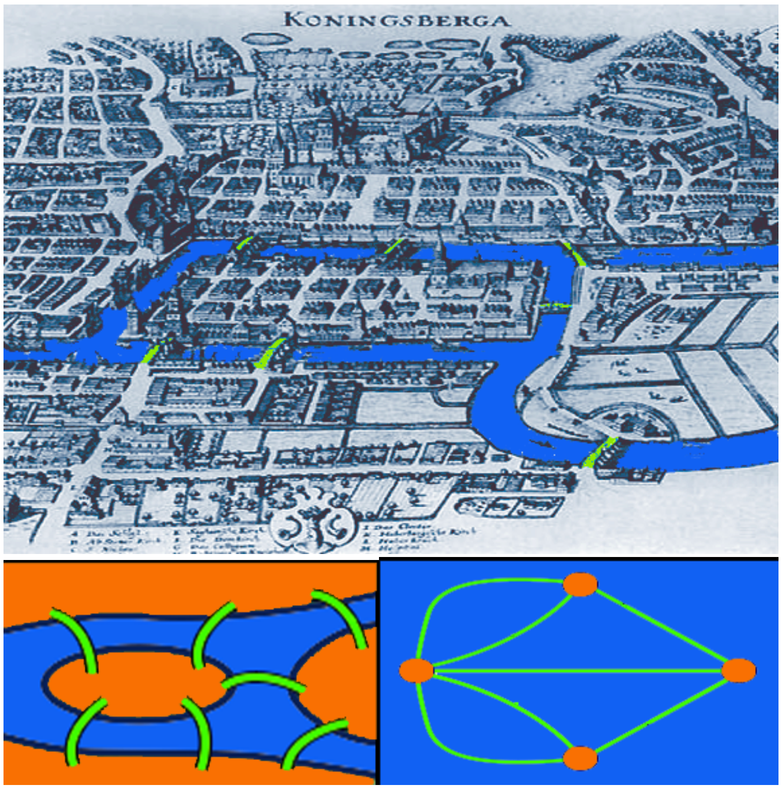
Excursion: Beginning of Graph Theory



Leonhard Euler (1707-1783)

Popular 18th century problem: *Is there a walk through Königsberg using each bridge exactly once?*

Euler Paths and Tours

Given an undirected connected graph $G = (V, E)$ with nodes V and edges E, we define:

Euler Path or Trail

is a path visiting each edge (of a graph) exactly once.

Euler Tour or Circuit or Cycle

is a path visiting each edge (of a graph) exactly once <u>and</u> ending at the starting point.

Euler Paths and Tours: Solution?

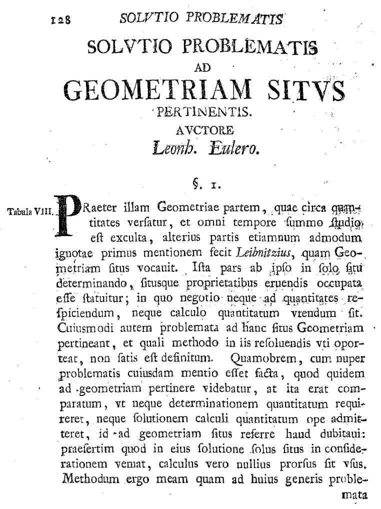
Leonhard Euler (1735) showed that the existence of such paths or tours depend on the degree of the nodes.

Euler Path

exists if there are exactly 0 or 2 nodes with uneven degree (i.e. number of attached edges).

Euler Tour

exists iff the graph is connected and has no nodes of odd degree.



128 *SOLVTIO PROBLEMATIS*

SOLVTIO PROBLEMATIS
AD
GEOMETRIAM SITVS
PERTINENTIS.
AVCTORE
Leonh. Eulero.

§. 1.

Tabula VIII. PRaeter illam Geometriae partem, quae circa quantitates verſatur, et omni tempore ſummo ſtudio eſt exculta, alterius partis etiamnum admodum ignotae primus mentionem fecit *Leibnitzius*, quam Geometriam ſitus vocauit. Iſta pars ab ipſo in ſolo ſitu determinando, ſitusque proprietatibus eruendis occupata eſſe ſtatuitur; in quo negotio neque ad quantitates reſpiciendum, neque calculo quantitatum vtendum ſit. Cuiusmodi autem problemata ad hanc ſitus Geometriam pertineant, et quali methodo in iis reſoluendis vti oporteat, non ſatis eſt definitum. Quamobrem, cum nuper problematis cuiusdam mentio eſſet facta, quod quidem ad geometriam pertinere videbatur, at ita erat comparatum, vt neque determinationem quantitatum requireret, neque ſolutionem calculi quantitatum ope admitteret, id ad geometriam ſitus referre haud dubitaui: praeſertim quod in eius ſolutione ſolus ſitus in conſiderationem veniat, calculus vero nullius prorſus ſit vſus. Methodum ergo meam quam ad huius generis problemata

Euler Paths and Tours on directed graphs

In directed graphs, that means, that edges have only one direction in which they can be crossed...

Euler Tour

exists iff the in-degree of each node is equal to its out-degree and if the graph is a strongly connected component, i.e. every node is reachable from every other node via a directed path.

Euler Path

exists iff there at most one node with *in-degree* - *out-degree* $= 1$ exists at most one node with *out-degree* - *in-degree* $= 1$.

Note, in graph theory the terms *node* and *vertex* (pl. vertices) are used interchangeably, as are *directed edge* and *arc*.

Variation: Hamiltonian Paths and Tours

Hamiltonian Path (or traceable path)

is a path that visits every node (of a graph) exactly once.

Hamiltonian Tour

is a Hamiltonian path ending at its starting point.

Problems:

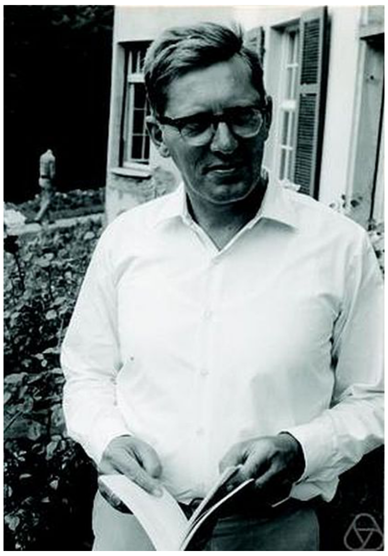
- determining whether such a path/tour exists is NP-complete.
- Hamiltonian Tours are a special case of the Traveling Salesman Problem.

William Rowan Hamilton (1805-1865)

Nicolaas de Bruijn

In 1946 Nicolaas de Bruijn got interested in the superstring problem:

- *Find the shortest circular superstring that contains all possible k-mers as substrings.*
- There are n^k k-mers for an alphabet of size $n = |\Sigma|$.
- For example, the number DNA-triplets: $n^k = 4^3 = 64$.


Nicolaas de Bruijn (1918-2012)

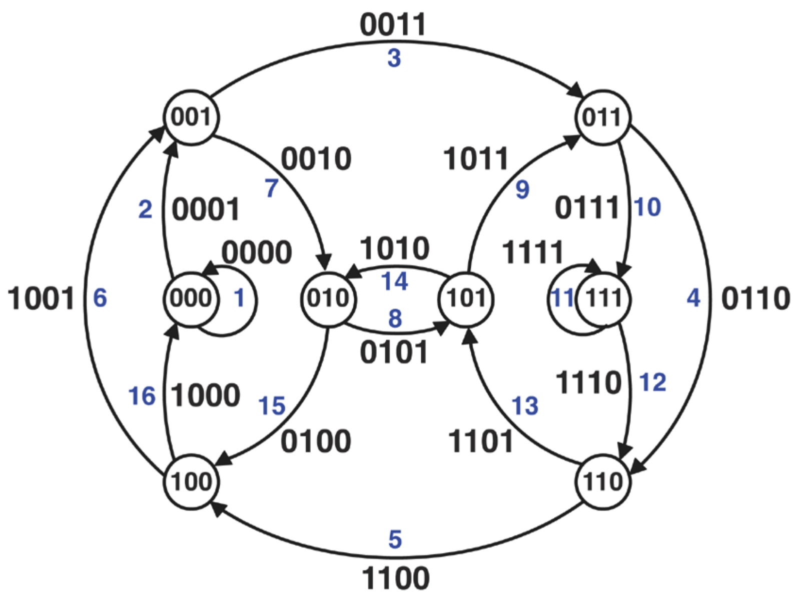
De Bruijn Graphs

De Bruijn Graph

- nodes: for all possible $(k-1)$-mers
- edges: directed links between nodes **a** and **b** if the $k-2$-long prefix of **b** is the suffix of **a**.
- Note: A Eulerian Tour exists because every node have one in-edge and one out-edge for each character in Σ.

