Lectures 4, 5, 6, and 7: Sequence Assembly

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

Sequencing the Human Genome

The sequencing the human genome was a tremendous scientific accomplishment, requiring large-scale collaboration between computational and life sciences.

However, the *intellectual* breakthrough that lead to successful sequencing came from computer scientists, not life scientists.

We will study the basic technology underlying all sequencing projects, and compare the somewhat different experimental strategies employed by the two groups.

Find the Computer Scientists!

The Sequence of the Human Genome

J. Craig Venter,[1*] Mark D. Adams,[1] Eugene W. Myers,[1] Peter W. Li,[1] Richard J. Mural,[1] Granger G. Sutton,[1] Hamilton O. Smith,[1] Mark Yandell,[1] Cheryl A. Evans,[1] Robert A. Holt,[1] Jeannine D. Gocayne,[1] Peter Amanatides,[1] Richard M. Ballew,[1] Daniel H. Huson,[1] Jennifer Russo Wortman,[1] Qing Zhang,[1] Chinnappa D. Kodira,[1] Xiangqun H. Zheng,[1] Lin Chen,[1] Marian Skupski,[1] Gangadharan Subramanian,[1] Paul D. Thomas,[1] Jinghui Zhang,[1] George L. Gabor Miklos,[2] Catherine Nelson,[3] Samuel Broder,[1] Andrew G. Clark,[4] Joe Nadeau,[5] Victor A. McKusick,[6] Norton Zinder,[7] Arnold J. Levine,[7] Richard J. Roberts,[8] Mel Simon,[9] Carolyn Slayman,[10] Michael Hunkapiller,[11] Randall Bolanos,[1] Arthur Delcher,[1] Ian Dew,[1] Daniel Fasulo,[1] Michael Flanigan,[1] Liliana Florea,[1] Aaron Halpern,[1] Sridhar Hannenhalli,[1] Saul Kravitz,[1] Samuel Levy,[1] Clark Mobarry,[1] Knut Reinert,[1] Karin Remington,[1] Jane Abu-Threideh,[1] Ellen Beasley,[1] Kendra Biddick,[1] Vivien Bonazzi,[1] Rhonda Brandon,[1] Michele Cargill,[1] Ishwar Chandramouliswaran,[1] Rosane Charlab,[1] Kabir Chaturvedi,[1] Zuoming Deng,[1] Valentina Di Francesco,[1] Patrick Dunn,[1] Karen Eilbeck,[1] Carlos Evangelista,[1] Andrei E. Gabrielian,[1] Weiniu Gan,[1] Wangmao Ge,[1] Fangcheng Gong,[1] Zhiping Gu,[1] Ping Guan,[1] Thomas J. Heiman,[1] Maureen E. Higgins,[1] Rui-Ru Ji,[1] Zhaoxi Ke,[1] Karen A. Ketchum,[1] Zhongwu Lai,[1] Yiding Lei,[1] Zhenya Li,[1] Jiayin Li,[1] Yong Liang,[1] Xiaoying Lin,[1] Fu Lu,[1] Gennady V. Merkulov,[1] Natalia Milshina,[1] Helen M. Moore,[1] Ashwinikumar K Naik,[1] Vaibhav A. Narayan,[1] Beena Neelam,[1] Deborah Nusskern,[1] Douglas B. Rusch,[1] Steven Salzberg,[12] Wei Shao,[1] Bixiong Shue,[1] Jingtao Sun,[1] Zhen Yuan Wang,[1] Aihui Wang,[1] Xin Wang,[1] Jian Wang,[1] Ming-Hui Wei,[1] Ron Wides,[13] Chunlin Xiao,[1] Chunhua Yan,[1] Alison Yao,[1] Jane Ye,[1] Ming Zhan,[1] Weiqing Zhang,[1] Hongyu Zhang,[1] Qi Zhao,[1] Liansheng Zheng,[1] Fei Zhong,[1] Wenyan Zhong,[1] Shiaoping C. Zhu,[1] Shaying Zhao,[12] Dennis Gilbert,[1] Suzanna Baumhueter,[1] Gene Spier,[1] Christine Carter,[1] Anibal