: Suffix Arrays

- What if we need to check many queries?
  - We don't need to check every page of the phone book to find 'Schatz'
  - Sorting alphabetically lets us immediately skip 96% (25/26) of the book *without any loss in accuracy*

- Sorting the genome: Suffix Array (Manber & Myers, 1991)
  - Sort every suffix of the genome



Split into n suffixes



Sort suffixes alphabetically

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15;

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

Lo →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15;

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Lo →

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Lo →

Hi →

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11;

Lo →

Hi →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC

Lo →

Hi →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9;

Lo →
Hi →

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Searching the Index

- Strategy 2: Binary search
  - Compare to the middle, refine as higher or lower

- Searching for GATTACA
  - Lo = 1; Hi = 15; Mid = (1+15)/2 = 8
  - Middle = Suffix[8] = CC
    => Higher: Lo = Mid + 1

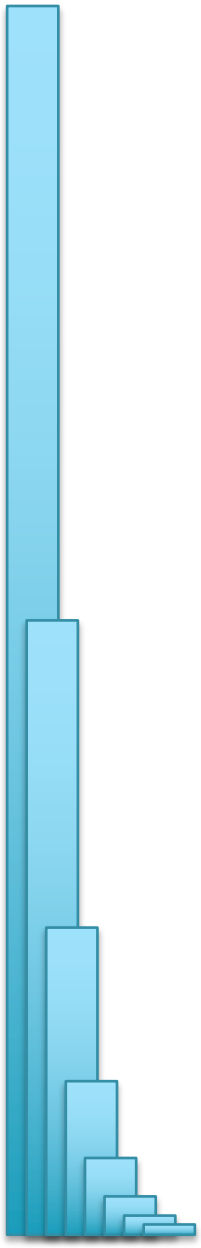
  - Lo = 9; Hi = 15; Mid = (9+15)/2 = 12
  - Middle = Suffix[12] = TACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 11; Mid = (9+11)/2 = 10
  - Middle = Suffix[10] = GATTACC
    => Lower: Hi = Mid - 1

  - Lo = 9; Hi = 9; Mid = (9+9)/2 = 9
  - Middle = Suffix[9] = GATTACA…
    => Match at position 2!

#	Sequence	Pos
1	ACAGATTACC…	6
2	ACC…	13
3	AGATTACC…	8
4	ATTACAGATTACC…	3
5	ATTACC…	10
6	C…	15
7	CAGATTACC…	7
8	CC…	14
9	GATTACAGATTACC…	2
10	GATTACC…	9
11	TACAGATTACC…	5
12	TACC…	12
13	TGATTACAGATTACC…	1
14	TTACAGATTACC…	4
15	TTACC…	11

Binary Search Analysis



- Binary Search

  Initialize search range to entire list

  mid = (hi+lo)/2; middle = suffix[mid]

  if query matches middle: done

  else if query < middle: pick low range

  else if query > middle: pick hi range

  Repeat until done or empty range [WHEN?]

- Analysis
  - More complicated method
  - How many times do we repeat?
    - How many times can it cut the range in half?
    - Find smallest x such that: $n/(2^x) \leq 1$; $x = \lg_2(n)$ [32]

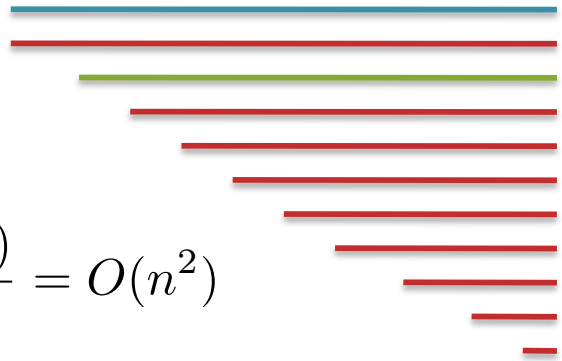
- Total Runtime: O(m lg n)
  - More complicated, but much faster!
  - Looking up a query loops 32 times instead of 3B

[How long does it take to search 6B or 24B nucleotides?]

Suffix Array Construction

- How can we store the suffix array?

