Burrows-Wheeler Alignment

Rupali Patwardhan

Genome 540

February 15 2011

Outline

- Massively parallel sequencing
- Alignment algorithms
- Burrows-Wheeler Alignment (BWA)
  - BWT
  - Suffix array
  - Backward search
- References

Massively parallel sequencing

- Illumina / SOLiD
- Very high throughput
  - Illumina: 60+ million read-pairs/lane, 16 lanes/run
- Short, error-prone reads
  - around 100bp
  - error-rates 1-2%

Massively parallel sequencing

- Most common applications
  - Medical re-sequencing (cases / unaffected controls)
  - Gene expression (RNA-Seq)
  - TF binding sites, epigenetic marks (ChIP-Seq)
  - Structural variation
  - Chromatin organization, etc., etc.

- Common first step = Alignment of reads to a reference genome

Many aligners out there …

- Bfast
- BioScope
- Bowtie
- BWA
- CLC bio
- CloudBurst
- Eland/Eland2
- GenomeMapper
- GnuMap
- Karma
- MAQ
- MOM
- Mosaik
- MrFAST/MrsFAST
- NovoAlign
- PASS
- PerM
- Phaster
- RazerS
- RMAP
- SSAHA2
- Segemehl
- SeqMap
- SHRiMP
- Slider/SliderII
- SOAP/SOAP2
- Srprism
- Stampy
- vmatch
- ZOOM
- ……

List adapted from a presentation by Heng Li

Need some indexing strategy

- Index the query
  - E.g. MAQ

- Index the reference
  - E.g. SOAP2, BWA, Bowtie

Burrows-Wheeler Alignment (BWA)

- By Heng Li and Richard Durbin

  (Bioinformatics, 2009)

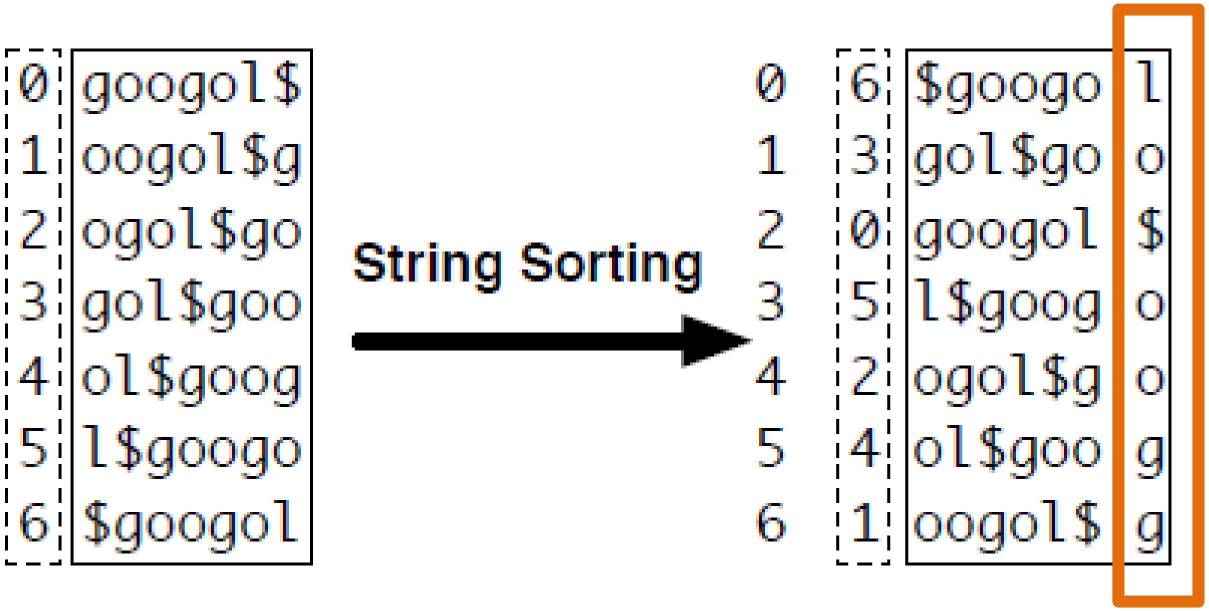
- Concepts
  - Burrows-Wheeler Transform
  - Suffix arrays

Burrows-Wheeler Transform

- *Burrows M and Wheeler D (1994)*
- Reversible permutation of text to allow better compression (e.g. bzip2)
- Algorithms exist to perform fast search on BW-transformed data

