Read Alignment Algorithms

Choosing what to use and why

- Tom Bair
- thomas-bair@uiowa.edu

Importance of choosing the best method for alignment

- The correct choice can dramatically change the mapping percentages
  - Ion torrent 68% (bowtie) to 95% (TMAP)
- The correct choice can allow you to actually find what you are looking for
  - Bowtie for chip-seq but not for SNP discovery

- Intelligently make tradeoffs in
  - Speed
  - Memory utilization
  - Accuracy
  - Ease of use

Overview

- Less focus on the computer science aspects
  - Just enough to be dangerous
- Concentrate on the application issues
  - Don't worry about the implementation
    - Except the details that impact use

Mapping vs Assembly

Mapping finds the n best matches of a particular read to a reference

Assembly finds the n best overlaps among the reads and determines the most likely genome based on the given data and inserts gaps where inconclusive data exists

What are we mapping against

- HGP
  - 100's of DNA samples, a small subset picked no one knows which ones were selected
- Celera
  - A few different samples including C . Venter
- Heavy Euro-centric little actual diversity

- What if there is a deletion in the original samples?
  - Your reads will not map
- SNP's count towards mismatches
  - Difficult to align in regions with many potential mismatches

What should be used?

- "Given a world population of over six billion people, we estimated that a complete human pan-genome would include an additional 19–40 Mb of novel sequences over the reference genome." Nature Biotechnology 28, 57–63 (2010)

- Novel rearrangements and locations with many snps compound the difficulty of mapping
  - Most common allele, most ancient, preserve coding?
  - Is it possible that the reference has disease alleles
    - Yes

Could also do assembly

- Difficult with current read lengths
  - Nanopore technologies
    - Any day now :)
- May be feasible soon
  - Cost still an issue
    - Larger computational resources
    - Storage
      - WGS vs exon

Assembly algorithms

- De Bruijn graphs
  - Velvet, ABYSS
- OLC – overlap layout consensus
  - Phrap, newbler
- Typically need high memory machines
  - 1-2 terabyte machines common for large genomes

Mapping algorithms

- Hash Reads
  - Eland, MAQ
- Hash Genomes
  - BFAST, Mosaik
- BWT (Burrows Wheeler Transform)
  - First create an efficient index
  - BWA, Bowtie

Hash

- Data structure to store and lookup segments efficiently

Hash Reads

- Variable memory footprint
- Usually smaller overall
- Have to generate the hash each time

Hash Genome

- Generate once
- Large but consistent memory requirement

- How to find mismatches with hashes
  - ‘spaced seeds’
  - Parts are not required to match
  - 6 hashes for 2 mismatches
  - 28 hashes for 50% of 3 mismatches
- Once general position determined exact alignment done by smith-waterman

BWT

- Use BWT to reorder the genome
  - Identical segments are near each other
- BWT used in several compression methods
- BWT methods can be faster for similar sensitivity
  - Much faster for reduced sensitivity
- BWA 5 edits /100bp read
- Bowtie 1-2 edits

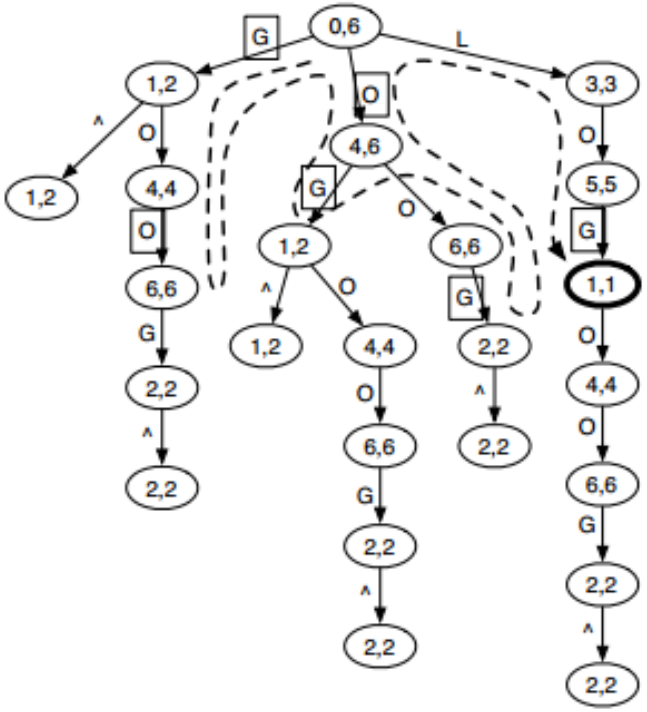


Fig. 1. Prefix trie of string 'GOOGOL'. Symbol ∧ marks the start of the string. The two numbers in a node give the SA interval of the string represented by the node (Section 2.3). The dashed line shows the route of the brute-force search for a query string 'LOL', allowing at most one mismatch. Edge labels in squares mark the mismatches to the query in searching. The only hit is the bold node [1, 1] which represents string 'GOL'.

