BIO306: Bioinformatics

Lecture 15

Review

Wenfei JIN PhD
jinwf@sustc.edu.cn
Department of Biology, SUSTech

Outline

- Lecture 1. Introduction of bioinformatics
- Lecture 2. NGS and Reads mapping
- Lecture 3. FM-index
- Lecture 4. Haplotype and linkage disequilibrium
- Lecture 5. Data basics and Data Structure
- Lecture 6. Reads mapping and output
- Lecture 7. Variant calling and output

Outline

- Lecture 8,9. Identifying disease associated variants
- Lecture 10,11. Gene expression profiling and RNA-seq
- Lecture 12. Epienetics and epigenome
- Lecture 13. Gene Ontology and enrichment analysis
- Lecture 14. Population genetics

Lecture 1

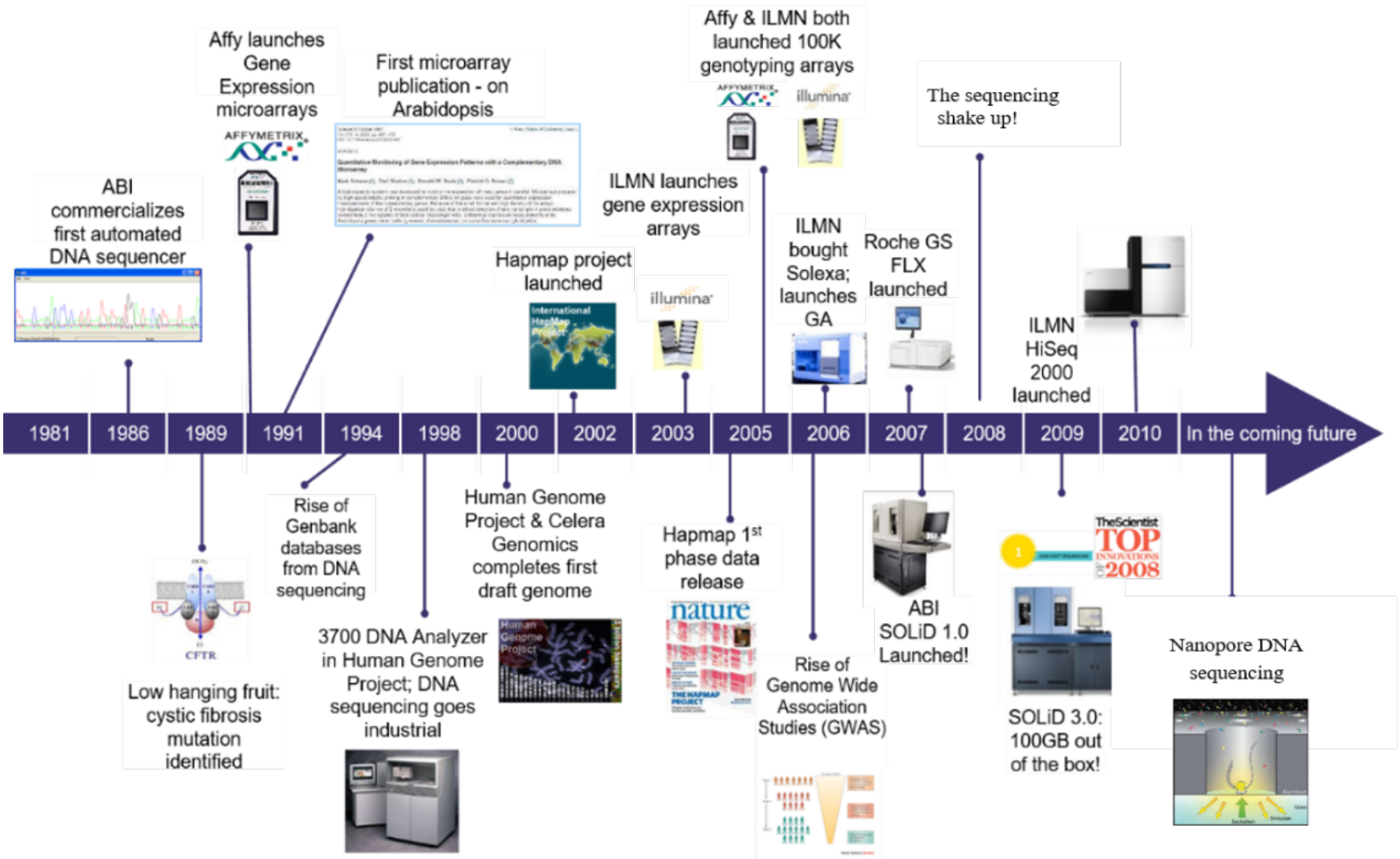
- Emergence of bioinformatics
  - Data Driver
  - Facilitate by development of computer science
- Major topics
  - Sequence analysis
  - Gene and protein expression
  - Bioimaging and structure prediction
  - Network and systems biology

Emergence of bioinformatics

- 1951-1953: Sanger sequenced insulin
- 1960s: Margaret Oakley Dayhoff applied computer to analyze protein data

