# CS481/CS583: Bioinformatics Algorithms

Can Alkan
EA509
calkan@cs.bilkent.edu.tr

**http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/**

# KMER DATA STRUCTURES

# Baseline problem

**In-memory representation of a large set of short k-mers:**

**e.g.**

`ACTGAT`

`GTATGC`

`ATTAAA`

`GAATTG`

. . .

***Slide by Rayan Chikhi***

# Applications

- Genome assembly
- Error-correction of DNA sequencing data
- Detection of similarity between sequences
- Detection of distances between datasets
- Alignment
- Pseudoalignment / quasi-mapping
- Detection of taxonomy
- Indexing large collections of sequencing datasets
- Quality control
- Detection of events (e.g., SNPs, indels, CNVs, alt. transcription)
- ...

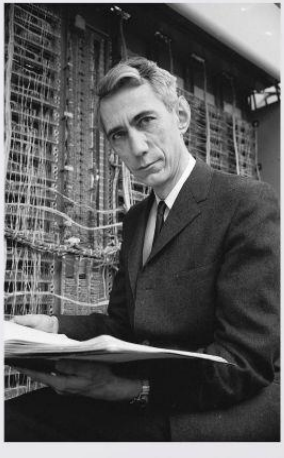
***Slide by Rayan Chikhi***

# k-mers

**Sequences of k consecutive letters**, e.g. ACAG or TAGG for k=4

N.G. de Bruijn (1946), de Bruijn sequences [1]

C. Shannon (1948), information theory [2]



*Slide by Rayan Chikhi*

# Framing the problem

**Large set of k-mers : $10^6$ - $10^{11}$ elements**

**k in [11; $10^3$]**

*Slide by Rayan Chikhi*

# Operations to support

- **Construction** (from a disk or stream)
- **Membership** ("is X in the set?")
- **Iteration** (enumerate all elements in the set)
- ...

**Extensions:**

- Associate value(s) to k-mers (e.g. abundance)