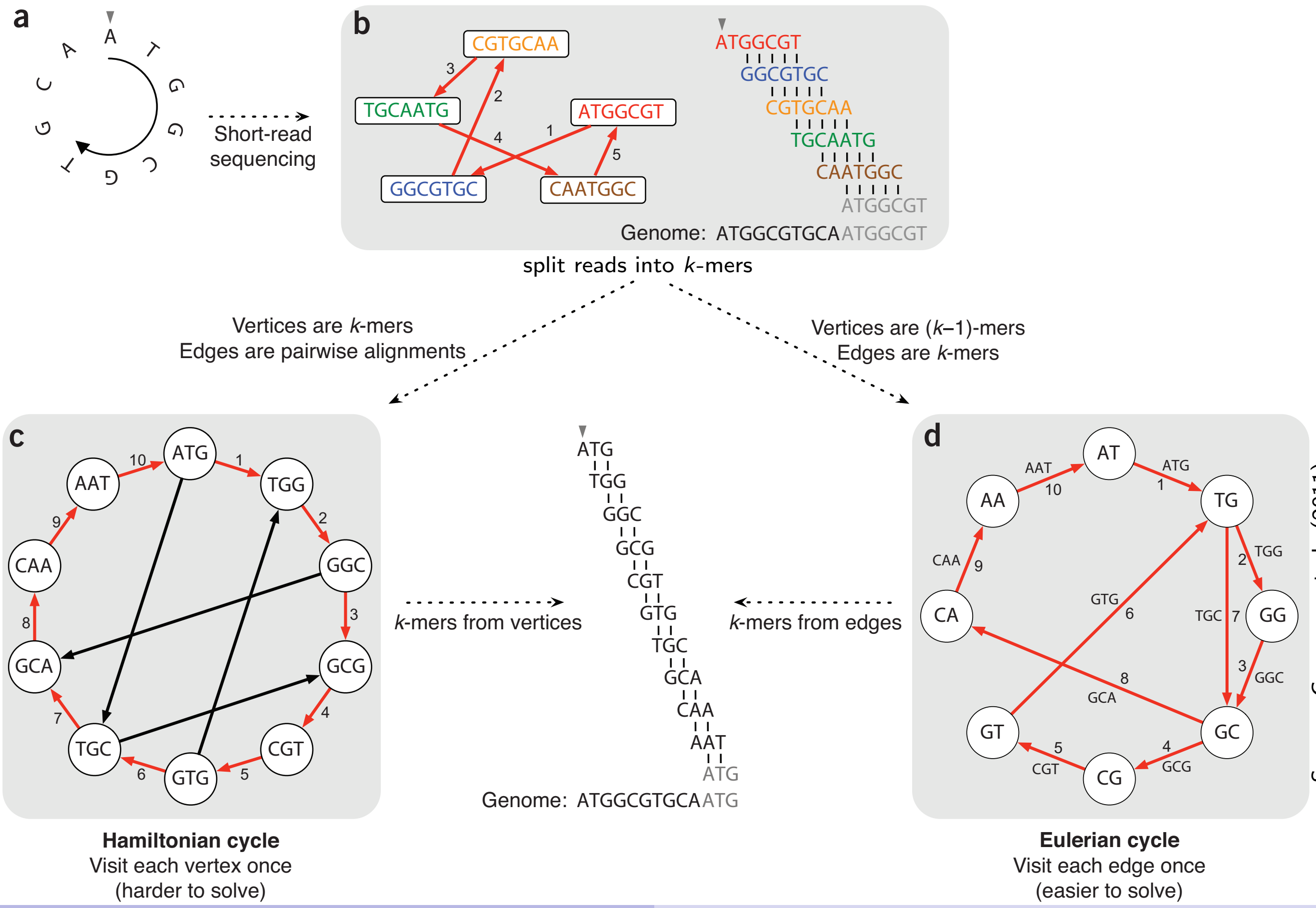
De Bruijn Graph Example

De Bruijn Graph for $k = 4$ and $\Sigma = \{0, 1\}$:



- The nodes are labeled with 000, 001, 010, 011, 100, 101, 110, 111.
- The edges are labeled linking the overlapping node labels, e.g. 100$\rightarrow$ 001 with 1001.
- An Eulerian Tour exists, each node has in-degree and out-degree 2.
- The Eulerian Tour (marked by blue numbers) spells out the circular superstring: 0000110010111101.

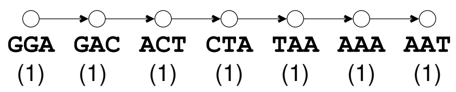
De Bruijn Graphs in Sequence Assembly

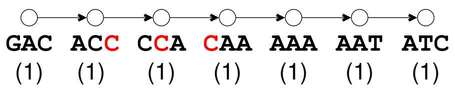


Source: Compeau et al. (2011)

Sequence De Bruijn Graph (from reads to graph)

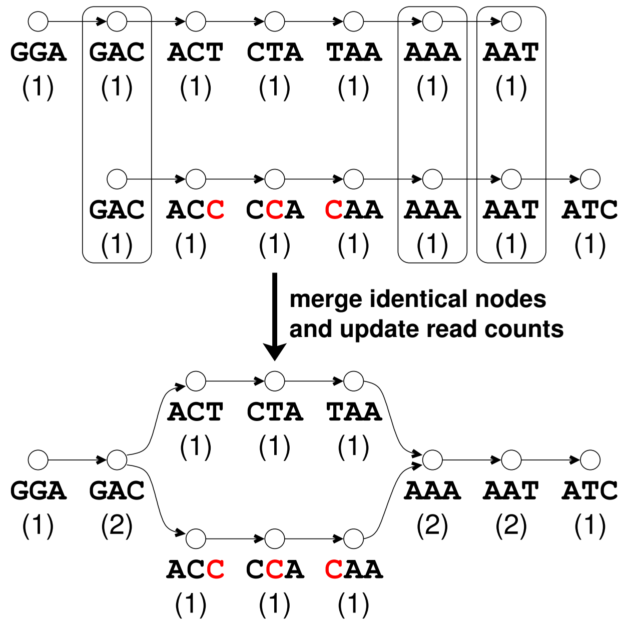
```
read 1:                                read 2:
G  G  A  C  T  A  A  A  T              G  A  C  C  A  A  A  T  C
G  G  A                                G  A  C
   G  A  C                                A  C  C
      A  C  T                                C  C  A
         C  T  A                                C  A  A
            T  A  A                                A  A  A
               A  A  A                                A  A  T
                  A  A  T                                A  T  C
```





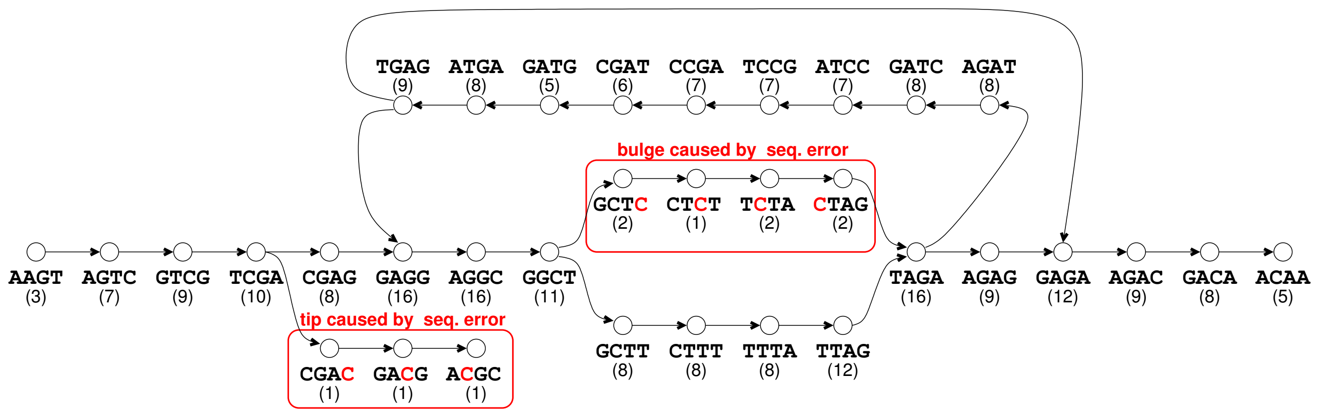
- construct a de Bruijn graph for each read with $k - 1 = 3$
- split the read into overlapping $k - 1$mers
- and construct the de Bruijn graph
- count 1 for the read at each node

Sequence De Bruijn Graph (merging identical nodes)



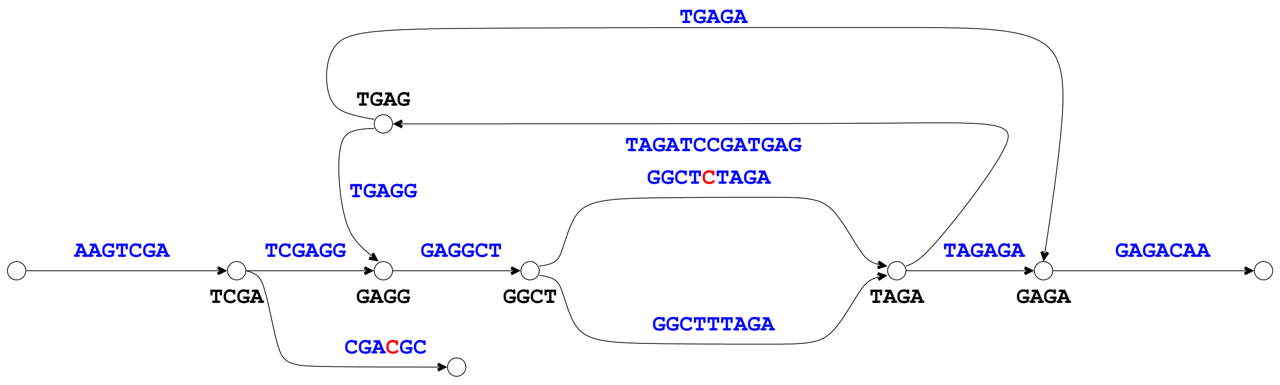
- take all 'read graphs'
- ...and merge identical nodes
- ...gaining a large de Bruijn graph
- sum up the read counts at the merged node
- do repeat this for all other reads

Sequence De Bruijn Graph (reducing the graph to contigs)



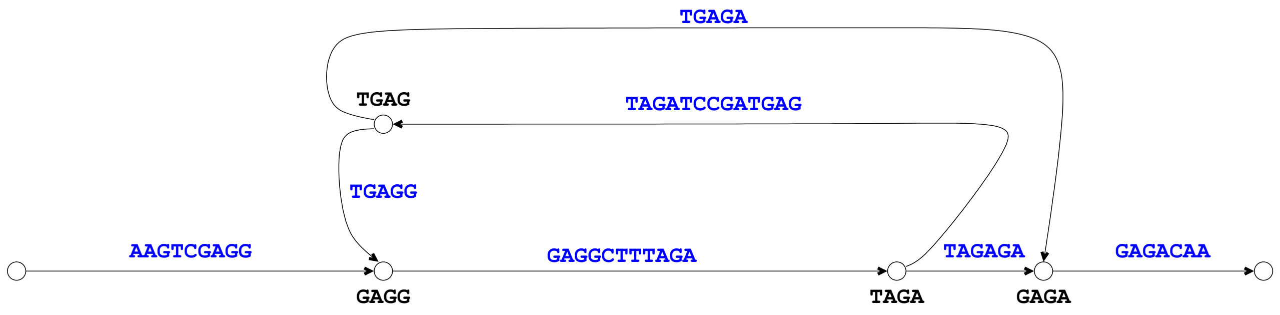
- sequencing errors cause tips and bulges with low read counts and can be identified using thresholds

Sequence De Bruijn Graph (reducing the graph to contigs)