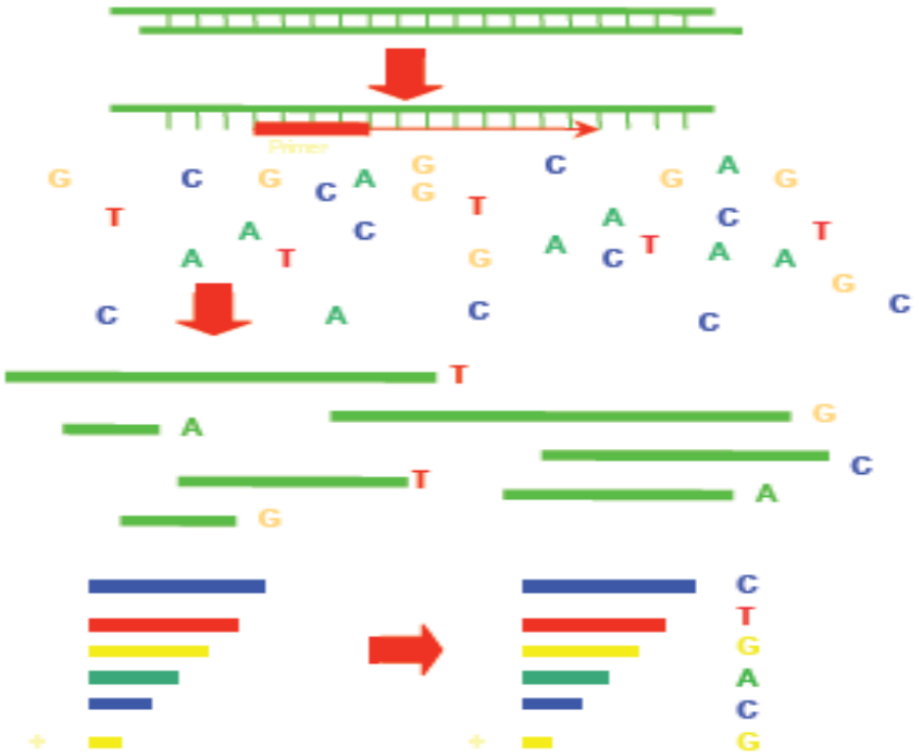
- 1990: BLAST (NCBI)
- 1990s: rise of bioinformatics
  - The protein sequence and Structure
  - Microarray
  - DNA/RNA sequencing

