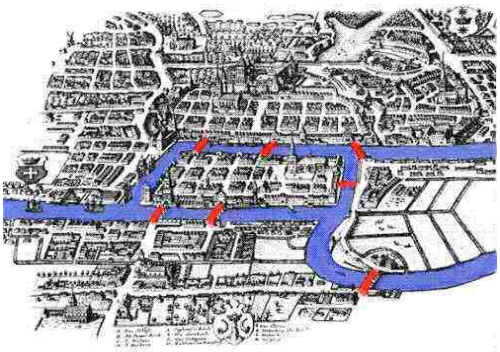
Sequencing and assembly

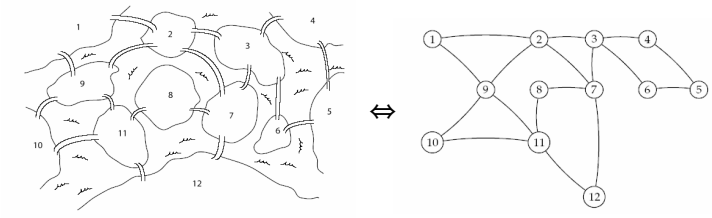
A little graph theory

The Seven Bridges of Königsberg



The Seven Bridges of Königsberg

- Find a tour through Königsberg that crosses every bridge exactly once
- Euler: route inside land doesn't matter, just the sequence of crossings
- Abstraction: Graph
  - Land masses are vertices (nodes)
  - Bridges are Edges



$G = \{ V, E \}$
$V = \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
$E = \{ (1,2), (1,9), (2,9), (2,3), (2,7), (3,4), (3,6), (3,7), (4,5), (5,6), (7,8), (7,12)\ (8,11), (9,10), (9,11), (11,12)\}$

Graphs

- Many (overlapping) classes including:
  - Directed (each edge has a direction) or undirected
  - Weighted (each edge has a numeric weight) or unweighted
  - Connected (a path exists between any pair of vertices)
- Can reduce some problems to finding a particular path or cycle in a particular graph

Some path problems

- In a (possibly weighted) graph, find:
  - Shortest path between two vertices
  - Longest path between two vertices
    - (global sequence alignment)

Eulerian Path

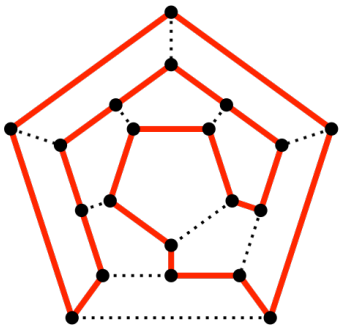
- Königsberg bridge problem: find a path that visits each edge exactly once
- Not possible for the real Königsberg, Euler showed that for such a path to exist the graph must have exactly zero or two nodes of odd degree
- Such a path is now called an *Eulerian Path*, and an algorithm exists to find it it $O(|E|)$ time

Hamiltonian path

- A path that visits every *vertex* exactly once
- Hamiltonian cycle: returns to the starting vertex
- Both decision problems are NP-complete*

*a solution can be verified in polynomial time,
but there is no know fast algorithm to find a solution

Hamiltonian cycle



Traveling salesman problem

- Given a list of cities with known pairwise distances between them, find the shortest tour that visits every city exactly once
- Equivalent to finding the shortest Hamiltonian cycle in a complete weighted graph
  - thus decision problem is NP-hard
    (and in fact, NP-complete)

Sequencing DNA

Sequencing

- Goal: determine sequence of nucleotides in a DNA molecule
- Limitations of current methods:
  - Require many copies of the fragment to be sequenced
  - Can only sequence a limited number of bases for a given DNA molecule
- Consensus sequence of these identical short fragments: sequencing "reads"

Sequencing longer molecules

- Shotgun sequencing
  - Break DNA into random fragments (in a way that yields overlapping fragments)
  - Sequence from one or both ends of the short fragments
- Assembly
  - Resolve original sequence from fragments

Assembly as a string problem

- All reads came from the same string, thus we seek some *superstring* of the reads (a string which contains every read as a substring)
  - There are (infinitely) many possible superstrings
  - Which one do we want?

Assembly as a string problem

- All reads came from the same string, thus we seek some *superstring* of the reads (a string which contains every read as a substring)
  - There are (infinitely) many possible superstrings
  - Which one do we want? Makes sense to seek the shortest superstring of the data

Shortest superstring problem

- Input: a set of strings s_1, ... s_n.
- Output: a string s that contains all of s_1, ... s_n as substrings, and which has the smallest possible length of all such superstrings

Graph representation solution

- Vertices: the n strings
- Edges: the edge between two nodes is - $overlap(s_i, s_j)$
  - $overlap(s_i, s_j)$ is the length of the longest prefix of s_j which is a suffix of s_i.
  - Thus, pairs with large overlap have small weights
- Find the shortest path that visits every vertex exactly once

Complexity

- Shortest Hamiltonian tour in a weighted graph is the Traveling Salesman Problem, which is NP-complete
- And... we can show that *any* solution to the SCS problem requires solving the Hamiltonian path problem, and thus is NP-complete

Shortest Common Superstring

- Problems
  - No efficient solution
  - Doesn't allow for errors in sequencing reads
  - Repeats: shortest reconstruction may not be correct reconstruction

Fragment assembly strategy

- Overlap-layout-consensus
  - Overlap: find potentially overlapping reads
  - Layout: order the reads
  - Consensus: merge reads into a single sequence, correcting errors (hopefully)

Overlap

- Find best match between a suffix of a read and a prefix of another
  - But not an exact match, sequencing errors occur at 1% to 5% of positions depending on technology
  - How can we find high scoring non-exact matches?

