Scalable Solutions for DNA Sequence Analysis

Michael Schatz

March 11, 2010
Argonne National Lab
Outline

1. Genome Assembly by Analogy
2. DNA Sequencing and Genomics
3. MapReduce for Sequence Analysis
   1. K-mer counting
   2. Read Mapping & Genotyping
   3. Genome Assembly
Shredded Book Reconstruction

- Dickens accidentally shreds the first printing of *A Tale of Two Cities*
  - Text printed on 5 long spools

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, ...

- How can he reconstruct the text?
  - 5 copies x 138,656 words / 5 words per fragment = 138k fragments
  - The short fragments from every copy are mixed together
  - Some fragments are identical
Greedy Reconstruction

The repeated sequence make the correct reconstruction ambiguous

- It was the best of times, it was the [worst/age]

Model sequence reconstruction as a graph problem.
de Bruijn Graph Construction

- $D_k = (V,E)$
  - $V = \text{All length-}k\ \text{subfragments} \ (k < l)$
  - $E = \text{Directed edges between consecutive subfragments}$
    - Nodes overlap by $k-1$ words

- Locally constructed graph reveals the global sequence structure
  - Overlaps between sequences implicitly computed

---

Original Fragment | Directed Edge
---|---
It was the best of | It was the best of

---

de Bruijn, 1946
Idury and Waterman, 1995
Pevzner, Tang, Waterman, 2001
de Bruijn Graph Assembly

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness.

A unique Eulerian tour of the graph reconstructs the original text.

If a unique tour does not exist, try to simplify the graph as much as possible.
Generally an exponential number of compatible sequences
- Value computed by application of the BEST theorem (Hutchinson, 1975)

\[
\mathcal{W}(G, t) = (\det L) \left\{ \prod_{u \in V} (r_u - 1)! \right\} \left\{ \prod_{(u,v) \in E} a_{uv}! \right\}^{-1}
\]

\( L = n \times n \) matrix with \( r_u-a_{uu} \) along the diagonal and \(-a_{uv}\) in entry \( uv \)

\( r_u = d^+(u)+1 \) if \( u=t \), or \( d^+(u) \) otherwise

\( a_{uv} \) = multiplicity of edge from \( u \) to \( v \)

Assembly Complexity of Prokaryotic Genomes using Short Reads.
Genomics

Your genome influences (almost) all aspects of your life

- Anatomy & Physiology: 10 fingers & 10 toes, organs, neurons
- Diseases: Sickle Cell Anemia, Down Syndrome, Cancer
- Psychological: Intelligence, Personality, Bad Driving

Your environment also influences your life

- Genome as a recipe, not a blueprint
Genomics across the Tree of Life

Selected Genomes

- *M. gallopavo* (Folkerts et al., 2010*)
- *A. dorsata* (Ruepell et al., 2010*)
- *V. destructor* (Cornman et al., 2010*)
- *N. ceranae* (Cornman et al., 2009)
- *B. taurus* (Zimin et al., 2009)
- *C. papaya* (Ming et al., 2008)
- *X. oryzae* (Salzberg et al., 2008)
- *T. vaginalis* (Carlton et al., 2007)
- Drosophila (Drosophila 12 genomes consortium, 2007)
- *B. malayi* (Ghedin et al., 2007)
- *A. aegypti* (Nene et al., 2007)
- Campylobacter (Fouts et al., 2005)

* In preparation or under review
DNA Sequencing

Genome of an organism encodes the genetic information in long sequence of 4 DNA nucleotides: ACGT

- Bacteria: ~3 million bp
- Humans: ~3 billion bp

Current DNA sequencing machines can generate 1-2 Gbp of sequence per day, in millions of short reads

- Per-base error rate estimated at 1-2% (Simpson et al., 2009)
- Sequences originate from random positions of the genome

Recent studies of entire human genomes analyzed 3.3B (Wang, et al., 2008) & 4.0B (Bentley, et al., 2008) 36bp reads