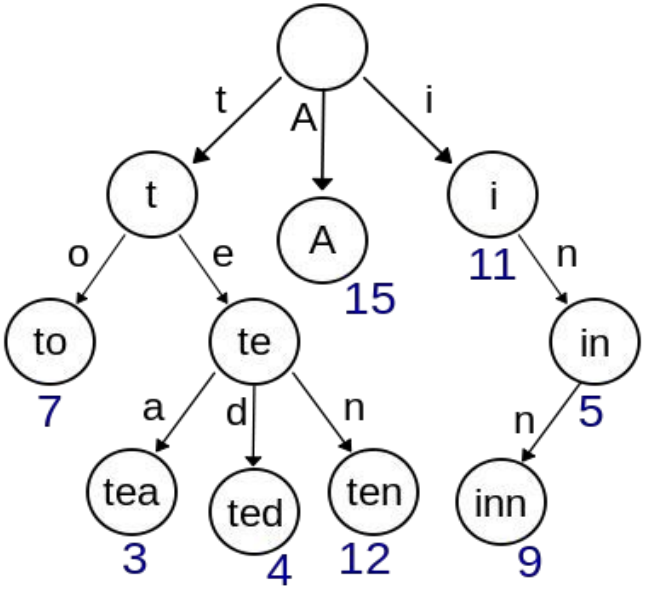
- Navigate the de Bruijn graph

*Slide by Rayan Chikhi*

# Membership test: Tries

**Worst case**

- **O(k) insertion**
- **O(k) deletion**
- **O(k) search**




**Also supports indexing**

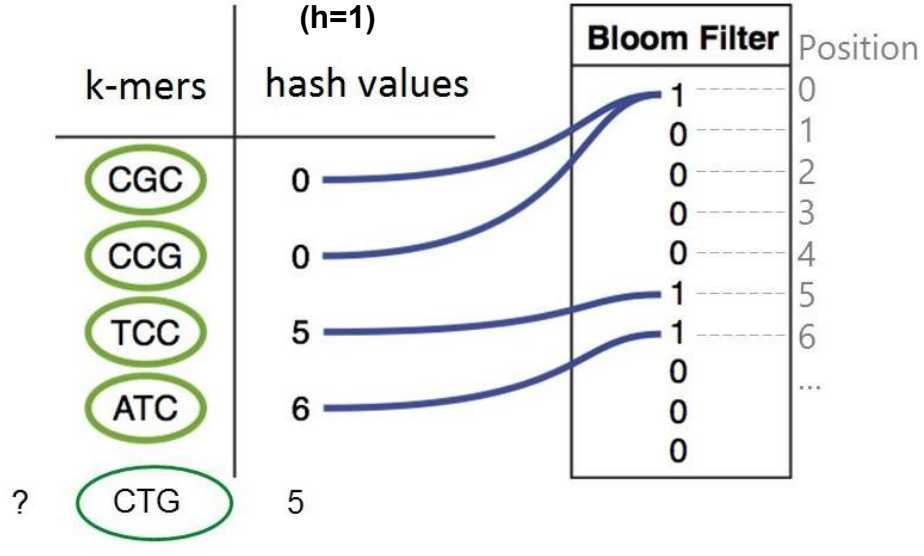
*Slide by Rayan Chikhi*

# Membership test: Bloom filters

**Init a bit array**
**Take h hash functions**
**Insertion: put 1's at positions given by hash functions**

**Query: are there 1's at all positions given by hash functions?**

- **O(hk) insertion**
- **(O(hk) deletion)**
- **O(hk) query**



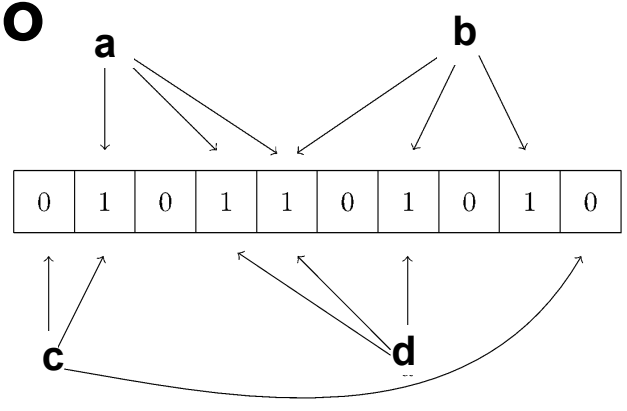

*Slide by Rayan Chikhi*

# Bloom filter

- Bloom filter encodes a set of k-mers
- Uses a bit array B of length m and d hash functions
  - to insert x, we set $B[h_i(x)] = 1$, for i=1,…,d
  - to query y, we check if $B[h_i(y)]$ all equal 1, for i=1,…,d
- Need an estimate for n, the number of k-mers to insert

# Bloom filter example

- a and b **are inserted in to a Bloom filter with m = 10, n=2, d=3**
- c is not in the set, since **some bits are 0**
- d has not **been inserted, but is still reported in the set, a false positive**
- Bloom filters have no false negatives



# Bloom filter

- Storing n k-mers in m bit array with d hash functions has a false positive rate of

$$\approx(1-e^{-d\ n/m})^d$$

- Given n and m, the optimal d is $\approx m/n \ln(2)$
- Example m = 8n, d=5 gives 2.16% fpr
  m = 6n, d=4 gives 5.6% fpr
  m = 4n, d=3 gives 14.6% fpr
- m=8n, corresponds to storing 1 byte per k-mer

