The hour of power: a suffix array worship

Side-ship Reverend
Steve Hoffmann
steve@bioinf.uni-leipzig.de

December 21, 2007

A short history of sequence detection

Timeline

1953	Discovery of DNA Structure
1977	Sanger Sequencing (radioactive labels)
1983	Invention of PC Reaction
1987	Dye Terminator Sequencing
1990	Pyrosequencing
1990	Human Genome Project
2003	Human Genome Project announces completion

Sanger Method (revisited)

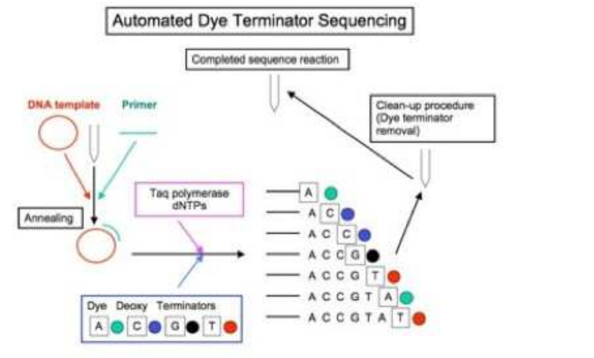


Figure: The introduction of dye-terminators allows for 1-cycle-sequencing. This technique was used to sequence the human genome.

Next generation sequencing

Quickly emerging technologies

system	acquired by	placed	price
Solexa	Illumina	2005/06	$400000
454	Roche	late 2006	$500000
SOLiD	ABI	June 2007	$600000

Solexa (1)

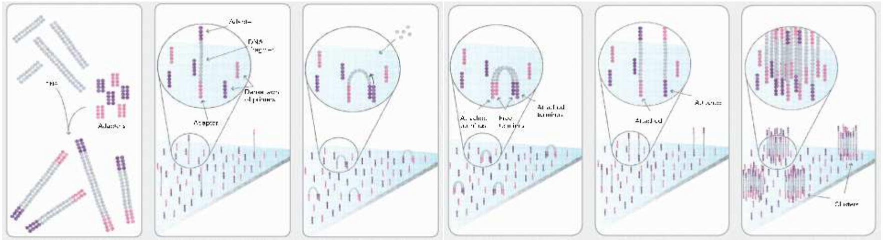


Figure: Fragment DNA is bound to adapters and immobilized. Solid phase bridge amplification results in dense clusters of double stranded DNA.

Solexa (2)

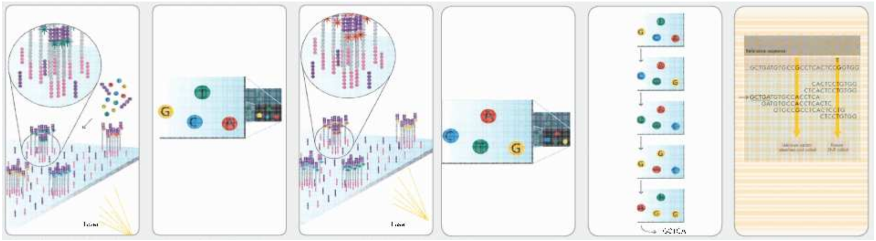


Figure: Sequences detected by laser light over multiple chemistry cycles.

Next generation

Hard facts [... according to friendly sales representatives]

	read size (bp)	Mbp/run	time (h)	accuracy (%)
Solexa	25	1000	60	99.90
454	250	100	7.5	99.50
SOLiD	35	1000	60	99.94

Data! Solexa produces $4'000'000$ reads per run. Today!

Basic concepts of sequence analysis

If you believe the representative ....

- Boyer-Moore
- Knuth-Morris-Pratt
- Bitap

In fact, data is less reliable! For real world 454 data exact pattern matches fails for more than 30% of the reads using a sliding window - exact pattern approach!

Basic concepts of sequence analysis (2)

"Seeding" methods

- BLAST2.0: HSP contains 2 matches of length $|W|$ exceeding threshold T. Largely heuristics supported by SW. Short seqs?

