



# **Bowtie: A Highly Scalable Tool for Post-Genomic Datasets**

Ben Langmead ([langmead@cs.umd.edu](mailto:langmead@cs.umd.edu))

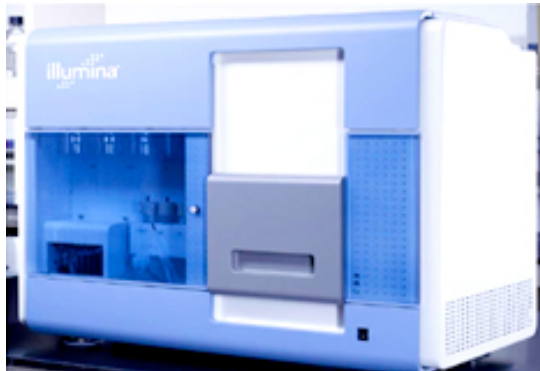
Work with: Cole Trapnell, Mihai Pop, Steven Salzberg

NCBI Seminar

November 10, 2008

# Supply

- High-throughput sequencing produces short DNA sequences (“reads”) in huge volumes at low cost
- Costs are down, adoption is up, next-next generation is coming soon



**Illumina**



**ABI**



**454**

# Demand

- Researchers have no problem putting this data to work



**1000 Genomes**



**Global Ocean Survey**



**Human Microbiome**

- Future growth may be multiplicative
  - Spatial resolution: tissues, individuals, geography
  - Temporal resolution: circadian, seasonal, lifetimes

# Short Read Applications

- Genotyping

Goal: identify variations

```
...CCATAG      TATGCGCCC      CGGA AATT T      GGTATAC...
...CCAT      CTATATGCG      TCGGA AATT      CGGTATAC
...CCAT GGCTATATG      CTATCGG AAA      GCGGTATA
...CCA AGGCTATAT      CCTATCGGA      TTGCGGTA      C...
...CCA AGGCTATAT      GCCCTATCG      TTTGCGGT      C...
...CC AGGCTATAT      GCCCTATCG      AAATTTGC      ATAC...
...CC TAGGCTATA      GCGCCCTA      AAATTTGC      GTATAC...
...CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC...
```

- RNA-seq, ChIP-seq, Methyl-seq

Goal: classify, measure significant peaks

```
...CC
...CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC...
GAAATTTGC
GGAAATTTG
CGGAATTT
CGGAATTT
TCGGAATT
CTATCGGAAA
CCTATCGGA TTTGCGGT
GCCCTATCG AAATTTGC
GCCCTATCG AAATTTGC ATAC...
```

# Short Read Applications

```
...CCATAG      TATGCGCCC      CGGAAATTT      GGTATAC...
...CCAT      CTATATGCG      TCGGAAATT      CGGTATAC
...CCAT GGCTATATG      CTATCGGAAA      GCGGTATA
...CCA AGGCTATAT      CCTATCGGA      TTGCGGTA      C...
...CCA AGGCTATAT      GCCCTATCG      TTTGCGGT      C...
...CC  AGGCTATAT      GCCCTATCG      AAATTTGC      ATAC...
...CC  TAGGCTATA      GCGCCCTA      AAATTTGC      GTATAC...
...CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC...
```

Finding the alignments is typically the performance bottleneck

```
...CC
      GAAATTTGC
      GGAAATTTG
      CGGAAATTT
      CGGAAATTT
      TCGGAAATT
      CTATCGGAAA
      CCTATCGGA      TTTGCGGT
      GCCCTATCG      AAATTTGC
      GCCCTATCG      AAATTTGC      ATAC...
...CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC...
```

# Short Read Alignment

---

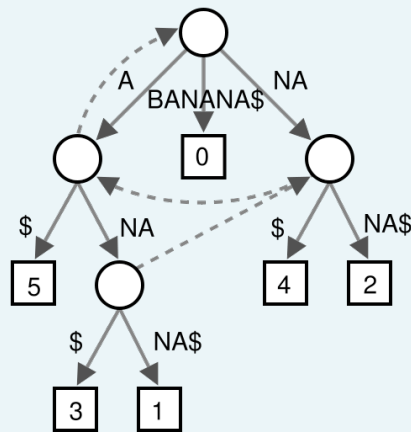
- Given a reference and a set of reads, report at least one “good” local alignment for each read if one exists
  - Approximate answer to: where in genome did read originate?
- What is “good”? For now, we concentrate on:
  - Fewer mismatches is better
  - Failing to align a low-quality base is better than failing to align a high-quality base

...TGATCATA... better than ...TGATCATA...  
GATCAA GAGAA

...TGATATA... better than ...TGATcaTA...  
GATcaT GTACAT

# Indexing

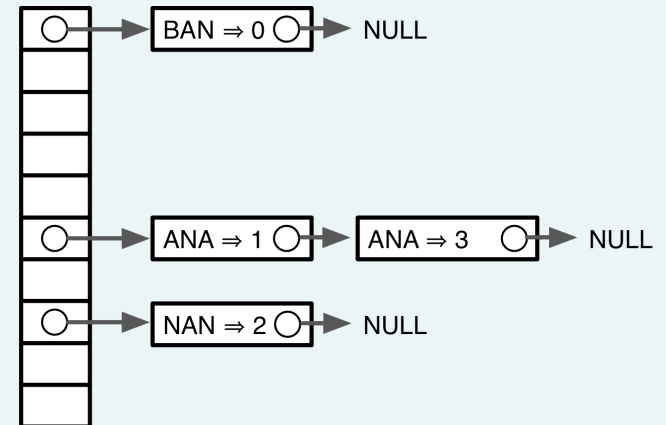
- Genomes and reads are too large for direct approaches like dynamic programming
- *Indexing* is required



**Suffix tree**

6	\$
5	A\$
3	ANA\$
1	ANANA\$
0	BANANA\$
4	NA\$
2	NANA\$

**Suffix array**



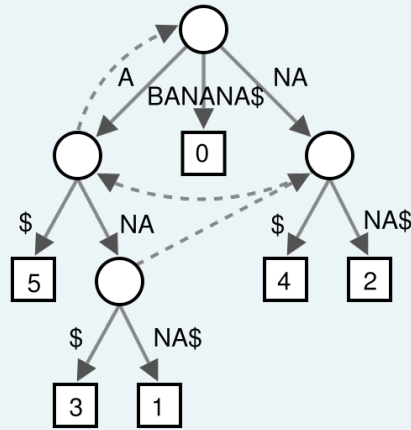
**Seed hash tables**

Many variants, incl. spaced seeds

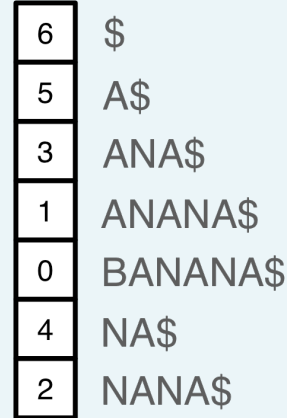
- Choice of index is key to performance

# Indexing

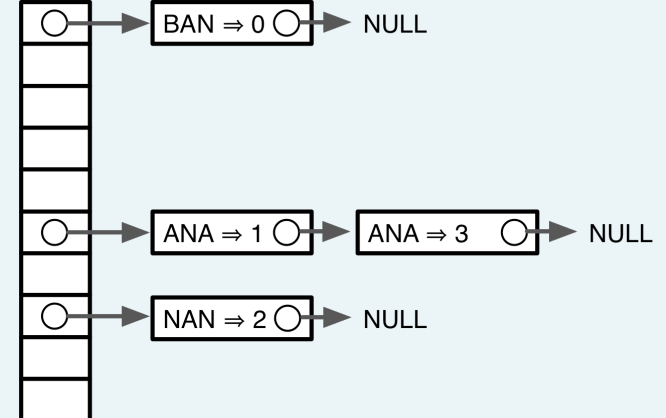
- Genome indices can be big. For human:



> 35 GBs



> 12 GBs



> 12 GBs

- Large indices necessitate painful compromises
  1. Require big-memory machine
  2. Use secondary storage
  3. Build new index each run
  4. Subindex and do multiple passes

# Burrows-Wheeler Transform

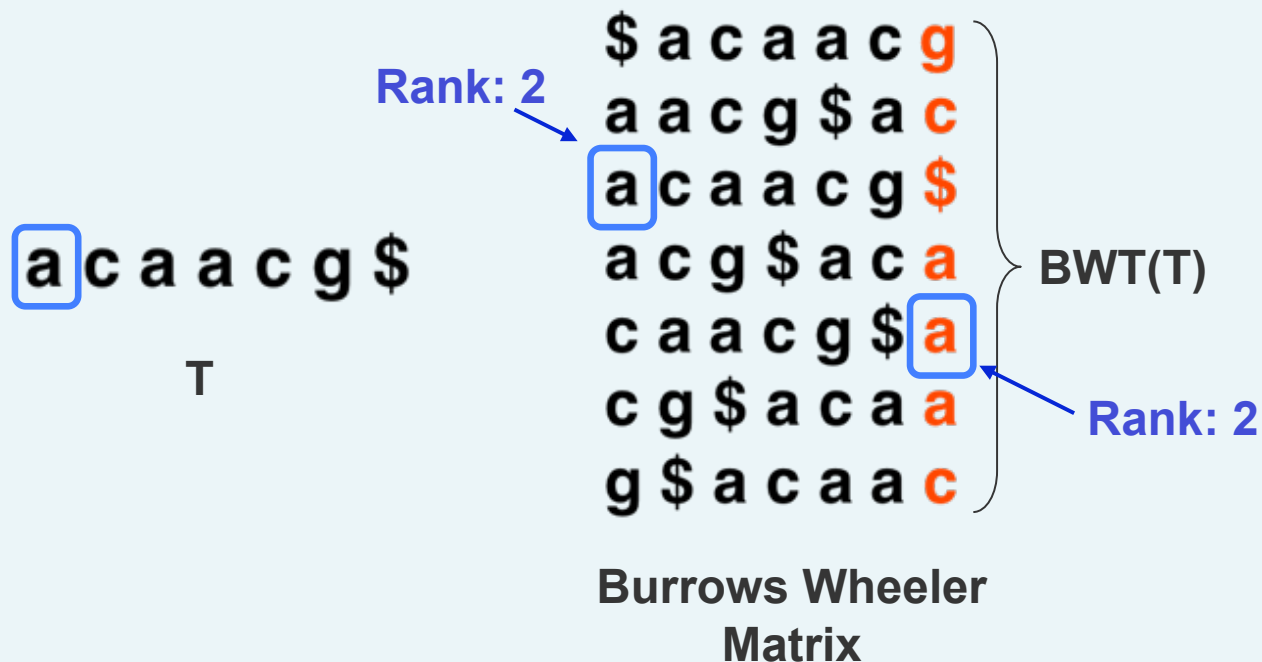
- Reversible permutation used originally in compression