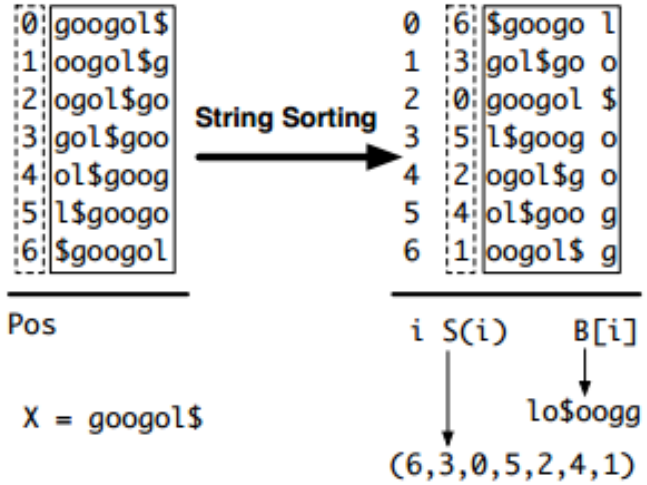


Fig. 2. Constructing suffix array and BWT string for X = googol$. String X is circulated to generate seven strings, which are then lexicographically sorted. After sorting, the positions of the first symbols form the suffix array $(6, 3, 0, 5, 2, 4, 1)$ and the concatenation of the last symbols of the circulated strings gives the BWT string lo$oogg.

implemented in BWT-SW (Lam et al., 2008). We adapted its source code to make it work with BWA.

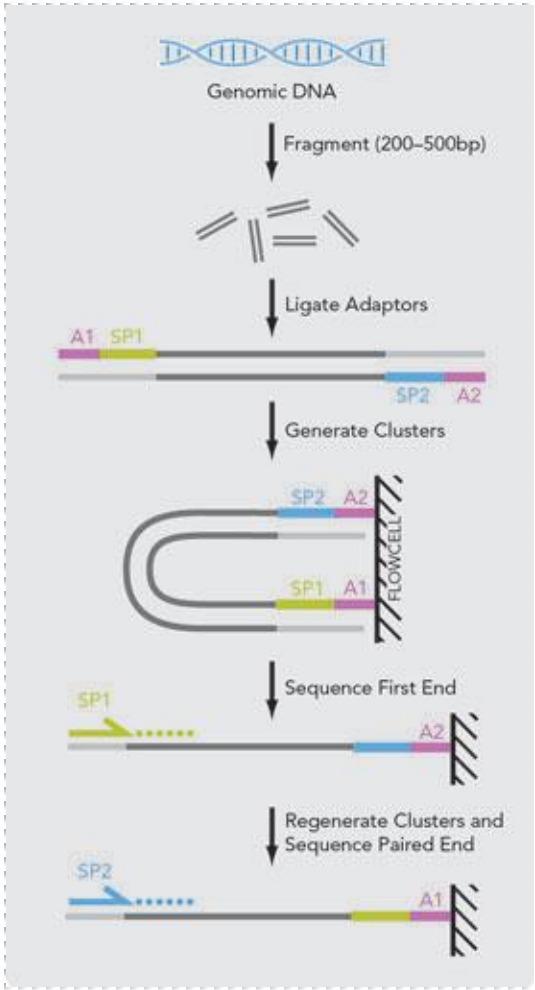
2.3 Suffix array interval and sequence alignment

If string W is a substring of X, the position of each occurrence of W in X will occur in an interval in the suffix array. This is because all the suffixes that have W as prefix are sorted together. Based on this observation, we

Strengths of available mappers

- Bowtie
  - Fast, easy, low memory, intolerant of gaps and many mismatches or errors
- BWA
  - Fast(ish), two step alignment, low memory, more forgiving of errors and gaps
- BLAST
  - Slow, easy, very tolerant of mismatches or gaps

Single-end			Paired-end			
Program	**Time (s)**	**Conf (%)**	**Err (%)**	**Time (s)**	**Conf (%)**	**Err (%)**
Bowtie-32	1271	79.0	0.76	1391	85.7	0.57
BWA-32	823	80.6	0.30	1224	89.6	0.32
MAQ-32	19797	81.0	0.14	21589	87.2	0.07
SOAP2-32	256	78.6	1.16	1909	86.8	0.78
Bowtie-70	1726	86.3	0.20	1580	90.7	0.43
BWA-70	1599	90.7	0.12	1619	96.2	0.11
MAQ-70	17928	91.0	0.13	19046	94.6	0.05
SOAP2-70	317	90.3	0.39	708	94.5	0.34
bowtie-125	1966	88.0	0.07	1701	91.0	0.37
BWA-125	3021	93.0	0.05	3059	97.6	0.04
MAQ-125	17506	92.7	0.08	19388	96.3	0.02
SOAP2-125	555	91.5	0.17	1187	90.8	0.14


Genomic DNA
Fragment (200–500bp)
Ligate Adaptors
A1
SP1
SP2
A2
Generate Clusters
SP2
A2
SP1
A1
FLOWCELL
Sequence First End
SP1
A2
Regenerate Clusters and
Sequence Paired End
SP2
A1

Dealing with paired end

Input insert size or calculate

BWA

- Find all the good matches in a batch (256k)
- calculate the distance between those matches
- For the matches that only have one good hit
  - anchor and then smith-waterman on unanchored

How to evaluate new products

- Understand what type of mapper it is
- Figure out inherent limitations of that approach
  - What twists have been developed to impact limitations
- What novel approaches are developed.
  - Do they matter for you application
    - Color space mapping
    - IUPAC coding
- Persistent optimization can beat clever methods
- Read characteristics influence mapper choice