  [How many characters are in all suffixes combined?]



$$S = 1 + 2 + 3 + \cdots + n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

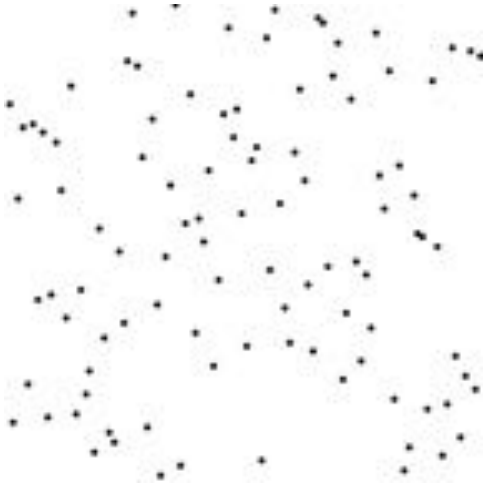
  - Hopeless to explicitly store 4.5 billion billion characters
  - Instead use implicit representation
    - Keep 1 copy of the genome, and a list of sorted offsets
    - Storing 3 billion offsets fits on a server (12GB)

- Searching the array is very fast, but it takes time to construct
  - This time will be amortized over many, many searches
  - Run it once "overnight" and save it away for all future queries