Overlap: alignment

- Optimal overlap alignment
  - However reads often have lower quality at ends
- Filtration approach
  - Find pairs of reads that share a common k-mer
  - Extend using local or global alignment
  - Ignore if similarity is below some threshold

Overlap: alignment

- Dealing with quality
  - Let local alignment handle it, or incorporate quality into alignment (PHRAP)
  - Trim low quality regions
- Some k-mers occur extremely frequently (repeats)
  - Discard k-mers that occur more frequently than some amount (derived from the expected sequence coverage)

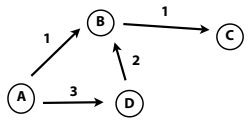
Layout

- Overlap graph: nodes are reads, edges are similarity scores
- Layout: find a path through the graph that explains every read, while maximizing quality of overlap (alignment score)
  - Still the Hamiltonian path problem

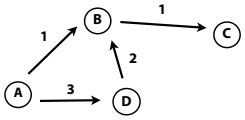
Layout: A greedy algorithm

- Iteratively add heaviest edges to path, as long as they are consistent
- In particular (simple):
  - Sort edges by weight
  - For each sorted edge, add it only if it would not result in the path branching

Layout example

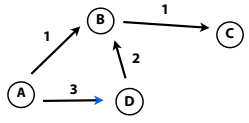


Layout example



Sorted edges
(A,D)
(D,B)
(A,B)
(B,C)
(C,D)

Layout example

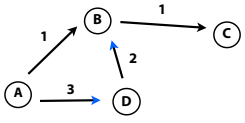


Sorted edges
(A,D)
(D,B)
(A,B)
(B,C)
(C,D)

Add (A,D)

Path: A→D

Layout example

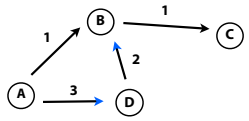


Sorted edges
(A,D)
(D,B)
(A,B)
(B,C)
(C,D)

Add (D,B)

Path: A→D →B

Layout example

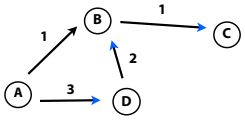


Sorted edges
(A,D)
(D,B)
(A,B)
(B,C)
(C,D)

Reject (A,B)

Path: A→D →B

Layout example



Sorted edges
(A,D)
(D,B)
(A,B)
(B,C)
(C,D)

Accept (B,C)

Path: A→D →B→C

Consensus

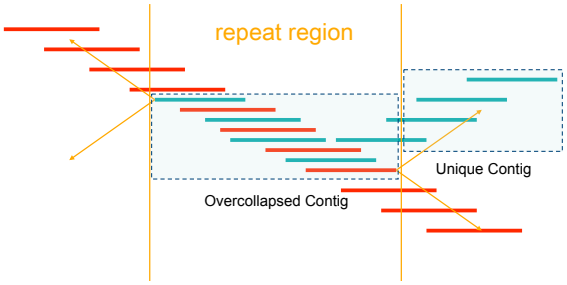
- Pairwise alignments between reads will specify a set of letters believe to represent the same position
  - Simple: use the letter that occurs the most
  - Complex: derive a multiple alignment, weight by quality

Practical problems: continuity

- If not all of the sequence is represented in reads, may not be able to resolve the whole sequence (the graph may not be connected)
- The result is a set of contigs
- Other methods would be needed the order and orientation of the contigs

Practical problems: repeats

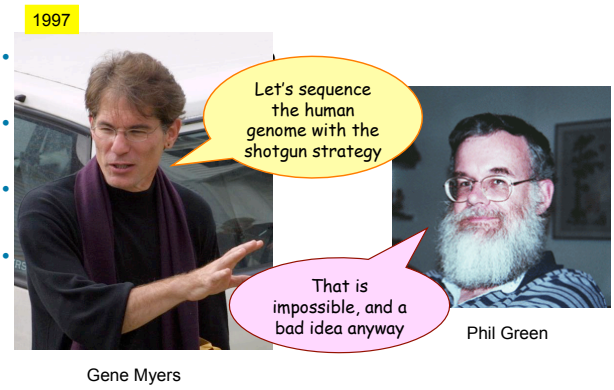
- If repetitive regions are longer than the read length, cannot be resolved
- Merge reads *up to* potential repeat boundaries (need to detect repeat boundaries in layout and break paths)



Even larger fragments

- This strategy was developed and used successfully for sequencing small regions (~50kb)
- How do we scale up to a whole genome?

History of WGA



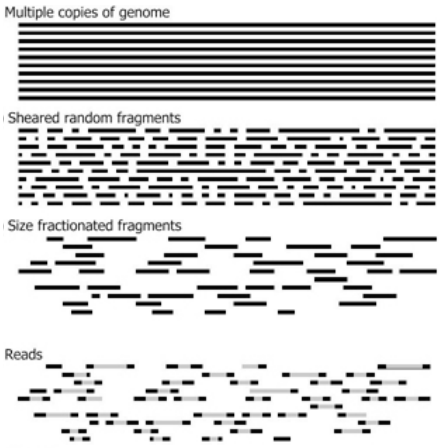
(I stole this slide verbatim from Serafim Batzoglou)

Whole genome shotgun sequencing

(Celera's assembly of the human genome)

Whole genome shotgun

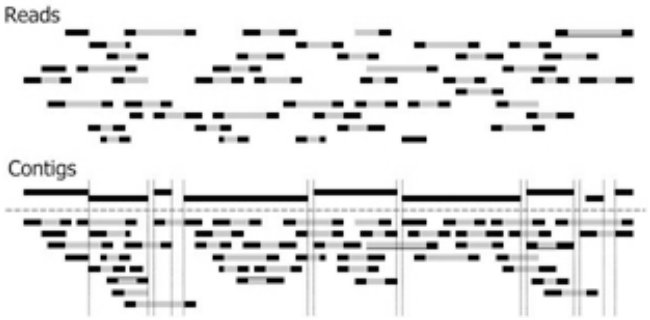
- Randomly fragment genomic DNA
- Select for fragments of a certain size
- Insert fragment into a plasmid, grow up bacteria to create many copies of each fragment
- Sequence from each end of the fragment