  Reduction of T *from 13 to 11* **triples** *runtime*

- FASTA: matching, neighboring k-tupel are joined using a narrow-band SW.
- BLAT: k-mer indices too inaccurate to map short sequences.

Goals

to find a method, that ...

- quickly maps large sets of differently sized sequences to reference genomes.
- takes full advantage of the accuracy
- allows mismatches
- avoids expensive alignment techniques

One possibly could build an index structure of the reference genome to speed up exact **and approximate** pattern matches. The construction of the index structure is time and space critical.

Index structure?

Suffix array.

What is a suffix, acutally?

Definition

A suffix is a substring that **starts at any** position and **ends with the last** position of a sequence.

`THISISASUFFIX$`

Obviously, a sequence of length m contains m suffixes.

Example

- `THISISASUFFIX$` is a suffix of `THISISASUFFIX$`
- `SUFFIX$` is a suffix of `THISISASUFFIX$`

A suffix array looks like a Phonebook!

```
Sequence

         0  1  2  3  4  5  6  7  8  9  10 11 12
.  .  .  T  A  C  T  G  A  T  G  G  C  T  G  A  .  .  .
```

```
Suffix array

    0     A  C  T  G  A  T  G  G  C  T  G  A
    1     A  T  G  G  C  T  G  A
    2     A
    3     C  T  G  A  T  G  G  C  T  G  A
    4     C  T  G  A
    5     G  A  T  G  G  C  T  G  A
    6     G  A
    7     G  C  T  G  A
    8     G  G  C  T  G  A
    9     T  A  C  T  G  A  T  G  G  C  T  G  A
    10    T  G  A  T  G  G  C  T  G  A
    11    T  G  A
    12    T  G  G  C  T  G  A
```

Figure: The suffix array is a *very simple* data structure with a lexicographical order. Lookups with binary search!

suffix arrays

Let $S := s_0\ s_1 \dots s_{n-1}$ be a sequence of length n over alphabet $\mathcal{A}$. Let $\prec_{\mathcal{A}}$ denote the lexicographical order induced by the alphabet order of $\mathcal{A}$. $S_i := s_i s_{i+1} \dots s_{n-1}$ denotes the i^{th} suffix of S

Definition (Suffixarray)

The suffixarray R is a table of length n such that

$$S_{R[i]} \preceq_{\mathcal{A}} S_{R[j]} \quad 0 \leq i \leq j \leq n-1 \tag{1}$$

With lexicographical order follows directly:

Corrollary

$\forall i,j \quad S_{R[i]} =_{\mathcal{A}} S_{R[j]} \implies i = j$

Genesis

The human genome has 2.3 billion basepairs (suffixes).

Theorem (Hoare)

A quicksort that partitions around a single randomly selected pivotal element sorts n **distinct** *items in*

$$2nH_n + O(n) \approx 1.386\ n \log n \tag{2}$$

expected comparisons.

Solution 1: Multikey quicksort, take some days off and enjoy life.
Solution 2: Linear method?

A trivial observation?

1	2	3	4	5	6	7	8	9	10	11	12
	R			R			R				R
M	I	S	S	I	S	S	I	P	P	I	$

Suffixes marked R are smaller compared to their successors.

building suffix arrays in linear time (1)

Lemma

Let S_i be of type R if $S_i \prec S_{i+1}$ and of type L otherwise. All suffixes can than be classified as R and L in $O(n)$.

Proof.

By case distinction.

1. Iff $s_i \neq s_{i+1}$: compare s_i and $s_i i + 1$
2. Iff $s_i = s_{i+1}$: find the smallest $j > i$ such that $s_j \neq s_i$. Iff $s_j > s_i$, then suffixes $S_i, S_{i+1}, \cdots, S_{j-1}$ are of type R and vice versa.

□

building suffix arrays in linear time (2)

Lemma

A type R suffix is lexicographically greater than a type L suffix that begins with the same first character.

Proof.

