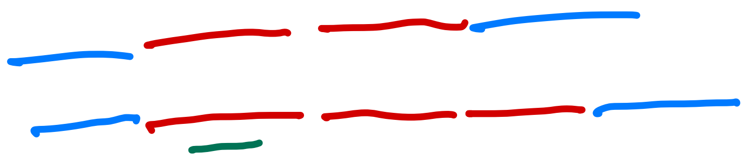
CSE 566 Spring 2023

Genome Assembly

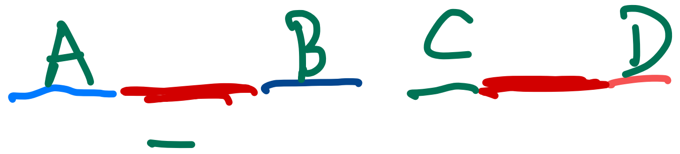
Instructor: Mingfu Shao

Genome Assembly

- Input: sequencing reads
- Output: genome, i.e., full-length sequences of chromosomes
- Challenges:
  - Sequencing errors
  - Repeats in the genome



2 copies or 3 copies ?



A-B, C-D

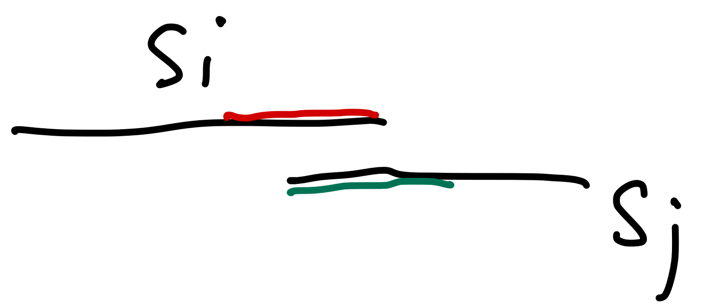
or A-D, C-B ?

Data Structures

- Two major data structures:
  - Overlap Graph
  - de Bruijn Graph (dBG)
- Idea: model the overlaps in reads

Overlap Graph

- A directed graph for a set of reads $S = \{S_1, S_2, \cdots, S_n\}$
  - Each node corresponds to a read
  - Edge from S_i to S_j if S_i overlaps with S_j
  - Weight of edge: the degree of overlap
- Two definitions of overlap
  - Exact overlap: the suffix of S_i exactly matches a prefix of S_j and the length of the match is at least l.
  - Approximate overlap: the optimal alignment score between a suffix of S_i and a prefix of S_j is at least t.

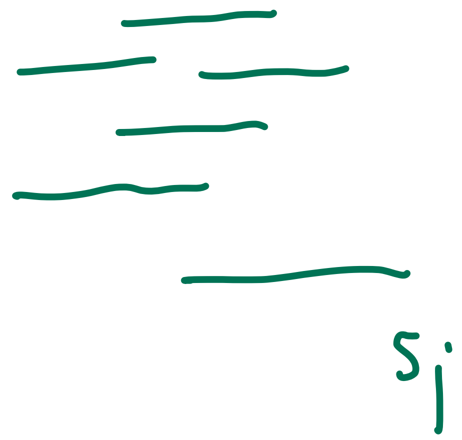


Finding All Exact Overlaps

- Input: $S = \{S_1, S_2, \cdots, S_n\}$

$O(n^2)$

- Output: all pairs that overlap exactly by at least l



- Algorithm:
  - Build the generalized suffix tree for $S = \{S_1, S_2, \cdots, S_n\}$
  - Search each S_j : after l letters, report all strings S_i such that #i is under the subtree.

(S_i, S_j)