**Space: m = 1.44n log2(ε) where ε is the false positive rate**

# Counting k-mers

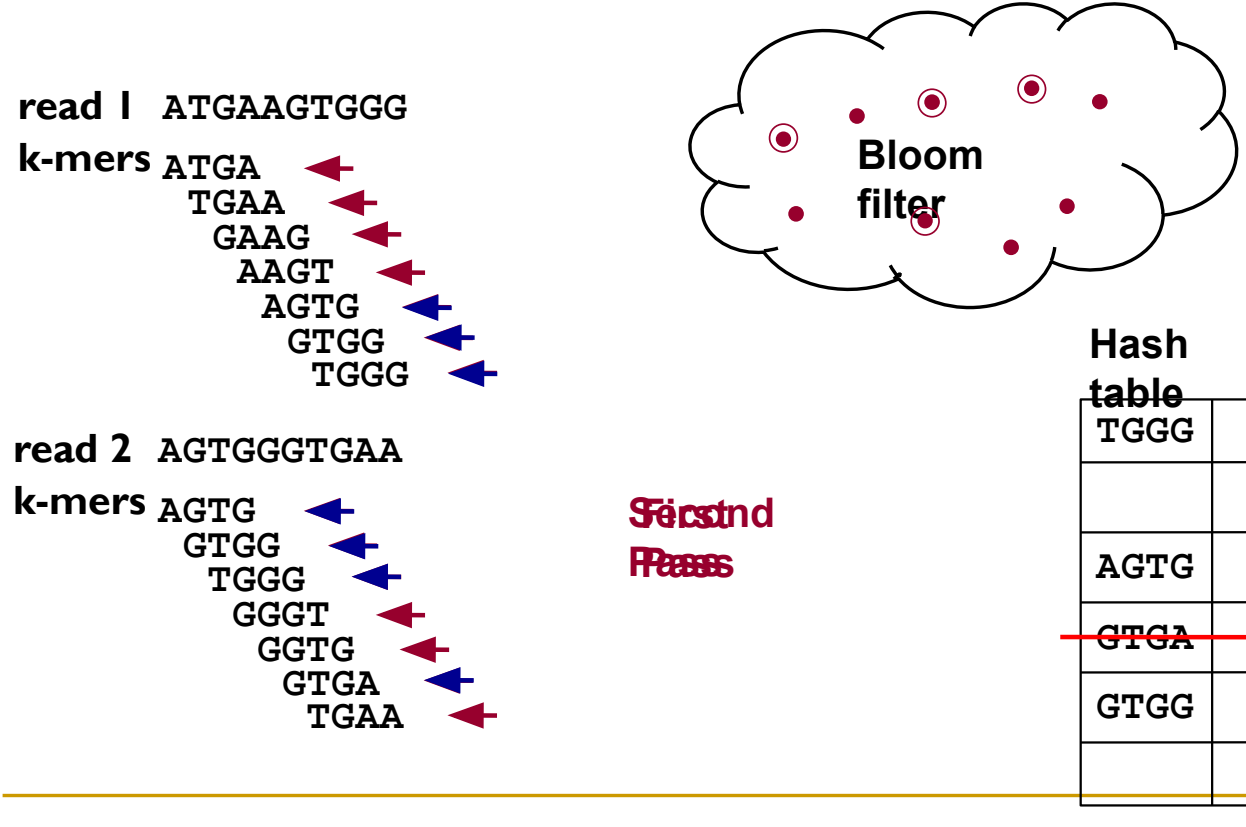
- Simple method

  Store each k-mer in a
  hash table with a counter

- Memory needed
  - store canonical k-mers
  - 2 bits for each of A,C,G,T
  - k/4 bytes per k-mer (k=31, 8 bytes)
  - 1-2 bytes per counter
  - +10% hash table overhead
- For a genome of size G, expect to see up to G distinct k-mers (2.5-3 billion for Human)
- ~ 36 Gb of memory

# Memory-efficient k-mer counting

- Use a Bloom filter and a hash table



# Algorithm

- This scheme guarantees
  - k-mers seen twice will be in the hash table
  - some unique k-mers will slip through
  - second pass gives accurate counts and allows to discard false positives
- Memory usage
  - full for k-mers in hash table (~ 9 bytes)
  - minimal for k-mers in bloom filter (~ .5-1 bytes)

# Results whole genome

- 25-mers in 36 bp reads
- 2.37 billion distinct 25-mers in hg18
- 12.18 billion 25-mers in the sequencing data
  - 9.35 billion unique
  - 2.83 billion with coverage 2 or greater

| Program | Time (hrs) | Memory (G) |
|---|---|---|
| BFCounter | 23.82 | 42 |
| Naïve | > 26.83 | >128 |