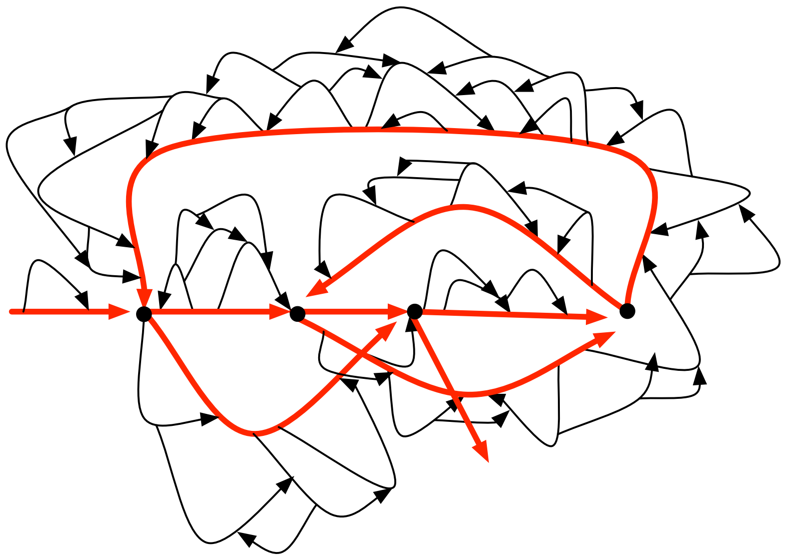
- sequencing errors cause tips and bulges with low read counts and can be identified using thresholds
- compactify the graph by merging the branch labels (rf. de Bruijn graph definition) on non-branching paths
- delete the obsolete nodes and re-link the branching nodes

Sequence De Bruijn Graph (reducing the graph to contigs)



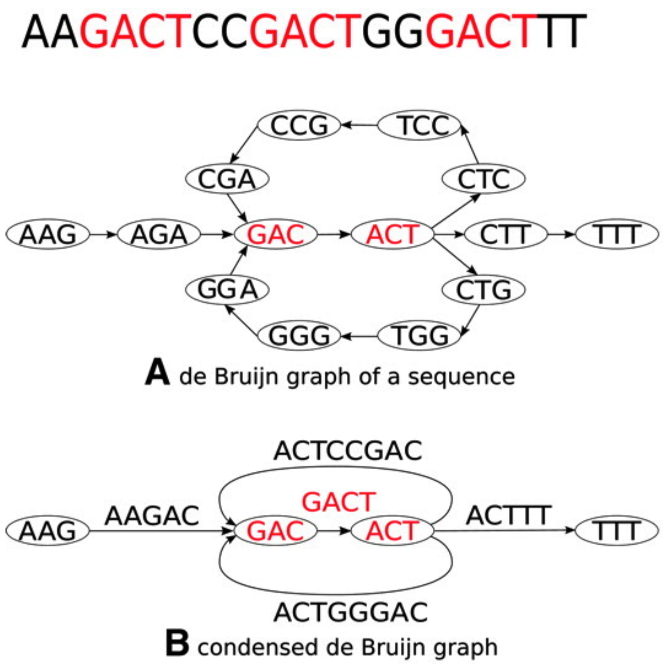
- sequencing errors cause tips and bulges with low read counts and can be identified using thresholds
- compactify the graph by merging the branch labels (rf. de Bruijn graph definition) on non-branching paths
- delete the obsolete nodes and re-link the branching nodes
- remove the tips and bulges with low read counts (sequencing errors)
- re-compactify the graph
- all the labels on the non-branching paths are our reconstructed contigs

De Bruijn Graphs for Assembly with Sequencing Errors



- The number of reads participating in bulges and tips tell us which are the frequent, and thus likely true ones.
- Bulges and tips with few reads are removed.

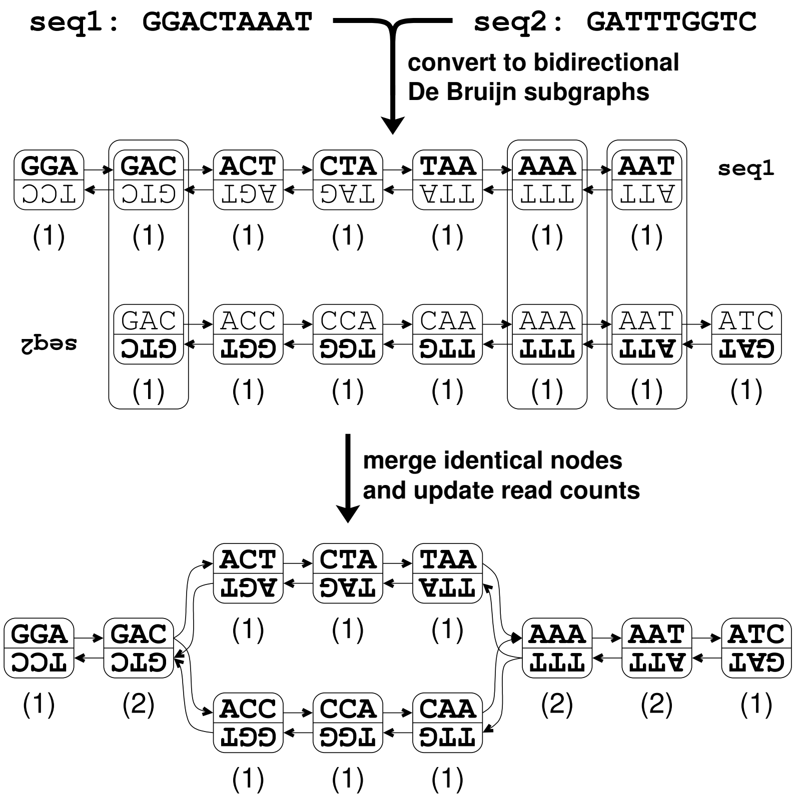
De Bruijn Graphs for Assembly with Repeats



Source: Chaisson et al. (2009)

- In the case of repeats, Euler paths are not possible, because the edges of the repeated region have to be used repeatedly.
- Typically the order in which edges from a repeat have to be followed cannot be determined. Then the paths have to be kept as separate contigs.
- Exception: if we have reads or paired-end information, which reach longer than the repeat, this helps to order the contigs.

Bidirected de Bruijn Graphs (Medvedev et al. 2007)


seq1: GGACTAAAT
seq2: GATTTGGTC
convert to bidirectional De Bruijn subgraphs
seq1
seq2
merge identical nodes and update read counts

- Problem: How do we know which strand our read is from? We don't!
- With bidirected de Bruijn Graphs one can cover k-mers and their complements.
- Note:
  At no time two nodes with identical strings can exist in one (sub)graph, and both strings have to be treated equally.

Some measures on genome sequencing and assembly

Coverage

Coverage describes the average number of times a nucleotide in the template DNA has been sequenced which is equivalent to the number of reads that cover each nucleotide on average.

$$coverage = \frac{\sum_{i \in \{all\ reads\}} length\ of\ read\ i}{length\ of\ template\ or\ genome}$$

Rule of thumb: the higher the better!

Some measures on genome sequencing and assembly

The quality of an assembly is hard to measure. Typically several values are used like

- maximum contig/scaffold length
- average contig/scaffold length
- combined total length
- the N50 or the NG50 value

N50 and NG50

N50 value

All contigs/scaffold are ordered descending in size. Starting from the largest contig/scaffold add their lengths. The N50 value is the length of the first contig, for which this sum of contig lengths covers $\geq$ 50% of the total length of contigs/scaffolds, i.e. the entire assembly.

Rule of thumb: the longer the better!

NG50 value

All contigs/scaffold are ordered descending in size. Starting from the largest contig/scaffold add their lengths. The NG50 value is the length of the first contig, for which this sum of contig lengths covers $\geq$ 50% of the total length of the sequenced genome.

Rule of thumb: the longer the better!

Sometimes other percentages than 50% are used leading to, e.g. N70 etc.

Reference based assembly

The whole procedure gets much easier if we have a reference genome available.

- mapping using search tools like BLAST or BLAT
- dynamic programming (e.g. Smith-Waterman) with pre-filtering to keep the candidate regions small, using
  - *hash-based k-mer index*
  - *spaced-seeds index*
- Approaches using the *Burrows-Wheeler-Transform* (BWT) of the reference sequence,

and the mapped reads are then summarized to contigs using consensus approaches.

Hash-based approaches

- Hash-based approaches typically require matching seed sequences (one or several) to identify candidate regions to be checked
- often contiguous seeds are used (e.g. perfectly matching words of length k)

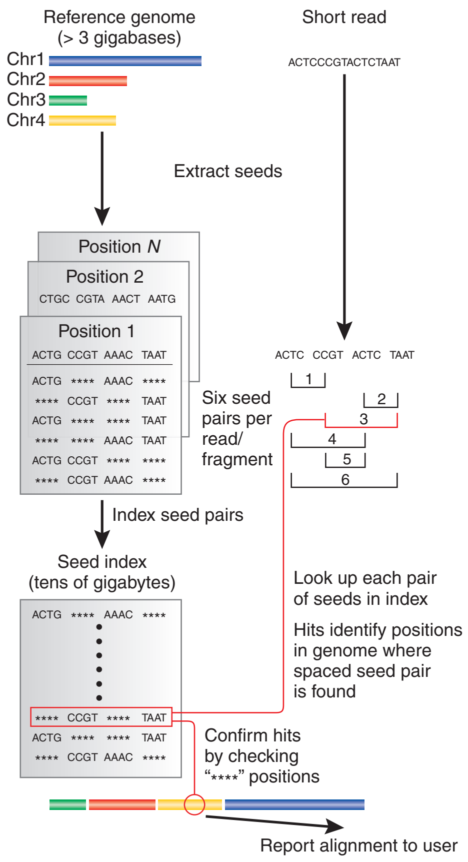
```
          1 1 1 1 1 1 1 1 1              seed encoding
...A C T A T C A T C G T A C A C A T...  reference sequence
          T C A T C G T A C              seed sequence (len=9)
   A C T A T C A T T G T A C A C A T     query sequence
```

- Another way is to use *spaced seeds*, i.e. that only certain letters in a longer word have to match.

```
   1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 1     seed encoding
...A C T A T C A T C G T A C A C A T...  reference sequence
   A . . . T C A . . . T A C . . A T     spaced seed (weight=9, len=17)
   A C T A T C A T T G T A C A C A T     query sequence
```

- It has been shown that the use of *spaced seeds* is much more sensitive, missing less hits. Especially, when using sets of spaced seeds.