The milestones in genome research

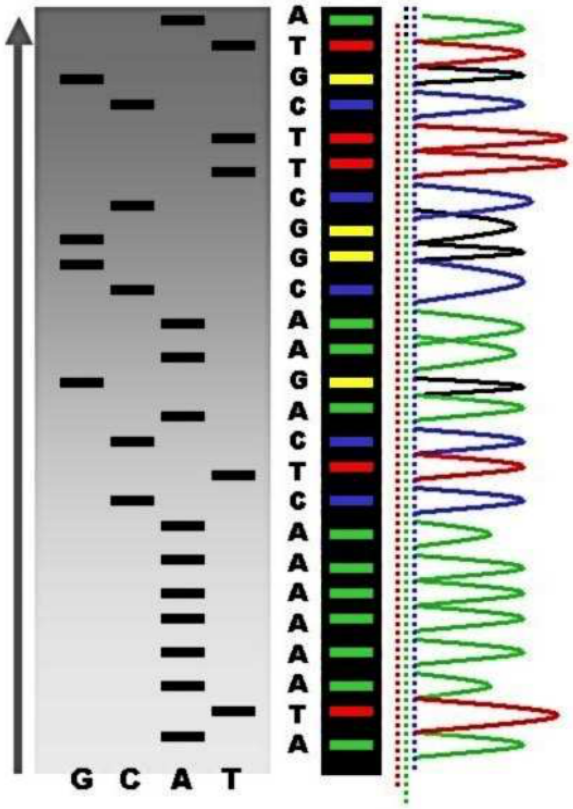


Sanger sequencing: dye-terminator sequencing

1986: 4 Reactions to 1 Lane
fluorescently labelled ddNTPs

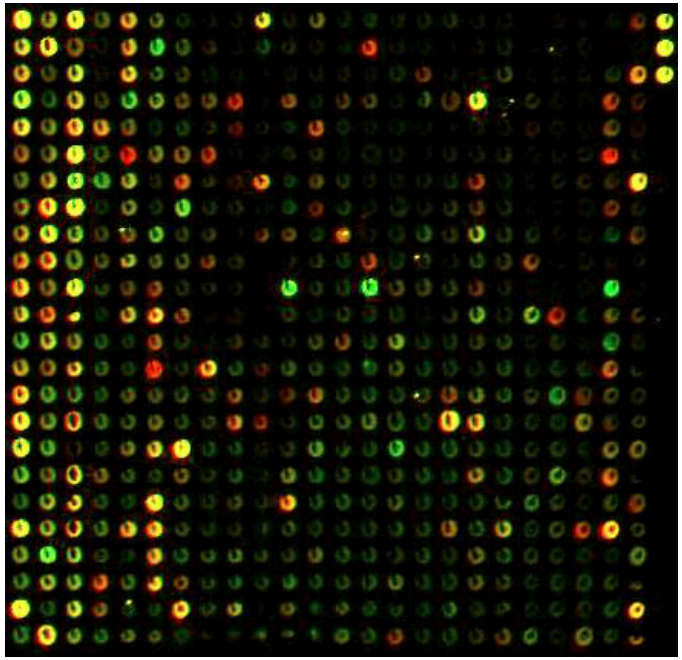


Sequencing Reaction Products

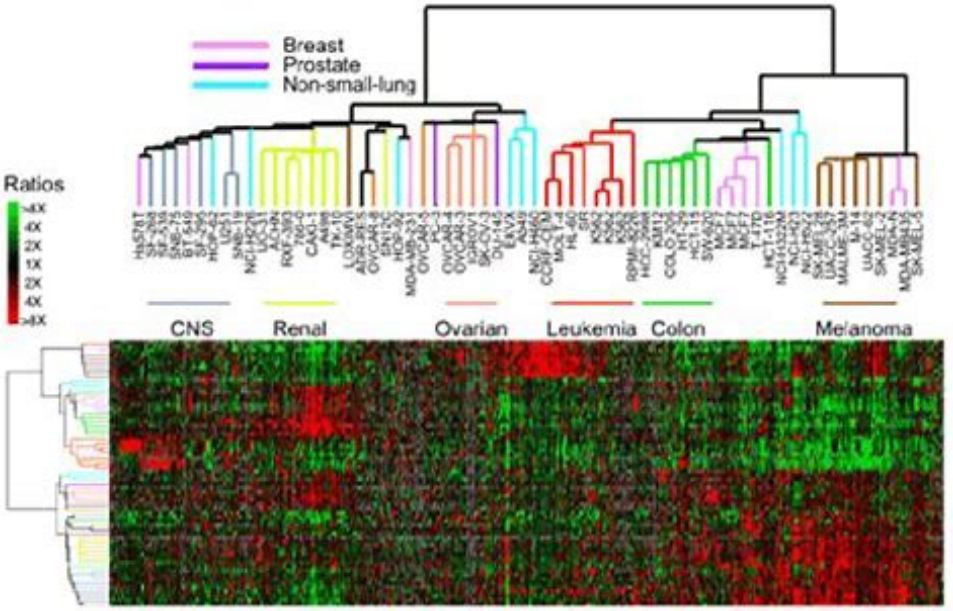


Progression of Sequencing Reaction

Microarray



Microarray image



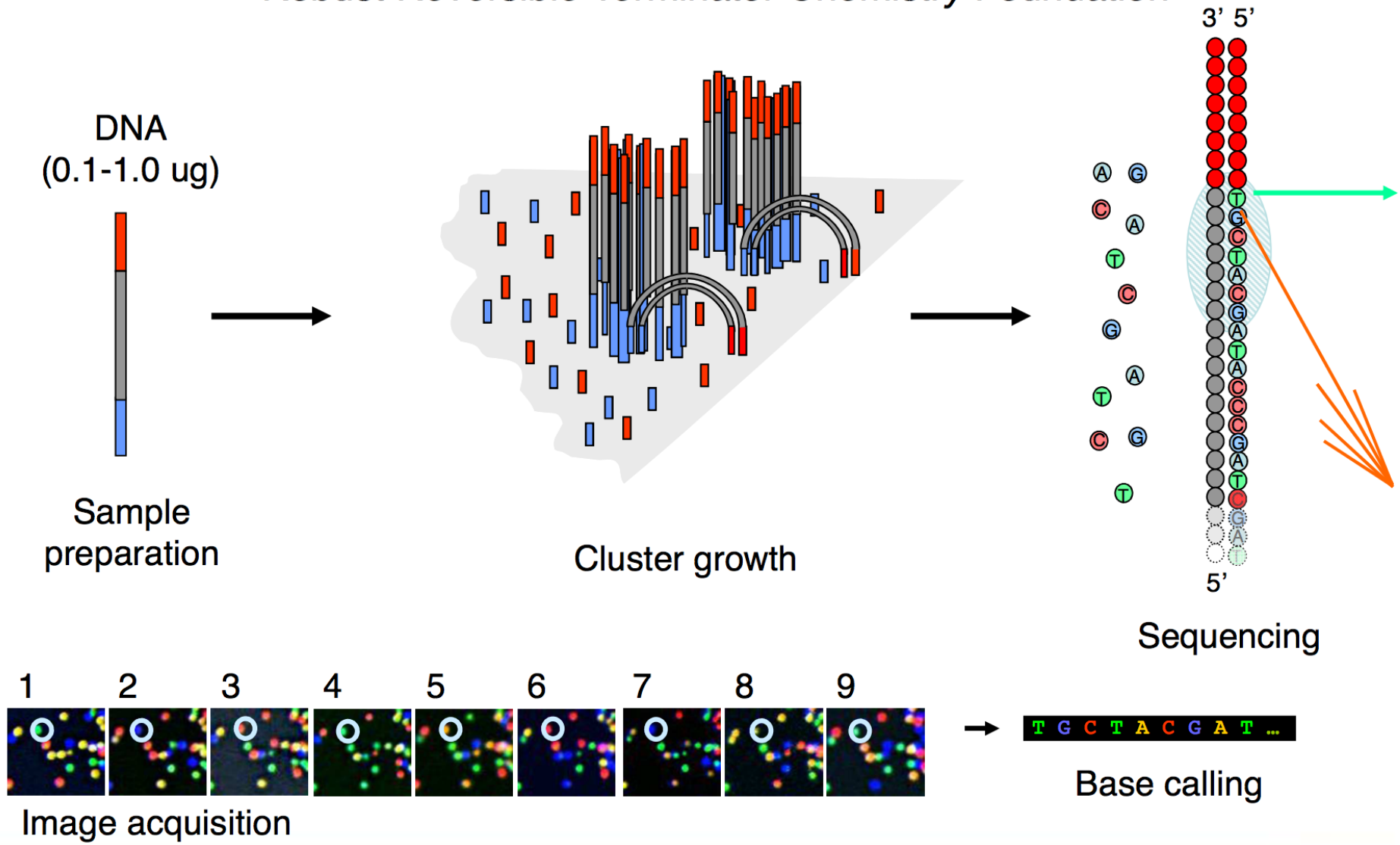
Gene expression clustering

Lecture 2. NGS and Reads mapping

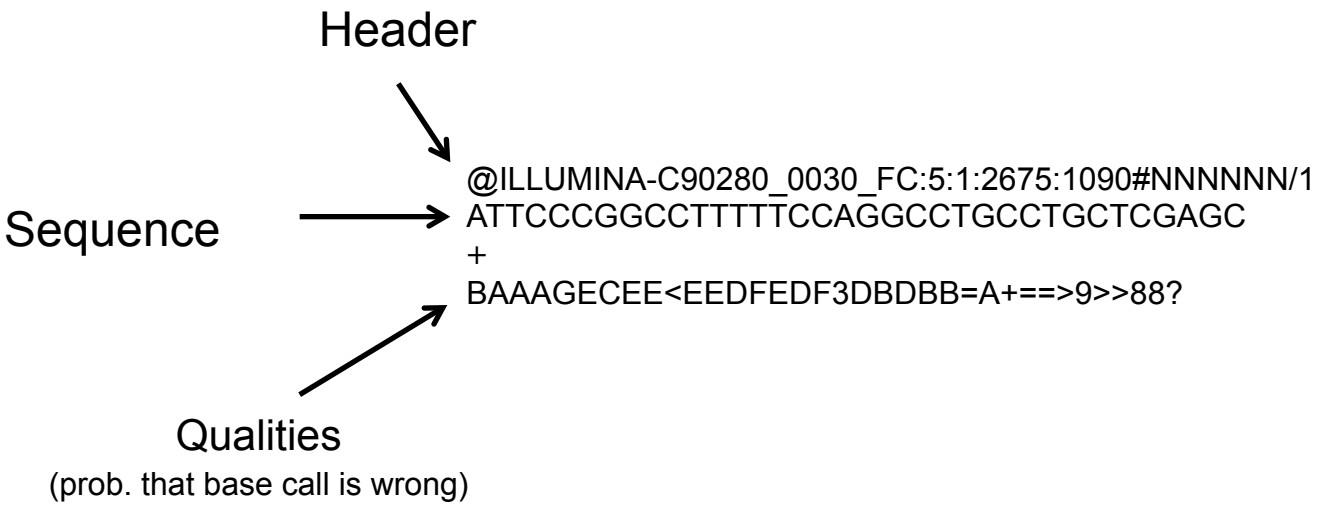
- Next Generation Sequencing (NGS)
- High-throughput sequencing
- Massive parallel sequencing
  - They share the technical paradigm of massive parallel sequencing via spatially separated, clonally amplified DNA templates or single DNA molecules in a flow cell.
  - Two categories (clone amplification and single DNA molecular)

Illumina Sequencing Technology

Robust Reversible Terminator Chemistry Foundation



NGS data



One character encodes a number using ascii table (0-255)

This number (*Q*) can be converted to *P*