# Approximate membership test: Counting Quotient Filter

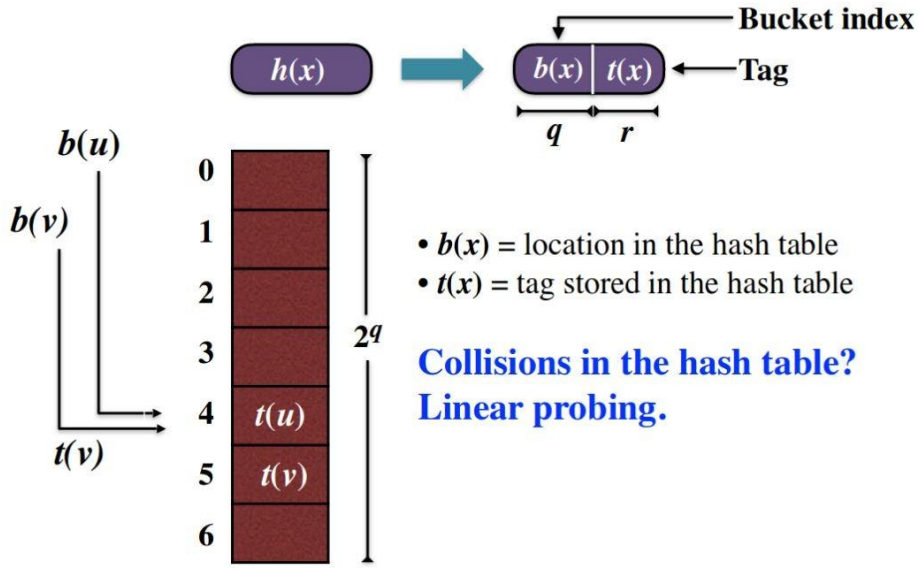



Fig: P. Pandley, SIGMOD 2017
https://pdfs.semanticscholar.org/6bde/f4a86108309086de4071c9d28d97565a84a4.pdf

**Hybrid between a *compact hash table* and a Bloom Filter.**

**Approximate membership**

- **O(k) insertion**
- **O(k) deletion**
- **O(k) query**

Pandley, Bender, Johnson, Patro, *SIGMOD 2017*

***Slide by Rayan Chikhi***

# Multi-set Membership Test: Sequence Bloom Trees

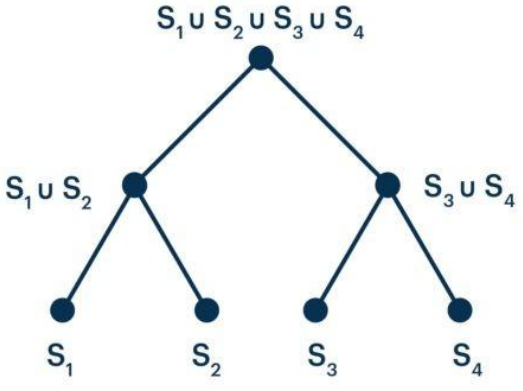



Fig: https://www.sevenbridges.com/sequence-bloom-trees-principles/

**Application:** fast sequence search in 1000's of RNA-seq experiments

**Leaf**: Bloom Filter of a sequencing dataset
**Internal nodes**: Bitwise union of children BF's

- Representation of sets of k-mer sets
- Approximate membership across all datasets in O(hits) instead of O(datasets)
- No k-mer iteration
- Insertion/deletion of complete datasets
- Whole structure resides on disk

Solomon, Kingsford, *Nat Biotech 2017*
Sun, Harris, Chikhi, Medvedev, *RECOMB 2017*
Solomon, Kingsford, *RECOMB 2017*