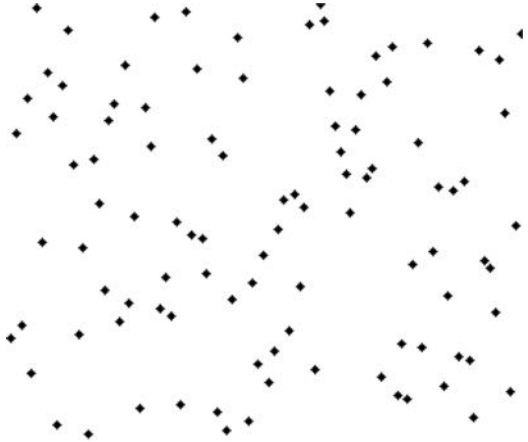
Pos
6
13
8
3
10
15
7
14
2
9
5
12
1
4
11

TGATTACAGATTACC

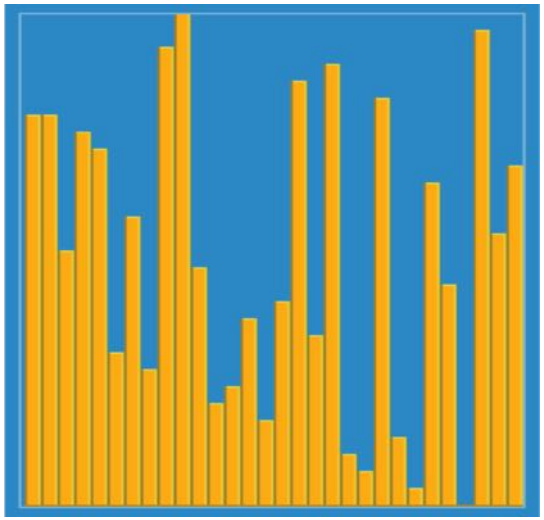
Quadratic Sorting Algorithms



Selection Sort
Move next smallest into position



Bubble Sort
Swap up bigger values over smaller



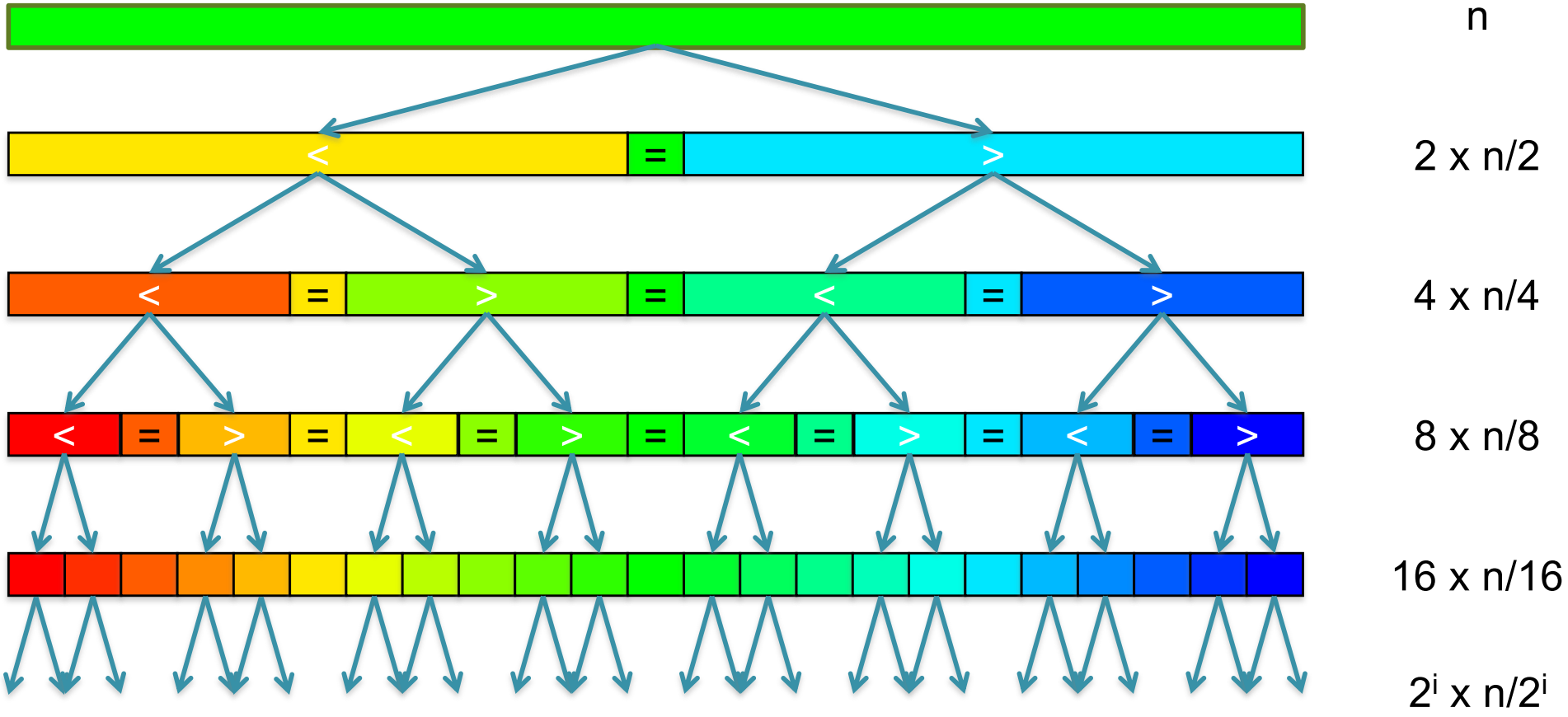
Insertion Sort
Slide next value into correct position

These algorithms will work, but are very slow for 3B suffixes!

How can we go faster?

Divide and Conquer

- Selection sort is slow because it rescans the entire list for each element
  - How can we split up the unsorted list into independent ranges?
  - Hint 1: Binary search splits up the problem into 2 independent ranges (hi/lo)
  - Hint 2: Assume we know the median value of a list



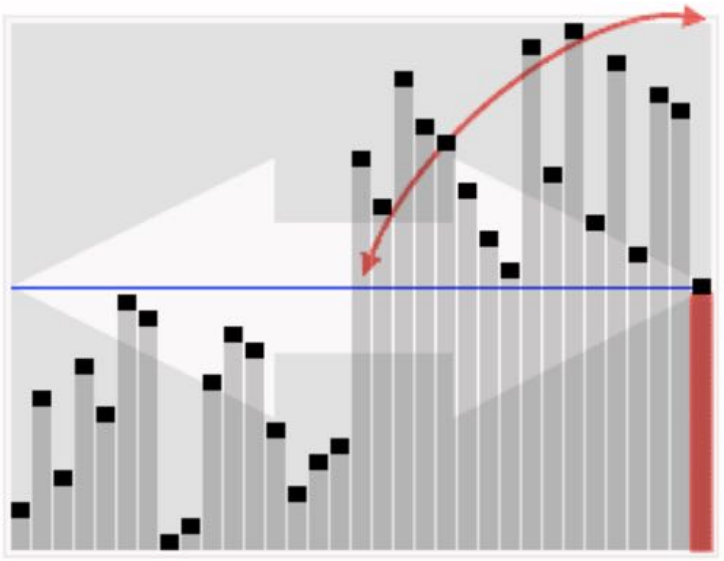
[How many times can we split a list in half?]