Phred-scale

$Q = -10 * \log_{10} P$

P=10^(-Q/10)

Reads mapping

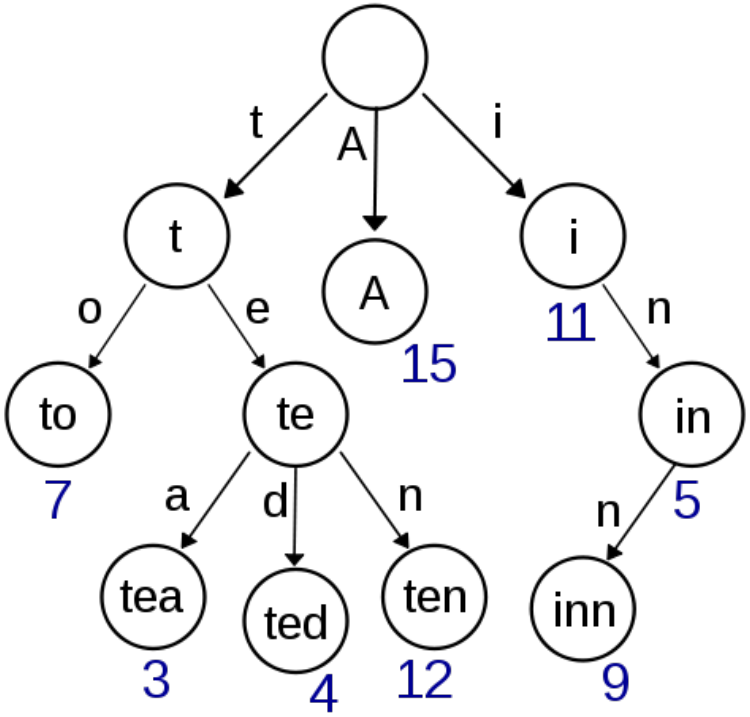
- Adapting hashed seed-extend algorithms to work with shorter reads
  - Improve seed matching sensitivity
    - BLAST, ELAND, SOAP, MAQ, RMAP, ZOOM
    - SSAHA2, BLAT, ELAND2
  - Above and/or Improve speed of local alignment for seed extension
    - Shrimp2, CLCBio
- Uses a Trie structure to search a reference sequence
  - BWA, Bowtie, SOAP2

Trie structure

- A trie (pronounced “try”) is a search tree—an ordered tree data structure that is used to store a dynamic set or associative array where the keys are usually strings.
  - Trie (prefix Tree)
  - Suffix Tree
  - Suffix Array

Trie

- Smallest tree such that:
  - Each edge is labeled with a character $c \in \Sigma$
  - A node has at most one outgoing edge labeled c, for $c \in \Sigma$
  - Each key is "spelled out" along some path starting at the root
- Natural way to represent either a *set* or a *map* where keys are strings