***Slide by Rayan Chikhi***

# Multi-set Membership Test: Bloom Filter Tries

Add *gcgccaggaatc* -> BURST

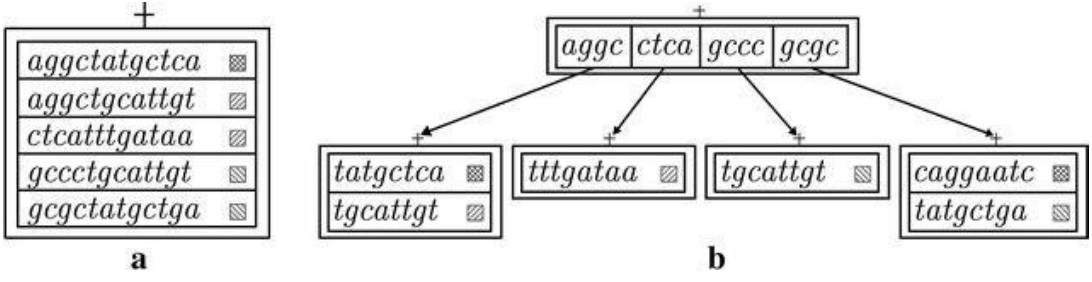



Fig:BFT article

**Principle: cut k-mers into chunks, insert in a burst trie, Bloom Filters added for speed**

- **Representation of sets of k-mer sets**
- **Tailored to pan-genomes: a single k-mer belongs to many sets**
- **Explicit dBG operations support**

**Application: indexing and compression of pan-genomes**

Holley, Wittler, Stoye, *WABI 2016*

**Alternative: colored de Bruijn graphs**

***Slide by Rayan Chikhi***

# De Bruijn Graphs

- *n*-dimensional directed graph of *m* symbols
  - $m^n$ vertices: all possible length-*n* sequences of *m* symbols
  - Edges between vertices *v* and *w* if *sequence(w)* can be generated by *shifting sequence(v)* by one character and add one new character
  - $S = \{s_1, s_2, \ldots, s_m\}$
  - $V = S^n = \{(s_1, \ldots, s_1, s_1), (s_1, \ldots, s_1, s_2), \ldots, (s_m, \ldots, s_m, s_m)\}$
  - $E = \{((v_1, v_2, \ldots, v_n), (w_1, w_2, \ldots, w_n)): v_2=w_1, v_3=w_2, \ldots, v_n=w_{n-1}\}$

# De Bruijn Graph for DNA Assembly

- $m = 4$ (A, C, G, T)
- $n = k$ (k-mer size)
- $4^k$ *potential* vertices
  - In reality if *k* is sufficiently large, upper bound is genome size
  - Twin vertices: vertices with sequences that are reverse-complement of each other
    - *AAAA* twin of *TTTT*

# De Bruijn Assemblers

- Currently the most common for short read seq: Euler, ALLPATHS-LG, Velvet, ABySS, SOAPdenovo
- Divide reads into k-mers
  - Build graph from k-mers
    - Put an edge if there is k-1 bp prefix-suffix match
  - Error correction
  - Eulerian path
- The first parts (graph construction & correction) is essentially common to all these assemblers, with a few implementation differences (e.g., parallelization/distributed computing in ABySS)

# A quick example

**TAGTCGAGGCTTTAGATCCGATGAGGCTTTAGAGACAG**

AGTCGAG CTTTAGA CGATGAG CTTTAGA
GTCGAGG TTAGATC ATGAGGC GAGACAG
GAGGCTC ATCCGAT AGGCTTT GAGACAG
AGTCGAG TAGATCC ATGAGGC TAGAGAA
TAGTCGA CTTTAGA CCGATGA TTAGAGA
CGAGGCT AGATCCG TGAGGCT AGAGACA
TAGTCGA GCTTTAG TCCGATG GCTCTAG
TCGACGC GATCCGA GAGGCTT AGAGACA
TAGTCGA TTAGATC GATGAGG TTTAGAG
GTCGAGG TCTAGAT ATGAGGC TAGAGAC
AGGCTTT ATCCGAT AGGCTTT GAGACAG
AGTCGAG TTAGATT ATGAGGC AGAGACA
GGCTTTA TCCGATG TTTAGAG
CGAGGCT TAGATCC TGAGGCT GAGACAG
AGTCGAG TTTAGATC ATGAGGC TTAGAGA
GAGGCTT GATCCGA GAGGCTT GAGACAG