- ~100 GB of compressed sequence data
## The Evolution of DNA Sequencing

<table>
<thead>
<tr>
<th>Year</th>
<th>Genome</th>
<th>Technology</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>Venter et al.</td>
<td>Sanger (ABI)</td>
<td>$300,000,000</td>
</tr>
<tr>
<td>2007</td>
<td>Levy et al.</td>
<td>Sanger (ABI)</td>
<td>$10,000,000</td>
</tr>
<tr>
<td>2008</td>
<td>Wheeler et al.</td>
<td>Roche (454)</td>
<td>$2,000,000</td>
</tr>
<tr>
<td>2008</td>
<td>Ley et al.</td>
<td>Illumina</td>
<td>$1,000,000</td>
</tr>
<tr>
<td>2008</td>
<td>Bentley et al.</td>
<td>Illumina</td>
<td>$250,000</td>
</tr>
<tr>
<td>2009</td>
<td>Pushkarev et al.</td>
<td>Helicos</td>
<td>$48,000</td>
</tr>
<tr>
<td>2009</td>
<td>Drmanac et al.</td>
<td>Complete Genomics</td>
<td>$4,400</td>
</tr>
</tbody>
</table>

(Pushkarev et al., 2009)

---

Critical Computational Challenges: Alignment and Assembly of Huge Datasets
Hadoop MapReduce

- MapReduce is the parallel distributed framework invented by Google for large data computations.
  - Data and computations are spread over thousands of computers, processing petabytes of data each day (Dean and Ghemawat, 2004)
  - Indexing the Internet, PageRank, Machine Learning, etc…
  - Hadoop is the leading open source implementation

- Benefits
  - Scalable, Efficient, Reliable
  - Easy to Program
  - Runs on commodity computers

- Challenges
  - Redesigning / Retooling applications
  - Not Condor, Not MPI
  - Everything in MapReduce
K-mer Counting

- Application developers focus on 2 (+1 internal) functions
  - **Map**: input $\rightarrow$ key:value pairs
  - **Shuffle**: Group together pairs with same key
  - **Reduce**: key, value-lists $\rightarrow$ output

ATGAACCTTA

- (ATG:1) (ACC:1)
- (TGA:1) (CCT:1)
- (GAA:1) (CTT:1)
- (AAC:1) (TTA:1)

GAACAACCTTA

- (GAA:1) (AAC:1)
- (AAC:1) (ACT:1)
- (ACA:1) (CTT:1)
- (CAA:1) (TTA:1)

TTTAGGCAAC

- (TTT:1) (GGC:1)
- (TTA:1) (GCA:1)
- (TAG:1) (CAA:1)
- (AGG:1) (AAC:1)

Shuffle

- ACA:1
- ATG:1
- CAA:2
- GCA:1
- TGA:1
- TTA:3

Map, Shuffle & Reduce

All Run in Parallel
Hadoop Architecture

- **Hadoop Distributed File System (HDFS)**
  - Data files partitioned into large chunks (64MB), replicated on multiple nodes
  - NameNode stores metadata information (block locations, directory structure)

- **Master node (JobTracker)** schedules and monitors work on slaves
  - Computation moves to the data, rack-aware scheduling

- **Hadoop MapReduce system won the 2009 GreySort Challenge**
  - Sorted 100 TB in 173 min (578 GB/min) using 3452 nodes and 4x3452 disks
Short Read Mapping

- Given a reference and many subject reads, report one or more “good” end-to-end alignments per alignable read
  - Find where the read most likely originated
  - Fundamental computation for many assays
    - Genotyping
    - RNA-Seq
    - Methyl-Seq
    - Structural Variations
    - Chip-Seq
    - Hi-C-Seq
- Desperate need for scalable solutions
  - Single human requires >1,000 CPU hours / genome
Crossbow
http://bowtie-bio.sourceforge.net/crossbow

• Align billions of reads and find SNPs
  – Reuse software components: Hadoop Streaming

• Map: Bowtie (Langmead et al., 2009)
  – Find best alignment for each read
  – Emit (chromosome region, alignment)