Advantage of Trie

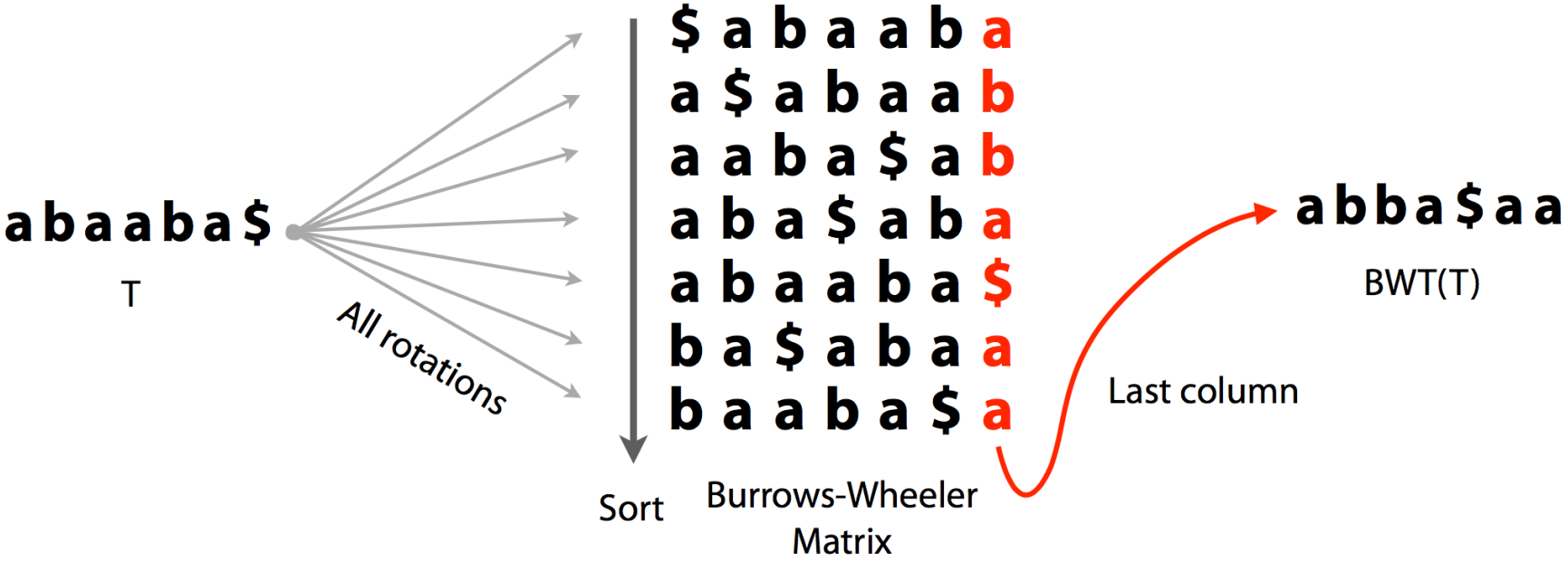
- Looking up data in a trie is faster in the worst case, O(m) time (where m is the length of a search string), compared to an imperfect hash table.
- There are no collisions of different keys in a trie.
- Buckets in a trie, which are analogous to hash table buckets that store key collisions, are necessary only if a single key is associated with more than one value.
- There is no need to provide a hash function or to change hash functions as more keys are added to a trie.
- A trie can provide an alphabetical ordering of the entries by key.

Burrows–Wheeler transform（BWT）

- What is BWT?

Burrows–Wheeler transform（BWT)

- Block-sorting compression
- BWT rearranges a character string into runs of similar characters.
- Rotation->sorting->last column



Lecture 3&5. Data structure and FM-index

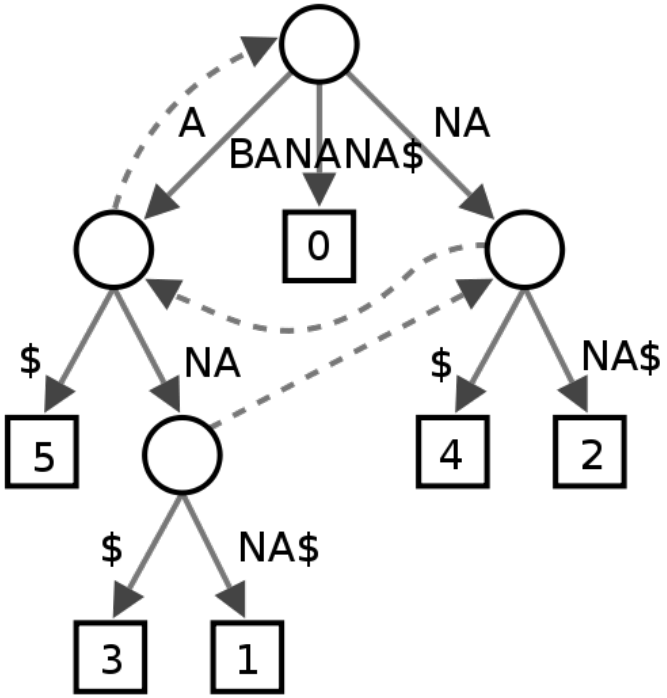
- Suffix Tree?
- Suffix array?
- How BWT is reversible?
- How is it FM index?

Suffix tree

Compressed trie containing all the suffixes of the given text as their keys and positions in the text as their values. Suffix trees allow particularly fast implementations of many important string operations.

Suffix tree

- The suffix tree for the string S of length n is defined as a tree such that
  - The tree has exactly n leaves numbered from 1 to n.
  - Except for the root, every internal node has at least two children.
  - Each edge is labeled with a non-empty substring of S.
  - No two edges starting out of a node can have string-labels beginning with the same character.
  - The string obtained by concatenating all the string-labels found on the path from the root to leaf i spells out suffix S[i..n], for i from 1 to n.