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

# Burrows-Wheeler Transform

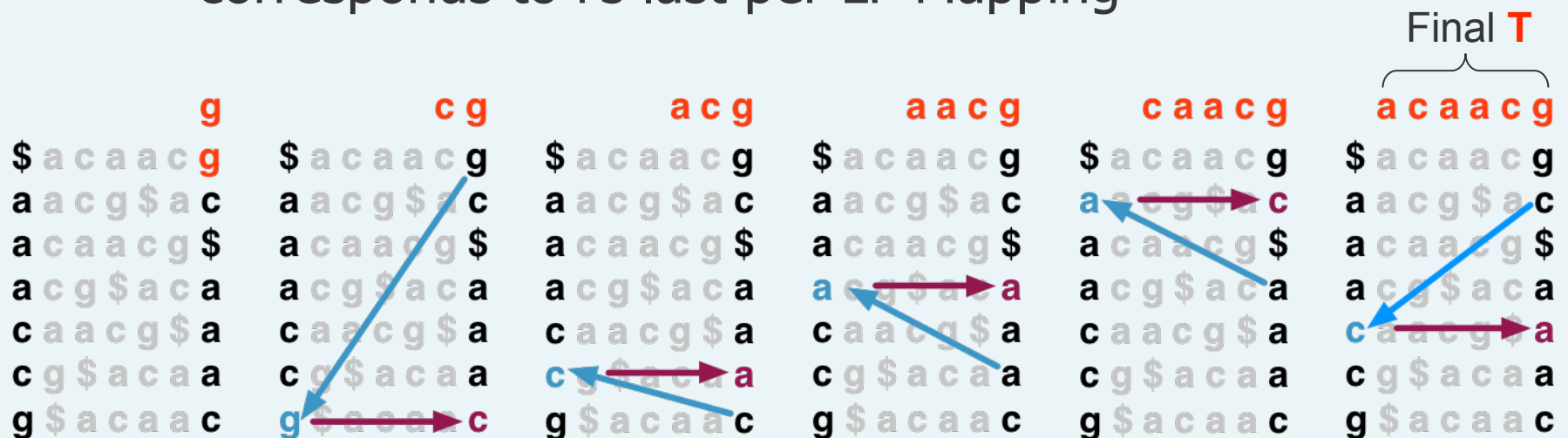
- Property that makes BWT(T) reversible is “LF Mapping”
  - $i^{\text{th}}$  occurrence of a character in **L**ast column is same *text* occurrence as the  $i^{\text{th}}$  occurrence in **F**irst column



# Burrows-Wheeler Transform

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

$$\mathbf{T} = \mathbf{BWT}[\mathbf{LF}(i)] + \mathbf{T}; i = \mathbf{LF}(i)$$
  - Where  $\mathbf{LF}(i)$  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

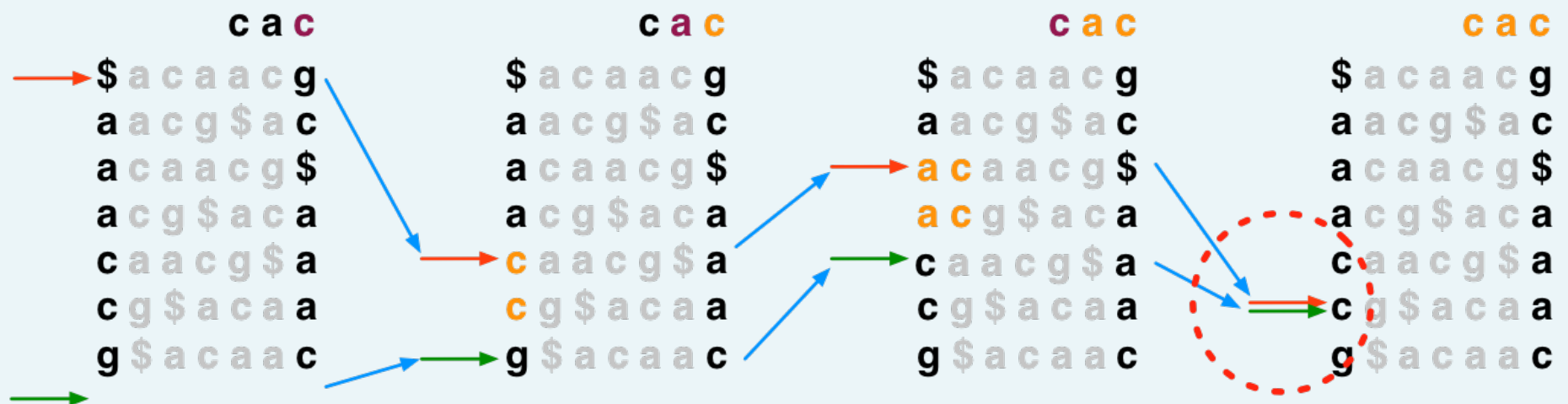
---

- 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



- If range becomes empty (**top** = **bot**) the query suffix (and therefore the query) does not occur in the text

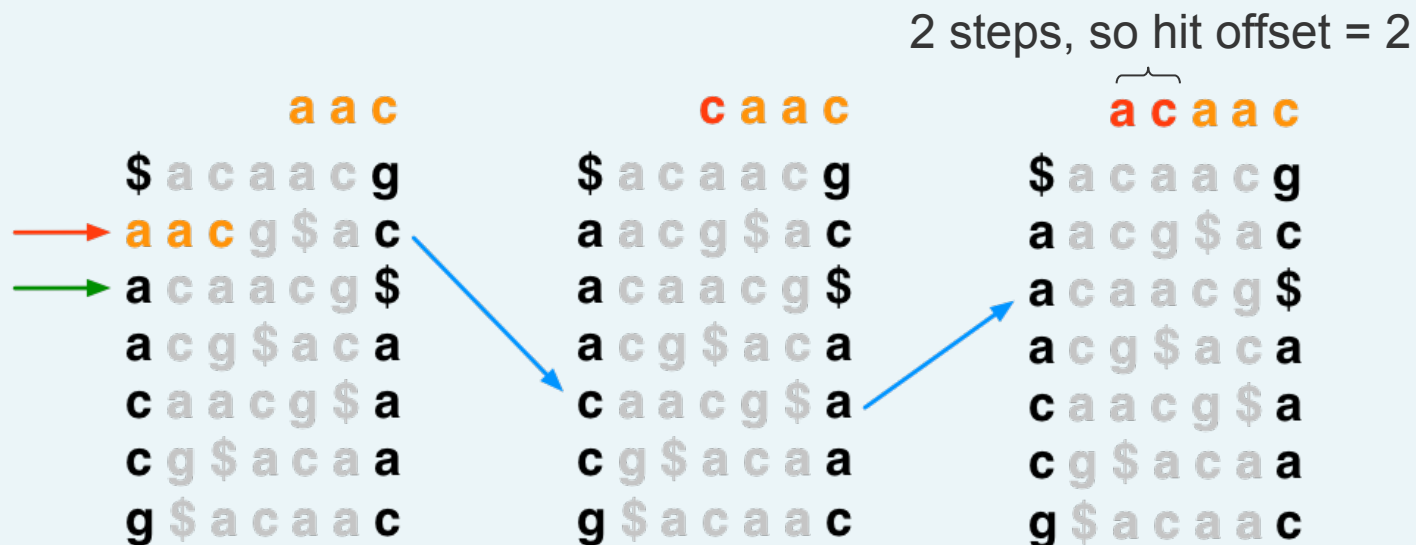
# Rows to Reference Positions

- Once we know a row contains a legal alignment, how do we determine its position in the reference?