Hash-based mapping



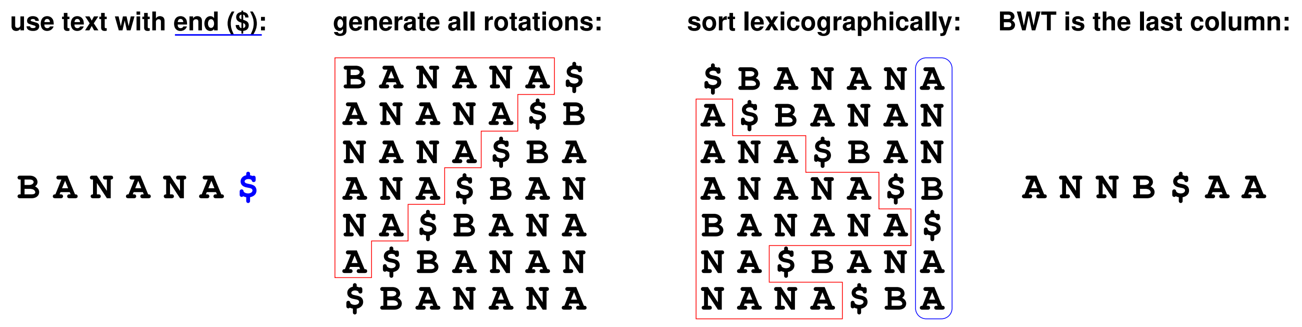
- mapping with candidate filtering based on (spaced) seed matches is easy to implement
- however, to generate a typical seed index is memory-intense (about 50GB for the human genome of 3 Gbp)
- the example uses six spaced seeds (1111111100000000, 0000000011111111, 0000111100001111, 1111000011110000, 0000111111110000, 1111000000001111)
- from all candidates the actual best hit position of the read has to be found by alignment and reported

Source: Trapnell+Salzberg (2009)

Burrows-Wheeler-Transform (BWT) - encoding

Methods using the Burrows-Wheeler-Transform (BWT) for mapping generate the BWT from the text, e.g. the genome, adding an start (^) and end character ($):

(Note, in the example we assume that $ sorts before the letters.)



use text with end ($):	**generate all rotations:**	**sort lexicographically:**	**BWT is the last column:**
	B A N A N A $	$ B A N A N A	
	A N A N A $ B	A $ B A N A N	
	N A N A $ B A	A N A $ B A N	
B A N A N A $	A N A $ B A N	A N A N A $ B	A N N B $ A A
	N A $ B A N A	B A N A N A $	
	A $ B A N A N	N A $ B A N A	
	$ B A N A N A	N A N A $ B A	

Originally, the Burrows-Wheeler-Transform (BWT) has been introduced in the field of data compression, because (a) the BTW compresses better than the original text and (b) one can decode the original text from the BWT.

Burrows-Wheeler-Transform (BWT) - decoding

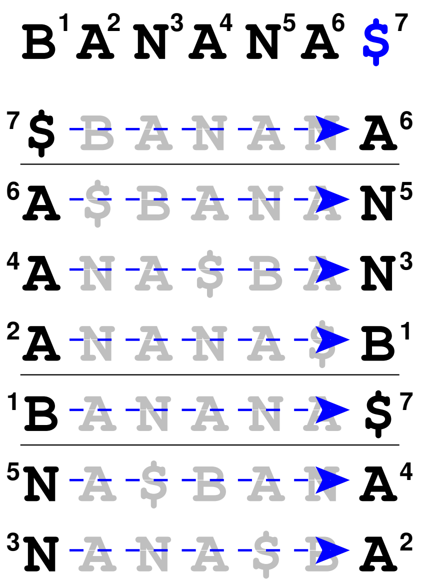
From the BTW the original text can easily decoded:

start from the BTW	sort	we know 1st +last column	rotate (BTW front)	sort	rotate (BTW front)
A	$	$ . . . A	A $ . . .	$ B . . .	A $ B . . .
N	A	A . . . N	N A . . .	A $ . . .	N A $ . . .
N	A	A . . . N	N A . . .	A N . . .	N A N . . .
B	A	A . . . B	B A . . .	A N . . .	B A N . . .
$	B	B . . . $	$ B . . .	B A . . .	$ B A . . .
A	N	N . . . A	A N . . .	N A . . .	A N A . . .
A	N	N . . . A	A N . . .	N A . . .	A N A . . .

and sort	add BTW and sort	add BTW and sort	add BTW and sort	...until the matrix has its width again
$ B A	$ B A N	$ B A N A	$ B A N A N	$ B A N A N A
A $ B	A $ B A	A $ B A N	A $ B A N A	A $ B A N A N
A N A	A N A $	A N A $ B	A N A $ B A	A N A $ B A N
A N A	A N A N	A N A N A	A N A N A $	A N A N A $ B
B A N	B A N A	B A N A N	B A N A N A	B A N A N A $
N A $	N A $ B	N A $ B A	N A $ B A N	N A $ B A N A
N A N	N A N A	N A N A $	N A N A $ B	N A N A $ B A

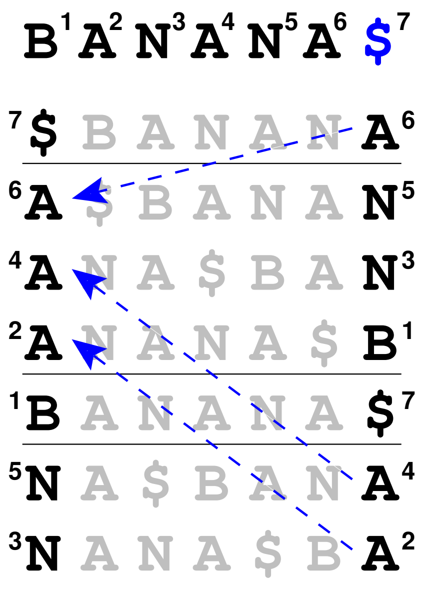
The original text can be found in the line ending with $.

Burrows-Wheeler-Transform (BWT) - some observations



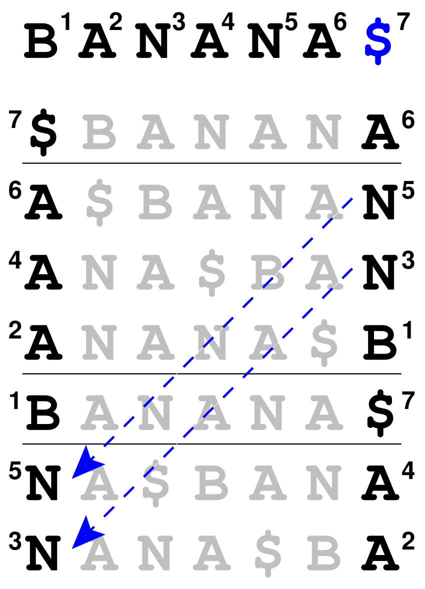
- a letter in the 1st column is easy to find, because they are sorted
- the letter in the last column preceeds the one in the 1st column (thus, searching for words starts at the last letter)

Burrows-Wheeler-Transform (BWT) - some observations



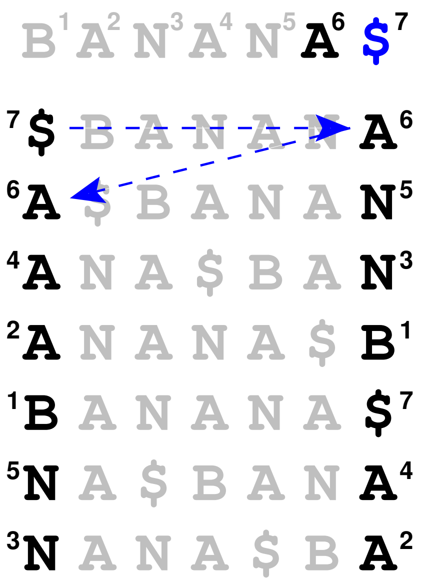
- a letter in the 1st column is easy to find, because they are sorted
- the letter in the last column preceeds the one in the 1st column (thus, searching for words starts at the last letter)
- the order of occurrence of a single letter in the last and the 1st column is the same (the 2nd A in the one is the 2nd A in the other)

Burrows-Wheeler-Transform (BWT) - some observations



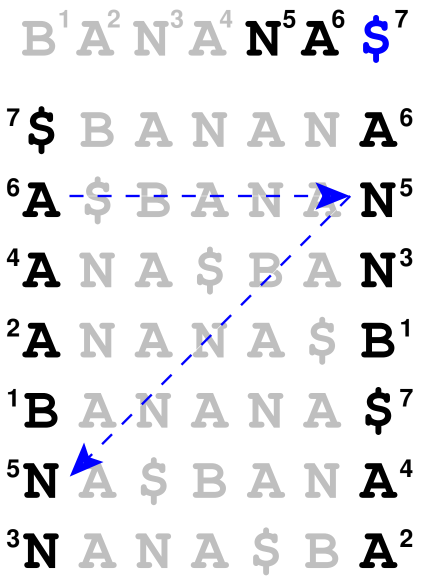
- a letter in the 1st column is easy to find, because they are sorted
- the letter in the last column preceeds the one in the 1st column (thus, searching for words starts at the last letter)
- the order of occurrence of a single letter in the last and the 1st column is the same (the 2nd A in the one is the 2nd A in the other)

Burrows-Wheeler-Transform (BWT) - easier decoding



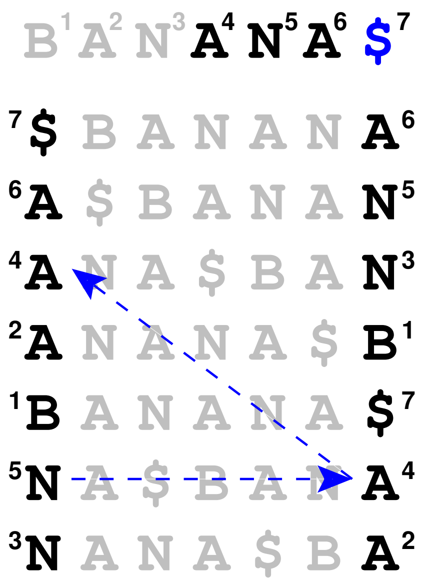
- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.