Cravchik,[1] Trevor Woodage,[1] Feroze Ali,[1] Huijin An,[1] Aderonke Awe,[1] Danita Baldwin,[1] Holly Baden,[1] Mary Barnstead,[1] Ian Barrow,[1] Karen Beeson,[1] Dana Busam,[1] Amy Carver,[1] Angela Center,[1] Ming Lai Cheng,[1] Liz Curry,[1] Steve Danaher,[1] Lionel Davenport,[1] Raymond Desilets,[1] Susanne Dietz,[1] Kristina Dodson,[1] Lisa Doup,[1] Steven Ferriera,[1] Neha Garg,[1] Andres Gluecksmann,[1] Brit Hart,[1] Jason Haynes,[1] Charles Haynes,[1] Cheryl Heiner,[1] Suzanne Hladun,[1] Damon Hostin,[1] Jarrett Houck,[1] Timothy Howland,[1] Chinyere Ibegwam,[1] Jeffery Johnson,[1] Francis Kalush,[1] Lesley Kline,[1] Shashi Koduru,[1] Amy Love,[1] Felecia Mann,[1] David May,[1] Steven McCawley,[1] Tina McIntosh,[1] Ivy McMullen,[1] Mee Moy,[1] Linda Moy,[1] Brian Murphy,[1] Keith Nelson,[1] Cynthia Pfannkoch,[1] Eric Pratts,[1] Vinita Puri,[1] Hina Qureshi,[1] Matthew Reardon,[1] Robert Rodriguez,[1] Yu-Hui Rogers,[1] Deanna Romblad,[1] Bob Ruhfel,[1] Richard Scott,[1] Cynthia Sitter,[1] Michelle Smallwood,[1] Erin Stewart,[1] Renee Strong,[1] Ellen Suh,[1] Reginald Thomas,[1] Ni Ni Tint,[1] Sukyee Tse,[1] Claire Vech,[1] Gary Wang,[1] Jeremy Wetter,[1] Sherita Williams,[1] Monica Williams,[1] Sandra Windsor,[1] Emily Winn-Deen,[1] Keriellen Wolfe,[1] Jayshree Zaveri,[1] Karena Zaveri,[1] Josep F. Abril,[14] Roderic Guigó,[14] Michael J. Campbell,[1] Kimmen V. Sjolander,[1] Brian Karlak,[1] Anish Kejariwal,[1] Huaiyu Mi,[1] Betty Lazareva,[1] Thomas Hatton,[1] Apurva Narechania,[1] Karen Diemer,[1] Anushya Muruganujan,[1] Nan Guo,[1] Shinji Sato,[1] Vineet Bafna,[1] Sorin Istrail,[1] Ross Lippert,[1] Russell Schwartz,[1] Brian Walenz,[1] Shibu Yooseph,[1] David Allen,[1] Anand Basu,[1] James Baxendale,[1] Louis Blick,[1] Marcelo Caminha,[1] John Carnes-Stine,[1] Parris Caulk,[1] Yen-Hui Chiang,[1] My Coyne,[1] Carl Dahlke,[1] Anne Deslattes Mays,[1] Maria Dombroski,[1] Michael Donnelly,[1] Dale Ely,[1] Shiva Esparham,[1] Carl Fosler,[1] Harold Gire,[1] Stephen Glanowski,[1] Kenneth Glasser,[1] Anna Glodek,[1] Mark Gorokhov,[1] Ken Graham,[1] Barry Gropman,[1] Michael Harris,[1] Jeremy Heil,[1] Scott Henderson,[1] Jeffrey Hoover,[1] Donald Jennings,[1] Catherine Jordan,[1] James Jordan,[1] John Kasha,[1] Leonid Kagan,[1] Cheryl Kraft,[1] Alexander Levitsky,[1] Mark Lewis,[1] Xiangjun Liu,[1] John Lopez,[1] Daniel Ma,[1] William Majoros,[1] Joe McDaniel,[1] Sean Murphy,[1] Matthew Newman,[1] Trung Nguyen,[1] Ngoc Nguyen,[1] Marc Nodell,[1] Sue Pan,[1] Jim Peck,[1] Marshall Peterson,[1] William Rowe,[1] Robert Sanders,[1] John Scott,[1] Michael Simpson,[1] Thomas Smith,[1] Arlan Sprague,[1] Timothy Stockwell,[1] Russell Turner,[1] Eli Venter,[1] Mei Wang,[1] Meiyuan Wen,[1] David Wu,[1] Mitchell Wu,[1] Ashley Xia,[1] Ali Zandieh,[1] Xiaohong Zhu[1]