# Rows to Reference Positions

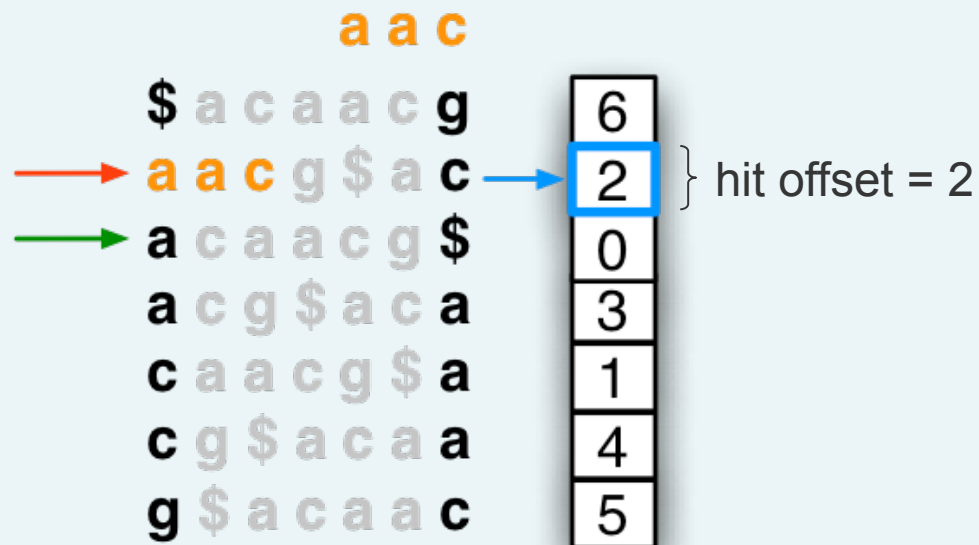
- Naïve solution 1: Use “walk-left” to walk back to the beginning of the text; number of steps = offset of hit



- Linear in length of text in general – too slow

# Rows to Reference Positions

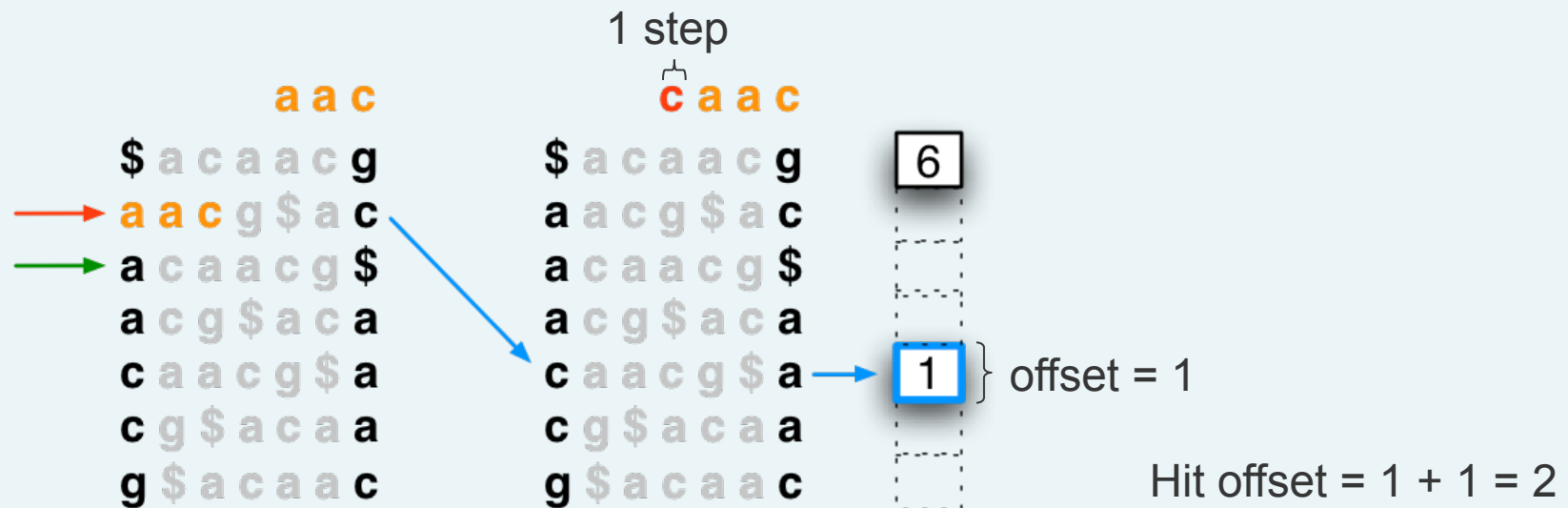
- Naïve solution 2: Keep whole suffix array in memory. Finding reference position is a lookup in the array.



- Suffix array is ~12 gigabytes for human – too big

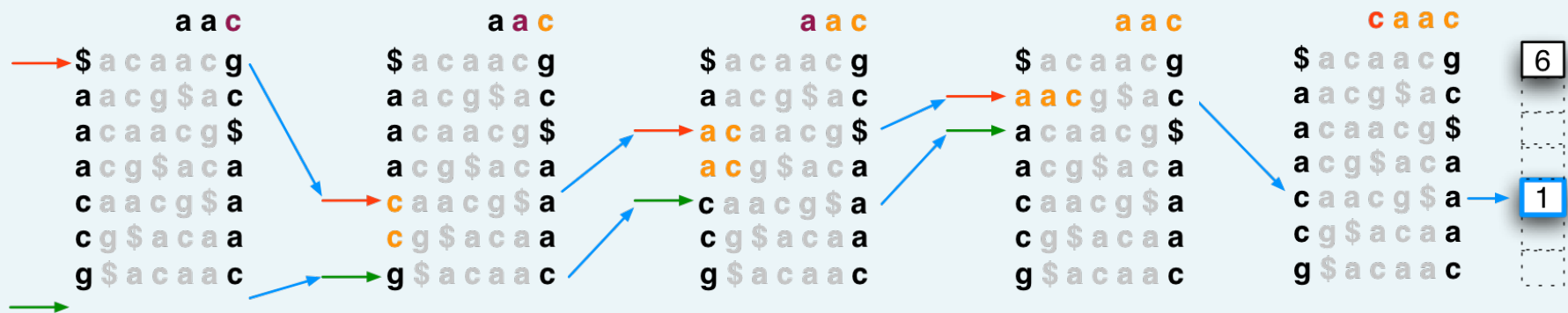
# Rows to Reference Positions

- Hybrid solution: Store *sample* of suffix array; “walk left” to next sampled (“marked”) row to the left
  - Due to Ferragina and Manzini



- Bowtie marks every 32<sup>nd</sup> row by default (configurable)

# Put It All Together

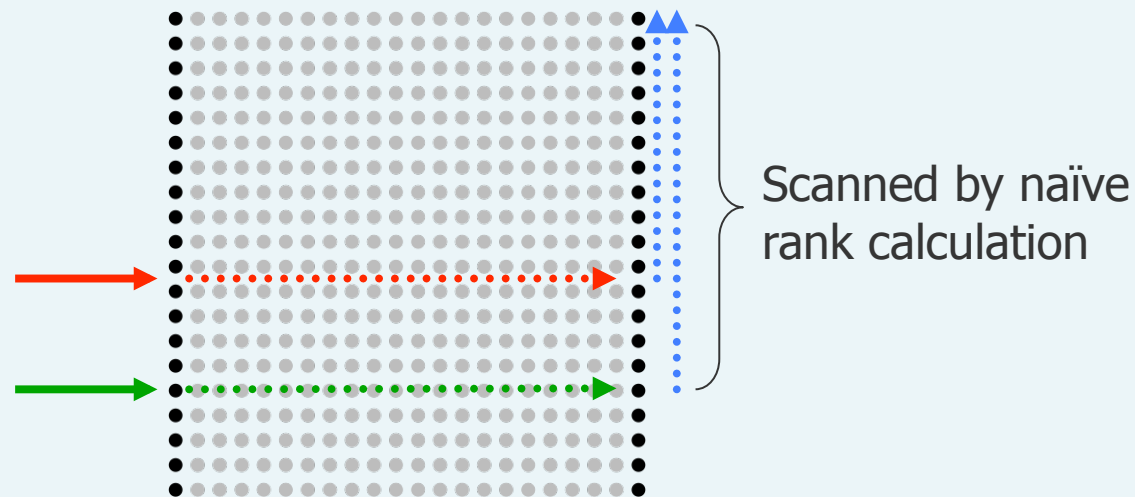


- Algorithm concludes: "aac" occurs at offset 2 in "acaacg"