• Shuffle: Hadoop
  – Group and sort alignments by region

• Reduce: SOAPsnp (Li et al., 2009)
  – Scan alignments for divergent columns
  – Accounts for sequencing error, known SNPs
## Performance in Amazon EC2

http://bowtie-bio.sourceforge.net/crossbow

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
<th>Cores</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Loading</td>
<td>3.3 B reads</td>
<td>106.5 GB</td>
<td>$10.65</td>
</tr>
<tr>
<td>Data Transfer</td>
<td>1h :15m</td>
<td>40 cores</td>
<td>$3.40</td>
</tr>
<tr>
<td>Setup</td>
<td>0h : 15m</td>
<td>320 cores</td>
<td>$13.94</td>
</tr>
<tr>
<td>Alignment</td>
<td>1h : 30m</td>
<td>320 cores</td>
<td>$41.82</td>
</tr>
<tr>
<td>Variant Calling</td>
<td>1h : 00m</td>
<td>320 cores</td>
<td>$27.88</td>
</tr>
<tr>
<td>End-to-end</td>
<td>4h : 00m</td>
<td></td>
<td>$97.69</td>
</tr>
</tbody>
</table>

Analyze an entire human genome for ~$100 in an afternoon. Accuracy validated at >99%

**Searching for SNPs with Cloud Computing.**

Related Approaches

**CloudBurst**

Highly Sensitive Short Read Mapping with MapReduce

100x speedup on 96 cores @ Amazon

(Schatz, 2009)

**MUMmerGPU**

High Throughput Sequence Alignment using GPGPUs

~10x speedup on nVidia GTX 8800

(Schatz, Trapnell, et al., 2007)

(Trapnell & Schatz, 2008)
Short Read Assembly

- Genome assembly as finding an Eulerian tour of the de Bruijn graph
  - Human genome: >3B nodes, >10B edges

- The new short read assemblers require tremendous computation
  - Velvet (Zerbino & Birney, 2008) serial: > 2TB of RAM
  - ABBySS (Simpson et al., 2009) MPI: 168 cores x ~96 hours
  - SOAPdenovo (Li et al., 2010) pthreads: 40 cores x 40 hours, >140 GB RAM
K-mer Counting

- Application developers focus on 2 (+1 internal) functions
  - Map: input \(\rightarrow\) key:value pairs
  - Shuffle: Group together pairs with same key
  - Reduce: key, value-lists \(\rightarrow\) output

ATGAACCTTA
- (ATG:1) (ACC:1)
- (TGA:1) (CCT:1)
- (GAA:1) (CTT:1)
- (AAC:1) (TTA:1)

GAACAACTTA
- (GAA:1) (AAC:1)
- (AAC:1) (ACT:1)
- (ACA:1) (CTT:1)
- (CAA:1) (TTA:1)

TTTAGGCAAC
- (TTT:1) (GGA:1)
- (TTA:1) (GCA:1)
- (TAG:1) (CAA:1)
- (AGG:1) (AAC:1)

Map, Shuffle & Reduce All Run in Parallel
Graph Construction

- Application developers focus on 2 (+1 internal) functions
  - **Map**: input → key:value pairs
  - **Shuffle**: Group together pairs with same key
  - **Reduce**: key, value-lists → output

### Map, Shuffle & Reduce
All Run in Parallel
Graph Compression

- After construction, many edges are unambiguous
  - Merge together compressible nodes
  - Graph physically distributed over hundreds of computers
Distributed Graph Processing

Input:
- Graph stored as node tuples

Map
- For all nodes, re-emit node tuple
- For all neighbors, emit value tuple

Shuffle
- Collect tuples with same key

Reduce
- Add together values, save updated node tuple

Example:
- **Input:**
  - A: (N E: B W: 42)
  - B: (N E: I, J, K W: 33)

- **Map:**
  - A: (N E: B W: 42)
  - B: (V A 42)
  - B: (N E: I, J, K W: 33)

- **Shuffle:**
  - B: (N E: I, J, K W: 33)
  - B: (V A 42)

- **Reduce:**
  - B: (N E: I, J, K W: 75)
Iterative Path Compression

Iteratively identify and collapse the beginning of each chain

Map:
- Emit messages to the neighbors of the head of each chain

Reduce:
- Update links, node label

Requires $S$ MapReduce cycles, where $S$ is the length of the longest simple path

- **B. anthracis**: $L=5.2\text{Mbp}$, $S=268,925$
- **H. sapiens chr 22**: $L=49.6\text{Mbp}$, $S=33,832$
- **H. sapiens chr 1**: $L=247.2\text{Mbp}$, $S=37,172$
Fast Path Compression