Progress in Genome Sequencing

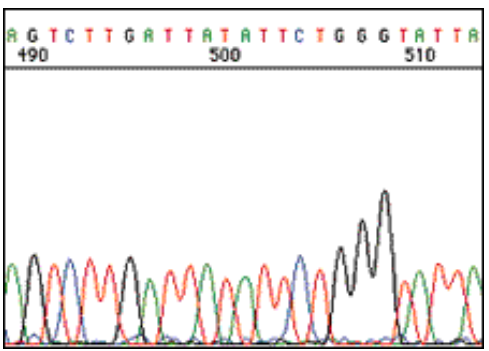
Amazing progress in the scale of sequencing projects has been achieved, largely through automation:

- Phage Phi-x174: 5kb, 1977.
- Bacteriophage lambda: 50kb, 1982.
- Haemophilus influenzae: 1.8Mb, July 1995.
- Drosophila: 180Mb, March 2000.
- Human: 3 billion bases, February 2001.

Interest now revolves around faster/cheaper rather than larger.

DNA Sequencing Machines

Traditional sequencing machines use the same basic principles as the original Gilbert-Sanger method. There has been tremendous progress in automating the procedure, however.

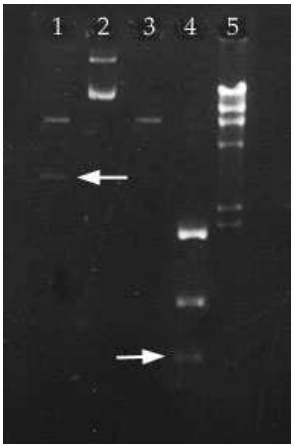


Read lengths have gotten only slightly longer with time, perhaps from 500 bp to 700 bp, and are now getting *shorter*.

Gilbert-Sanger Method

The sample to be sequenced is replicated in four distinct bins, using a distinct fluorescently labeled PCR primer. The bin associated with a given base x is given a mixture of functional and non-functional versions of x, where non-functional bases terminate transcription. This creates labeled fragments of all sizes ending in x.

Using gel electrophoresis, the fragments are separated by length. The presence or absence of a labeled band in each lane denotes whether the sequence has the given base in each position.



Modern capillary machines use smaller amounts of reagents and avoid problems with wandering lanes.

In the good regions of a read, the base error rate should be below 2%.

Short Read Sequencing

New sequencing technologies (Solexa, ABI/Solid, Helicos, 454/Illumina) work by imaging beads/molecules affixed to a surface, watching as they give off light during DNA copying operations.

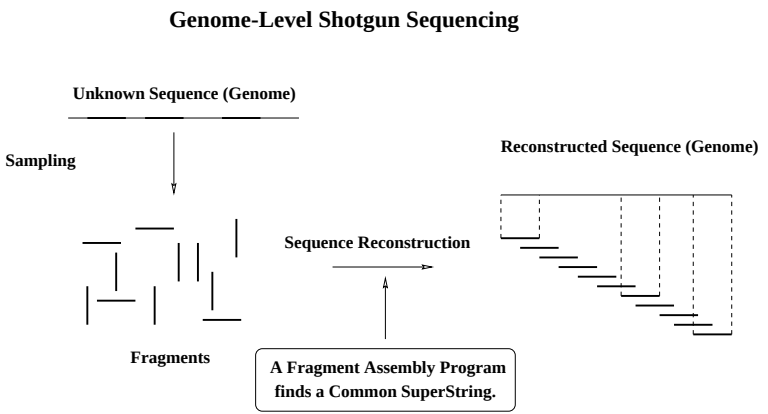
These yield massive numbers of very short reads (25-35 bases) or large numbers of short reads (100-400 bases) at relatively low cost.

These are particularly useful for sequencing variants; goal: the $1,000 genome

De novo sequencing is still best done using longer “short” (454) reads.

Fragment Assembly

In *shotgun sequencing*, whole genomes are sequenced by making clones, breaking them into small pieces, and putting the pieces together again based on overlaps.



Note that the fragments are *randomly* sampled, and thus no positional information is available.

Gaps

Since we rely on fragment overlaps to identify their position, we must sample sufficient fragments to ensure enough overlaps.

Let T be the length of the target molecule being sequenced using n random fragments of length l, where we recognize all overlaps of length t or greater.