QuickSort Analysis

```
QuickSort(Input: list of n numbers)
// see if we can quit
if (length(list)) <= 1): return list

// split list into lo & hi
pivot = median(list)
lo = {}; hi = {};
for (i = 1 to length(list))
if (list[i] < pivot): append(lo, list[i])
else:                 append(hi, list[i])

// recurse on sublists
return (append(QuickSort(lo), QuickSort(hi))
```



http://en.wikipedia.org/wiki/Quicksort

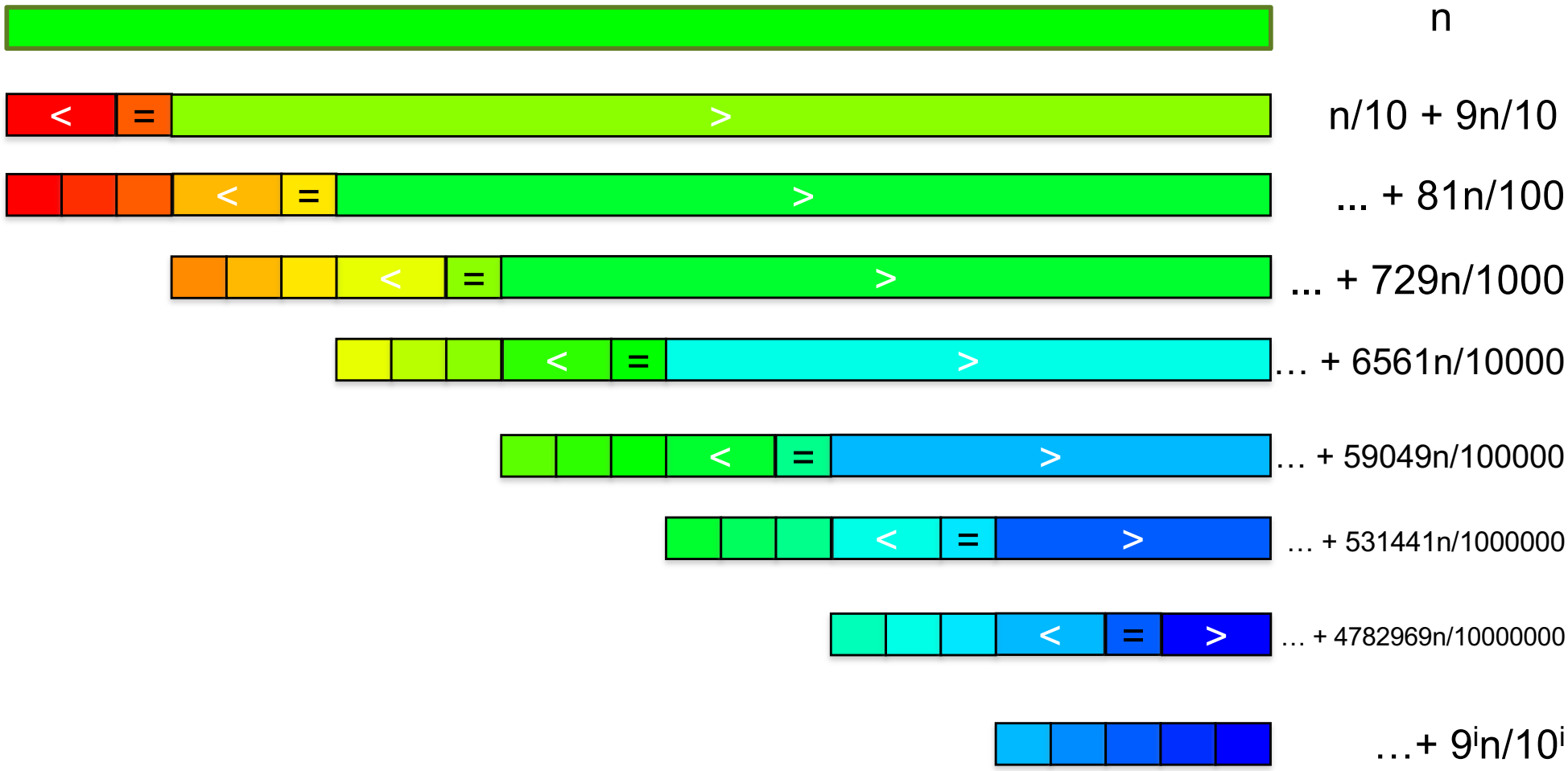
Analysis (Assume we can find the median in O(n))

$$T(n) = \begin{cases} O(1) & \text{if } n \leq 1 \\ O(n) + 2T(n/2) & \text{else} \end{cases}$$

$$T(n) = n + 2(\frac{n}{2}) + 4(\frac{n}{4}) + \cdots + n(\frac{n}{n}) = \sum_{i=0}^{lg(n)} \frac{2^i n}{2^i} = \sum_{i=0}^{lg(n)} n = O(n \lg n)$$

Picking the Median

- What if we miss the median and do a 90/10 split instead?



[How many times can we cut 10% off a list?]