By contradiction. Let $S_i \prec S_j$ be of type R and L respectively, begining with the same character c. Write $S_i := c\alpha c_1\beta$ and $S_j := c\alpha c_2\gamma$ with $c_1 \neq c_2$.

1. α contains a first character $c_3 \neq c \overset{R,L}{\Longrightarrow} c_3 > c$ and $c_3 < c$.
2. otherwise $\Longrightarrow c_1 \geq c \wedge c_2 \leq c \Longrightarrow c_1 \geq c_2$.

□

building suffix arrays in linear time (3)

A quite magical algorithm

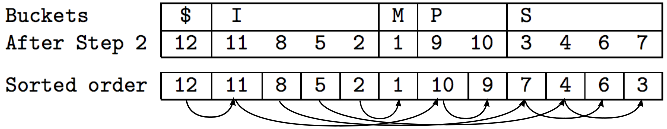
Let B be the sorted array of all suffixes of type R

1. Bucket all suffixes according to their first character in a bucket array A ($O(n)$).
2. Scan B. Move each suffix encountered to the current end of its bucket and advance the end ($O(|B|) \leq O(n)$).
3. Scan A. For each $A[i]$, iff $S_{A[i]-1}$ is type L move to current front and advance the front ($O(n)$).

building suffix arrays in linear time (4)

1	2	3	4	5	6	7	8	9	10	11	12
	R			R			R				R
M	I	S	S	I	S	S	I	P	P	I	$

Order of Type R	12	8	5	2



	$	I				M	P		S			
Buckets After Step 2	12	11	8	5	2	1	9	10	3	4	6	7
Sorted order	12	11	8	5	2	1	10	9	7	4	6	3

Figure: Sorting a suffix array in linear time. A sorted list of type *R* suffixes suffices to sort **all** suffixes of *S*

suffix arrays allow for binary searches!

Once the suffix array is generated, we already have a data structure that allows pattern matches in $O(n \log n)$.

But we can do better!

First take a look at the **longest common prefixes (lcps)** of the suffix array.

the lcp table is easy to compute - but not trivial!

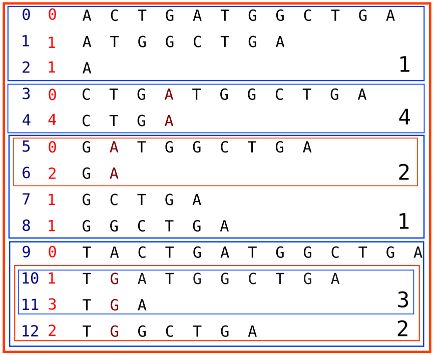
Enhanced suffix array

0	0	A C T G A T G G C T G A
1	1	A T G G C T G A
2	1	A
3	0	C T G A T G G C T G A
4	4	C T G A
5	0	G A T G G C T G A
6	2	G A
7	1	G C T G A
8	1	G G C T G A
9	0	T A C T G A T G G C T G A
10	1	T G A T G G C T G A
11	3	T G A
12	2	T G G C T G A

Figure: the lcp table stores the *lcp-length* of S_i and S_{i+1} at $lcp[i+1]$

A first enhancement: lcp intervals

Enhanced suffix array



0	0	A C T G A T G G C T G A	
1	1	A T G G C T G A	
2	1	A	1
3	0	C T G A T G G C T G A	
4	4	C T G A	4
5	0	G A T G G C T G A	
6	2	G A	2
7	1	G C T G A	
8	1	G G C T G A	1
9	0	T A C T G A T G G C T G A	
10	1	T G A T G G C T G A	
11	3	T G A	3
12	2	T G G C T G A	2

Figure: Enhanced suffix arrays: lcp intervals will speed up your search!

child intervals implicitly contain **a suffix tree**

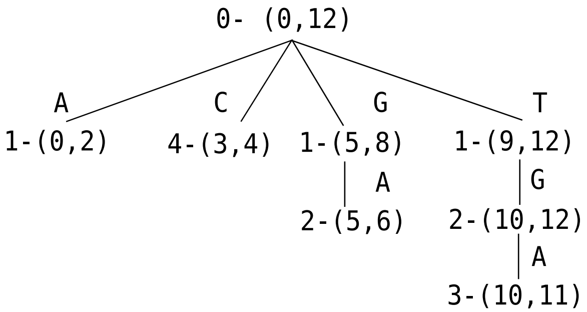


Figure: Child intervals create implicitly generate a tree-like structure: the lcp-tree is a representative of a *suffix tree* [Stefan Kurtz, 2004]