The *Lander-Waterman* equation gives the expected number of *gaps* g as

$$g = ne^{-n(l-t)/T}$$

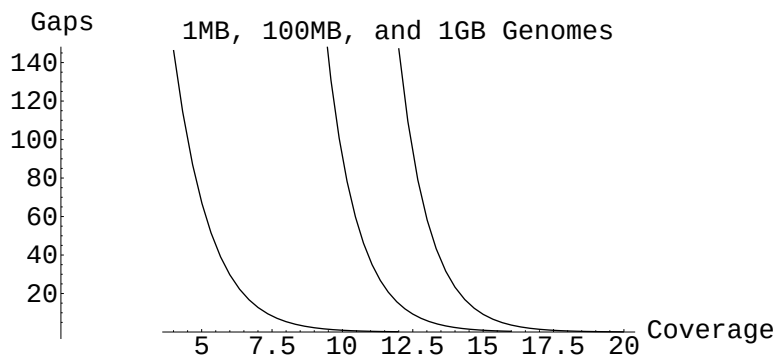
Where does the e come from?

Suppose we have as many fragments as bases, i.e. $T = n$ and each fragment is length 1. The probability p that base i is *not* sampled is

$$p = (\frac{n-1}{n})^n \rightarrow \frac{1}{e}$$

Coverage

The *coverage* of a sequencing project is the ratio of the total sequenced fragment length to the genome length, i.e. nl/T.



Gaps are very difficult and expensive to close in any sequencing strategy, meaning that very high coverage is necessary to use shotgun sequencing on a large genome.

Sequencing Strategies

The effectiveness of a genome sequencing strategy depends upon the degree of *coverage*, the length of the inserts, and the auxiliary *mapping* information available to help assembly.

The DNA fragments or *clones* are replicated by inserting them into a living organism, the *cloning vector*.

Small fragments (40,000 bp) can be cut and pasted into a bacterial *cosmid*. Bigger fragments (up to 2,000,000 bp) can be replicated as a bacterial or yeast artificial chromosome, a *BAC* or *YAC*.

Clone Sizes and Double-Ended Reads

After sequencing both ends of a given insert, we know roughly how far apart they should be in the final assembly.

Selecting the right mix of insert sizes can simplify assembly.

Small inserts give tight assembly constraints, but big inserts help us build a scaffolding across the entire genome.

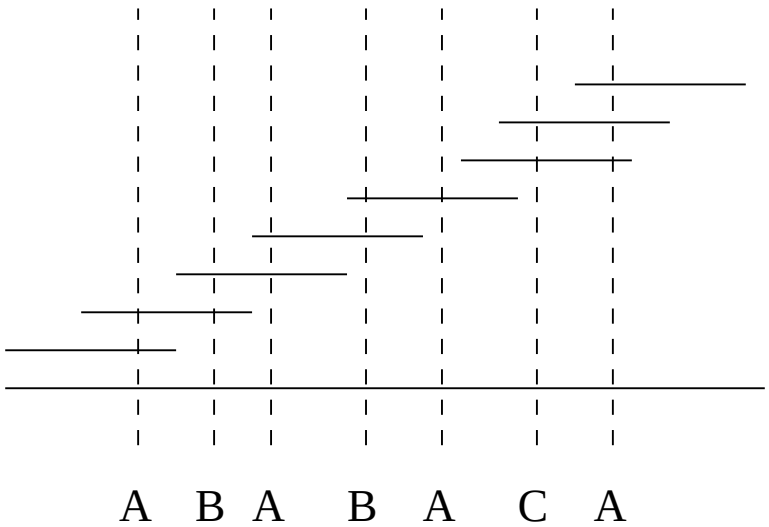
The internals of clones can be sequenced, but it is much more expensive than end sequencing. Thus it is done only in the closing gaps.

Mapping Data

The high coverage necessary to sequence large genomes without gaps frightened most laboratories away from pure shotgun sequencing strategies.

A different approach is to construct a *map* showing where each clone lies on the human genome, and use this map to guide end sequencing and assembly.

Mapping data can be based on (1) using hybridization to detect the presence or absence of a given short sequence (*STS*) in a given clone, or (2) using *restriction enzymes* to cut each clone at a given pattern, and looking for similar fragment lengths.