# Checkpointing in FM Index

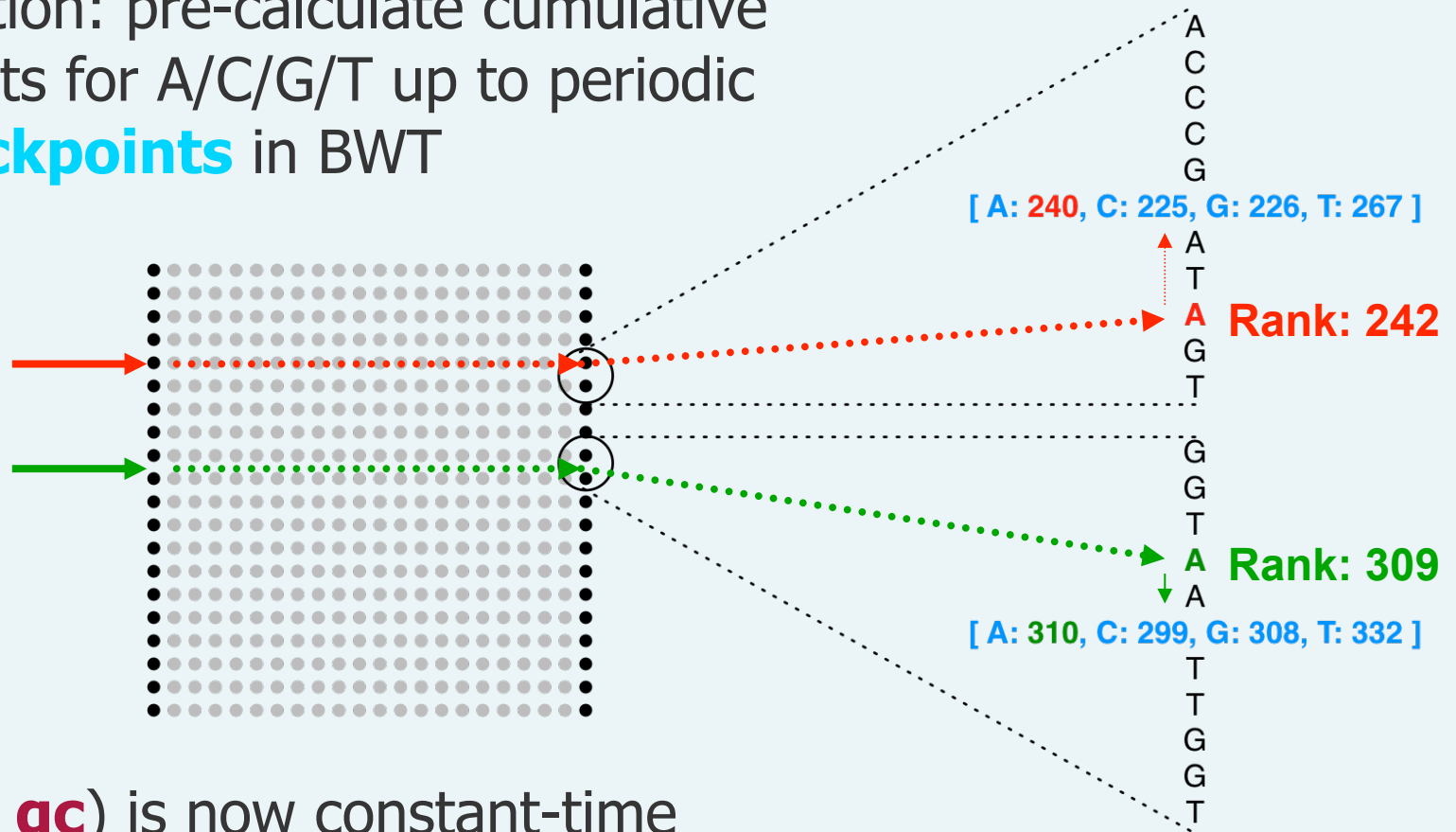
---

- $LF(i, qc)$  must determine the *rank* of  $qc$  in row  $i$
- Naïve way: count occurrences of  $qc$  in all previous rows
  - This  $LF(i, qc)$  is linear in length of text – too slow



# Checkpointing in FM Index

- Solution: pre-calculate cumulative counts for A/C/G/T up to periodic **checkpoints** in BWT



- **LF**(i, **qc**) is now constant-time  
(if space between checkpoints is considered constant)

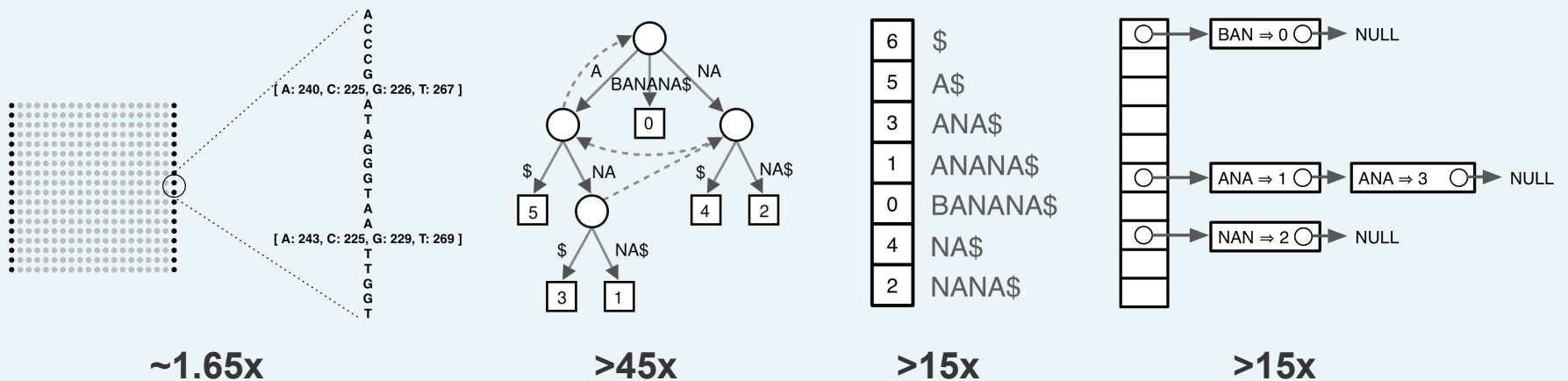
# FM Index is Small

- 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



# FM Index in Bioinformatics

---

- Oligomer counting
  - Healy J *et al*: Annotating large genomes with exact word matches. *Genome Res* 2003, 13(10):2306-2315.
- Whole-genome alignment
  - Lippert RA: Space-efficient whole genome comparisons with Burrows-Wheeler transforms. *J Comp Bio* 2005, 12(4):407-415.
- Smith-Waterman alignment to large reference
  - Lam TW *et al*: Compressed indexing and local alignment of DNA. *Bioinformatics* 2008, 24(6):791-797.

# Short Read Alignment

---

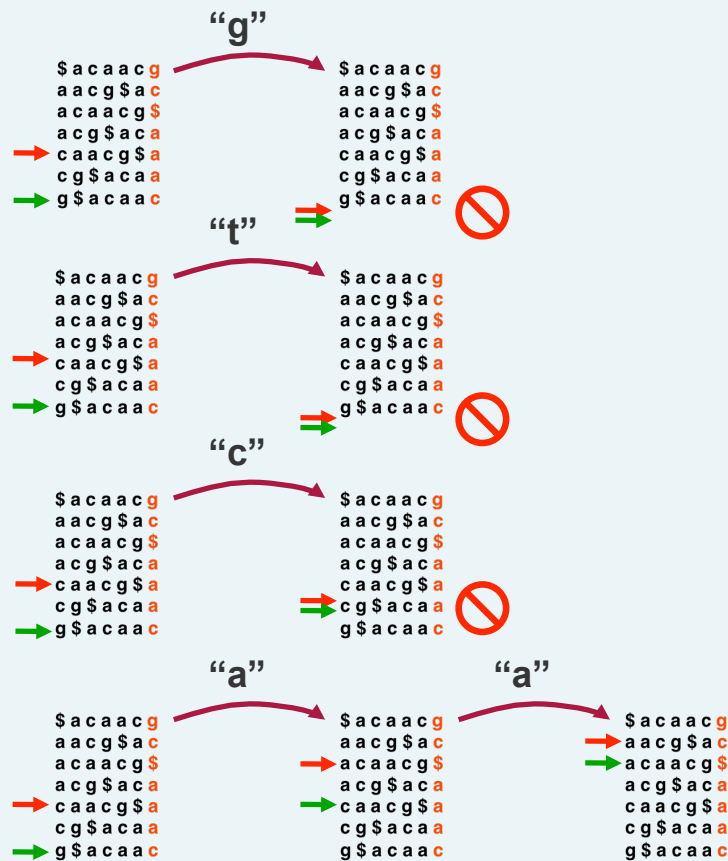
- FM Index finds exact sequence matches quickly in small memory, but short read alignment demands more:
  - Allowances for mismatches
  - Consideration of quality values
- Lam *et al* try index-assisted Smith-Waterman
  - Slower than BLAST
- We tried index-assisted “seed-and-extend”
  - Competitive with other aligners, but not much faster
- Bowtie’s solution: *backtracking quality-aware search*





# Backtracking

- May not be so lucky



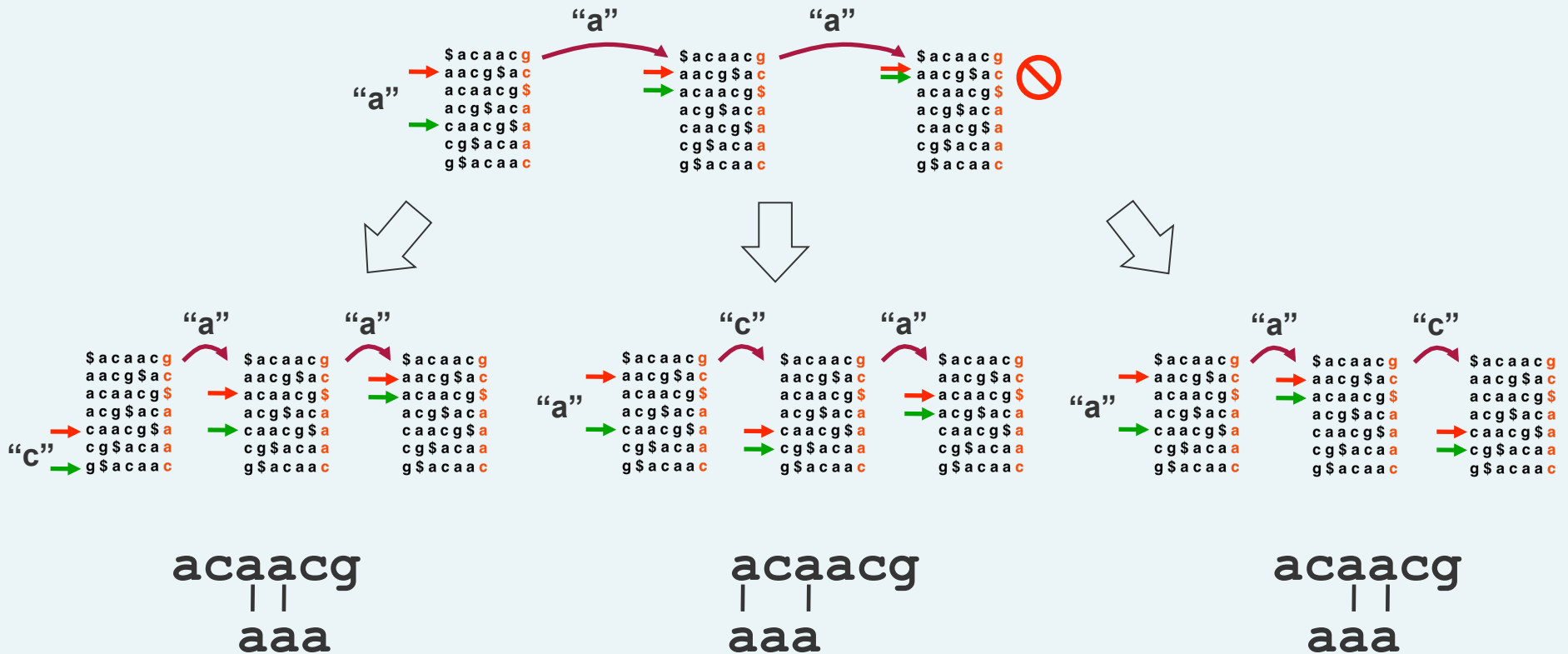
Found this alignment (eventually):

```

acaacg
 |  |
agc
  