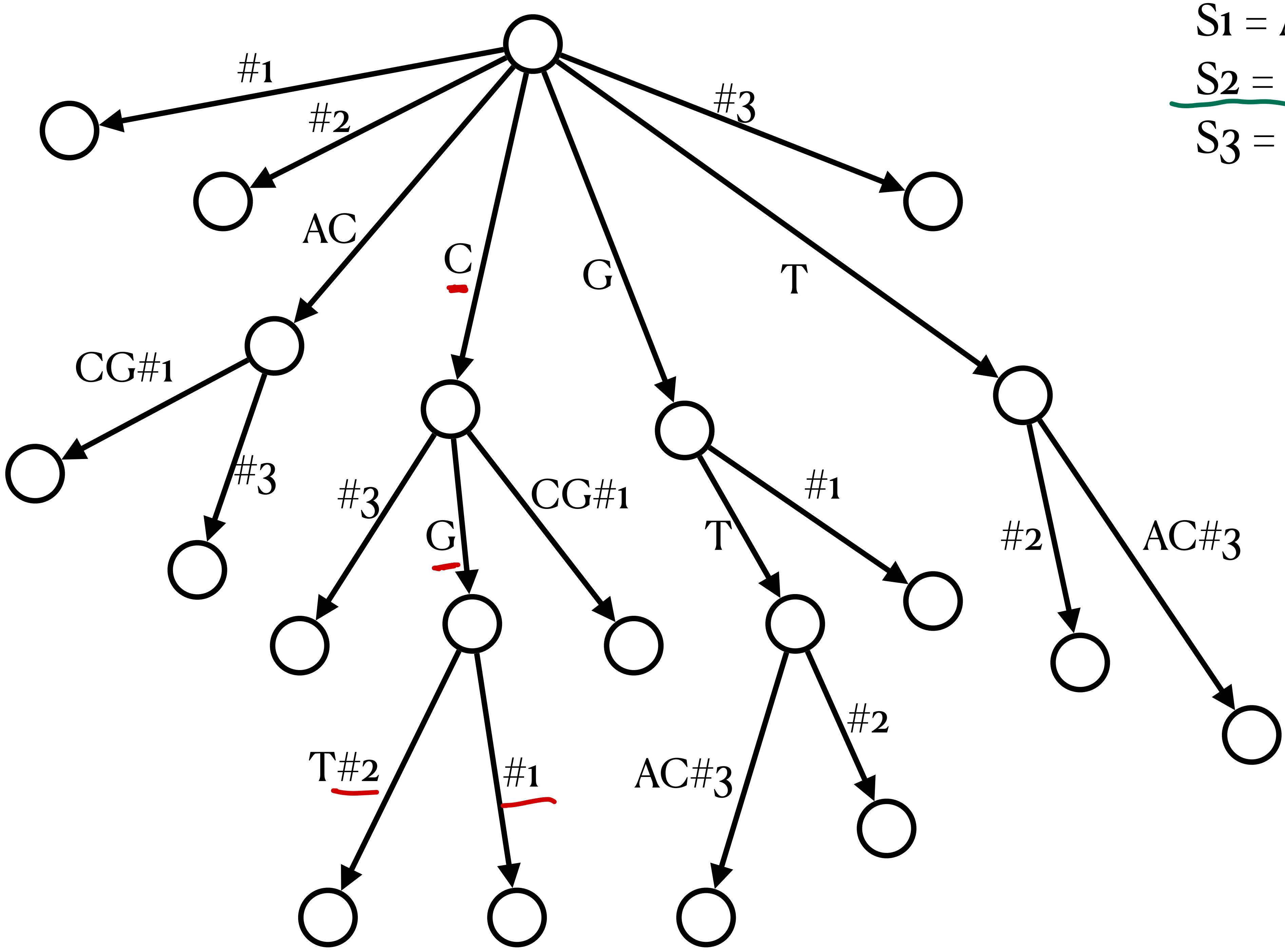
- Running time: $O(N + E)$, where $N = \sum_{i=1}^{n} |S_i|$ and E represents the total number of overlapping pairs.

Example

$l = 2$

S_1 = ACCG#1
S_2 = CGT#2
S_3 = GTAC#3

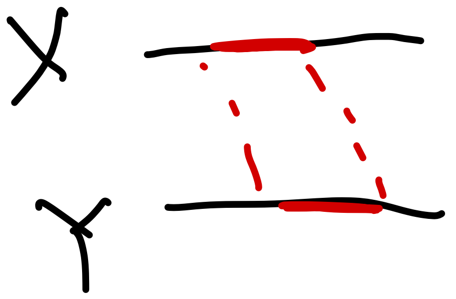
$S_j = S_2$



Finding Approximate Overlap

$\underline{O(n^2)}$

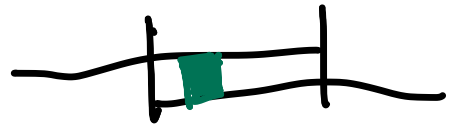
- Input: two strings *X* and *Y*
- Output: optimal alignment between suffix of *X* and prefix of *Y*
- Assume unit gap cost (columns are independent)
- Algorithm: dynamic programming



```
X: AATAGACATTCGATC
           |||  |||||
Y:         CAT-CGGTCACTG
```

Finding All Approximate Overlaps

$O(|S_i| \cdot |S_j|)$



- Running time: $\underline{O(\sum_{i,j} |S_i| \cdot |S_j|)} = \underline{O(N^2)}$
- Extremely slow.
- For sequencing data with low error rate, it is reasonable to assume each pair of overlapped sequences shares at least a substring of length k (aka kmer); then only compare pairs that satisfy this condition (using a hash table).