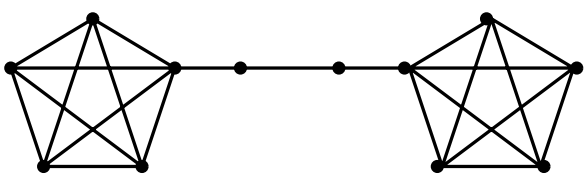


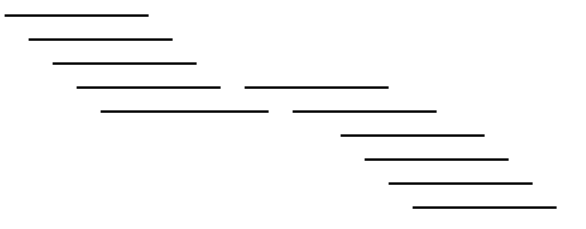
With a good enough map, the required coverage might go down to 2 or 3.

The Algorithmics of Mapping

Note that the correct ordering is a Hamiltonian path on the clones. Reconstructing clone order from mapping data tends to be an NP-complete.

However, the difficulty is due to errors and ambiguity in the mapping data since the problem of recognizing *interval graphs* can be done in linear time!





Celera vs. the Public Consortium

The public consortium used a sequencing strategy based on mapping the clones first.

Celera used hundreds of high-throughput sequencing machines to obtain enough coverage to shotgun sequence the human genome.

Why is Assembly Difficult?

The most natural notion of assembly is to order the fragments so as to form the shortest string containing all of them.

However, the problem of finding the shortest common superstring of a set of strings is NP-complete.

A B R A C
A C A D A
A D A B R
D A B R A
R A C A D

```
A B R A C A D A B R A
A B R A C
    R A C A D
      A C A D A
          A D A B R
            D A B R A
```

Even Worse

Even worse, we have to deal with significant errors in the sequence fragments.

Even worse, genomes tend to have many *repeats* (approximate copies of the same sequence), which are very hard to identify and reconstruct.

Due to repeats, the shortest common superstring is typically *shorter* than the real sequence.

Overlap Detection

Even worse, the size of the problem is very large. Celera's Human Genome sequencing project contained roughly 26.4 million fragments, each about 550 bases long.

To decide what overlaps what, we *could* compare each fragment against each other fragment via $O(n^2)$ dynamic programming, but faster methods are needed.

$$(26.4 \text{ million})^2 \times (550)^2 = 2.1 \times 10^{20} \text{ operations!}$$

Celera's assembly involved 500 million trillion base to base comparisons, requiring over 20,000 CPU hours.

Thus efficient overlap detection is critical, more critical than the NP-complete part of the problem!

Errors and Overlap Detection

Overlap detection must be tolerant of sequencing error, but even an error rate of 2% means one should be able to find fairly long exact matches in a long overlap.

Consider an overlap of length 100. Two errors in each read mean four errors in this region. Even if spaced equally apart, there must be an exact match of at least length 19.

Tries and Trees

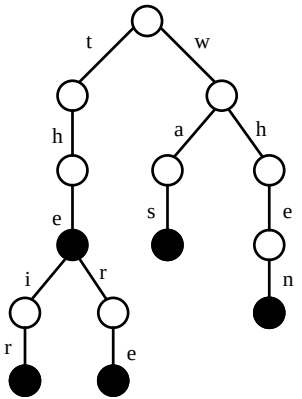
There are several interesting data structures for speeding up exact pattern matches in strings..

A *trie* is a data structure which permits access to any string s in an n word dictionary in $O(|s|)$ time for constant-sized alphabets.

This is optimal and independent of the dictionary size!

Note that binary search of an n word dictionary would take $O(n|s|)$ time.

A trie has a node for each character position, with prefixes shared:



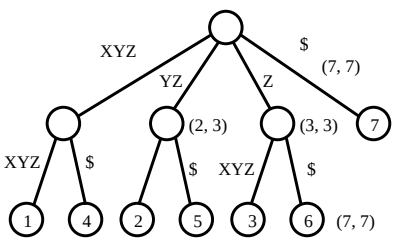
Searching in a trie is easy: just match the character and traverse down the correct path.

Building the trie is also easy: insert a new string by matching until you fail, then split the last node.

Suffix Trees

A very special set of patterns to put in a trie are the complete set of all *suffixes* of a string.

X Y Z X Y Z $
Y Z X Y Z $
Z X Y Z $
X Y Z $
Y Z $
Z $
$



With such a tree, we can perform *substring* searches efficiently, since every substring is the prefix of some suffix. Further, the set of all instances of a given substring t are the leaves of the subtree rooted at t, and can be found by DFS.