Features of suffix tree

- The construction of such a tree for the string *S* takes time and space linear in the length of S.
- Once constructed, several operations can be performed quickly, for instance locating a substring in S, locating a substring if a certain number of mistakes are allowed, locating matches for a regular expression pattern etc.
- Suffix trees also provide one of the first linear-time solutions for the longest common substring problem.
- These speedups come at a cost: storing a string's suffix tree typically requires significantly more space than storing the string itself.

Suffix array

- a suffix array is a sorted array of all suffixes of a string.
  - a data structure used, among others, in full text indices, data compression algorithms and within the field of bibliometrics.
  - as a simple, space efficient alternative to suffix trees

Definition of suffix array

- Let S=S[1]S[2]...S[n] be a string and let S[i,j] denote the substring of S ranging from i to j.

- The suffix array A of S is now defined to be an array of integers providing the starting positions of suffixes of S in lexicographical order. This means, an entry A[i] contains the starting position of the i-th smallest suffix in S and thus for all $1<I \leq n$: $S[A[i-1],n]<S[A[i],n]$.

Example of suffix array(1)

Consider the text S= `banana$` to be indexed:

i	1	2	3	4	5	6	7
$S[i]$	b	a	n	a	n	a	$

Suffix	i
banana$	1
anana$	2
nana$	3
ana$	4
na$	5
a$	6
$	7

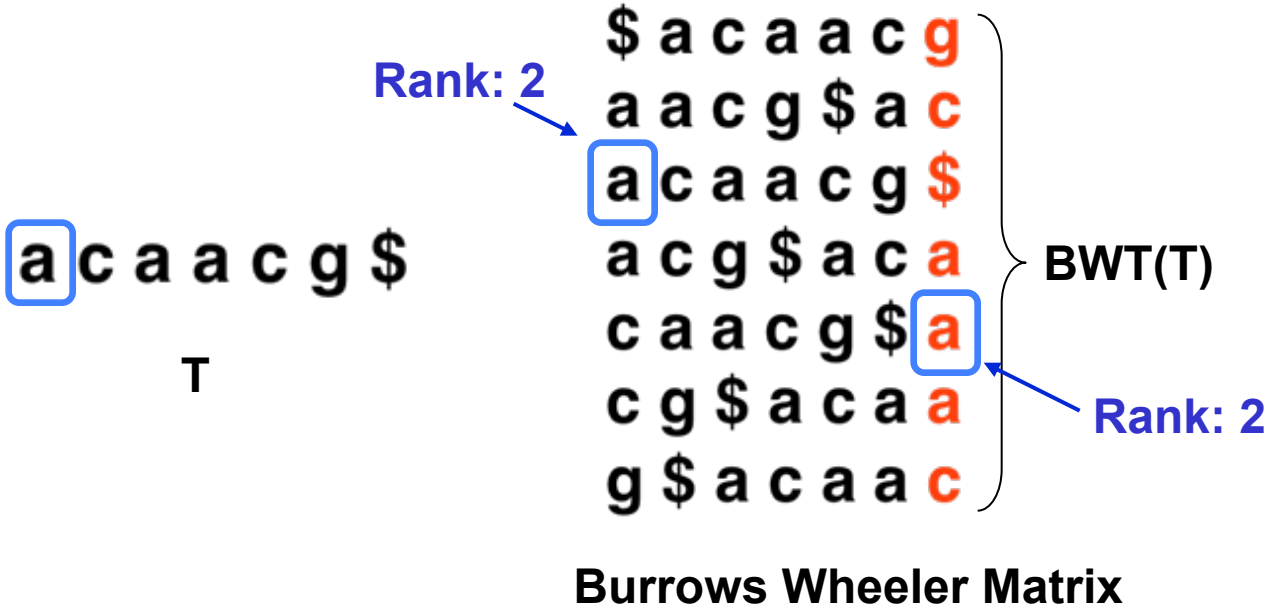
Suffix	i
$	7
a$	6
ana$	4
anana$	2
banana$	1
na$	5
nana$	3

The text ends with the special sentinel letter $ that is unique and lexicographically smaller than any other character. The text has the following suffixes.

These suffixes can be sorted in ascending order.

LF Mapping

- Property that makes BWT(T) reversible is "LF Mapping"
  - i^{th} occurrence of a character in Last column is same text occurrence as the i^{th} occurrence in First column



BWT reversing

Reverse BWT(T) starting at right-hand-side of *T* and moving left

Start in first row. *F* must have **$**. *L* contains character just prior to **$**: $\mathbf{a_0}$

$\mathbf{a_0}$: LF Mapping says this is same occurrence of **a** as first **a** in *F*. Jump to row *beginning* with $\mathbf{a_0}$. *L* contains character just prior to $\mathbf{a_0}$: $\mathbf{b_0}$.

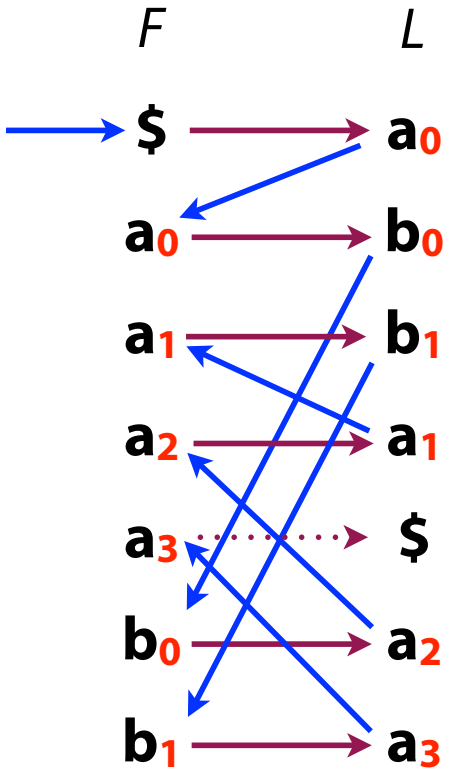
Repeat for $\mathbf{b_0}$, get $\mathbf{a_2}$