$e = 1\%$

$k = 20\text{-}50$

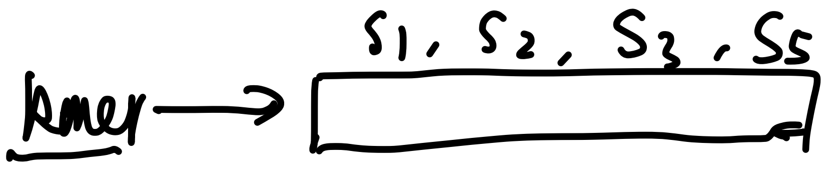
$15\text{-}100$

- For sequencing data with high error rate: no satisfying methods (one of my research topic).

$\uparrow$

$10\% - 15\%$

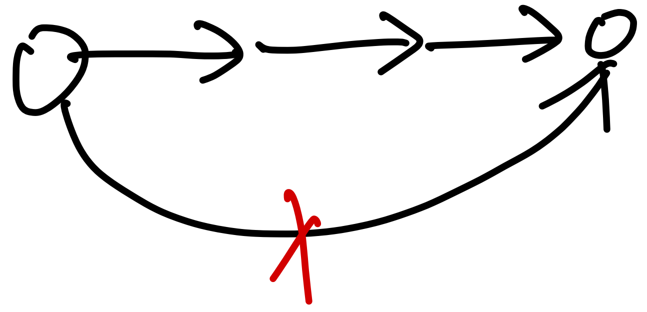
HiFi $< 1\%$

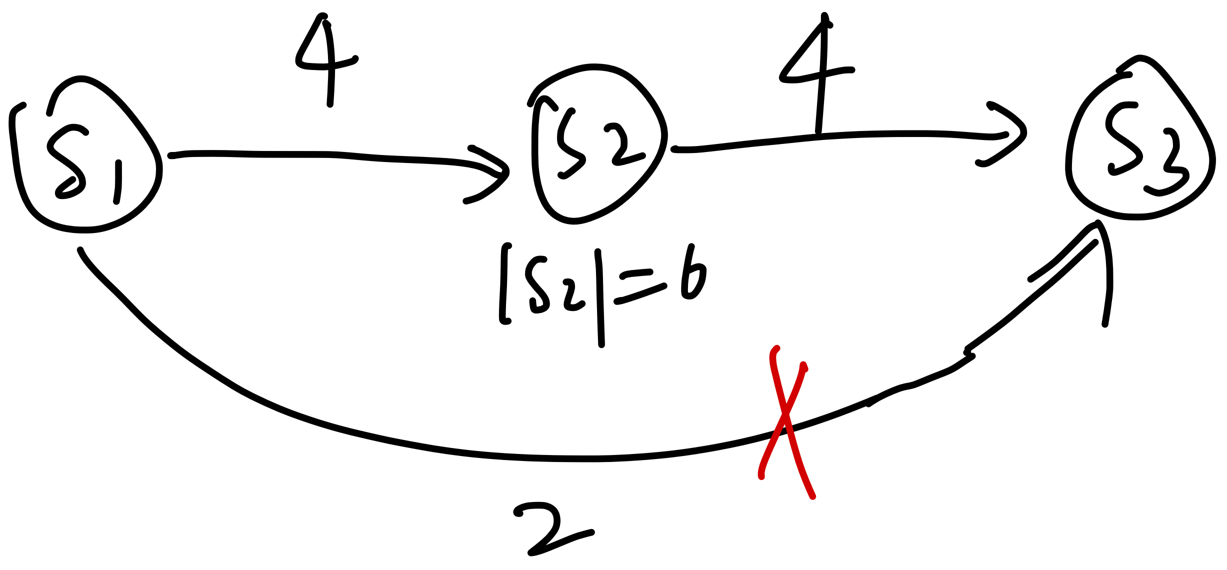


Remove Redundancy (Layout)

- Redundancy: edges that can be inferred through transitivity.
- Layout (remove redundancy):
  - Remove edges that skip one node
  - Remove edges that skip two node