Linear-Size Suffix Trees

A suffix tree can be stored in linear space, by collapsing degree-1 nodes into paths, and paths into references to the original string.

The incremental insertion algorithm to build a suffix tree might take $O(n^2)$ time to build the tree, because finding the split-point for each insertion might require $O(n)$ time in matching.

However, there are more sophisticated algorithms (Weiner's, Ukkonen's, McCreight's) which can build the *entire tree* in linear time.

Exploiting Suffix Trees: Longest Common Substring

Given two strings s_1 and s_2, what is the longest *contiguous* substrings they have in common.

Example: *live*stock and sea*live*r

The naive $O(nm)$ algorithm fully tests each alignment of s_1 against s_2.

In 1970, Knuth conjectured that a linear-time algorithm was impossible. Can you prove him wrong?

Longest Common Substring Algorithm

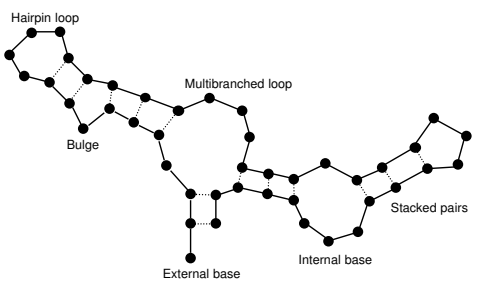
Build a suffix tree of the length $n+m+1$ concatenated string $s' = s_1 \# s_2$, where $\#$ does not occur in either string.

Label each leaf node of the suffix tree with the name of the string it is contained in. Label each internal node with the union of the labels of its descendents.

By doing a DFS on the $O(n+m+1)$ node tree, we can find the deepest node which has both an s_1 and s_2 descendant. This defines the longest common substring!

Exploiting Suffix Trees: Palindromes

A *palindrome* reads the same forwards and backwards: *A man, a plan, a canal – Panama* or *Madam I'm Adam*.
In DNA sequence analysis, a palindrome is a sequence equal to its reverse complement, e.g. *GAATTC*.
Such palindromes bind/fold to create *secondary structures* in sequences, which often have biological significance.



Finding Palindromes

How can we find the longest self-binding substring in a DNA sequence? Use the longest common substring algorithm with s_1 as the input sequence and s_2 as its reverse complement sequence!

This does not guarantee that the lcs of these strings starts/ends in the same place, so does not necessarily find a palindrome.

However, after we augment the suffix tree so as to answer *lowest common ancestor* queries in constant time...

...we can find the longest sub-palindrome in linear time by asking the 'length' of the LCA of $S[i]$ and $S[i+n+1]$ for each $1 \leq i \leq n$.

Exploiting Suffix Trees: Circular String Linearization

Certain genomic structures (*plasmids*) have closed circular DNA molecules rather than linear molecules.

To look such a molecule up in a database, we much find a *canonical* place to break it to leave a linear string.

The most obvious place to break it uniquely is so as to always leave the *lexicographically* smallest string, i.e. the string which appears first in sorted order.

Building and sorting all n such strings takes $O(n^2)$ time. Can you do better?

Linear Time Linearization

Break the string arbitrarily to create a linear string L.
Now build the suffix tree for string $S = LL\#$. This is linear in the size of the input.
Example: $L = gcttcaat$ so $S = gcttcaatgcttcaat\#$.
Do a traversal down from the root, always picking the lexicographically smallest character. Assume that $\#$ is at the end of the alphabet.

Suffix Arrays

The *suffix array* is an amazing data structure for efficiently searching whether S is a substring of string T.
For a given string T, we construct the lexicographically sorted array of all its *suffixes*.
For $T = mississippi$, the suffix array is:

```
11 : i
 8 : ippi
 5 : issippi
 2 : ississippi
 1 : mississippi
10 : pi
 9 : ppi
 7 : sippi
 4 : sissippi
 6 : ssippi
 3 : ssissippi
```

Searching in a Suffix Array

Since every substring is the prefix of some suffix, Substring search now reduces to binary search in this array. Example: is "sip" a substring of T?

Binary search in a suffix array takes $O(m \lg n)$, where n is the length of T and m the length of the matched substring.

With auxilliary data, this can be improved to $O(\lg n + m)$.

Note that we can just as easily find *all* the occurrences of a given string S in T by binary searching just before/after S.