Challenges

- Nodes stored on different computers
- Nodes can only access direct neighbors

Randomized List Ranking

- Randomly assign $\bar{H}/T$ to each compressible node
- Compress $\bar{H} \rightarrow T$ links

Performance

- Compress all chains in log(S) rounds (<20)
- If <1024 nodes to compress (from any number of chains), assign them all to the same reducer (save 10 rounds)

Randomized Speed-ups in Parallel Computation.
Node Types

- **Isolated nodes (10%)**
  - Contamination

- **Tips (46%)**
  - Clip short tips

- **Bubbles/Non-branch (9%)**
  - Pop bubbles

- **Dead Ends (.2%)**
  - Split forks

- **Half Branch (25%)**
  - Unzip

- **Full Branch (10%)**
  - Thread reads, cloud surfing

(Chaisson, 2009)
Contrail
http://contrail-bio.sourceforge.net

Scalable Genome Assembly with MapReduce

- **Genome**: *E. coli* 4.6Mbp bacteria
- **Input**: 20M 36bp reads, 200bp insert
- **Preprocessor**: Quality-Aware Error Correction

### Assembly of Large Genomes with Cloud Computing.
Selected Related Work

**AutoEditor & AutoJoiner**
Improving Genome Assemblies without Resequencing  
(Gajer, Schatz, Salzberg, 2004)  
(Carlton et al., 2007)

**Hawkeye**
Assembly Visualization & Analytics  
(Schatz, Phillippy, Shneiderman, Salzberg, 2007)

**PhyloTrac**
Integrated survey analysis of prokaryotic communities  
(Schatz, Phillippy, et al., 2010)

**Graph Summarization**
Revealing Biological Modules via Graph Summarization.  
(Navlakha, Schatz, Kingsford, 2008)

**Assembly Forensics**
Finding the Elusive Mis-assembly  
(Phillippy, Schatz, Pop, 2008)

**Transgenic Hunt**
Characterization of Insertion Sites in Rainbow Papaya  
(Suzuki et al., 2008)
Research Directions

• Scalable Sequencing
  – Genomes, Metagenomes, *-Seq, Personalized Medicine
  – How do we survive the tsunami of sequence data?
    o Efficient indexing & algorithms, multi-core & multi-disk systems

• Practically Parallel
  – Managing n-tier memory hierarchies, crossing the PRAM chasm
  – How do we solve problems with 1000s of cores?
    o Locality, Fault Tolerance, Programming Languages & Parallel Systems

• Computational Discovery
  – Abundant data and computation are necessary, but not sufficient
  – How do we gain insight?
    o Modeling, Machine Learning, Databases, Visualization & HCI
Summary

“NextGen sequencing has completely outrun the ability of good bioinformatics people to keep up with the data and use it well… We need a MASSIVE effort in the development of tools for ‘normal’ biologists to make better use of massive sequence databases.”

Jonathan Eisen – JGI Users Meeting – 3/28/09

• Computational Biology
  – Make the problems of genotyping and assembly of large genomes from short reads feasible and accessible to individual researchers

• High Performance Computing
  – Developed Novel Parallel Algorithms for MapReduce and Multicore systems
Acknowledgements

Advisor
Steven Salzberg

UMD Faculty
Mihai Pop, Art Delcher, Amitabh Varshney, Carl Kingsford, Ben Shneiderman, James Yorke, Jimmy Lin, Dan Sommer

CBCB Students
Adam Phillippy, Cole Trapnell, Saket Navlakha, Ben Langmead, James White, David Kelley
Thank You!

http://www.cbcb.umd.edu/~mschatz
Genome Coverage

Idealized assembly

• Uniform probability of a read starting at a given position
  – \( p = \frac{G}{N} \)

• Poisson distribution in coverage along genome
  – Contigs end when there is no overlapping read

• Contig length is a function of coverage and read length
  – Short reads require much higher coverage