S1 = ACCGTA
S2 = CGTACT
S3 = TACTGAT





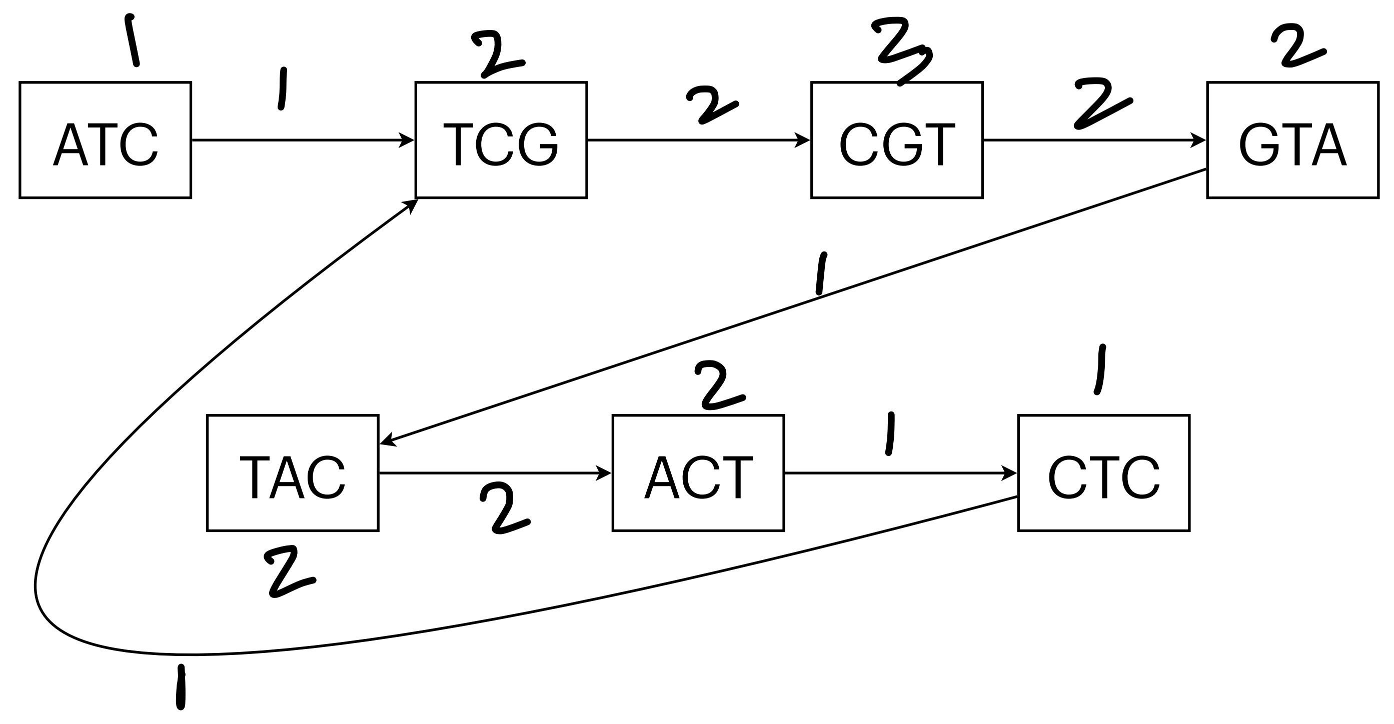
de Bruijn Graph

- A directed graph for a set of reads $S = \{S_1, S_2, \cdots, S_n\}$
    - Each distinct (k-1)-mer corresponds to a node
    - Each distinct k-mer corresponds to an edge, pointing from its first (k-1)-mer to its second (k-1)-mer
    - Weight of node = number of appearances of the (k-1)-mer
    - Weight of edge = number of appearances of the k-mer
- Construction in $O(N)$ time (with hash tables).