Building and Storing Suffix Arrays

Amazingly we need only store the original string and the sorted start positions to do the search! The jth character of the ith prefix is at $T[start[i] + j - 1]$.

But how fast can be built the suffix array of an n character string?

Radix sorting n strings of n characters can be done in $O(n^2)$, linear in the size of the input.

Building Suffix Arrays Efficiently

But what is really amazing is that suffix arrays can be built in both linear time and space!

First, build a *suffix tree* in $O(n)$ time.

Performing a lexicographic depth first search of a suffix tree yields a suffix array.

Suffix arrays use many times less space than suffix trees (say $3n$ vs. $17n$ bytes), which is often the dominating factor in large text search problems.

Constructing the Overlap Graph

Through clever use of suffix arrays, the entire overlap graph can be built in near-linear time.

After building the array of all suffixes of all fragments, potential overlaps will share a prefix of a suffix, and hence be near each other in sorted order.

Accepting a fragment pair as overlapping may require several significant long matches.

Since there are 4^k possible DNA sequences of length k, and n places for such k-mers to start if $|T| = n$, matches start to get significant if $k > \log_4 n$.

For human, longer than 16-mers start to get interesting, so we can expect to find significant exact matches.

Complications

Several engineering issues arise in building any assembler:

- *Chimera detection* – Certain fragments are, in fact concatenations of sequences from two different regions.
- *Multiple sequence alignment* – We can remove single base sequencing errors by voting in overlapped regions.
- *Identifying repeat regions* – Mapping data and ad hoc algorithms are essential to help resolve repeats.
- *Integrating distance and clone constraints into assembly* – This requires careful laboratory work (preferably automated) to ensure accurate input.

Gel Reading Programs

Calling sequence bases from sequence trace data is not a trivial problem for several reasons, including (1) varying density of the gel along a lane, (2) varying density of the gel across lanes, (3) separation between bands shrinks as we move along the lane.

Phred, by Phil Green, is the most famous base-calling program for automated sequencer traces. It assigns an error probability to each base called and yielded 40-50% lower errors than the software included with Applied Biosystems (ABI) sequencing machines.

Phred's base-calling pipeline consists of several phases:

1. It predicts peak locations using the assumption that fragments should be locally evenly spaced. This helps determine the correct number of bases in a region where peaks are not well resolved, noisy or displaced.
2. Observed peaks are identified in the trace and matched to the predicted peak locations. Some are omitted and some are split, yielding the main base sequence.
3. Unmatched peaks are analyzed to see if they represent bases, and if so is inserted into the sequence.

The quality score Q is based on an estimate of the probability of error p, where $Q = -10 \log_{10} p$.

Genome-Scale Assemblers

Assemblers for bacterial-sized genomes and beyond (such as those used by TIGR and Celera) must use sophisticated data structures to avoid comparing every pair of fragments.

Typically, an initial assembly of ‘unitigs’ of high-quality, long overlaps is constructed.

The expected number of fragments at every position is a function of coverage. Stacks of fragments which are too high likely indicate improperly combined repeat regions.

Sequence Scaffolds

These unitigs are assembled into a ‘scaffold’ of pieces. Most reads are in ‘mate pairs’, since the left and right ends of clones of sizes up to 10 kb or so have both been sequenced. The scaffold is assembled using this pairing and distance information.

Other less high-quality reads and unitigs can now be positioned on this scaffold. Remaining gaps can be closed by sequencing the clones spanning different contigs of the scaffold.

Other Sequencing Tricks

Celera used the public consortium's Human sequence data from Genbank. To avoid being fooled by its assembly, they computationally shredded this sequence into artificial reads and incorporated this as raw data into its own project.

Certain sequencing projects do not attempt to sequence full genomic DNA. To identify the genes, they translate back the *RNA* transcripts of genes into *cDNA* and sequence this.

Although the genes are a very important part of what we are looking for, a drawback such expressed sequence tags (ESTs) is that more highly expressed genes are significantly over-represented, making it hard to find rare genes.

Population Sequencing

Sequencing a mixed population (i.e. fragments from multiple individuals of a given species) should not compromise assembly because of the overwhelming in-species similarity, while differences in bases helps study variation (single nucleotide polymorphisms or SNPs)

Once a quality human reference genome become known, the sequence variants of an individual can largely be identified by aligning short-reads to this target genome and noting single-base mismatches.