Mapping Sequencing Reads to a Reference Genome

High Throughput Sequencing

Example applications:
- Sequencing a genome (DNA)
- Sequencing a transcriptome and gene expression studies (RNA)
- ChIP (chromatin immunoprecipitation)

Example platforms:
- 454
- Illumina
- SOLiD

RNA
Sequencing Output

- Hundreds of millions of sequencing reads, each ~100 nts in length
- We need to map each read to the genome, i.e., determine the region of the genome each read corresponds to

Nucleic Acid Sequencing

G-5

Nucleic Acid Sequencing

G-6

Nucleic Acid Sequencing

G-7

Nucleic Acid Sequencing

G-8
Nucleic Acid Sequencing

>CGTAGTACAGCTACGTATATGGGGCTACGATGTTACATTACGTAGCTATCGGACTTAGCCATCG
>CGCCTATAGGCTATATGCATATAGCAGTATAGCTATAGCGGACTTAGCCATCG

Mapping to Reference Genome

Reference Genome
CGTAGTACAGCTACGTATATGGGGCTACGATGTTACATTACGTAGCTATCGGACTTAGCCATCG

Sequencing Read

Burrows-Wheeler Transform

ATCATTAAATCATG

Efficient Substring Search

ATCATTAAATCATG

TCA
Efficient Substring Search

Range of rows starting with $A$ [1,7)
Thus, the substring is in the reference sequence at indices {6,7,8,0,11,3}
Efficient Substring Search

Range of rows starting with
A  \([1,7)\)

Range of rows starting with
CA  \([7,9)\)

Thus, the substring
is in the reference sequence at indices  \(\{10,2\}\)

Thus, the substring
is in the reference sequence at indices  \(\{9,1\}\)
Efficient Substring Search

<table>
<thead>
<tr>
<th>ATCATTAAATCATG$</th>
<th>GTAA$CCTTTTAAAA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCA</td>
<td></td>
</tr>
<tr>
<td>$ATCATTAATCATG$</td>
<td>14</td>
</tr>
<tr>
<td>AAATCATG$ATCATT</td>
<td>6</td>
</tr>
<tr>
<td>ATCATG$ATCATTAA</td>
<td>7</td>
</tr>
<tr>
<td>ATCATG$ATCATTAA</td>
<td>8</td>
</tr>
<tr>
<td>ATCATTAAATCATG$</td>
<td>0</td>
</tr>
<tr>
<td>ATG$ATCATTAATAC</td>
<td>11</td>
</tr>
<tr>
<td>ATG$ATCATTAATAC</td>
<td>3</td>
</tr>
<tr>
<td>AAATCATG$ATCATT</td>
<td>10</td>
</tr>
<tr>
<td>ATATCAATG$ATCAT</td>
<td>5</td>
</tr>
<tr>
<td>ATG$ATCATTAATAC</td>
<td>9</td>
</tr>
<tr>
<td>ATG$ATCATTAATAC</td>
<td>1</td>
</tr>
<tr>
<td>TAAATCATG$ATCAT</td>
<td>12</td>
</tr>
<tr>
<td>TTAAATCATG$ATCA</td>
<td>4</td>
</tr>
</tbody>
</table>

Range of rows starting with A: [1,7)
Range of rows starting with CA: [7,9)
Range of rows starting with TCA: [11,13)

Precompute Helper Information

| GTAA$CCTTTTAAAA |

Returns the number of nucleotide characters in the BWT that are lexicographically less than c.

```
public int getNumberCharactersLessThan(char c);
```

- `getNumberCharactersLessThan('$');` returns 0
- `getNumberCharactersLessThan('C');` returns 7
- `getNumberCharactersLessThan('T');` returns 10

Returns the number of occurrences of nucleotide character c in the BWT up to but not including index i.

```
public int getNumberOccurrencesPriorToIndex(char c, int i);
```

- `getNumberOccurrencesPriorToIndex('A', 5);` returns 2
- `getNumberOccurrencesPriorToIndex('A', 13);` returns 4
- `getNumberOccurrencesPriorToIndex('T', 8);` returns 2

How can we compute this?

Each step of the search must be fast! O(1) time.
Efficient Substring Search

```
ATCATTAAATCATG$ GTAA$CCTTTTTAAA
  TCA
    $ATCATTAAATCATG 14
    AAATCATG$ATCATT 6
    ATATCG$ATCATTA 7
    ATCATG$ATCATTA 8
    ATCATTAATCATG$ 0
    ATG$ATCATTAATC 11
    ATTAATCATG$ATC 2
    CATTAAATCATG$ATC 10
    G$ATCATTAATCATG 13
    TAAATCATG$ATC 5
    TCG$ATCATTAATC 9
    TATTAATCATG$ATC 1
    TG$ATCATTAATC 12
    TTTAATCATG$ATC 4
```

How can we compute these?

Another Example

```
ATCATTAAATCATG$ ATT
  T
    $ATCATTAAATCATG 14
    AAATCATG$ATCATT 6
    ATATCG$ATCATTA 7
    ATCATG$ATCATTA 8
    ATCATTAATCATG$ 0
    ATG$ATCATTAATC 11
    ATTAATCATG$ATC 3
    CATG$ATCATTAATC 10
    TTAAATCATG$ATC 2
    G$ATCATTAATCATG 13
    TAAATCATG$ATC 5
    TCG$ATCATTAATC 9
    TATTAATCATG$ATC 1
    TG$ATCATTAATC 12
    TTTAATCATG$ATC 4
```

A Substring Not in the Reference

```
ATCATTAAATCATG$ GTAA$CCTTTTTAAA
  GTC
    $ATCATTAAATCATG 14
    AAAATCATG$ATCATT 6
    ATATCG$ATCATTA 7
    ATCATG$ATCATTA 8
    ATCATTAATCATG$ 0
    ATG$ATCATTAATC 11
    ATTAATCATG$ATC 2
    CATTAAATCATG$ATC 10
    G$ATCATTAATCATG 13
    TAAATCATG$ATC 5
    TCG$ATCATTAATC 9
    TATTAATCATG$ATC 1
    TG$ATCATTAATC 12
    TTTAATCATG$ATC 4
```

Efficient Substring Search with Errors

Suppose we tried to map a read of length 60 nts to a reference genome and found that the read did not map. Perhaps the read contains one or two errors from the sequencing process.

• Break the read up into three pieces and map each of the three pieces to the reference.
• If none of the three pieces map, then there are at least three errors in the read.
• If one or more of the pieces map, then we use that mapping to perform a fast alignment.