Repeat for $\mathbf{a_2}$, get $\mathbf{a_1}$

Repeat for $\mathbf{a_1}$, get $\mathbf{b_1}$

Repeat for $\mathbf{b_1}$, get $\mathbf{a_3}$

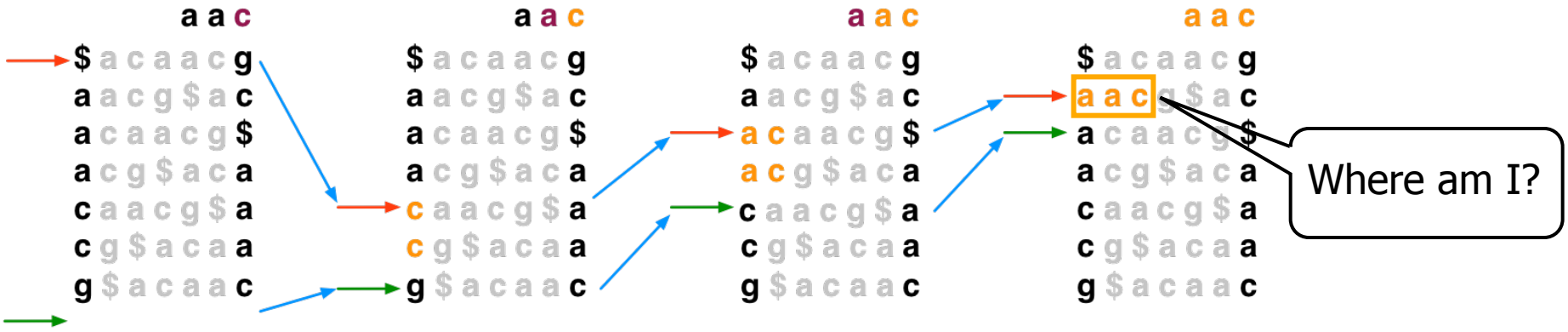
Repeat for $\mathbf{a_3}$, get **$**, done



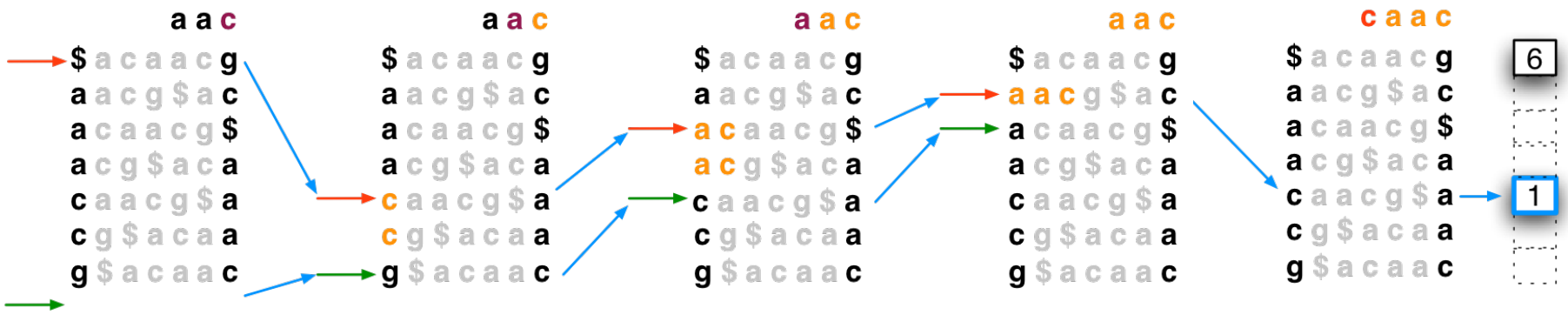
Reverse of chars we visited = $\mathbf{a_3\ b_1\ a_1\ a_2\ b_0\ a_0\ \$}$ = *T*

FM Index

- FM Index: an index combining the BWT *with a few small auxilliary data structures*
- "FM" supposedly stands for "Full-text Minute-space."



Reads mapping



- Algorithm concludes: “aac” occurs at offset 2 in “acaacg”

Lecture 4. Haplotype and linkage disequilibrium

- What is Haplotype?
- How to construct haplotypes?
- What is linkage disequilibrium?
- What is complete LD?
- What is perfect LD?
- What is LD block?

- Haplotype is a group of alleles in an organism that are inherited together from a single parent

- Linkage disequilibrium is the non-random association of alleles at different loci in a given population

Lecture 6. Reads mapping and output

- Reads depth
- Mapping Quality (MAPQ)
- Sequence alignment/Map format (SAM)
- samtools

Coverage/Depth

- Coverage/depth is how many times that your data covers the genome (on average)

$$C=N*L/G$$

- Example:
  - N : number of reads: 10M
  - L: Read length: 50
  - G: Genome size: 5 M bases
  - C = 10*50/5 = 100X

- On average there are 100 reads covering each position in the genome

Column 5: MAPQ - Mapping Quality

- Phred score, identical to the quality measure in the fastq file. quality **Q**, probability **P**:

$$P = 10 \wedge (-Q / 10.0)$$

If **Q=30**, **P=1/1000** → on average, one of out 1000 alignments will be wrong

As good as this sounds it is not easy to compute such a quality.