Burrows-Wheeler-Transform (BWT) - easier decoding



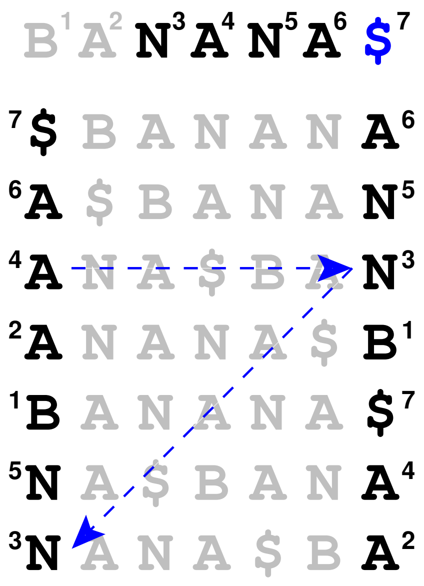
- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters

Burrows-Wheeler-Transform (BWT) - easier decoding



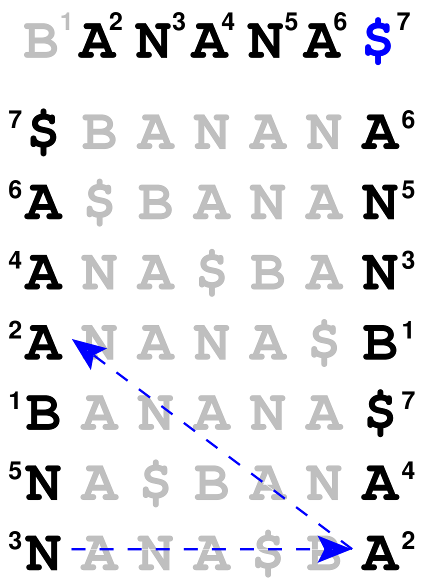
- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters

Burrows-Wheeler-Transform (BWT) - easier decoding



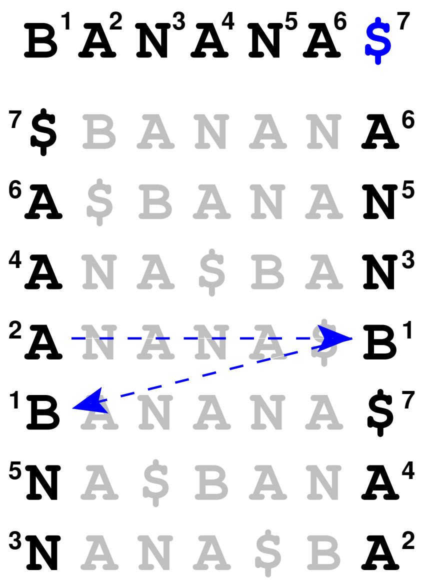
- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters

Burrows-Wheeler-Transform (BWT) - easier decoding



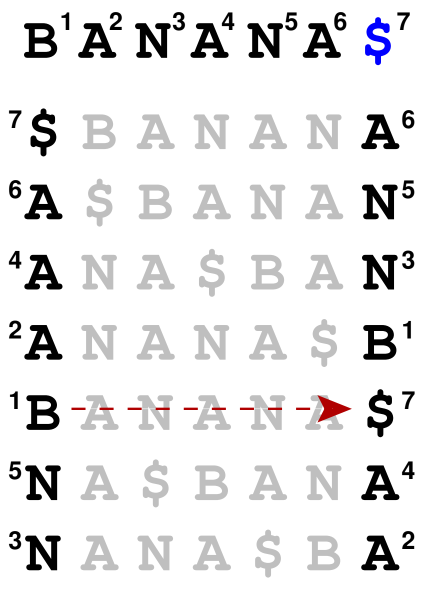
- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters

Burrows-Wheeler-Transform (BWT) - easier decoding



- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters

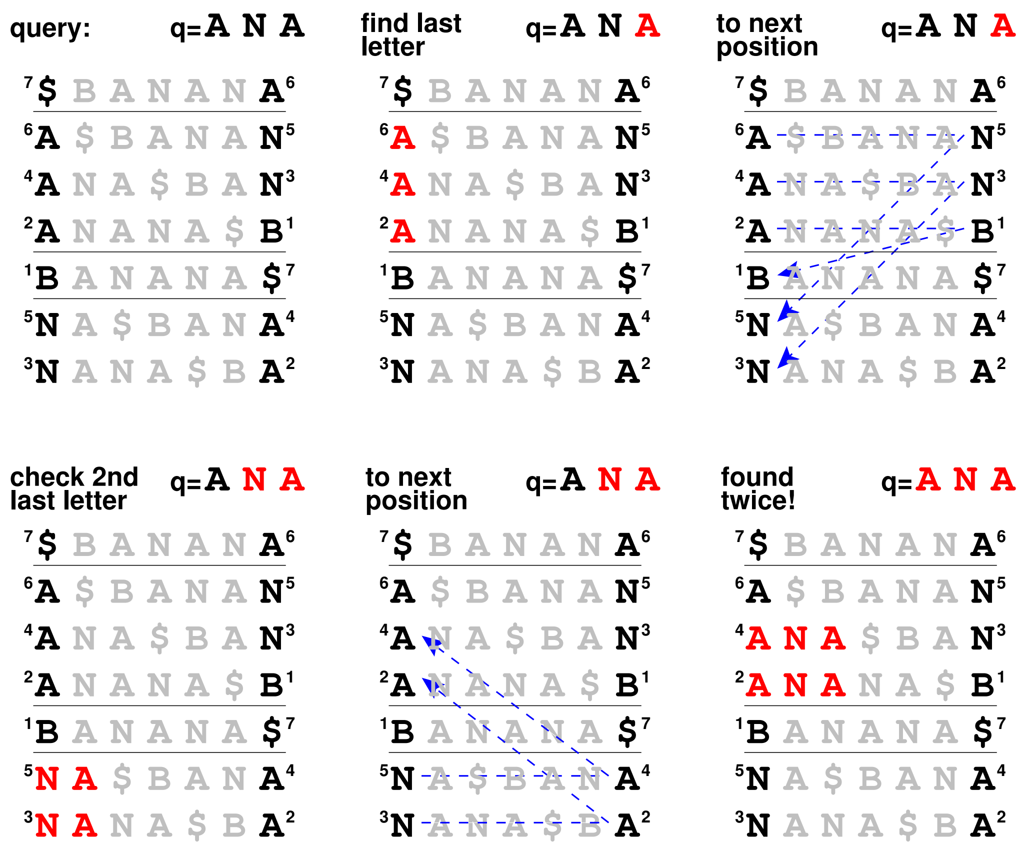
Burrows-Wheeler-Transform (BWT) - easier decoding



- from these observations about the last-column-first-column property, we can derive an easier way to decode a BWT
- We start from the end character ($) in the first column.
- We determine the preceeding character by checking the last column of the same row.
- Then we jump the the according character in the first column,
- and we note down the decoded character.
- Now we can repeat that to determine all preceeding characters
- and finish when we meet the end character ($) again

Burrows-Wheeler-Transform (BWT) - searching

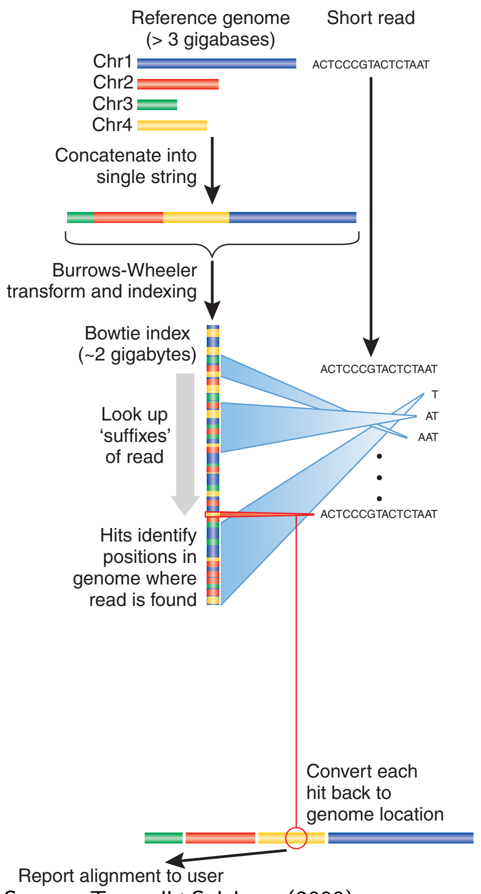
Searching from the last letter to the first of the search string (q=ANA):



Burrows-Wheeler-Transform (BWT) - approximate matches

- this way exact matches can be found easily
- to find approximate matches,
- everytime a mismatch is detected,
- a backtrace is done, introducing changes at any position
- at the beginning only one change and later more if still no match is found

BWT-based mapping


Reference genome
(> 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Concatenate into single string
Burrows-Wheeler transform and indexing
Bowtie index (~2 gigabytes)
Look up 'suffixes' of read
ACTCCCGTACTCTAAT
T
AT
AAT
ACTCCCGTACTCTAAT
Hits identify positions in genome where read is found
Convert each hit back to genome location
Report alignment to user

Source: Trapnell+Salzberg (2009)

- BWT-based is harder to implement than seed based approaches
- however, it is less memory intense (only (about 1-2GB for the human genome of 3 Gbp) and much faster
- on the other hand, seed based approaches have been shown to be much more sensitive, and thus able to match more reads correctly

Genome Analysis

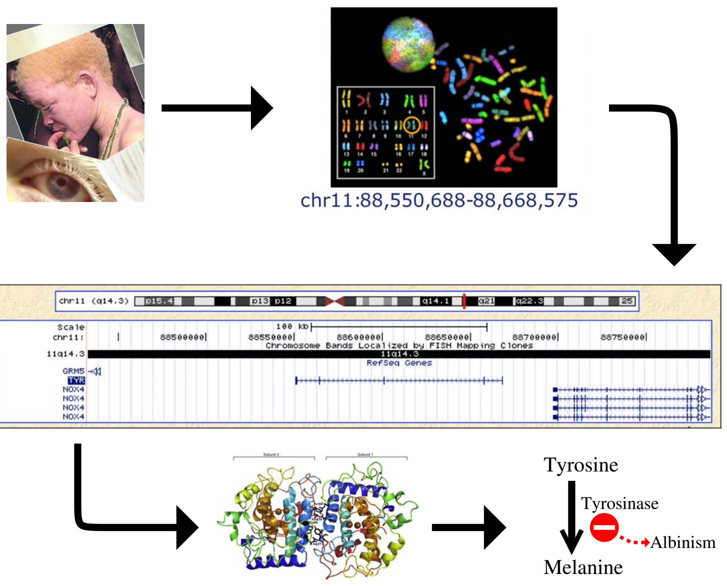
Genomics is Changing Biology - pre-genomic era

In the pre-genomic era:

1 gene →
1 research project

or

1 gene ⇒
1 master/PhD
thesis



Genomics is Changing Biology - post-genomic era

Today, in the post-genomic era:

WHERE is a functionality encoded

WHAT functionality is encoded?