*Slide courtesy of Dan Zerbino*

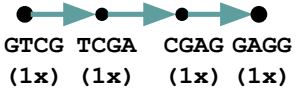
# A quick example

```
 AGTCGAG CTTTAGA  CGATGAG CTTTAGA
  GTCGAGG  TTAGATC  ATGAGGC    GAGACAG
     GAGGCTC   ATCCGAT AGGCTTT GAGACAG
 AGTCGAG    TAGATCC ATGAGGC  TAGAGAA
TAGTCGA  CTTTAGA CCGATGA    TTAGAGA
    CGAGGCT  AGATCCG TGAGGCT  AGAGACA
TAGTCGA GCTTTAG TCCGATG  GCTCTAG
   TCGACGC    GATCCGA GAGGCTT AGAGACA
TAGTCGA    TTAGATC GATGAGG TTTAGAG
  GTCGAGG TCTAGAT   ATGAGGC  TAGAGAC
      AGGCTTT  ATCCGAT AGGCTTT GAGACAG
 AGTCGAG   TTAGATT  ATGAGGC   AGAGACA
       GGCTTTA  TCCGATG    TTTAGAG
    CGAGGCT TAGATCC  TGAGGCT   GAGACAG
 AGTCGAG  TTTAGATC  ATGAGGC TTAGAGA
     GAGGCTT  GATCCGA GAGGCTT  GAGACAG
```