Determination of lcp-intervals

Definition (lcp-interval)

An lcp-interval $[i,j]$, $0 \leq i < j \leq n$, has the lcp-value ℓ if

$$\begin{aligned} &lcp[i] < \ell, \\ &lcp[k] \geq \ell \; \forall k \; i+1 \leq k \leq j, \\ &lcp[k] = \ell, \; \exists \; k \\ &lcp[j+1] < \ell \end{aligned}$$

For the computer science guys: Obviously, the computation of lcp-intervals can be performed in $O(n)$ in a top-down scan of the lcp-table using simple stack operations. This is important for the generation of **child interval tables**.

From lcp intervals to child intervals

Definition (child interval)

Consider the lcp-interval $[i, j]$ in R with lcp $l > 0$. If $[i, j]$ encloses $[l, r]$ with an lcp $m > l$ and no other interval within $[i, j]$ encloses $[l, r]$, $[l, r]$ is called a **child interval** of $[i, j]$.

The child interval major concept: allows linear time transformation of suffix arrays into suffix trees. Child intervals can be determined in linear time using simple stack operations.
O(m) pattern matching

A sophisticated enhancement: suffix link intervals

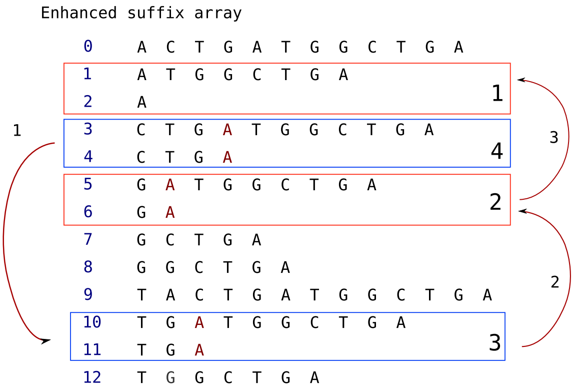


Figure: Enhanced suffix arrays: suffix link intervals will speed up your search even more!

enhancing suffix arrays with suffix links

Definition (suffix link)

Let $S_{R[i]} = aw$ be a suffix of S. The suffix link interval of $S_{R[i]}$ points to the position of the *subsequent* suffix $S_{R[i]+1} = w$ in R .

Definition (suffix link interval)

Let $[i, j]$ be an suffix array interval with a lcp $l > 0$. Each $k \in [i, j]$ corresponds to a suffix $S_{R[k]}$. The suffix link interval $[i', j']$ has an lcp-value of $l - 1$. It can be obtained using suffix links for all $R[k]$.

Suffix links can be obtained in O(1) and suffix intervals can be obtained in O(n).

Quick summary!

suffix arrays ...

1. are time and space critical
2. can be build with less than 8bit per index in linear time
3. can be enhanced with suffix link intervals in linear time
4. current implementation uses more than 12 bit per index
5. allow rapid exact pattern matches in $O(m)$ (!!)
6. can also be used for approximate pattern matching ...

O(m+n) matching statistics (MS) using suffix arrays

j	0	1	2	3	4	5	6	7	8	9
(l_j, p_j)	$(2,0)$	$(1,1)$	$(4,1)$	$(6,0)$	$(5,1)$	$(4,2)$	$(3,3)$	$(2,4)$	$(2,2)$	$(1,3)$

Figure: Matching statistics for a pattern $P = caacacacca$ matched agains suffix array $S = cacaccc$.