```

# Backtracking

- Relevant alignments may lie along multiple paths
  - E.g., Q = "aaa", T = "acaacg"



# Backtracking

---

- Bowtie's `-v <int>` option allows alignments with up to `<int>` mismatches
  - Regardless of quality values
  - Max mismatches allowed: 3
  - Equivalent to SOAP's\* `-v` option

\* Li R *et al*: SOAP: short oligonucleotide alignment program. *Bioinformatics* 2008, 24(5):713-714.

# Qualities

- When backtracking is necessary, Bowtie will backtrack to *leftmost just-visited position* with minimal quality

Sequence: G C C A T A C G G A T T A G C C

Phred Quals: 40 40 35 40 40 40 40 30 30 20 15 15 40 40 40 40

(higher number = higher confidence)

G	C	C	A	T	A	C	G	G	A	T	T	A	G	C	C
40	40	35	40	40	40	40	30	30	20	15	15	40	40	40	40

⊘ ←—————●

G	C	C	A	T	A	C	G	G	A	C	T	A	G	C	C
40	40	35	40	40	40	40	30	30	20	15	15	40	40	40	40

⊘ ←————●

G	C	C	A	T	A	C	G	G	G	C	T	A	G	C	C
40	40	35	40	40	40	40	30	30	20	15	15	40	40	40	40

✓ ←————●

- Greedy, depth-first, not optimal, but simple

# Qualities

- Bowtie supports a Maq\*-like alignment policy
  - $\leq N$  mismatches allowed in first L bases on left end
  - Sum of mismatch qualities may not exceed E
  - N, L and E configured with **-n**, **-l**, **-e**
  - E.g.:

G	C	C	A	T	A	C	G	G	G	C	T	A	G	C	C
40	40	35	40	40	40	40	30	30	20	15	15	40	25	5	5


L=12

E=50, N=2



If N < 2 

If E < 45 

If L < 9 and N < 2 

- Maq-like is Bowtie's default mode (N=2, L=28, E=70)

\* Li H, Ruan J, Durbin R: Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res* 2008.

# Implementation

---

- Free, Open Source under Artistic License
- C++
- Uses SeqAn\* library (<http://www.seqan.de>)
- Uses POSIX threads to exploit parallelism
- **bowtie-build** is the indexer
- **bowtie** is the aligner
- **bowtie-convert** converts Bowtie's alignment output format to Maq's **.map** format
  - Users may leverage tools in the Maq suite, e.g., **maq assemble**, **maq cns2snp**
  - Uses code from Maq

\* Doring A, Weese D, Rausch T, Reinert K: SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics* 2008, 9:11.

# http://bowtie-bio.sf.net

## Bowtie

An ultrafast memory-efficient short read aligner



**Bowtie** is an ultrafast, memory-efficient short read aligner. It aligns short DNA sequences (reads) to the human genome at a rate of 25 million reads per hour on a typical workstation with 2 gigabytes of memory. Bowtie indexes the genome with a Burrows-Wheeler index to keep its memory footprint small: 1.3 GB for the human genome. It supports alignment policies equivalent to [Maq](#) and [SOAP](#) but is much faster: about 35x faster than Maq and over 350x faster than SOAP when aligning to the human genome.



### ❖ 0.9.7 release - 11/8/08

- Added new reporting option `-m <int>` which suppresses all alignments for a particular read if more than `<int>` reportable alignments exist for it.
- Threads now buffer all alignments for a particular read/phase then output all alignments in one critical section. This guarantees that all alignments for a given read/phase appear in one consecutive block of the output, even when multiple threads are operating in parallel.
- Separated the quality-conversion and parsing aspects of the old `--solexa-quals` argument into separate arguments: `--solexa-quals` (quality conversion) and `--integer-quals` (parsing).
- `bowtie-convert` now handles the new (post-0.7.0) Maq alignment format. The new format allows Maq tools to handle reads up to 127 bases, whereas the old format was limited to 63 bases. Added a `-o` option to opt for the old Maq format.
- New `--refout` argument sends alignments to a set of files named `refXXXXX.map`, where `XXXXX` is the 0-padded index of the reference sequence aligned to. Useful for dealing with large datasets aligned to, e.g., the assembled human genome.
- Improved tutorial to use a simple simulated read set (included) to do SNP calls with Maq.
- Added `--nota` option to `bowtie-build`
- Fixed `make_h_sapiens_asm.sh` script to include mitochondrial DNA.

### ❖ TopHat released - 11/8/08

- Cole Trapnell has completed the initial release of [TopHat](#), a fast splice junction mapper for RNA-Seq reads. TopHat aligns RNA-Seq reads to mammalian-sized genomes using Bowtie, and then analyzes the mapping results to identify splice junctions between exons.

#### Site Map

[Home](#)  
[Getting started](#)  
[Manual](#)

#### Latest Release

[Bowtie 0.9.7](#) 11/8/08

#### Pre-built indexes

<a href="#">H. sapiens, NCBI 36.3, contigs</a>	2.1 GB
<a href="#">H. sapiens, NCBI 36.3, assembly</a>	2.0 GB
<a href="#">P. troglodytes, NCBI 2.1, contigs</a>	2.1 GB
<a href="#">R. norvegicus, NCBI 4.1, contigs</a>	1.9 GB
<a href="#">M. musculus, NCBI 37.1, contigs</a>	1.8 GB
<a href="#">C. familiaris, NCBI 2.1, contigs</a>	1.7 GB
<a href="#">G. gallus, NCBI 2.1, contigs</a>	753 MB
<a href="#">T. rubripes, Fugu Gen. Cons. v4</a>	273 MB
<a href="#">D. melanogaster, Flybase, r5.11</a>	121 MB
<a href="#">A. thaliana, TAIR, TAIR8</a>	92 MB
<a href="#">C. elegans, Wormbase, WS190</a>	77 MB

# Indexing Performance

---

- Bowtie employs a indexing algorithm\* that can trade flexibly between memory usage and running time
- For human (NCBI 36.3) on 2.4 GHz AMD Opteron:

Physical memory Target	Actual peak memory footprint	Wall clock time
16 GB	14.4 GB	4h:36m
8 GB	5.84 GB	5h:05m
4 GB	3.39 GB	7h:40m
2 GB	1.39 GB	21h:30m

\* Kärkkäinen J: Fast BWT in small space by blockwise suffix sorting. *Theor Comput Sci* 2007, 387(3):249-257.

# Comparison to Maq & SOAP

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie -v 2 (server)	15m:07s	15m:41s	33.8 M	1,149 MB	-	67.4
SOAP (server)	91h:57m:35s	91h:47m:46s	0.08 M	13,619 MB	351x	67.3
Bowtie (PC)	16m:41s	17m:57s	29.5 M	1,353 MB	-	71.9
Maq (PC)	17h:46m:35s	17h:53m:07s	0.49 M	804 MB	59.8x	74.7
Bowtie (server)	17m:58s	18m:26s	28.8 M	1,353 MB	-	71.9
Maq (server)	32h:56m:53s	32h:58m:39s	0.27 M	804 MB	107x	74.7

- PC: 2.4 GHz Intel Core 2, 2 GB RAM
- Server: 2.4 GHz AMD Opteron, 32 GB RAM
- Bowtie v0.9.6, Maq v0.6.6, SOAP v1.10
- SOAP not run on PC due to memory constraints
- Reads: FASTQ 8.84 M reads from 1000 Genomes (Acc: SRR001115)
- Reference: Human (NCBI 36.3, contigs)

# Comparison to Maq & SOAP

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie -v 2 (server)	15m:07s	15m:41s	<b>33.8 M</b>	1,149 MB	-	67.4
SOAP (server)	91h:57m:35s	91h:47m:46s	0.08 M	13,619 MB	<b>351x</b>	67.3
Bowtie (PC)	16m:41s	17m:57s	<b>29.5 M</b>	1,353 MB	-	71.9
Maq (PC)	17h:46m:35s	17h:53m:07s	0.49 M	804 MB	<b>59.8x</b>	74.7
Bowtie (server)	17m:58s	18m:26s	<b>28.8 M</b>	1,353 MB	-	71.9
Maq (server)	32h:56m:53s	32h:58m:39s	0.27 M	804 MB	<b>107x</b>	74.7