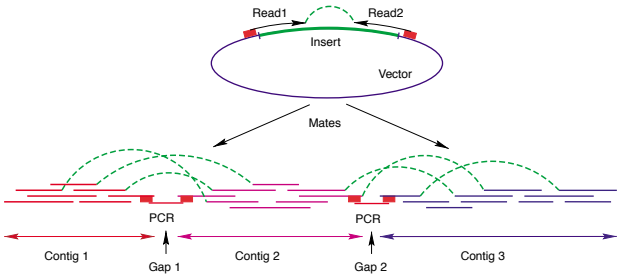
Shotgun assembly

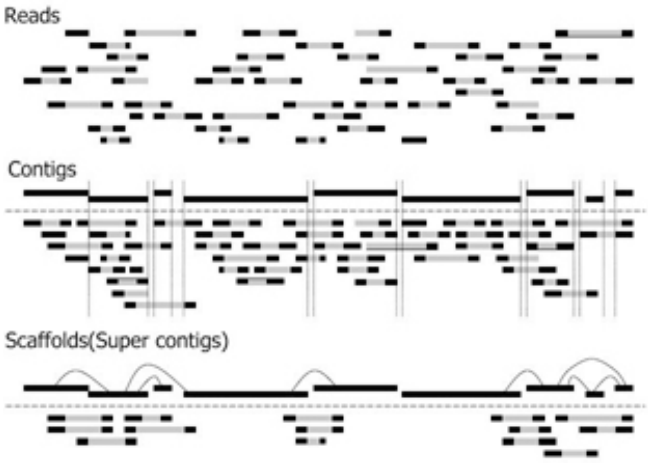
- Merge reads into contigs as described previously

Shotgun assembly

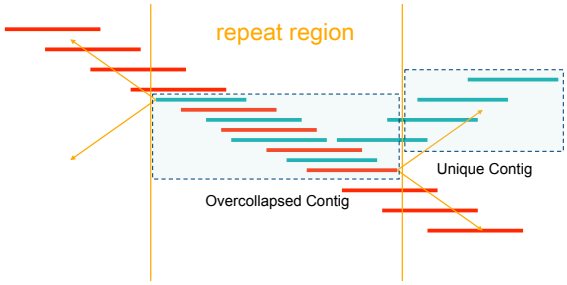


- Merge reads into contigs as described previously
- Order contigs into scaffolds
  - Mate-pair information provides order and approximate distance
  - Multiple insert sizes can make this much more effective

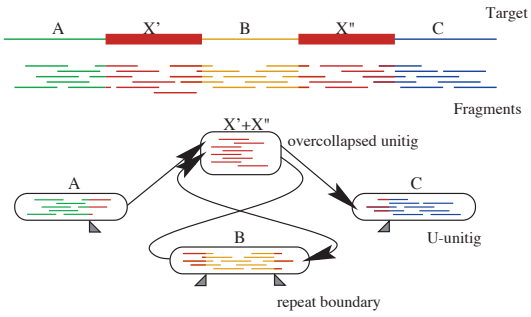


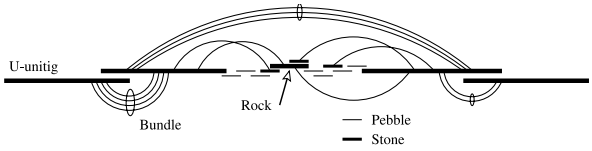


Main problem: resolving repeats



Resolving long repeats



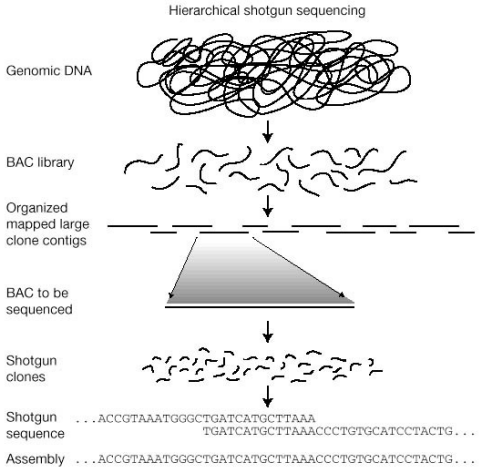


Hierarchical shotgun sequencing

(The public human genome project)

Hierarchical strategy

- Construct a set of large (100 to 200kb) clones, and sequence each independently with the shotgun approach
  - Clones are mapped and selected to provide an ordered tiling of the genome
- Devised to eliminate long-range misassembly and reduce the risk of short-range misassembly
- Allows targeting specific regions of the genome for greater sequencing depth



Assembling short reads

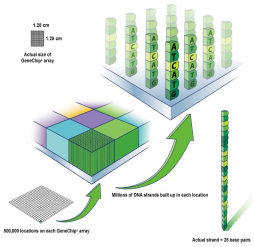
Assembling short reads

- Problems with O-L-C approach
  - Complexity: orders of magnitude more reads to deal with, challenging both for overlap and layout (even with heuristics approaches)
  - Repeats: difficulty resolving repeats longer than read size much more problematic as reads get much shorter
- So, we can either get a lot more efficient, or find a new approach

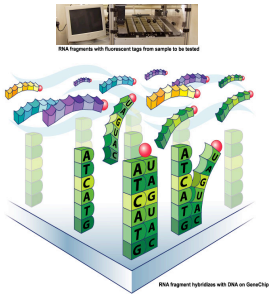
Inspiration: sequencing by hybridization

- What if we had a sequencing technology that could tell us all of the k-mers contained in a particular sequence?

Microarray (Affymetrix)

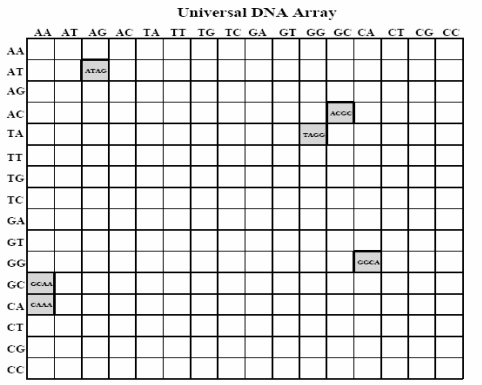


Millions of DNA strands build up on each location.



Tagged probes become hybridized to the DNA chip's microarray.