$$Q=-10\log_{10}P$$

Lecture 7. Variant calling and output

- Genetic variants
  - Single nucleotide variant (SNV)
  - Structure Variations
    - Copy number variation (CNV)
- Variants calling
  - Error from library prep (PCR artifacts)
  - Machine sequencing errors
  - Misalignment (mapping error)
  - Error in reference sequence

Simple approaches to variant calling: Varscan

- If the genotypes do not match, then their read counts are evaluated by one-tailed Fisher's exact test in a two-by-two table

	Reference	alternate
Tumor reads	Tumor reads 1	Tumor reads 2
Normal reads	Normal reads 1	Normal reads 2

- Fisher exact test is performed. If the P value is significant then
  - if the normal sample is called reference and the tumor sample is called alternate, then the variant is called somatic

The use of MAP in MAQ

Recall from the lecture that we call the posterior probabilities of the three genotypes given the data D, that is a column with n aligned nucleotides and quality scores of which k correspond to the reference a and $n-k$ to a variant nucleotide b.

$$
\begin{aligned}
p(G = \langle a,a \rangle | D) &\propto p(D|G = \langle a,a \rangle) p(G = \langle a,a \rangle) \\
&\propto \alpha_{n,k} \cdot (1-r)/2 \\
p(G = \langle b,b \rangle | D) &\propto p(D|G = \langle b,b \rangle) p(G = \langle b,b \rangle) \\
&\propto \alpha_{n,n-k} \cdot (1-r)/2 \\
p(G = \langle a,b \rangle | D) &\propto p(D|G = \langle a,b \rangle) p(G = \langle a,b \rangle) \\
&\propto \binom{n}{k} \frac{1}{2^n} \cdot r
\end{aligned}
$$

MAQ: Consensus Genotype Calling

A major problem in SNV calling is false positive heterozygous variants. It seems less probable to observe a heterozygous call at a position with a common SNP in the population. For this reason, MAQ uses a different prior (r) for previously known SNPs ($r = 0.2$) and "new" SNPs ($r = 0.001$).

Let us examine the effect of these two priors on variant calling. In R, we can write

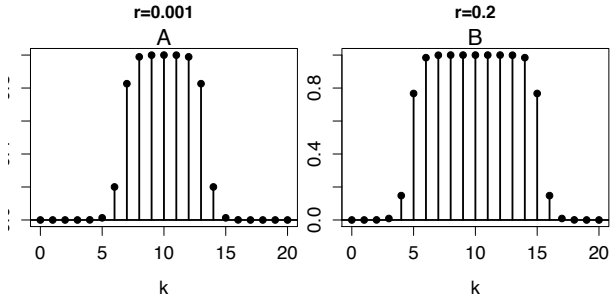
$$p(G = \langle a, b \rangle | D) \propto \binom{n}{k} \frac{1}{2^n} \cdot r$$

as

```
> dbinom(k,n,0.5) * r
```

where k is the number of non-reference bases, n is the total number of bases, and 0.5 corresponds to the probability of seeing a non-ref base given that the true genotype is heterozygous.

MAQ: Consensus Genotype Calling



- The figures show the posterior for the heterozygous genotype according to the simplified MAQ algorithm discussed in the previous lecture
- The prior $r = 0.0001$ means than positions with 5 or less ALT bases do not get called as heterozygous, whereas the prior with $r = 0.2$ means that positions with 5 bases do get a het call

Lecture 8,9. Identifying disease associated variants

- Gene mapping/Genetics association
- Common disease-common variant (CD-CV)
- Methods for identifying disease associated variants
- Challenges in sequencing-based genome-wide association study

Lecture 10,11. Gene expression profiling and RNA-seq

- What's the advantage of RNA-seq compared with microarray?
- What factors should we consider for RNA-seq data normalization?
- What's the advantage of single cell sequencing over bulk cells?

Lecture 12 epigenetics and epigenome

- Definition of epigenetics?
- How to detect genome-wide DNA methylation?
- How to detect genome-wide nucleosome positioning and chromatin accessibility?
- How to identify genome-wide TF binging sites? How to do the peak calling?
- What is Hi-C? How to identify the significant interaction ?

Lecture 13 Gene Ontology and enrichment analysis

- Gene ontology (GO) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species.

- GO is the framework for the model of biology. The GO defines concepts/classes used to describe gene function, and relationships between these concepts. It classifies functions along three aspects:

Lecture 13 Gene Ontology and enrichment analysis

- How to detect GO/pathway enrichment?

Lecture 14 Population genomics

- Effective population size (N_e)
  - The effective population size (N_e) is defined as the number of breeding individuals in an idealized population that shows the same allele frequency spectrum as the population under consideration.
- The main processes influencing allele frequencies
  - Genetic drift
  - Mutation
  - Natural selection
  - Gene flow

Lecture 14 Population genomics

- Exemplify one approach for investigate population structure/history
- Exemplify one approach for natural selection detection
- Exemplify one natural selection event including detection of the event

Statistic for the genome-wide detection of natural selection

1. Divergence rate and phylogenetic shadowing
2. Changed function-altering mutation, e.g., d_N/d_S or K_N/K_S
3. Polymorphism deviating from interspecies divergence e.g. Hudson-Kreitman-Aguade (HKA) test and McDonald-Kreitman (MK) test
4. Changed allele frequency spectrum e.g., Tajima's D
5. Increased derived allele frequencies
6. Extended haplotype homozygosity (EHH), e.g., iHS
7. Locus-specific population differentiation, e.g., F_{ST}
8. Biased ancestry contribution in admixed population.
9. Composite strategies. e.g. combine multiple factors and Likelihood-ratio test

Thank you for your attention!