Randomized Quicksort

- 90/10 split runtime analysis

$$T(n) = n + T(\frac{n}{10}) + T(\frac{9n}{10})$$

$$T(n) = n + \frac{n}{10} + T(\frac{n}{100}) + T(\frac{9n}{100}) + \frac{9n}{10} + T(\frac{9n}{100}) + T(\frac{81n}{100})$$

$$T(n) = n + n + T(\frac{n}{100}) + 2T(\frac{9n}{100}) + T(\frac{81n}{100})$$

$$T(n) = \sum_{i=0}^{\log_{10/9}(n)} n = O(n \lg n)$$

Find smallest x s.t.

$$(9/10)^x n \leq 1$$

$$(10/9)^x \geq n$$

$$x \geq \log_{10/9} n$$

- If we randomly pick a pivot, we will get at least a 90/10 split with very high probability
  - Everything is okay as long as we always slice off a fraction of the list

[Challenge Question: What happens if we slice 1 element]

Exact Matching Review & Overview

Where is GATTACA in the human genome?

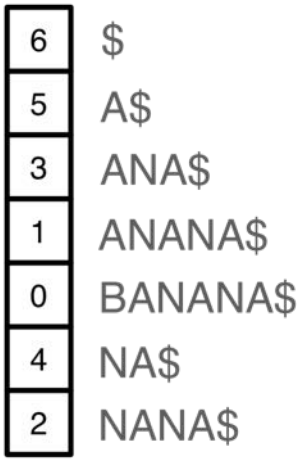
Brute Force (3 GB)

```
BANANA
BAN
  ANA
    NAN
      ANA
```

O(|q| * |r|)

Slow & Easy

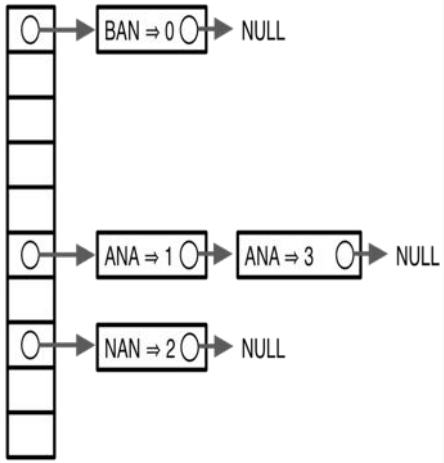
Suffix Array (>15 GB)



O(lg |r|)

Full-text index

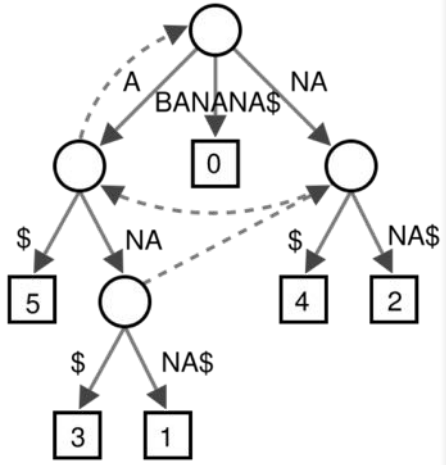
Hash Table (>15 GB)



O(1)

Fixed-length lookup

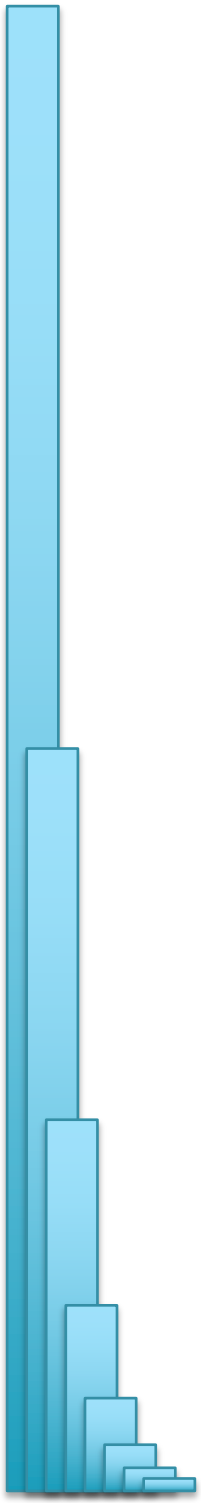
Suffix Tree (>51 GB)



O(|q|)

Full-text, but bulky

*** These are general techniques applicable to any search problem ***



Next Steps

1. Reflect on the magic and power of Suffix Arrays!

1. Assignment 8 due Friday November 16 @ 10pm