Algorithm

1. for pattern P_j greedily match each character until no child interval found or P completely matched
2. if $j > 0 \wedge l_j > 0$ call suffix link interval and match for pattern P_{j+1} else call [0,n].

relaxed alignments from matching statistics (1)

preliminary considerations

- each position of the MS holds list of genomic coordinates
- for "almost" accurate sequences few gaps expected
- generally better quality of proximal subsequences
- matching statistics - diagonal stretches of a DP matrix
- relaxation of MS might be necessary (e.g. homopolymers)
- idea: shortest path from left-most to right-most position.

A sophisticated enhancement: suffix link intervals

Matching statistics

	A	C	C	T	G	A	A	A	A	T	G	G	C
Length	2	1	4	3	2	1	1	1	5	4	3	2	1
Matchlist	1	2	2	3	4	1	1	1	5	6	7	8	2
		9	9	10	11	5	5						9
				12	12	12							
		d=1				d=3							

Relaxed alignment

```
    A C C T G A A A A T G G C
  T A - C T G A - - - T G G C T G A
```

Figure: A path through the matching statistics: an approximate alignment

Evading mismatches

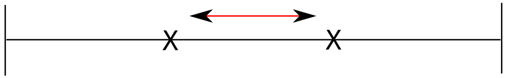


Figure: Information content increases as two mismatches (X) evade.

Assume two neighboring mismatches x_1, x_2 with $x_2 > x_1$. Obviously the information content (or relative entropy) between them increases as the evade with

$$\lim_{d(x_1,x_2)\to\max} \sum_{k=x_1+1}^{x_2-1} p_k \log(p_k \cdot |\mathcal{A}|) \longrightarrow \max \tag{3}$$

Emerging mismatches

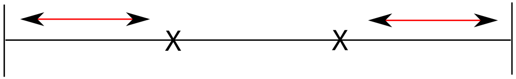


Figure: Combined information content increases as two mismatches (X) emerge.

Assume two neighboring mismatches x_1, x_2 with $x_2 > x_1$. Obviously the information content (or relative entropy) between them *decreases* as they emerge *but*

$$\lim_{d(x_1,x_2)\to\min} \sum_{k=0}^{x_1-1} p_k \log(p_k \cdot |\mathcal{A}|) + \sum_{k=x_2}^{m-1} p_k \log(p_k \cdot |\mathcal{A}|) \longrightarrow \max \quad (4)$$

Emerging mismatches

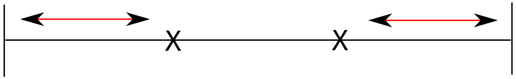


Figure: Combined information content increases as two mismatches (X) emerge.

Assume two neighboring mismatches x_1, x_2 with $x_2 > x_1$. Obviously the information content (or relative entropy) between them *decreases* as they emerge *but*

$$\lim_{d(x_1,x_2)\to\min} \sum_{k=0}^{x_1-1} p_k \log(p_k \cdot |\mathcal{A}|) + \sum_{k=x_2}^{m-1} p_k \log(p_k \cdot |\mathcal{A}|) \longrightarrow \max \quad (5)$$

Combining single match fragments

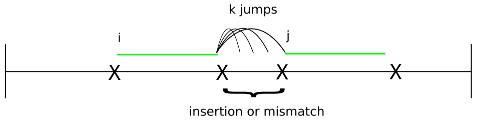


Figure: After obtaining the raw matching statistics, we try to assemble single match fragments

Using the matching statistics we can assemble the suffixes S_i, S_j of single fragment matches i, j with $S_j - S_j = k$ in O(k) for each S_i that matches to i . Obviously, k could be upper bounded by

$$k < \frac{\sigma(P[j..j + l_i - 1]) \cdot \delta_{\text{gapopen}}}{\delta_{\text{indel}}} \tag{6}$$

Subcritical matches result in misplacements

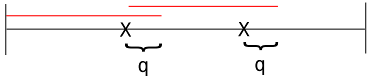


Figure: Substrings between mismatches are not long enough to have a critical information content. The matches in the matching statistics include the mismatches - misplacement

Due to the anatomy of the matching statstics (and the task: map to a reference genome) mismatches are more likely to occur at the end of a single match. Hence, for a match at position i and length l_i in the matching statistics, we have to devise a q-interval $[i + l_i - q, i + l_i]$, $0 \leq q \leq l_i$ and enumerate all possible sequences with k mismatches.