- Bowtie delivers about 30 million alignments per CPU hour

# Comparison to Maq & SOAP

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie -v 2 (server)	15m:07s	15m:41s	33.8 M	1,149 MB	-	<b>67.4</b>
SOAP (server)	91h:57m:35s	91h:47m:46s	0.08 M	13,619 MB	351x	<b>67.3</b>
Bowtie (PC)	16m:41s	17m:57s	29.5 M	1,353 MB	-	71.9
Maq (PC)	17h:46m:35s	17h:53m:07s	0.49 M	804 MB	59.8x	74.7
Bowtie (server)	17m:58s	18m:26s	28.8 M	1,353 MB	-	71.9
Maq (server)	32h:56m:53s	32h:58m:39s	0.27 M	804 MB	107x	74.7

- Disparity in reads aligned between Bowtie (67.4%) and SOAP (67.3%) is slight

# Comparison to Maq & SOAP

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie -v 2 (server)	15m:07s	15m:41s	33.8 M	1,149 MB	-	67.4
SOAP (server)	91h:57m:35s	91h:47m:46s	0.08 M	13,619 MB	351x	67.3
Bowtie (PC)	16m:41s	17m:57s	29.5 M	1,353 MB	-	<b>71.9</b>
Maq (PC)	17h:46m:35s	17h:53m:07s	0.49 M	804 MB	59.8x	<b>74.7</b>
Bowtie (server)	17m:58s	18m:26s	28.8 M	1,353 MB	-	<b>71.9</b>
Maq (server)	32h:56m:53s	32h:58m:39s	0.27 M	804 MB	107x	<b>74.7</b>

- Disparity in reads aligned between Bowtie (71.9%) and Maq (74.7%) is more substantial (2.8%)
  - Mostly because Maq `-n 2` reports some, but not all, alignments with 3 mismatches in first 28 bases
  - Fraction (<5%) of disparity is due to Bowtie's backtracking limit (a heuristic not discussed here)

# Comparison to Maq & SOAP

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie -v 2 (server)	15m:07s	15m:41s	33.8 M	<b>1,149 MB</b>	-	67.4
SOAP (server)	91h:57m:35s	91h:47m:46s	0.08 M	<b>13,619 MB</b>	351x	67.3
Bowtie (PC)	16m:41s	17m:57s	29.5 M	<b>1,353 MB</b>	-	71.9
Maq (PC)	17h:46m:35s	17h:53m:07s	0.49 M	<b>804 MB</b>	59.8x	74.7
Bowtie (server)	17m:58s	18m:26s	28.8 M	<b>1,353 MB</b>	-	71.9
Maq (server)	32h:56m:53s	32h:58m:39s	0.27 M	<b>804 MB</b>	107x	74.7

- Bowtie and Maq have memory footprints compatible with a typical workstation with 2 GB of RAM
  - Maq builds non-reusable spaced-seed index on reads; recommends segmenting reads into chunks of 2M (which we did)
- SOAP requires a computer with >13 GB of RAM
  - SOAP builds non-reusable spaced-seed index on genome

# Comparison to Maq w/ Poly-A Filter

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Bowtie speedup	Reads aligned (%)
Bowtie (PC)	16m:39s	17m:47s	29.8 M	1,353 MB	-	<b>74.9</b>
Maq (PC)	11h:15m:58s	11h:22m:02s	0.78 M	804 MB	<b>38.4x</b>	<b>78.0</b>
Bowtie (server)	18m:20s	18m:46s	28.8 M	1,352 MB	-	<b>74.9</b>
Maq (server)	18h:49m:07s	18h:50m:16s	0.47 M	804 MB	<b>60.2x</b>	<b>78.0</b>

- Maq documentation: reads with “poly-A artifacts” impair Maq’s performance
- Re-ran previous experiment after running Maq’s “catfilter” to eliminate 438K poly-A reads
- Maq makes up some ground, but Bowtie still >35x faster
- Similar disparity in reads aligned, for same reasons

# Multithreaded Scaling

	CPU time	Wall clock time	Reads per hour	Peak virtual memory footprint	Speedup
Bowtie, 1 thread (server)	18m:19s	18m:46s	28.3 M	<b>1,353 MB</b>	-
Bowtie, 2 threads (server)	20m:34s	10m:35s	<b>50.1 M</b>	<b>1,363 MB</b>	<b>1.77x</b>
Bowtie, 4 threads (server)	23m:09s	6m:01s	<b>88.1 M</b>	<b>1,384 MB</b>	<b>3.12x</b>

- Bowtie uses POSIX threads to exploit multi-processor computers
  - Reads are distributed across parallel threads
  - Threads synchronize when fetching reads, outputting results, etc.
  - Index is shared by all threads, so footprint does not increase substantially as # threads increases
- Table shows performance results for Bowtie v0.9.6 on 4-core Server with 1, 2, 4 threads

# 1000 Genomes Genotyping



```
...CCATAG      TATGCGCCC      CGGA AATT  CGGTATAC
...CCAT      CTATATGCG      TCGGA AATT  CGGTATAC
...CCAT GGCTATATG      CTATCGGAAA  GCGGTATA
...CCA AGGCTATAT      CCTATCGGA  TTGCGGTA  C...
...CCA AGGCTATAT      GCCCTATCG  TTTGCGGT  C...
...CC  AGGCTATAT      GCCCTATCG  AAATTTGC  ATAC...
...CC  TAGGCTATA  GCGCCCTA  AAATTTGC  GTATAC...
...CCATAGGCTATATGCGCCCTATCGGCAATTTGCGGTATAC...
```

- Bowtie aligns all 1000-Genomes (Build 2) reads for human subject NA12892 on a 2.4 Ghz Core 2 workstation with 4 GB of RAM with 4 parallel threads:
  - 14.3x coverage, 935 M reads, 42.9 Gbases
  - Running time: 14 hrs – 1 overnight

# Future Work

---


- Paired-end alignment
- Finding alignments with insertions and deletions
- ABI color-space support

# TopHat: Bowtie for RNA-seq

- TopHat is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads using Bowtie, and then analyzes the mapping results to identify splice junctions between exons.
  - Contact: Cole Trapnell (cole@cs.umd.edu)
  - <http://tophat.cbcb.umd.edu>


## TopHat

A splice junction mapper for short RNA-Seq reads



TopHat is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner [Bowtie](#), and then analyzes the mapping results to identify splice junctions between exons.

TopHat is a collaborative effort between the University of Maryland [Center for Bioinformatics and Computational Biology](#) and the University of California [Departments of Mathematics](#) and [Molecular and Cell Biology](#).



» **0.7.1 release - 11/08/2008**

The following issues have been fixed:

- o Maq 0.7.0 changed the Maq map file format. Bowtie 0.9.7 now supports both the new and old mapping format, and thus so now does TopHat. TopHat now checks the version of Maq on the system and uses the correct format.
- o Minor command line interface improvements
- o The `-x` option has been added to allow the use of FASTQ files that are scaled on the Solexa quality scale, as opposed to Phred (the default). Note that TopHat doesn't support FASTQ-int, only ASCII-encoded qualities are used.
- o The `-D` option has been added, allowing users to specify when to look for junctions within single islands, as opposed to just between two distinct islands
- o The `-Q` option allows the user to specify a Phred quality character below which the island consensus caller will use the reference base call. That is, TopHat will not allow SNPs to be called where base quality drops below a certain threshold.

### Site Map

- [Home](#)
- [Getting started](#)
- [Manual](#)

### Latest Release

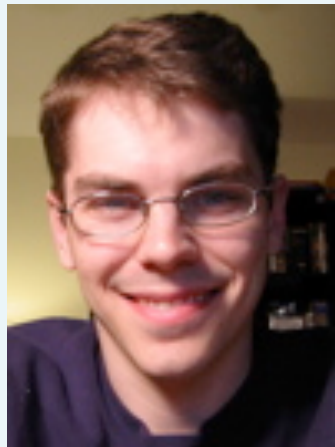
<a href="#">TopHat 0.7.1</a>	11/08/2008
------------------------------	------------

### Pre-built indexes

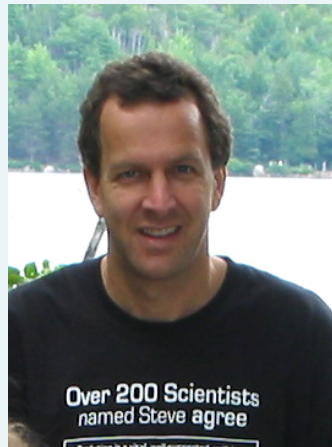
<a href="#">H. sapiens, NCBI 36.3, contigs</a>	2.1 GB
<a href="#">H. sapiens, NCBI 36.3, assembly</a>	2.0 GB
<a href="#">P. troglodytes, NCBI 2.1, contigs</a>	2.1 GB
<a href="#">R. norvegicus, NCBI 4.1, contigs</a>	1.9 GB

# Work With

---



Cole Trapnell



Steven Salzberg



Mihai Pop



# Extra Slides

---

# Bowtie Usage

---

Usage: bowtie [options]\* <ebwt\_base> <query\_in> [<hit\_outfile>]

<ebwt\_base> ebwt filename minus trailing .1.ebwt/.2.ebwt  
<query\_in> comma-separated list of files containing query reads  
(or the sequences themselves, if -c is specified)  
<hit\_outfile> file to write hits to (default: stdout)

Options:

-q query input files are FASTQ .fq/.fastq (default)  
-f query input files are (multi-)FASTA .fa/.mfa  
-c query sequences given on command line (as <query\_in>)  
-e/--maqerr <int> max sum of mismatch quals (rounds like maq; default: 70)  
-l/--seedlen <int> seed length (default: 28)  
-n/--seedmms <int> max mismatches in seed (0, 1 or 2, default: 2)  
-v <int> report end-to-end hits w/ <=v mismatches; ignore qualities  
-5/--trim5 <int> trim <int> bases from 5' (left) end of reads  
-3/--trim3 <int> trim <int> bases from 3' (right) end of reads  
-u/--qupto <int> stop after the first <int> reads  
-t/--time print wall-clock time taken by search phases  
--solexa-quals convert FASTQ qualities from solexa-scaled to phred  
--concise write hits in a concise format  
--maxns <int> skip reads w/ >n no-confidence bases (default: no limit)  
-o/--offrate <int> override offrate of Ebwt; must be >= value in index  
--seed <int> seed for random number generator  
--verbose verbose output (for debugging)  
--version print version information and quit

# Bowtie Indexer Usage

---

```
Usage: bowtie-build [options]* <reference_in> <ebwt_outfile_base>
  reference_in          comma-separated list of files with ref sequences
  ebwt_outfile_base    write Ebwt data to files with this dir/basename
Options:
  -f                   reference files are Fasta (default)
  -c                   reference sequences given on cmd line (as <seq_in>)
  --bmax <int>        max bucket sz for blockwise suffix-array builder
  --bmaxmultsqrt <int> max bucket sz as multiple of sqrt(ref len)
  --bmaxdivn <int>    max bucket sz as divisor of ref len
  --dcv <int>         diff-cover period for blockwise (default: 1024)
  --nodc              disable difference cover (blockwise is quadratic)
  -o/--offrate <int> SA index is kept every 2^offRate BWT chars
  -t/--ftabchars <int> # of characters in initial lookup table key
  --big --little      endianness (default: little, this host: little)
  --seed <int>        seed for random number generator
  --cutoff <int>      truncate reference at prefix of <int> bases
  -q/--quiet          verbose output (for debugging)
  -h/--help           print detailed description of tool and its options
  --version           print version information and quit
```

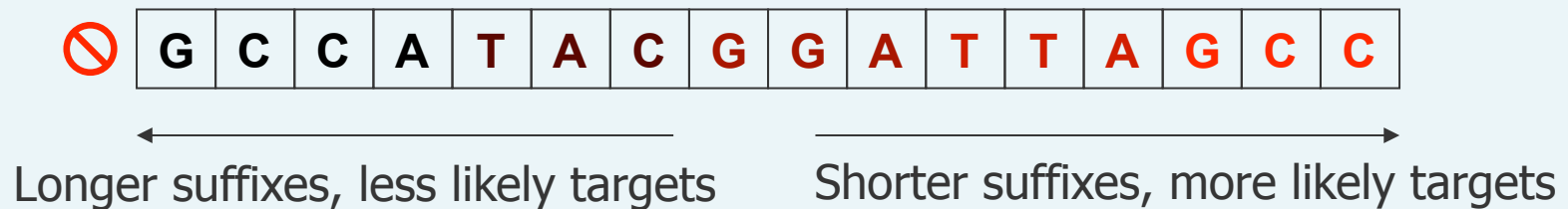
# Reporting

---

- k <int>** Report up to <int> valid alignments per read (default: 1). Validity of alignments is determined by the alignment policy (combined effects of -n, -v, -l, and -e). If many alignments are reported, they may be subject to stratification; see --best, --nostrata. Bowtie is designed to be very fast for small -k but **BOWTIE CAN BECOME VERY SLOW AS -k INCREASES.**
- a/--all** Report all valid alignments per read (default: off). Validity of alignments is determined by the alignment policy (combined effects of -n, -v, -l, and -e). Reported alignments may be subject to stratification; see --best, --nostrata. Bowtie is designed to be very fast for small -k; **BOWTIE CAN CAN BECOME VERY SLOW IF -a/--all IS SPECIFIED.**
- best** Reported alignments must belong to the best possible alignment "stratum" (default: off). A stratum is a category defined by the number of mismatches present in the alignment (for -n, the number of mismatches present in the seed region of the alignment). E.g., if --best is not specified, Bowtie may sometimes report an alignment with 2 mismatches in the seed even though there exists an unreported alignment with 1 mismatch in the seed. **bowtie IS ABOUT 3-5 TIMES SLOWER WHEN --best IS SPECIFIED.**
- nostrata** If many valid alignments exist and are reportable (according to the --best and -k options) and they fall into various alignment "strata", report all of them. By default, Bowtie only reports those alignments that fall into the best stratum, i.e., the one with fewest mismatches. **BOWTIE CAN BECOME VERY SLOW WHEN --nostrata IS COMBINED WITH -k OR -a.**

# Excessive Backtracking

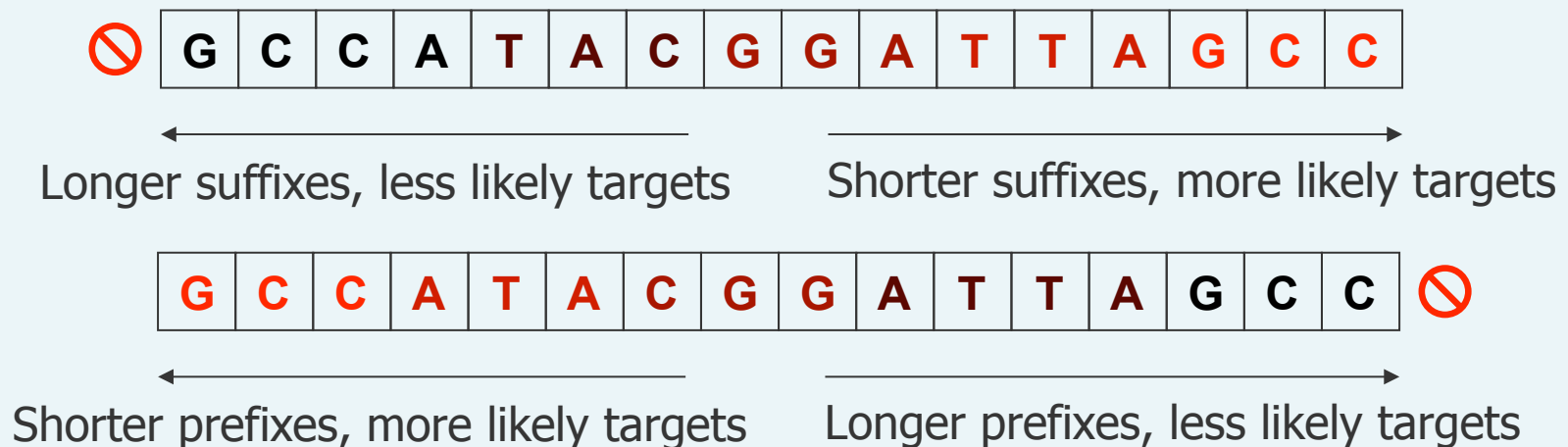
- Bowtie only backtracks if it can make progress, i.e., if **top**  $\neq$  **bot** after the backtrack
  - Rightmost positions are likeliest targets because shorter suffixes are likeliest to occur “by chance”



- When  $>1$  mismatch is allowed, such backtracks can easily dominate running time and make search slow

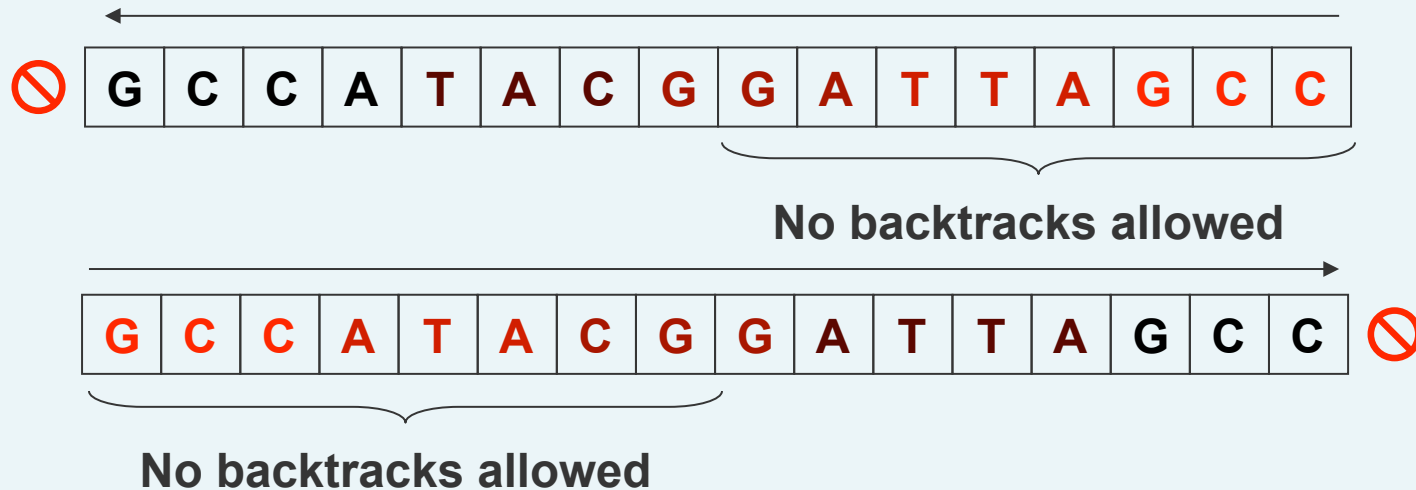
# Excessive Backtracking

- Solution: Double indexing
- We've considered matching from right to left, but what if left-to-right were possible too?