1 genome $\rightarrow$
1 PhD/Masters project

or

Dozens or hundreds of genomes $\rightarrow$
background for one research project

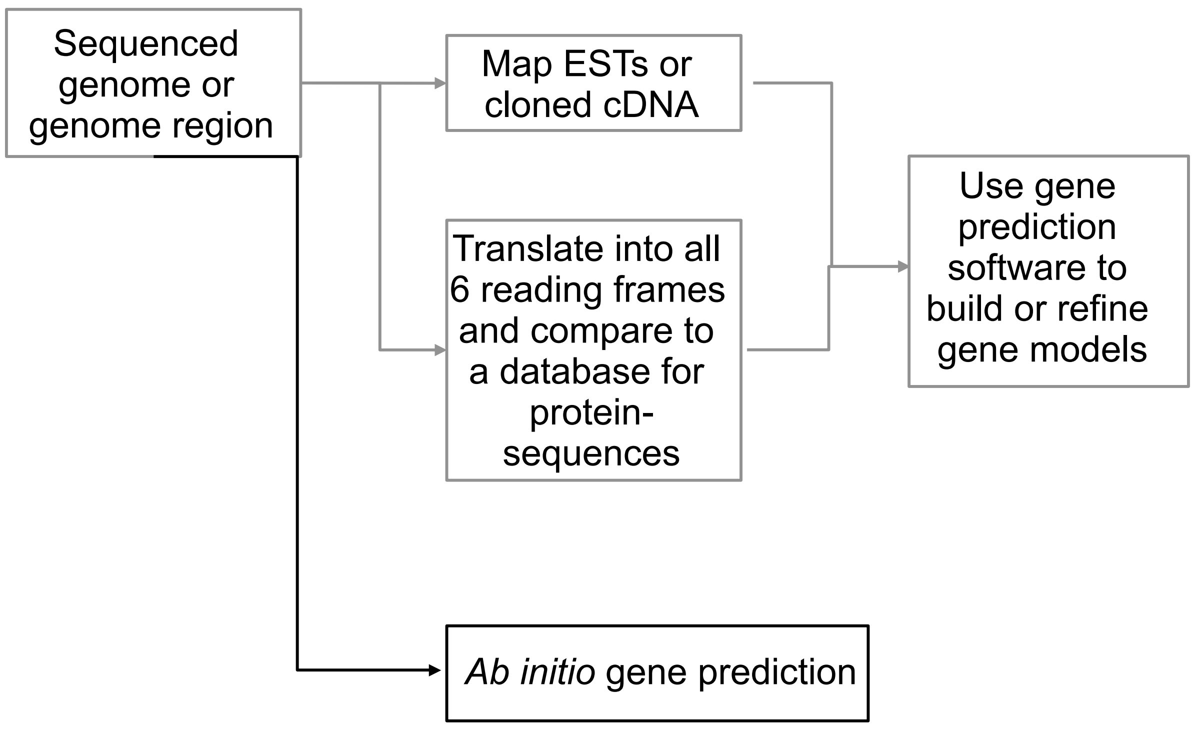
Newspapers: Genome deciphered, but...

```
agtctggagcccagaagggacacaccagcacagtctggtaggctacagca
gcaagtctctaaagaaaggctgagaacacccagaacaggagagttcaggt
ccaggatggccagcctgttccggtcctatctgccagcaatctggctgctg
ctgagccaactccttagagaaagcctagcagcagagctgaggggatgtgg
tccccgatttggaaaacacttgctgtcatattgccccatgcctgagaaga
cattcaccaccaccccaggagggtggctgctggaatctggacgtcccaaa
ggtgagagccctggactaccaaacaatcagaatgaggcctgaaaaaacag
gctccagatctcattgactgcctgtagtcaactcagactctactgtggct
agtgcctacaagtttgtggtttttcattgtaatgtgcttttattaaaagg
gtctcaccagaaatctcatgggaagttgggggtagaggagaagctgcagg
aaaaacagaagaacagcctcaccttggaggctcttggtgcctttcccacc
tggcagccagagatacagggtggagaaaacagggaatcctcagagaaggt
gactattctcaaacaccagcaggaggtgaggtgtactcccagacccccag
agataaatttaacaagaaaaatggaaagtattcatggataacaaatgtta
acatttggtgtcgacacatatagatgtatgttgcttctcagccttttagc
taagatccactgatatagatgtatagatttatgtggatcagatttgaata
tctgaatctttctattagaataacttgtgattataaaaaaaattccctat
tgctatccagtacataatgcattgaggttatcaatcaaaacagtgtcagg
gagagtcaaaagtggtatggtataggcatttagagggtcattagaagagt
gcatagaggggacaggatgaggagttagcatatccttaatattgtagtat
cttaaagtgccctactctaagtaagctaagttgttgaaatgttaggatac
ttgctgattctctctggtgtttaattacatggaggcaatgggtacattgt
ggtctaggcaaattgtataatttttctgatcctctttcacatgaatgttt
ttcctcacctttcattcctctcttttacttcacagaaatggtgtcaacct
ccaacaacaaagatggacaagccttaggtacgacatcagaattcattcct
aatttgtcaccagagctgaagaaaccactgtctgaagggcagccatcatt
gaagaaaataatactttcccgcaaaaagagaagtggacgtcacagatttg
atccattctgttgtgaagtaatttgtgacgatggaacttcagttaaatta
tgtacatagtagagtaatcatggactggacatctcatccattctcatatg
tattctcaatgacaaattcactgatgcccaattaaatgattgctgttt
```

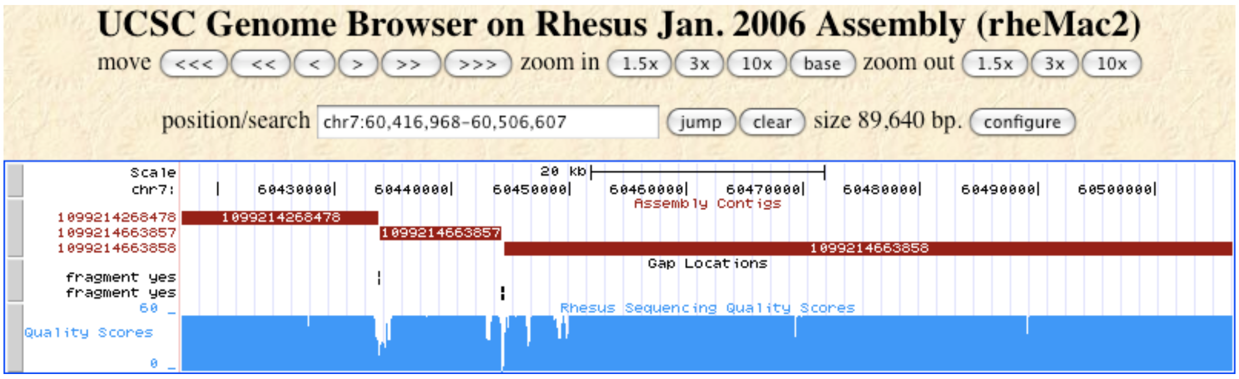
Procedures of Genome Analyses

- Gene annotation
- Alternative splicing (transcript variants, RNA-seq)
- Differential expression (transcript abundance, RNA-seq)
- Annotation of regulatory elements (ChIP-seq)
- Detection of genome variants (Single Nucleotide Polymorphisms or SNPs)
- Comparative genomics
- Repeat annotation

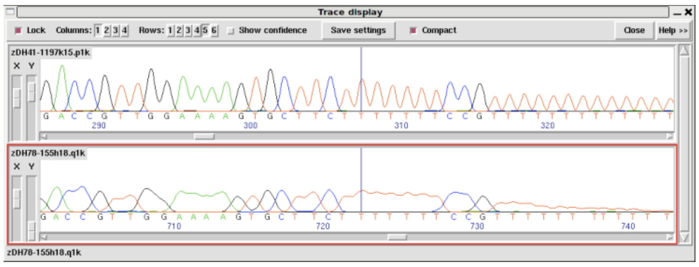
Gene annotation


Sequenced genome or genome region
Map ESTs or cloned cDNA
Translate into all 6 reading frames and compare to a database for protein-sequences
Use gene prediction software to build or refine gene models
Ab initio gene prediction

Sequencing quality: the Phred score


UCSC Genome Browser on Rhesus Jan. 2006 Assembly (rheMac2)

position/search chr7:60,416,968-60,506,607 size 89,640 bp.