Large-Scale Genome Assembly from Short Reads.
Two Paradigms for Assembly

a) Read Layout
R₁: GACCTACA
R₂: ACCTACAA
R₃: CCTACAAG
R₄: CTACAAGT
A: TACAAAGTT
B: ACAAGTTA
C: CAAGTTAG
X: TACAAAGTC
Y: ACAAGTTCC
Z: CAAGTTCCG

b) Overlap Graph

c) de Bruijn Graph

Large-Scale Genome Assembly from Short Reads.
Short Reads and Mate-pairs

- Explore the relationship between read length and contig N50 size
  - Perfect reads, lengths: 25, 35, 50, 100, 250, 500, 1000
  - Long reads are limiting case for short mated reads, perfectly compute the insert sequence

*Assembly Complexity of Prokaryotic Genomes using Short Reads.*
ABySS Results

• Assemble 42x 36bp reads

• Mate pairs double the size of the contigs
  – Insert size 210bp

• Identify 100k insertions and deletions
  – Pronounced deletion peak corresponds to Alu family of retrotransposons
Bidirectional de Bruijn Graph

- Designate a representative mer for each mer/rc(mer) pair
  - Use the lexigraphically smaller mer

- Bidirected edges record if connection is between forward or reverse mer

- In practice, keep separate adjacency lists for the forward and reverse mers

AAGG [CCTT]: AAG⁺ -> AGG⁺
ACTT [AAGA]: ACT⁺ -> AAG⁻
GCTT [AAGC]: AGC⁻ -> AAG⁻
AAG⁺ -> AGC⁺

(Medvedev et al, 2007)
Find Compressible Nodes

Input: Graph stored as \((n : (\text{nodeinfo}, ni))\)

Map:
- For all nodes, emit \((n : (\text{nodeinfo}, ni))\)
- If node \(n\) has unique predecessor \(p\), emit \((p : (\text{unique-pred}, n))\)

Reduce:
- If node \(n\) has unique successor \(s\), and received \((\text{unique-pred}, s)\),
  - Mark \(ni\) as compressible
- Save \((n : (\text{nodeinfo}, ni))\)
Error Correction

Sequencing error distorts graph structure

– Errors at end of read
  • Trim off ‘dead-end’ tips
  • B’ passes trim message to A

– Errors in middle of read
  • Pop Bubbles
  • B’ and B pass bubble messages to A
    – A is lexicographically smaller than C

– Recursively apply, rerun path compression between each iteration

Parallel Network Motif Finding
Repeat Analysis

- **X-cut**
  - Annotate edges with spanning reads
  - Separate fully spanned nodes
    - (Pevzner et al., 2001)

- **Scaffolding**
  - If mate pairs are available search for a path consistent with mate distance
  - Use message passing to iteratively collect linked and neighboring nodes

- Other simplifications possible

Parallel Frontier Search
Index reference using a suffix tree
- Each suffix represented by path from root
- Reorder tree along space filling curve

Map many reads simultaneously on GPU
- Find matches by walking the tree
- Find coordinates with depth first search

Performance on nVidia GTX 8800
- Match kernel was ~10x faster than CPU
- Search kernel was ~4x faster than CPU
- End-to-end runtime ~4x faster than CPU

Optimizing data intensive GPGPU computations for DNA sequence alignment.
Amazon Elastic MapReduce
EC2 Pricing

Amazon Elastic MapReduce currently is available in the US region only. Pay only for what you use – there is no minimum fee. Amazon Elastic MapReduce pricing is in addition to normal Amazon EC2 and Amazon S3 pricing.

<table>
<thead>
<tr>
<th>Standard Amazon EC2 Instances</th>
<th>Amazon EC2 Price per hour (On-Demand Instances)</th>
<th>Amazon Elastic MapReduce Price per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small (Default)</td>
<td>$0.10 per hour</td>
<td>$0.015 per hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.40 per hour</td>
<td>$0.06 per hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.80 per hour</td>
<td>$0.12 per hour</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>High CPUInstances</th>
<th>Amazon EC2 Price per hour (On-Demand Instances)</th>
<th>Amazon Elastic MapReduce Price per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>$0.20 per hour</td>
<td>$0.03 per hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.80 per hour</td>
<td>$0.12 per hour</td>
</tr>
</tbody>
</table>