Universal array for all 4-mers

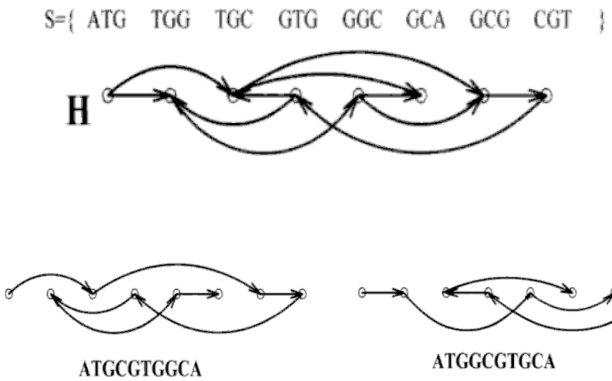


Inspiration: sequencing by hybridization

- What if we had a sequencing technology that could tell us all of the k-mers contained in a particular sequence?
  - Such as a universal microarray
- Can we re-construct a sequence from all of its k-mers?

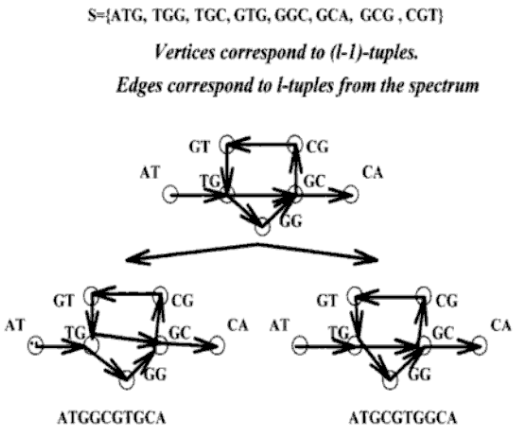
Solving the SBH problem

- We can solve it in nearly the same way as the SCS problem
- Build a graph in which
  - each node is a k-mer
  - each (directed) edge between nodes *s* and *t* means the suffix of *s* is the prefix of *t*.



Solving the SBH problem

- We can solve it in nearly the same way as the SCS problem
- Build a graph in which
  - each node is a k-mer
  - each (directed) edge between nodes *s* and *t* means the suffix of *s* is the prefix of *t*.
- Find a path through the graph that visits every vertex exactly once
  - Hamiltonian path, NP-complete

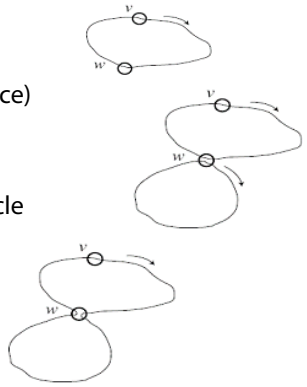


SBH as an Eulerian path problem

- Instead of making the vertices k-mers, make them all the k-1 mers
- Each k-mer then defines exactly one edge in the graph
- This is called a "de Bruijn" graph

Finding an Eulerian path

- Start at an arbitrary vertex *v* and form an arbitrary cycle (without using any edges twice)
- If the cycle is not Eulerian, it must contain some vertex *w* with unused edges, find a cycle from that vertex
- Combine and repeat



Major problems with SBH

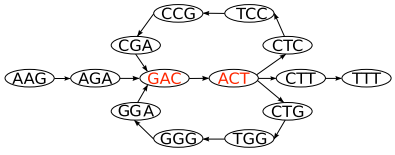
- Hard to detect exact k-mers (probes can hybridize with mismatches)
- Longer k-mers help, but this increases the array size exponentially

NGS ⇔ SBH

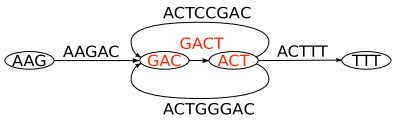
- Even with high-density microarrays, hard to build a universal array for a reasonable size k
- Ultra short read sequencing at high depth provides a more efficient way to find k-mers (for some k shorter than the read length)
- Thus: assemble short reads by using them to accurately determine the spectrum of a sequence and an Eulerian path approach

de Bruijn graphs for SR assembly

AAGACTCCGACTGGGACTTT

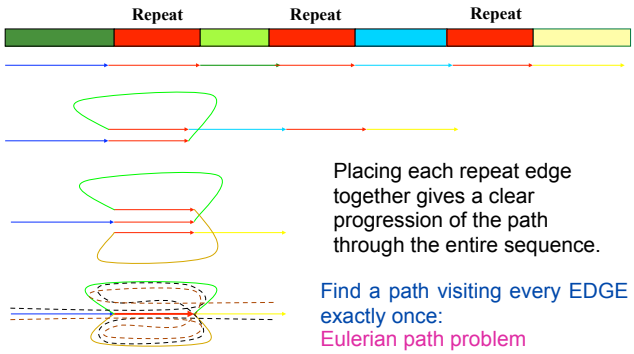


(a) de Bruijn graph of a sequence



(b) condensed de Bruijn graph

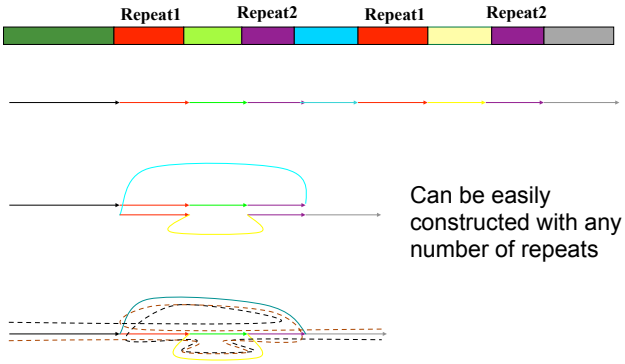
Repeat graph



Placing each repeat edge together gives a clear progression of the path through the entire sequence.