Example

k=4

S1 = ATCGTA
S2 = CGTACT
S3 = TACTCGT

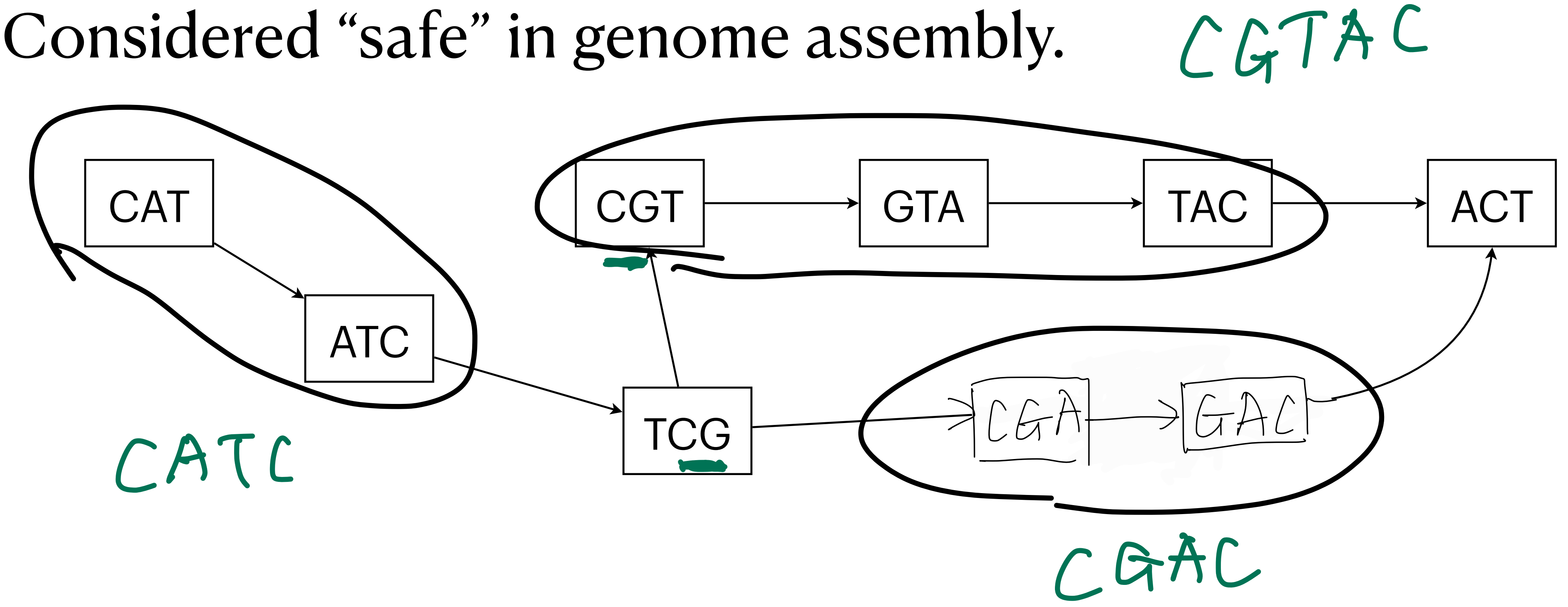


Comparison

	Overlap Graph	**dBG**
Construction	O(N^2)	O(N)
Space	O(#overlapping-sequences)	O(#unique-kmers); bounds: 4^k, O(N), and O(G)
Property	Loss-free	Lose information

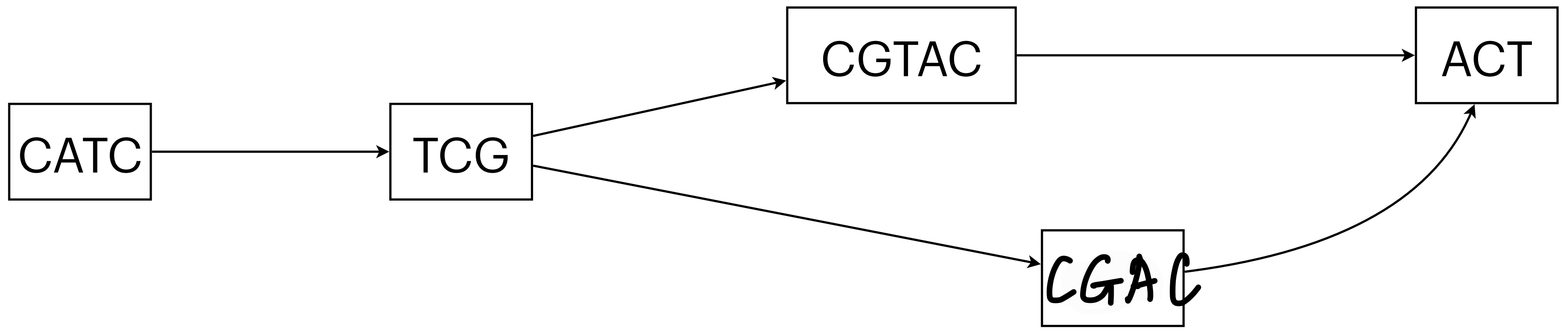
Contig

- Contig: a maximal path without branching (i.e., unambiguous path) in a graph (either overlap graph or dBG).
- Considered "safe" in genome assembly.



Compact de Bruijn Graph (cdBG)

- cdBG: merging each contig into a single node
- More space efficient.
- Problem: direct construction of cdBG
    - Bifrost, cuttlefish(2), BCALM(2)


CATC
TCG
CGTAC
ACT
CGAC