*Slide courtesy of Dan Zerbino*

# A quick example

**First read: GTCGAGG**

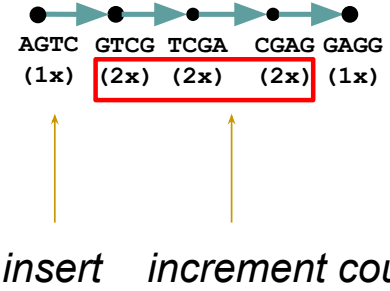



*Slide courtesy of Dan Zerbino*

# A quick example

**First read: GTCGAGG**

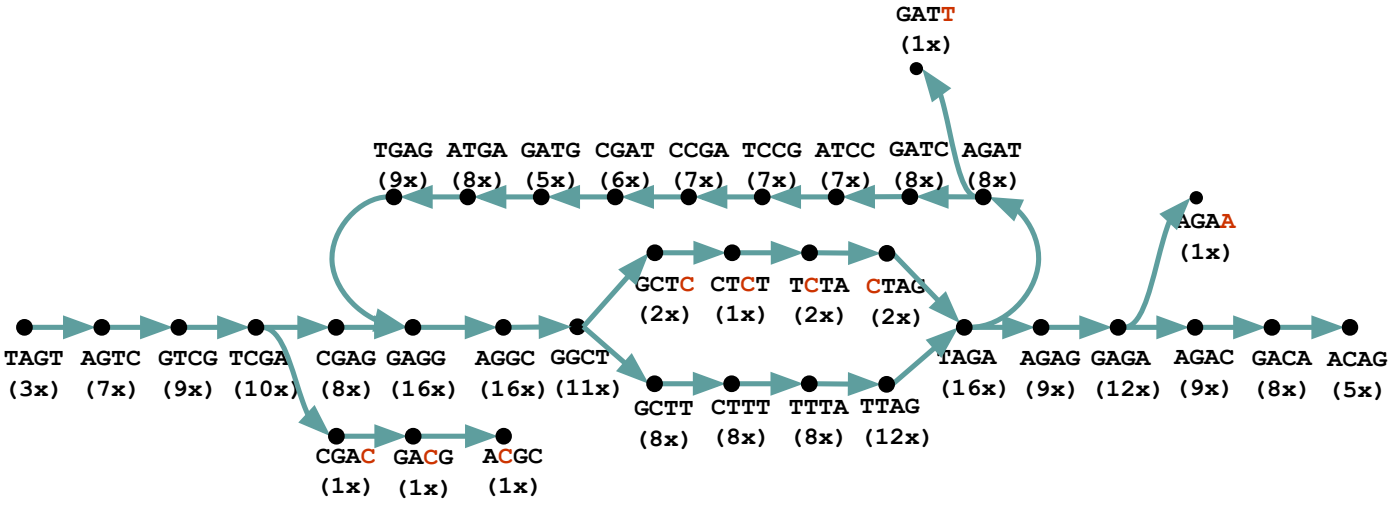
**Second read: AGTCGAG**




*Slide courtesy of Dan Zerbino*

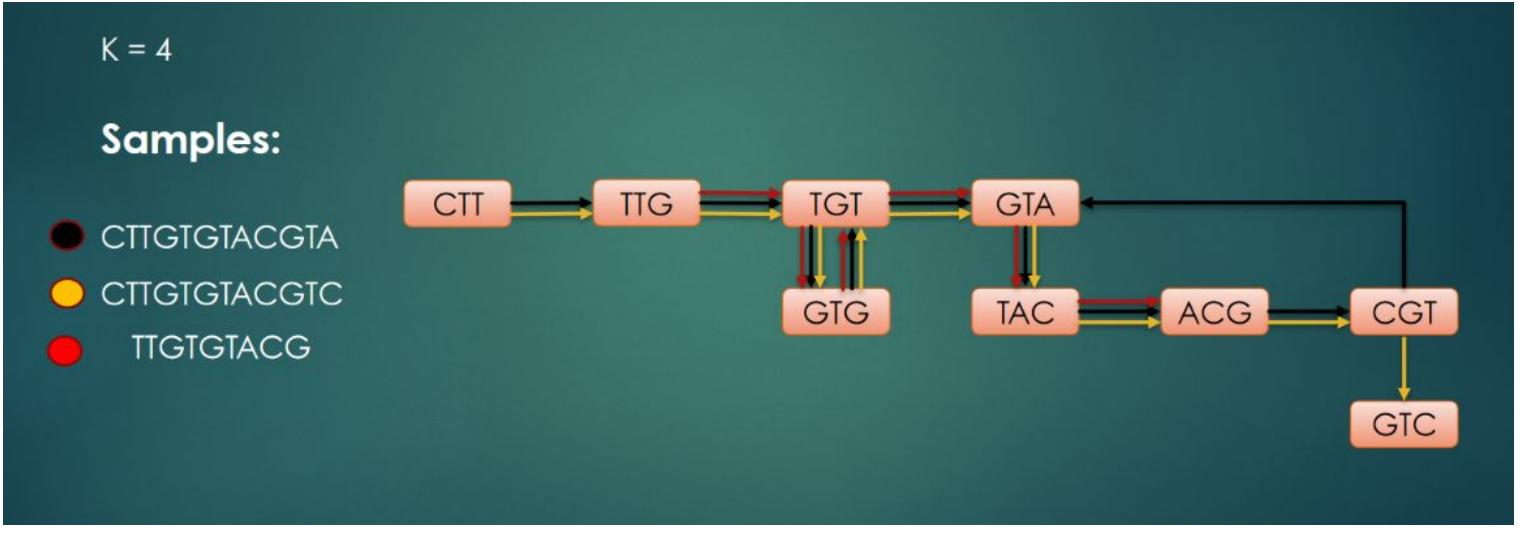
# A quick example

**All the others…**




*Slide courtesy of Dan Zerbino*

# Colored de Bruijn Graphs




*Slide by Rob Patro*

# For more

- "Data structures based on k-mers for querying large collections of sequencing data sets", Camille Marchet, Christina Boucher, Simon J. Puglisi, Paul Medvedev, Mikaël Salson and Rayan Chikhi
  - Genome Res, 2020: https://genome.cshlp.org/content/early/2020/12/16/gr.260604.119.abstract
  - bioRxiv: https://www.biorxiv.org/content/10.1101/866756v3