Find a path visiting every EDGE exactly once:
Eulerian path problem

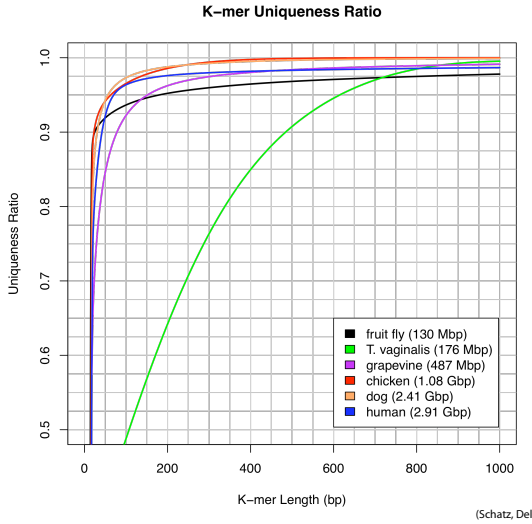
Repeat graph



Can be easily constructed with any number of repeats

Challenges

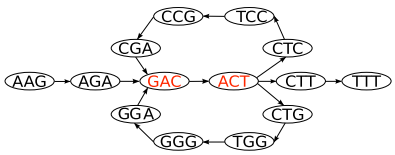
- Sequencing errors affect the graph substantially
  - Must correct somehow
- Resolving repeats
  - de Bruijn graph is simplified and transformed into a repeat graph
  - Goal of assembly: either transform the repeat graph so an Eulerian path can be found *or* find a simplified repeat graph (giving contigs)
  - Can't resolve tandem repeat counts



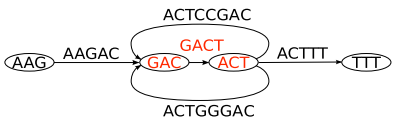
(Schatz, Delcher, and Salzberg 2010)

de Bruijn graph

AAGACTCCGACTGGGACTTT

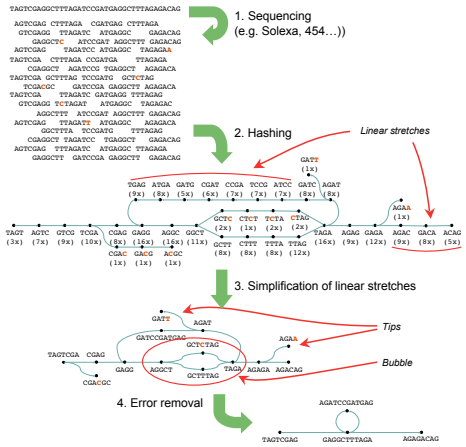


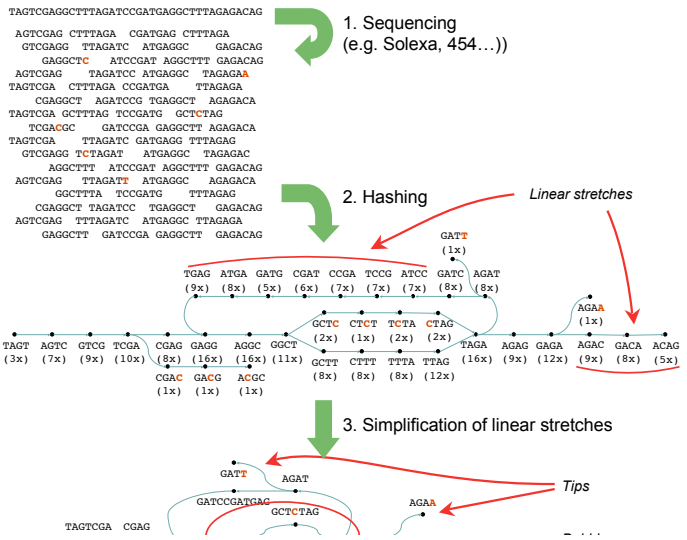
(a) de Bruijn graph of a sequence

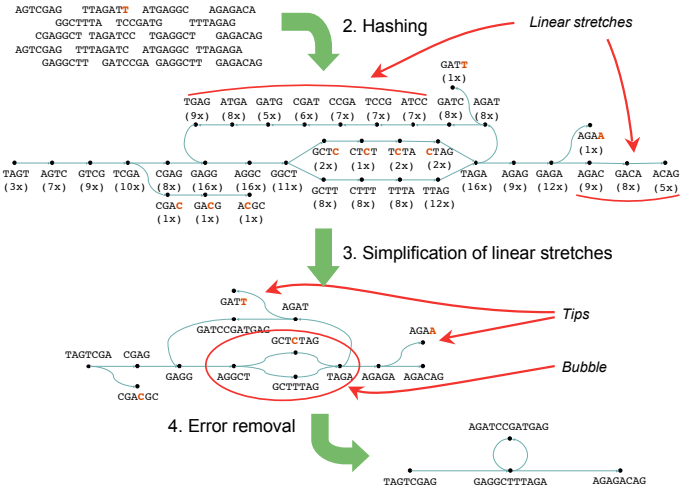


(b) condensed de Bruijn graph

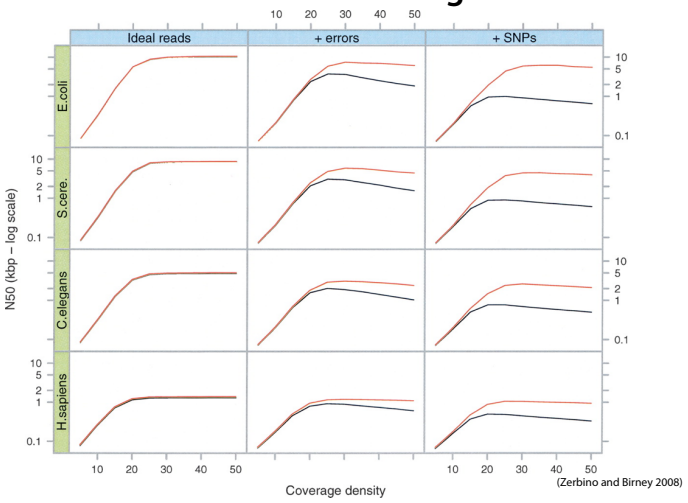
Velvet







Maximum N50 length

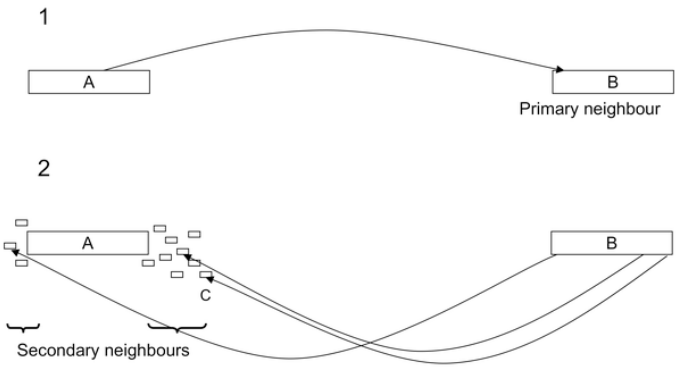


Connecting contigs

- Can extract unique contiguous regions from the graph, but length is limited by the read length and inherent repeat structure of the sequence
- Two ways to improve this:
  - Use some paired-end reads
  - Use some long reads