Burrows-Wheeler Transform

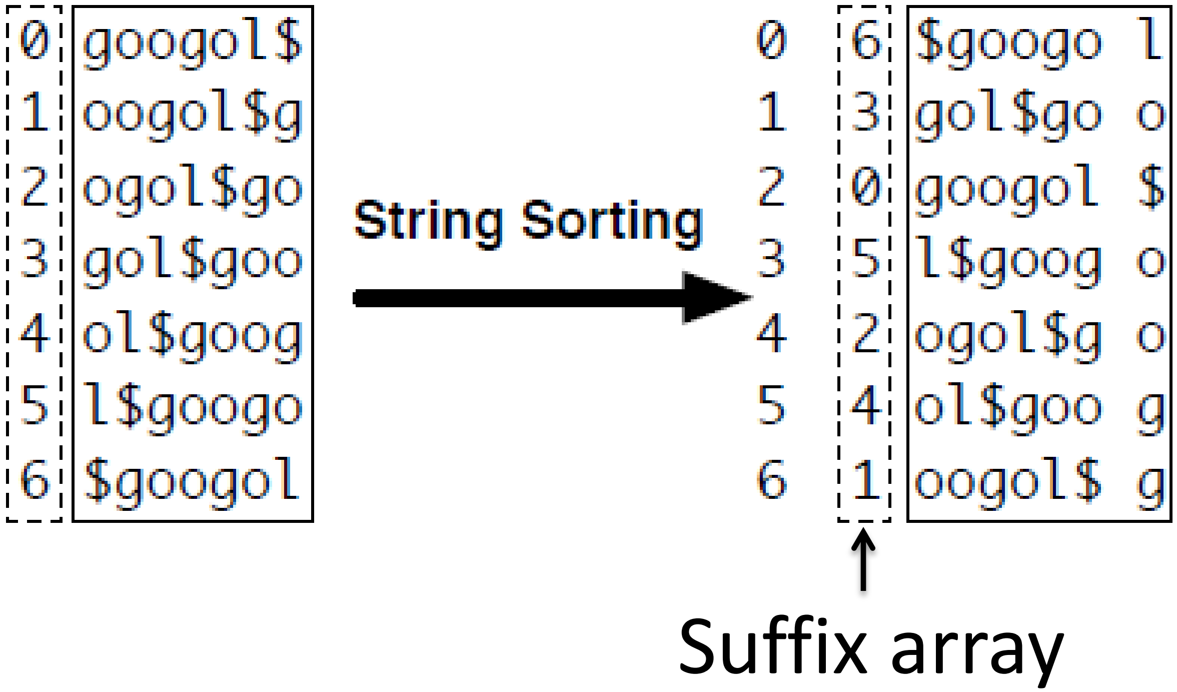
1. Original text = "googol"
2. Append '$' to mark the end = X = "googol$"
3. Sort all rotations of the text in lexicographic order
4. Take the last column.



0	googol$
1	oogol$g
2	ogol$go
3	gol$goo
4	ol$goog
5	l$googo
6	$googol

0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

Connection to suffix arrays

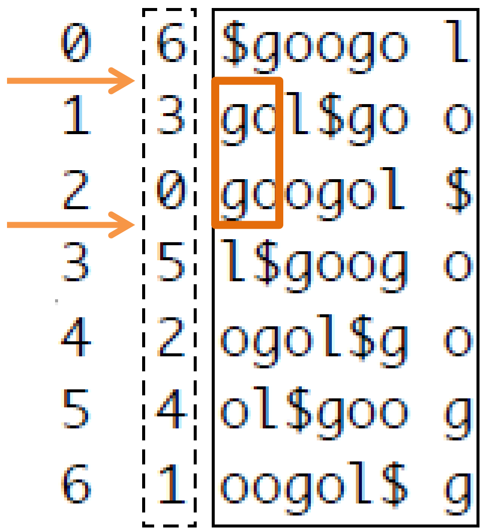


Suffix Array (SA) Interval

- All occurrences of substrings with a common suffix, W appear next to each other, defining an interval $[\ \underline{R}(W), \overline{R}(W)\ ]$

- Suffix array interval of “go” = [1,2]

W = “go”



0	6	$googo	l
1	3	gol$go	o
2	0	googol	$
3	5	l$goog	o
4	2	ogol$g	o
5	4	ol$goo	g
6	1	oogol$	g

X = googol$

0 3

Convince yourself for “go”

- Calculate the SA interval of the query W
- Can be done iteratively from the end of W:

$$\begin{aligned}\underline{R}(aW) &= C(a) + O(a, \underline{R}(W) - 1) + 1 \\ \overline{R}(aW) &= C(a) + O(a, \overline{R}(W))\end{aligned}$$

if W is an empty string, $\underline{R}(W) = 1$ and $\overline{R}(W) = n - 1$

0	6	\$googo	l
1	3	gol\$go	o
2	0	googol	\$
3	5	l\$goog	o
4	2	ogol\$g	o
5	4	ol\$goo	g
6	1	oogol\$	g

C(a) = Number of symbols in X[0, n-2] that are lexicographically smaller than ***a***

O(a, i) = Number of occurrences of ***a*** in B[0, i]

Inexact matching

- Search for SA intervals of substrings of X that match query W with no more than z mismatches
- Start from the end of W and keep moving toward the start
- Prefix with each possible base and look for match
- If match requires a base different from that in the real query, increment mismatch count
- Abort if mismatch count exceeds z

Inexact matching: bounded traversal

- To reduce search space, the backward search is bounded by the D(.) array where D(i) is the lower bound of the number of differences in W[0,i]

```
CALCULATED($W$)
    $z \leftarrow 0$
    $j \leftarrow 0$
    for $i = 0$ to $|W| - 1$ do
        if $W[j, i]$ is not a substring of $X$ then
            $z \leftarrow z + 1$
            $j \leftarrow i + 1$
        $D(i) \leftarrow z$
```

Inexact matching: bounded traversal

```
  InexRecur($W, i, z, k, l$)
      if $z < D(i)$ then
          return $\emptyset$
      if $i < 0$ then
          return $\{[k, l]\}$
      $I \leftarrow \emptyset$
*     $I \leftarrow I \cup$ InexRecur($W, i-1, z-1, k, l$)
      for each $b \in \{A, C, G, T\}$ do
          $k \leftarrow C(b) + O(b, k-1) + 1$
          $l \leftarrow C(b) + O(b, l)$
          if $k \leq l$ then
*             $I \leftarrow I \cup$ InexRecur($W, i, z-1, k, l$)
              if $b = W[i]$ then
                  $I \leftarrow I \cup$ InexRecur($W, i-1, z, k, l$)
              else
                  $I \leftarrow I \cup$ InexRecur($W, i-1, z-1, k, l$)
      return $I$
```

In practice …

- The algorithm for BWT, as described is quadratic in space and time
- In practice, the suffix array is constructed first using the algorithm by Hon et. al., 2007 and then BWT is performed