Summary

"Next generation" sequencing

1. 454, Solexa and SOLiD technologies emerging very fast
2. compared to Sanger: Gbp versus Mbp.
3. Sanger technology requires large infrastructure - expensive.
4. Read lengths might be too short and inaccurate to allow unique matching and assembly.
5. Solexa and 454 have announced to improve read lengths and quality by end of this year.

Summary

Currently ...

1. implementation uses relaxed alignment
2. suffix array construction: 15min - 8s
3. mapping speed: 20.000 454-reads (100 bp) in 16s
4. 500'000 solexa reads to Hs-Chr1 in 200s
5. disk space: Chr. 1 *H. sapiens* 5GB

Thanks!

We use suffix arrays to fix your bike!
Andrew Torda

454

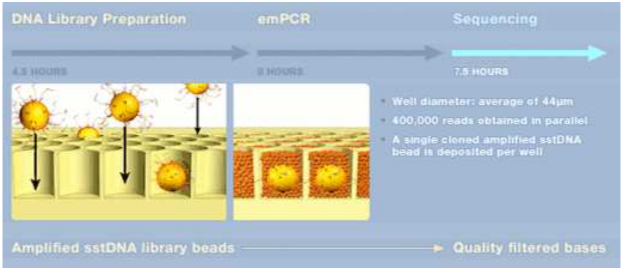

DNA Library Preparation
emPCR
Sequencing
4.5 HOURS
8 HOURS
7.5 HOURS
Well diameter: average of 44µm
400,000 reads obtained in parallel
A single cloned amplified sstDNA bead is deposited per well
Amplified sstDNA library beads
Quality filtered bases

Figure: sstDNA (ligated to adapters) captured on beads and clonally expanded in water-in-oil microreactors (emPCR) are caught in wells with a diameter of 44μm.

454

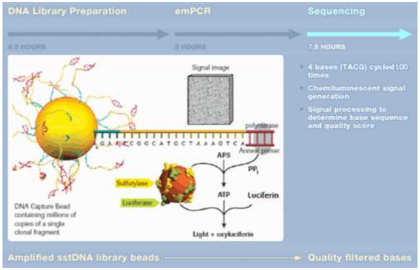

DNA Library Preparation
emPCR
Sequencing
4.5 HOURS
8 HOURS
7.5 HOURS
Signal image
4 bases (TACG) cycled 100 times
Chemiluminescent signal generation
Signal processing to determine base sequence and quality score
polymerase
GAAGCGGCATGCTAAAGTCA
Anneal primer
APS
PP$_i$
Sulfurylase
Luciferase
ATP
Luciferin
Light + oxyluciferin
DNA Capture Bead containing millions of copies of a single clonal fragment
Amplified sstDNA library beads
Quality filtered bases

Figure: Parallel sequencing of up to 400000 reads is performed on a single picotiter plate. Nucleotides are flowed sequentially across the plate. Pyrosequencing: light reaction is induced by Sulfurylases and Luciferases.

SOLiD

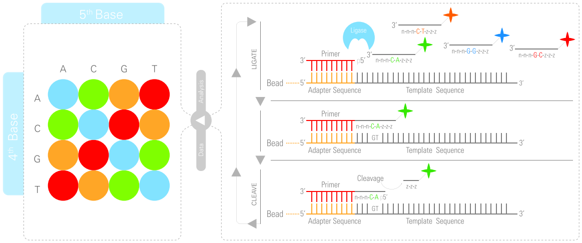


Figure: Sequencing by ligation. sstDNA is immobilized onto beads and enriched in a water-in-oil PCR. In each cycle a set of 4 dye-labeled probes is given to immobilized beads. Specificity is achieved by interogating base 4 and 5. Cleavage of the 3'-end allows for the subsequent ligation reaction. In subsequent cycles complementary primers with an offset of one allows to call bases 3,4 and so on.