Velevet: Pebble (paired ends)

- Primary scaffold
  - For a given *unique* node, use all mate-pairs to estimate distance from that node to other *unique* nodes
  - Iterate, to produce a set of estimated distances between all pairs of unique nodes
- Secondary scaffold
  - Infer secondary neighbors from primary neighbors



Assisted assembly

Comparative assembly

- Align reads to existing assembled genome to guide assembly
- Between species with sufficient similarity
  - Human reference used to assemble Neanderthal, Chimpanzee, Orangutan, ...
  - Elephant used to assemble Mammoth,
  - ...
- Within species
  - Multiple human genomes assembled using reference

Problems with comparative assembly

- Difficult to recover complex variation
  - Small scale local rearrangement (segmental duplications) can be very hard to accurately sequence
  - Paired reads can help to uncover rearrangements
- Harder to assemble any novel sequences not represented in the reference
- Hybrid approach: *de novo* assembly followed by referenced assembly

Gene boosted assembly

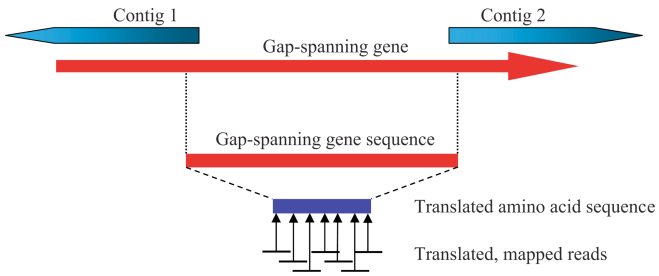
- Build an initial set of contigs and run gene prediction on them
- Where good predicted genes span contigs, use gene to orient contigs and fill in gaps by aligning reads to *predicted* amino acid sequence

Gene boosted assembly of *P. aeruginosa*

- Comparative assembly (using multiple references) for initials contigs
- Gene prediction on contigs
- For genes extending beyond or between contigs, align unassembled reads using tblastn
- For remaining unplaced reads, *de novo* assemble with velvet

(Salzberg et al. 2008)

Gene boosted assembly of *P. aeruginosa*



(Salzberg et al. 2008)

High-quality draft assemblies of mammalian genomes from massively parallel sequence data

Sante Gnerre[a], Iain MacCallum[a], Dariusz Przybylski[a], Filipe J. Ribeiro[a], Joshua N. Burton[a], Bruce J. Walker[a], Ted Sharpe[a], Giles Hall[a], Terrance P. Shea[a], Sean Sykes[a], Aaron M. Berlin[a], Daniel Aird[a], Maura Costello[a], Riza Daza[a], Louise Williams[a], Robert Nicol[a], Andreas Gnirke[a], Chad Nusbaum[a], Eric S. Lander[a,b,c,1], and David B. Jaffe[a,1]

[a]Broad Institute of MIT and Harvard, Cambridge, MA 02142; [b]Department of Biology, Massachusetts Institute of Technology, Cambridge, MA 02139; and [c]Department of Systems Biology, Harvard Medical School, Boston, MA 02115

Contributed by Eric S. Lander, November 23, 2010 (sent for review October 8, 2010)

Massively parallel DNA sequencing technologies are revolutionizing genomics by making it possible to generate billions of relatively short (~100-base) sequence reads at very low cost. Whereas such data can be readily used for a wide range of biomedical applications, it has proven difficult to use them to generate high-quality de novo genome assemblies of large, repeat-rich vertebrate genomes. To date, the genome assemblies generated from such data have fallen far short of those obtained with the older (but much more expensive) capillary-based sequencing approach. Here, we report the development of an algorithm for genome assembly, ALLPATHS-LG, and its application to massively parallel DNA sequence data from the human and mouse genomes, generated on the Illumina platform. The resulting draft genome assemblies have good accuracy, short-range contiguity, long-range connectivity, and coverage of the genome. In particular, the base accuracy is high (≥99.95%) and the scaffold sizes (N50 size = 11.5 Mb for human and 7.2 Mb for mouse) approach those obtained with capillary-based sequencing. The combination of improved sequencing technology and improved computational methods should now make it possible to increase dramatically the de novo sequencing of large genomes. The ALLPATHS-LG program is available at http://www.broadinstitute.org/science/programs/genome-biology/crd.

raised about the quality of de novo assemblies that can be constructed from such data (13).

Here, we describe an algorithm and software package ALLPATHS-LG for de novo assembly of large (and small) genomes. We demonstrate the power of the approach by applying it to massively parallel sequence data generated from both the human and the mouse genomes. The results approach the quality of assemblies obtainable with capillary-based sequencing in terms of completeness, contiguity, connectivity, and accuracy. The uncovered regions of the genome consist largely of repetitive sequences, with segmental duplications remaining a particularly important challenge. The results indicate that it should be possible to generate high-quality draft assemblies of large genomes at ~1,000-fold lower cost than a decade ago.

Results

Model for Input Data. De novo genome assembly depends both on the computational methods used and on the nature and quantity of sequence data used as input. For capillary-based sequencing, genome scientists ultimately converged around a fairly standard model, specifying the desired coverage from libraries of various insert sizes. For massively parallel sequencing data, we specify such a model in Table 1.

We adopted this model for several reasons. First, it requires

Allpaths-LG: Model for sequencing

Table 1. Provisional sequencing model for de novo assembly

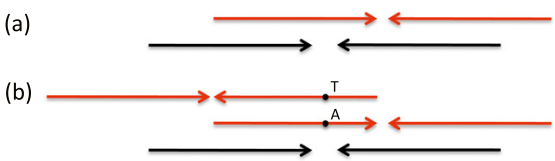
Libraries, insert types*	Fragment size, bp	Read length, bases	Sequence coverage, ×	Required
Fragment	180†	≥100	45	Yes
Short jump	3,000	≥100 preferable	45	Yes
Long jump	6,000	≥100 preferable	5	No‡
Fosmid jump	40,000	≥26	1	No‡

*Inserts are sequenced from both ends, to provide the specified coverage.
†More generally, the inserts for the fragment libraries should be equal to ~1.8 times the sequencing read length. In this way, the reads from the two ends overlap by ~20% and can be merged to create a single longer read. The current sequencing read length is ~100 bases.
‡Long and Fosmid jumps are a recommended option to create greater continuity.