Amazon EC2 and Amazon S3 charges are billed separately. Pricing for Amazon Elastic MapReduce is per instance-hour consumed for each instance type, from the time job flow began processing until it is terminated. Each partial instance-hour consumed will be billed as a full hour. For additional details on Amazon EC2 Instance Types, Amazon EC2 Reserved Instances Pricing, or Amazon S3 Pricing, follow the links below:

Amazon EC2 Instance Types
Amazon EC2 Reserved Instances Pricing
Amazon S3 Pricing
CloudBurst
http://cloudburst-bio.sourceforge.net

- Leverage Hadoop to build a distributed inverted index of k-mers and find end-to-end alignments

- 100x speedup over RMAP with 96 cores at Amazon EC2

CloudBurst: Highly Sensitive Read Mapping with MapReduce.
Schatz MC (2009) Bioinformatics. 25:1363-1369
CloudBurst: Highly Sensitive Read Mapping with MapReduce

(Schatz, 2009)

1. **Map: Catalog K-mers**
   - Emit every k-mer in the genome and non-overlapping k-mers in the reads
   - Non-overlapping k-mers sufficient to guarantee an alignment will be found

2. **Shuffle: Coalesce Seeds**
   - Hadoop internal shuffle groups together k-mers shared by the reads and the reference
   - Conceptually build a hash table of k-mers and their occurrences

3. **Reduce: End-to-end alignment**
   - Locally extend alignment beyond seeds by counting mismatches, or with Landau-Vishkin k-difference algorithm to allow for indels.
   - If read aligns end-to-end, record the alignment

---

**Human chromosome 1**

**Read 1**

**Read 2**

**Read 1, Chromosome 1, 12345-12365**

**Read 2, Chromosome 1, 12350-12370**
CloudBurst Results on Local CBCB Cluster

- Evaluation of CloudBurst running time while scaling the number of reads and the number of allowed mismatches while mapping to human chromosomes 1 (top) and 22 (bottom) on the local cluster with 24 cores.

- Colored lines indicate timings allowing 0 (fastest) through 4 (slowest) mismatches between a read and the reference.

- As the number of reads increases, the running time increases linearly.

- As the number of allowed mismatches increases, the running time increases super-linearly from the exponential increase in seed instances.
Comparison to RMAP

- CloudBurst running time compared to RMAP for mapping 7M reads, showing the speedup of CloudBurst running on 24 cores compared to RMAP running on 1 core.

- As the number of allowed mismatches increases, the relative overhead decreases allowing CloudBurst to meet and exceed 24x linear speedup.

- Produces identical results in a fraction of the time, especially for highly sensitive alignments.
Amazon EC2 Evaluation

- CloudBurst running times for mapping 7M reads to human chromosome 22 with at most 4 mismatches on the local and EC2 clusters.

- The 24-core Amazon High-CPU Medium Instance EC2 cluster is faster than the 24-core Small Instance EC2 cluster, and the 24-core local dedicated cluster.

- As the number of cores increase, the running time decreases with near linear speedup. The 96-core cluster is 3.5x faster than the 24-core, and 100x faster than a serial run of RMAP.
Burrows-Wheeler Transform

- Reversible permutation of the characters in a text

\[ \text{BWT}(T) \text{ is the index for } T \]

A block sorting lossless data compression algorithm.
Bowtie algorithm

Reference

BWT( Reference )

Query:
AATGATACGGCGACCACCGAGATCTA
Bowtie algorithm

Reference

BWT( Reference )

Query:
A AT G A T A C G G C G A C C A C C G A G AT C T A
Bowtie algorithm

Reference

BWT( Reference )

Query:
A A T G A T A C G G C G A C C A C C G A G A T C T A
Bowtie algorithm

Reference

BWT( Reference )

Query:

AATGATA C G G C G A C C C G A G A T C T A
Bowtie algorithm

Reference

BWT( Reference )

Query:

AATGA

TACGGCGACACCACCGAGATCTA
**Bowtie algorithm**

Reference

BWT( Reference )

Query:

A A T G

AT A C G G C G A C C A C C G A G A T C T A
Bowtie algorithm

Query: AATG

Reference

BWT( Reference )

Query: TACGGCGACCACCGAGATCTA
Bowtie algorithm

Reference

BWT( Reference )

Query:

AATGTTCACGGCGACCACCGAGATCTA