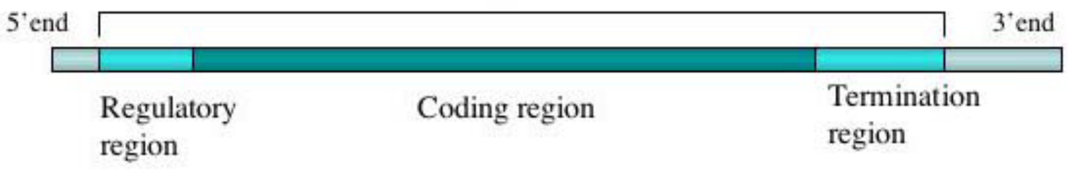


$$Q = -10 \log10(P)$$

P is the probability that your base call is incorrect

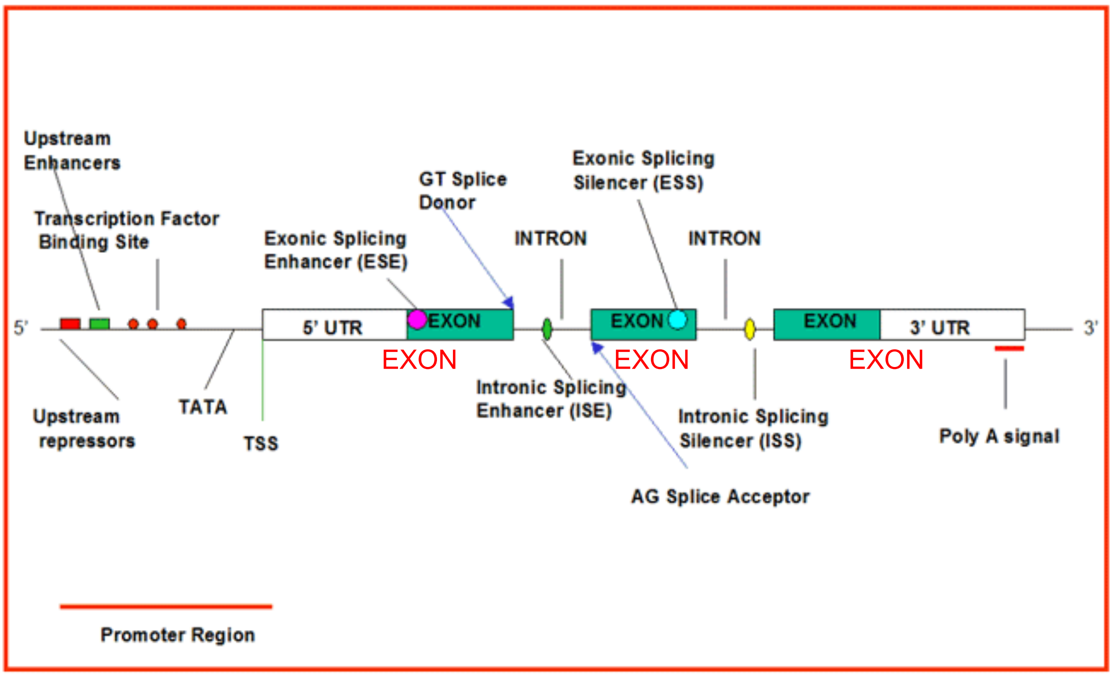
Gene Models: quick review

Prokaryotic gene model



Gene Models: quick review

A Eukaryotic gene model:

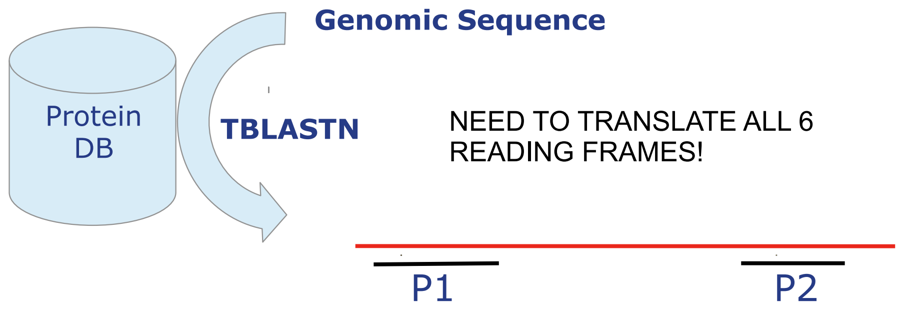

Upstream Enhancers
Transcription Factor Binding Site
Exonic Splicing Enhancer (ESE)
GT Splice Donor
Exonic Splicing Silencer (ESS)
INTRON
INTRON
5'
5' UTR
EXON
EXON
EXON
3' UTR
3'
EXON
EXON
EXON
Upstream repressors
TATA
TSS
Intronic Splicing Enhancer (ISE)
Intronic Splicing Silencer (ISS)
AG Splice Acceptor
Poly A signal
Promoter Region

Wikipedia

Remember: All parts remaining in the mRNA are exonic, not just the coding parts.

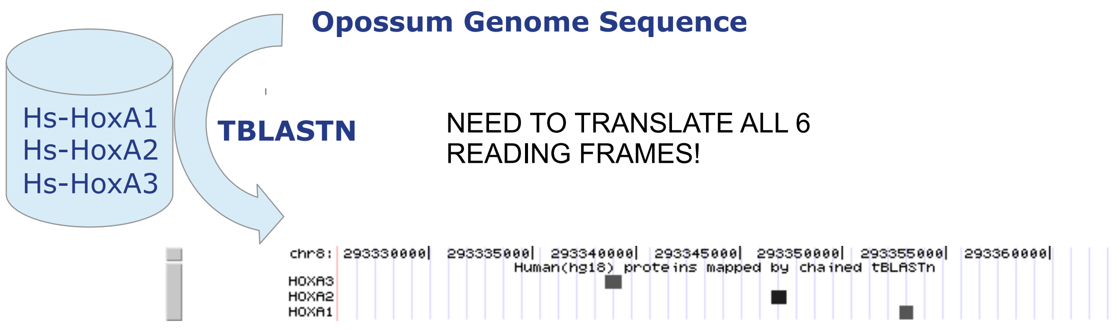
Homology based (extrinsic) methods

Alignment of translated DNA sequence to protein database



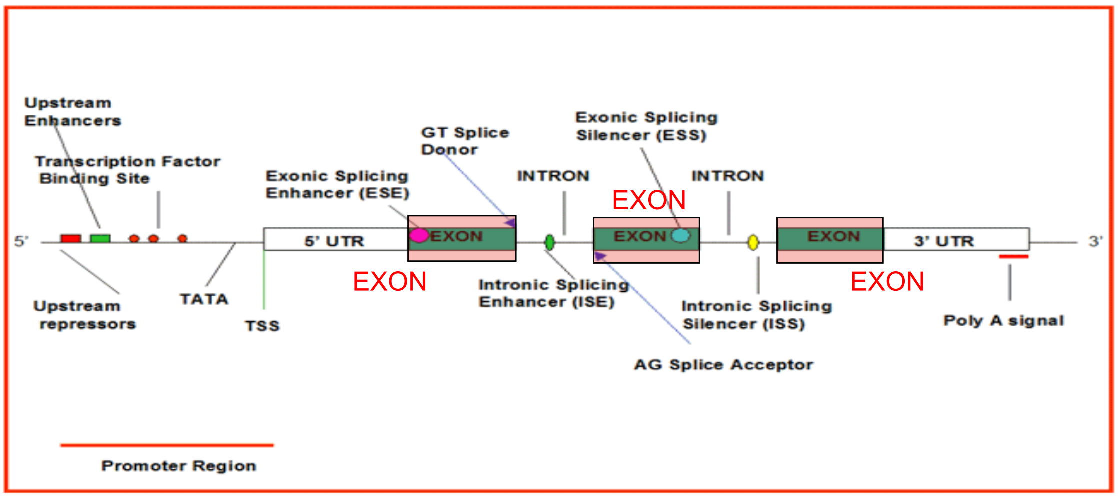
Homology based (extrinsic) methods

Alignment of translated DNA sequence to protein database


Opossum Genome Sequence
Hs-HoxA1
Hs-HoxA2
Hs-HoxA3
TBLASTN
NEED TO TRANSLATE ALL 6 READING FRAMES!
chr8: 293330000| 293335000| 293340000| 293345000| 293350000| 293355000| 293360000|
Human(hg18) proteins mapped by chained tBLASTn
HOXA3
HOXA2
HOXA1

Homology based (extrinsic) methods

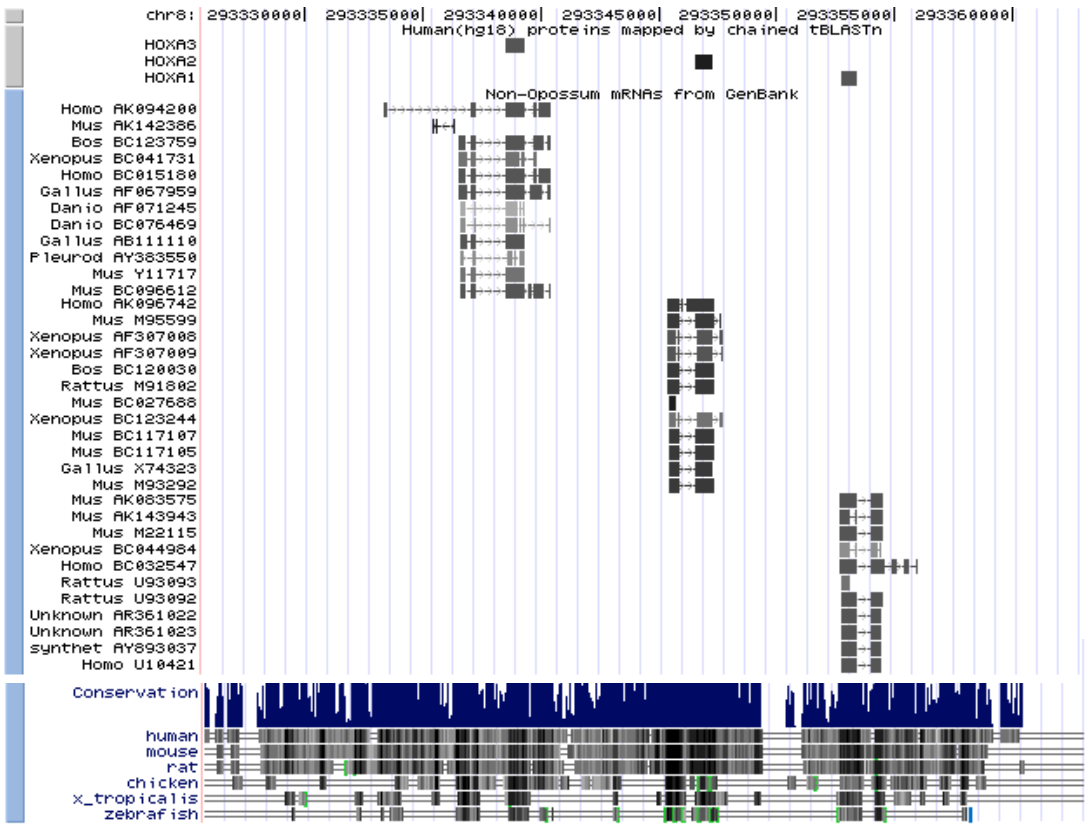
Proteins aligned to opossum genomic DNA:
pink regions depict the optimum that can be achieved


Upstream Enhancers
Transcription Factor Binding Site
Exonic Splicing Enhancer (ESE)
GT Splice Donor
Exonic Splicing Silencer (ESS)
INTRON
INTRON
EXON
5'
5' UTR
EXON
EXON
EXON
3' UTR
3'
EXON
EXON
Upstream repressors
TATA
TSS
Intronic Splicing Enhancer (ISE)
Intronic Splicing Silencer (ISS)
Poly A signal
AG Splice Acceptor
Promoter Region

Ultimate drawback: no information about 5 and 3 UTRs!