Allpaths-LG key innovations

- Better recipes and protocols for library prep, more even representation of sequences
- "read doubling" to span repetitive regions
- Hierarchical approach to low coverage regions

"Read doubling"



"Gap patching"

Table 3. Human and mouse assemblies

Assemblies:	Human			Mouse		
Assembly no.:	1	2	3	4	5	6
Sequence data:	Illumina	Illumina	ABI3730	Illumina	Illumina	ABI3730
Program:	ALLPATHS-LG	SOAP	Celera	ALLPATHS-LG	SOAP	ARACHNE
Completeness						
Covered, %	91.1	74.3	96.2	88.7	86.2	94.2
Captured, %	6.6	18.6	1.3	8.6	8.0	3.8
Uncaptured, %	2.3	7.0	2.5	2.7	5.7	2.0
Segmental duplication coverage, %	41.1	12.1	62.2	42.3	27.9	65.7
Exon bases covered, %	95.1	81.2	96.2	96.7	92.4	97.3
Continuity						
Contig N50, kb	24	5.5	109	16	16	25
Scaffold N50, kb	11,543	399	17,646	7,156	340	16,871
Contig accuracy						
Ambiguous bases, %	0.08	0	0	0.04	0	0
1-kb chunks vs. reference	NA12878 GRC	GRC	GRC	B6	B6	B6
(I) perfect	77.1			88.6	76.8	78.0
(II) ≤0.1% error rate	8.7			2.5	2.9	7.0
(III) ≤1%	10.2			5.7	6.1	11.7
(IV) ≤10%	3.1 3.6	5.5	3.6	2.8	11.8	2.4
(V) >10%	0.4 0.4	0.7	0.5	0.2	2.4	0.3
Base quality, from I–III	Q33			Q36	Q35	Q33
Misassembly % of 1-kb chunks, from IV–V	3.5 4.0	6.2	4.1	3.0	14.2	2.7
Scaffold accuracy						
Validity at 100 kb, %	99.1	99.5	99.7	99.0	98.8	99.1

Assemblathon 1: A competitive assessment of de novo short read assembly methods

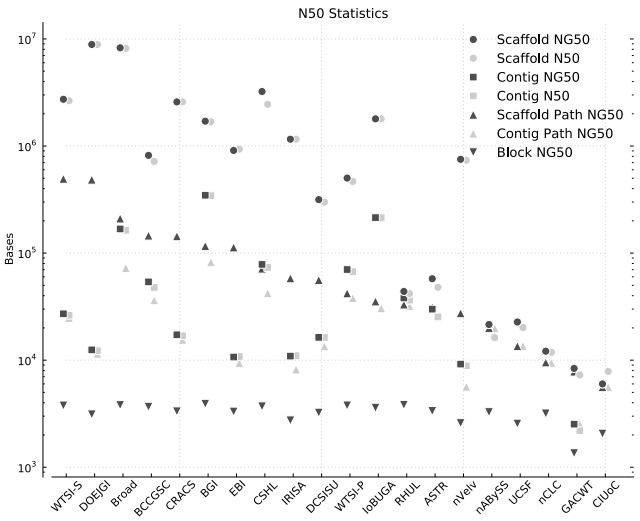
Dent A. Earl, Keith Bradnam, John St. John, et al.

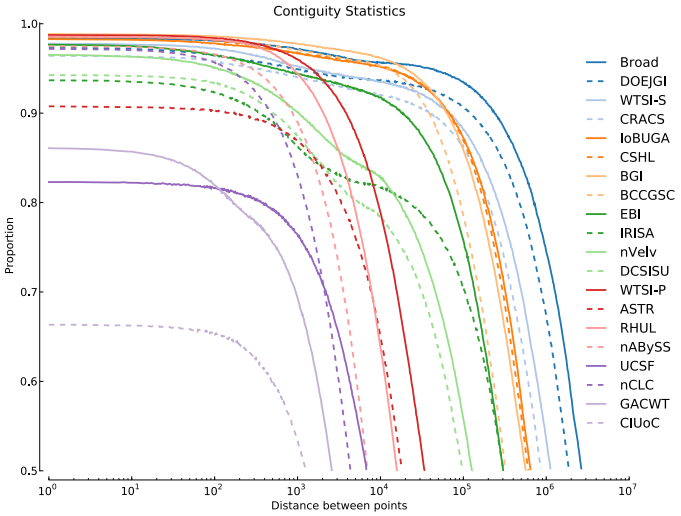
Genome Res. published online September 16, 2011
Access the most recent version at doi:10.1101/gr.126599.111

Two simulated genomes using evolver, simulated reads from first genome used for assembly

ID	Affiliations	Entries	Software	Used β
ASTR	Agency for Science, Technology and Research, Singapore	1	PE-Assembler	No
WTSI-P	Wellcome Trust Sanger Institute, UK	2	Phusion2, phrap	No
EBI	European Bioinformatics Institute, UK	2	SGA, BWA, Curtain, Velvet	No
WTSI-S	Wellcome Trust Sanger Insitute, UK	4	SGA	No
CRACS	Center for Research in Advanced Computing Systems, Portugal	3	ABySS	Yes
BCCGSC	BC Cancer Genome Sciences Centre, Canada	5	ABySS, Anchor	No
DOEJGI	DOE Joint Genome Insititute, USA	1	Meraculous	No
IRISA	L'IRISA (Institut de recherche en informatique et systèmes aléatoires), France	5	Monument	No
CSHL	CSHL (Cold Spring Harbor Laboratory), USA	2	Quake, Celera, Bambus2	No*
DCISU	Department of Computer Science, Iowa State University	1	PCAP	No
IoBUGA	Computational Systems Biology Laboratory, University of Georgia, USA	3	Seqclean, SOAPdenovo	No
UCSF	UC San Francicso, USA	1	PRICE	Yes
RHUL	Royal Holloway, University of London, UK	5	OligoZip	No
GACWT	The Genome Analysis Centre, Sainsbury Laboratory, and Wellcome Trust Centre for Human Genetics, UK	3	Cortex_con_rp	No
CIUoC	Department of Computer Science, University of Chicago, USA	1	Kiki	No
BGI	BGI, Shenzhen China	1	SOAPdenovo	No
Broad	Broad Institute	1	ALLPATHS-LG	No
nVelv	—	6	Velvet	No
nCLC	—	9	CLC	No
nABySS	—	6	ABySS	No