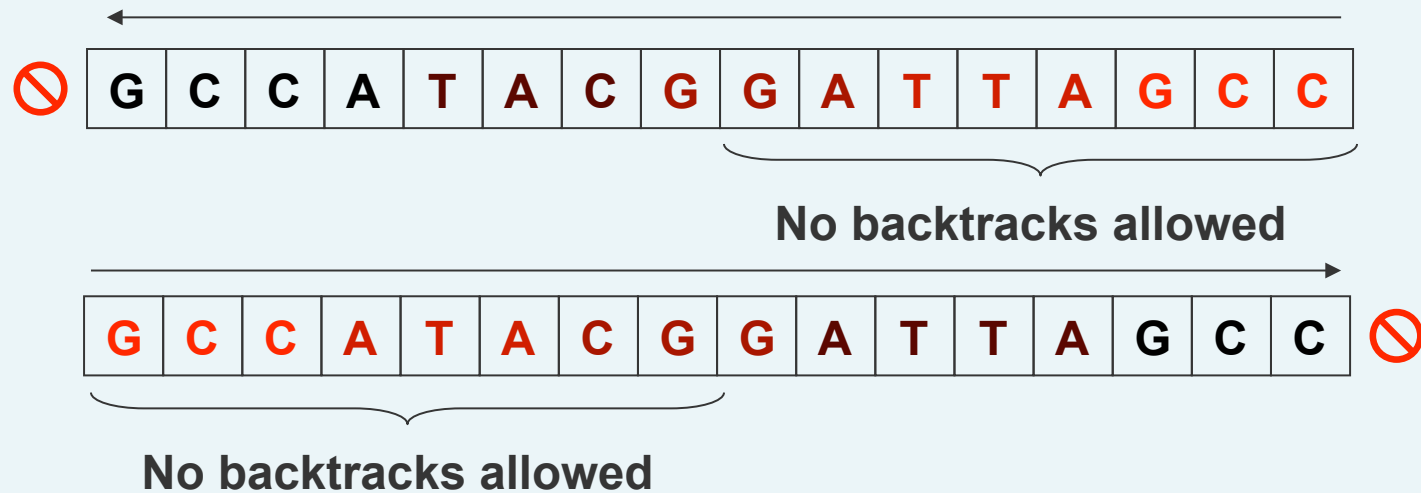


# Excessive Backtracking

- Suggests a multi-stage scheme that minimizes excessive backtracking in reddest regions
  - Workflow for up to 1-mismatch that matches in both directions & disallows backtracks in reddest regions:



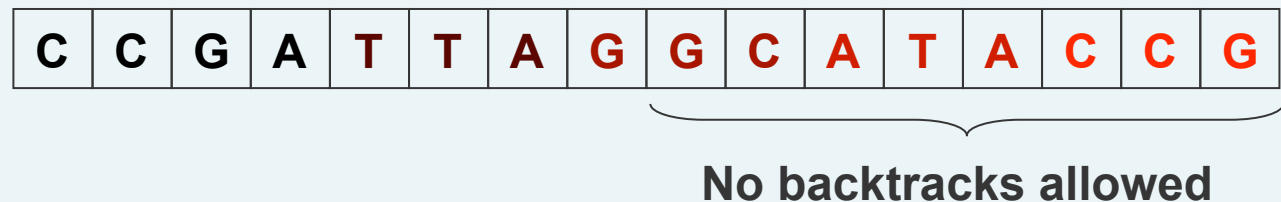
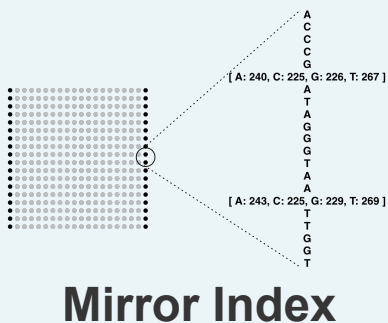
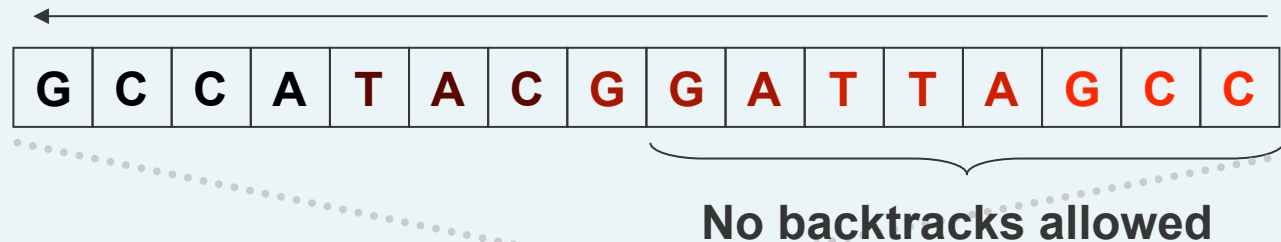
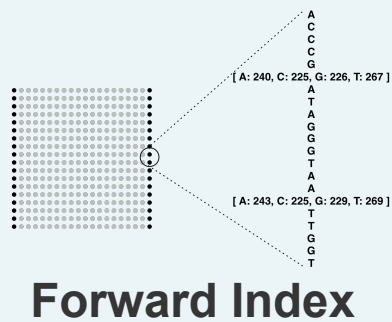
# Excessive Backtracking



- Minimizes backtracks by disallowing backtracks in reddest regions
- Maintains full sensitivity by matching in both directions

# Excessive Backtracking

- But how to match left-to-right?
- Double indexing:
  - Reverse read and use “mirror index”: index for reference with sequence reversed



# Demand

---

“True understanding of how genes function requires knowledge of their expression patterns, their impact on all other genes and their effects on DNA structure and modifications. **These data will have to be obtained across large numbers of cell types, individuals, environments and time points.**”

- Kahvejian, A., Quackenbush, J., Thompson, J.F.,  
“What would you do if you could sequence everything?” *Nat. Biotechnol.* 26, 1099 (Oct 2008)



# Wanted: Scalable Algorithms

---

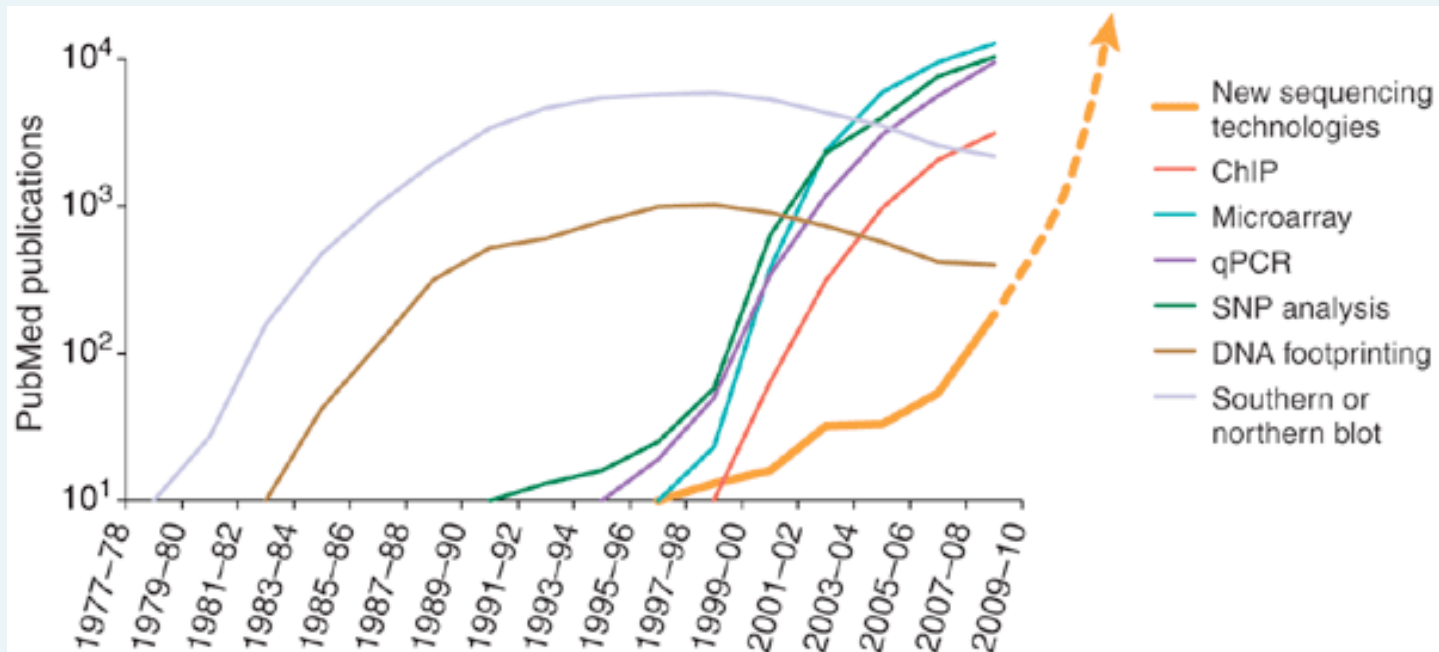
“...the overwhelming amounts of data being produced are the equivalent of taking a drink from a fire hose...”

“...**grant-awarding bodies should start focusing on the back-end bioinformatics** as much as the sequencing technology itself. And as the **bioinformatics bottleneck** threatens to limit instrument sales, **manufacturers as a group have a massive incentive to unblock it.**”

- Editorial, *Nature Biotechnology* 26, 1099 (Oct 2008)



# NGS Trend



PubMed was “searched in two-year increments for key words and the number of hits plotted over time.”

**From the following article**

[What would you do if you could sequence everything?](#)

Avak Kahvejian, John Quackenbush & John F Thompson

Nature Biotechnology 26, 1125 - 1133 (2008)

doi:10.1038/nbt1494