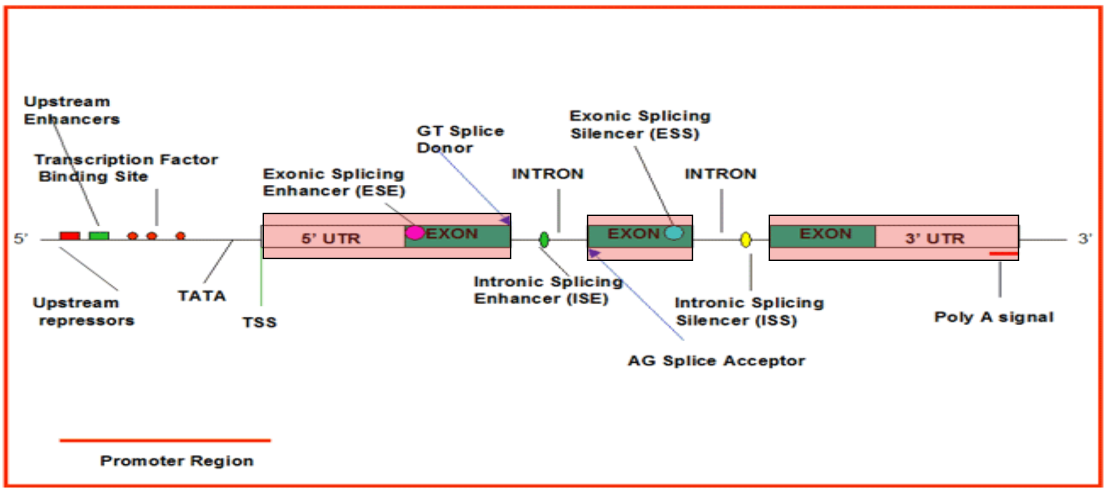
Homology based (extrinsic) methods

Alignment of transcribed sequences (cDNA, ESTs or mRNAs from RNAseq)



Homology based (extrinsic) methods

cDNAs or ESTs aligned to opossum genomic DNA: pink regions depict the optimum that can be achieved

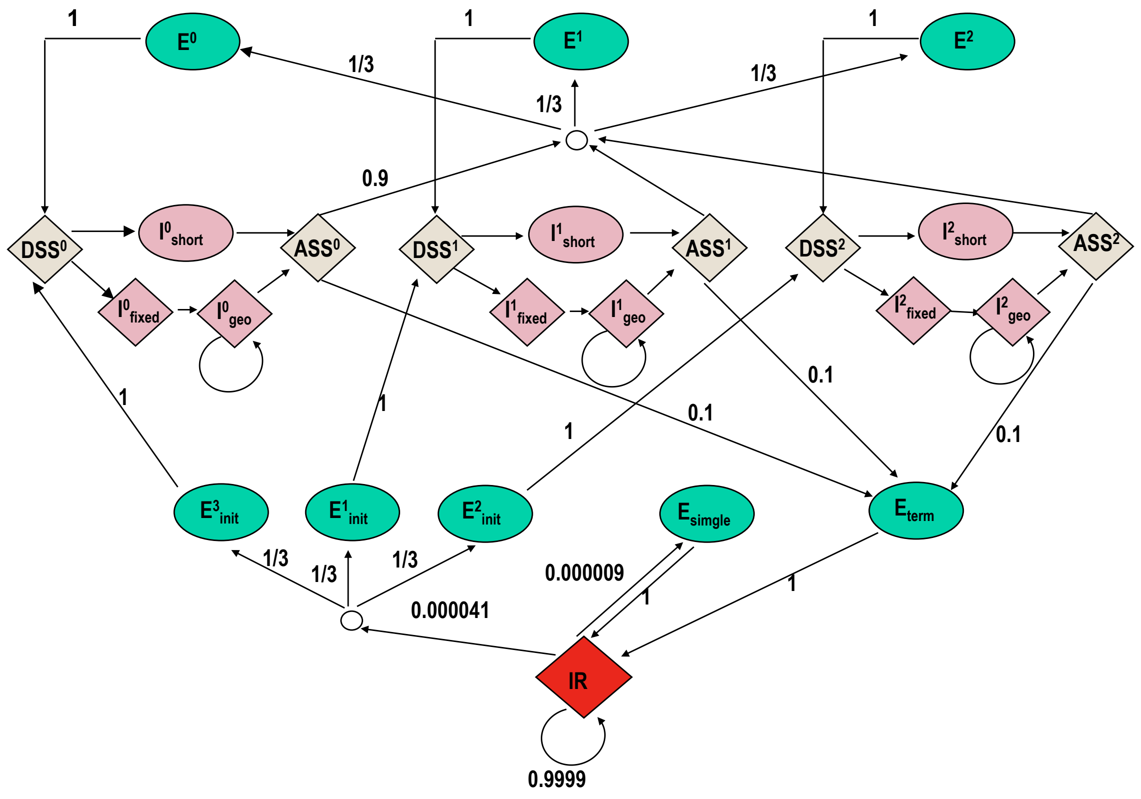

Upstream Enhancers
Transcription Factor Binding Site
GT Splice Donor
Exonic Splicing Silencer (ESS)
Exonic Splicing Enhancer (ESE)
INTRON
INTRON
5'
5' UTR
EXON
EXON
EXON
3' UTR
3'
Upstream repressors
TATA
TSS
Intronic Splicing Enhancer (ISE)
Intronic Splicing Silencer (ISS)
Poly A signal
AG Splice Acceptor
Promoter Region

Ab initio Gene Annotation (intrinsic methods)

- Nucleotide sequences are not random across the genome; instead, there are sequences features that are common among protein-coding genes
- Some of them are:
  - long open reading frames;
  - codon bias;
  - proximity to transcriptional and translational initiation motifs;
  - 3' polyadenylation sites and
  - splicing consensus sequences at putative intron-exon boundaries
- *Ab initio* gene discovery programs recognize such features to identify protein-coding genes. Most popular programs are based on Hidden Markov Models (HMMs)
- Irrespective of which discovery algorithm is used, all computationally identified putative genes must be confirmed by a second line of evidence before being elevated to gene status in the genome annotation

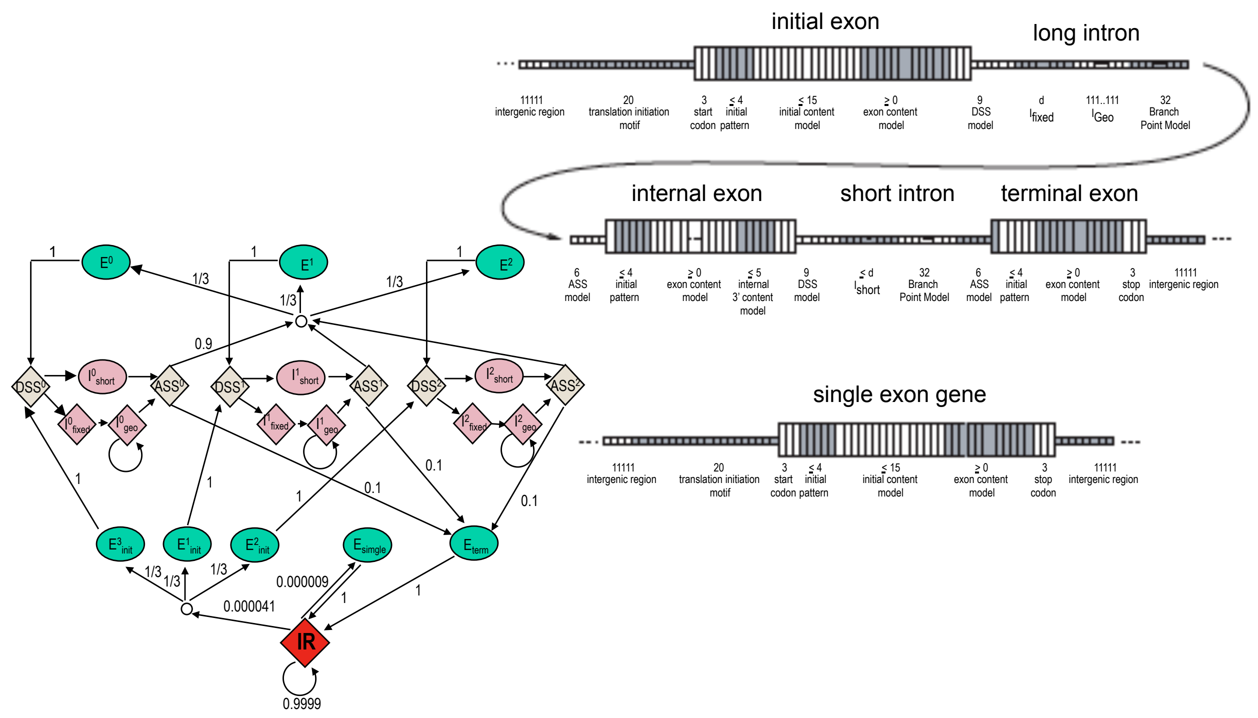
Ab Initio Gene Finding with HMMs

The states of AUGUSTUS and the possible transitions between them.



Modified from Stanke and Waack (2003), Bioinformatics

Ab Initio Gene Finding with HMMs



Modified from Stanke and Waack (2003), Bioinformatics

Gene Prediction: Ultima Ratio

Include ANY information that can point toward the presence of a gene into the prediction procedure:

- similarity to known proteins/expressed sequences (to identify exon candidates)
- similarity to non-coding parts of expressed sequences (to identify UTR candidates)
- presence of putative signal sequences to identify new exons or to refine exon borders
- sequence conservation in intra-specific or inter-specific genome comparisons
- **enrichment of words** that occur preferentially in coding sequences (codon bias, hexamers, etc)
- periodicity in the DNA sequence
- . . .