ID	Overall	CPNG50	SPNG50	Struct.	CC50	Subs.	Copy Num.	Cov. Tot.	Cov. Genic
Broad	31	2 (7.25e+04)	3 (2.11e+05)	3 (1244)	1 (2.66e+06)	4 (2.92e-06)	11 (6.71e-02)	6 (98.3)	1 (93.8)
BGI	37	1 (8.23e+04)	6 (1.17e+05)	6 (1878)	7 (5.66e+05)	11 (1.20e-05)	2 (6.75e-03)	1 (98.8)	3 (92.7)
WTSI-S	38	9 (2.48e+04)	1 (4.95e+05)	2 (475)	3 (1.14e+06)	1 (1.30e-07)	9 (5.74e-02)	8 (97.8)	5 (91.8)
DOEJGI	44	14 (1.15e+04)	2 (4.86e+05)	1 (456)	2 (1.89e+06)	3 (4.43e-07)	7 (5.42e-02)	11 (97.3)	4 (92.3)
CSHL	57	3 (4.23e+04)	8 (7.17e+04)	14 (5146)	6 (6.11e+05)	9 (1.02e-05)	6 (4.95e-02)	4 (98.5)	7 (89.1)
CRACS	58	11 (1.55e+04)	5 (1.44e+05)	4 (1666)	4 (8.61e+05)	2 (3.81e-07)	12 (6.82e-02)	14 (96.3)	6 (90.2)
BCCGSC	60	5 (3.63e+04)	4 (1.46e+05)	10 (2867)	8 (3.22e+05)	8 (7.00e-06)	15 (1.17e-01)	2 (98.7)	8 (88.9)
EBI	64	16 (9.39e+03)	7 (1.13e+05)	7 (2055)	9 (3.04e+05)	6 (5.17e-06)	1 (3.56e-03)	9 (97.7)	9 (88.5)
IoBUGA	65	7 (3.06e+04)	12 (3.54e+04)	15 (6310)	5 (6.47e+05)	15 (3.80e-05)	3 (8.38e-03)	6 (98.3)	2 (92.8)
RHUL	71	6 (3.20e+04)	13 (3.31e+04)	8 (2551)	15 (1.59e+04)	5 (3.52e-06)	5 (4.77e-02)	4 (98.5)	15 (67.4)
WTSI-P	74	4 (3.80e+04)	11 (4.21e+04)	13 (4895)	13 (3.41e+04)	14 (1.48e-05)	4 (4.38e-02)	2 (98.7)	13 (75.0)
DCSISU	99	12 (1.35e+04)	10 (5.61e+04)	12 (4319)	12 (9.75e+04)	13 (1.37e-05)	13 (6.91e-02)	15 (94.3)	12 (79.0)
nABySS	100	10 (1.99e+04)	16 (2.00e+04)	5 (1731)	16 (6.97e+03)	7 (5.96e-06)	19 (3.17e-01)	10 (97.5)	17 (57.2)
IRISA	103	17 (8.20e+03)	9 (5.82e+04)	11 (3725)	9 (3.04e+05)	17 (3.99e-05)	14 (7.61e-02)	16 (93.7)	10 (88.1)
ASTR	106	8 (2.52e+04)	14 (3.13e+04)	9 (2818)	14 (1.81e+04)	12 (1.28e-05)	18 (2.88e-01)	17 (90.9)	14 (68.5)
nVelv	114	18 (5.65e+03)	15 (2.75e+04)	18 (8626)	11 (1.27e+05)	18 (6.21e-05)	10 (6.22e-02)	13 (96.5)	11 (84.8)
nCLC	115	15 (9.47e+03)	18 (9.54e+03)	16 (7283)	18 (4.36e+03)	10 (1.11e-05)	8 (5.61e-02)	12 (97.2)	18 (55.4)
UCSF	138	12 (1.35e+04)	17 (1.35e+04)	20 (24987)	17 (6.84e+03)	20 (1.21e-04)	17 (2.30e-01)	19 (83.7)	16 (59.6)
GACWT	149	20 (2.53e+03)	19 (7.82e+03)	17 (8622)	19 (2.60e+03)	16 (3.86e-05)	20 (3.46e-01)	18 (86.4)	20 (48.0)
CIUoC	152	19 (5.60e+03)	20 (5.60e+03)	19 (11282)	20 (1.27e+03)	19 (1.11e-04)	16 (1.98e-01)	20 (78.5)	19 (48.9)



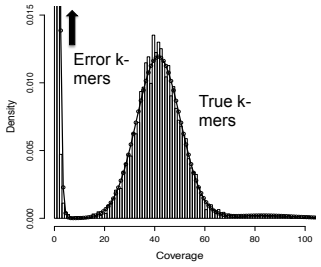


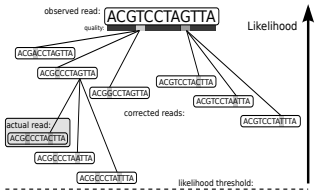
Next: Hybrid assembly

- Long (454, PacBio) reads can span gaps but are error prone and expensive
  - Use for scaffolding contigs assembled from short reads where more errors can be tolerated
  - Correct errors first using short-reads

Error correction is crucial

1. Count all "Q-mers" in reads
   - Fit coverage distribution to mixture model of errors and regular coverage
   - Automatically determines threshold for trusted k-mers
2. Correction Algorithm
   - Considers editing erroneous kmers into trusted kmers in decreasing likelihood
   - Includes quality values, nucleotide/nucleotide substitution rate





Quake: quality-aware detection and correction of sequencing reads.
Kelley, DR, Schatz, MC, Salzberg SL (2010) *Genome Biology.* 11:R116