Algorithms for Bioinformatics Autumn 2011

VELI MÄKINEN

HTTP://WWW.CS.HELSINKI.FI/EN/COURSES/582670/2011/S/K/1

Lecture 5

GRAPH ALGORITHMS AND DNA SEQUENCING

Part of the slides are from http://bix.ucsd.edu/bioalgorithms/slides.php

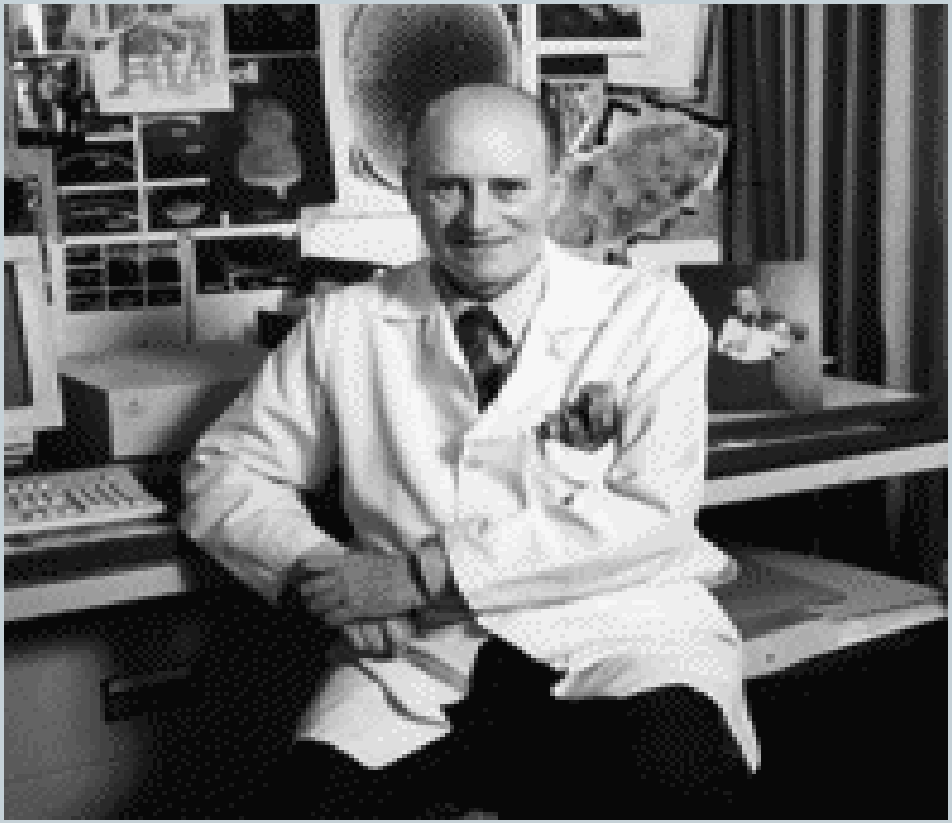
Part I

INTERVAL GRAPHS (SKIPPED, WILL BE COVERED ON MONDAY 10.10. LECTURE)

Beginning of Graph Theory in Biology

Benzer's work

- Developed deletion mapping
- "Proved" linearity of the gene
- Demonstrated internal structure of the gene



Seymour Benzer, 1950s

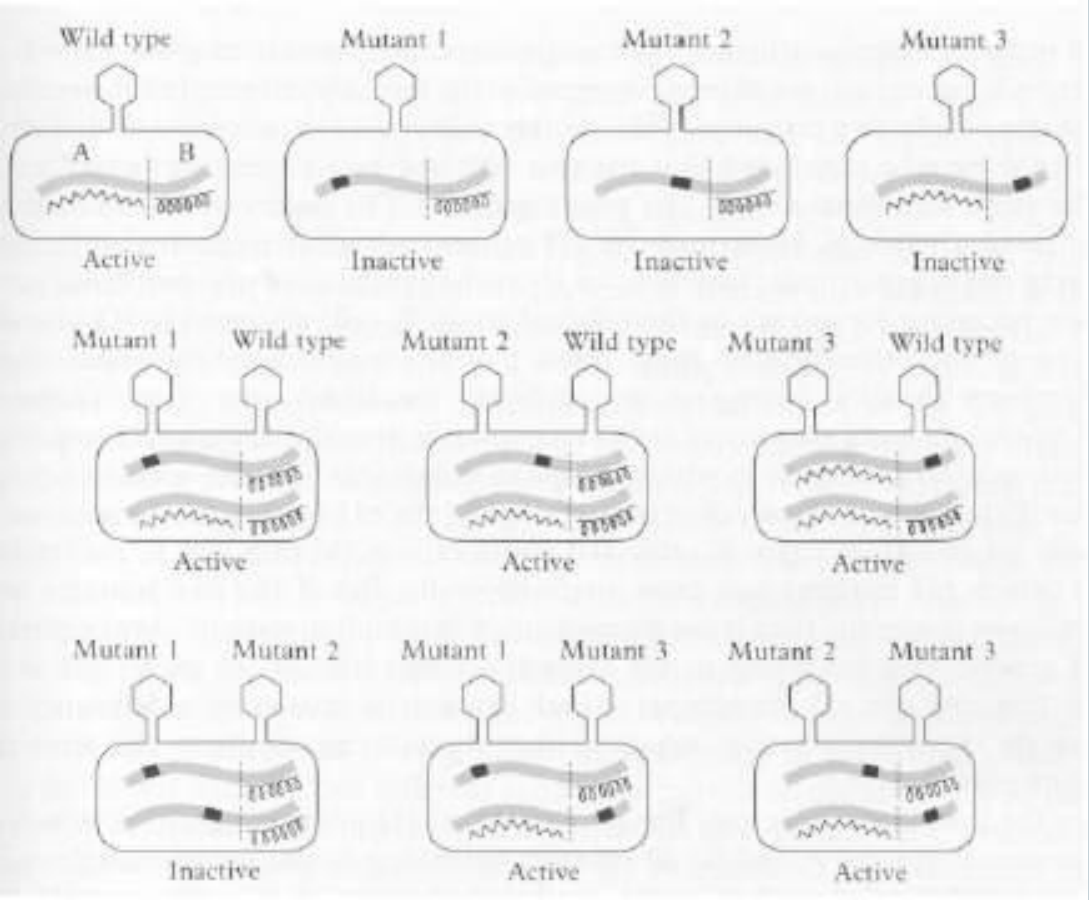
Viruses Attack Bacteria

- Normally bacteriophage T4 kills bacteria
- However if T4 is mutated (e.g., an important gene is deleted) it gets disabled and looses an ability to kill bacteria
- Suppose the bacteria is infected with two different mutants each of which is disabled – would the bacteria still survive?
- Amazingly, a pair of disable viruses can kill a bacteria even if each of them is disabled.
- How can it be explained?

Benzer's Experiment

- Idea: infect bacteria with pairs of mutant T4 bacteriophage (virus)
- Each T4 mutant has an unknown interval deleted from its genome
- If the two intervals overlap: T4 pair is missing part of its genome and is disabled – bacteria survive
- If the two intervals do not overlap: T4 pair has its entire genome and is enabled – bacteria die

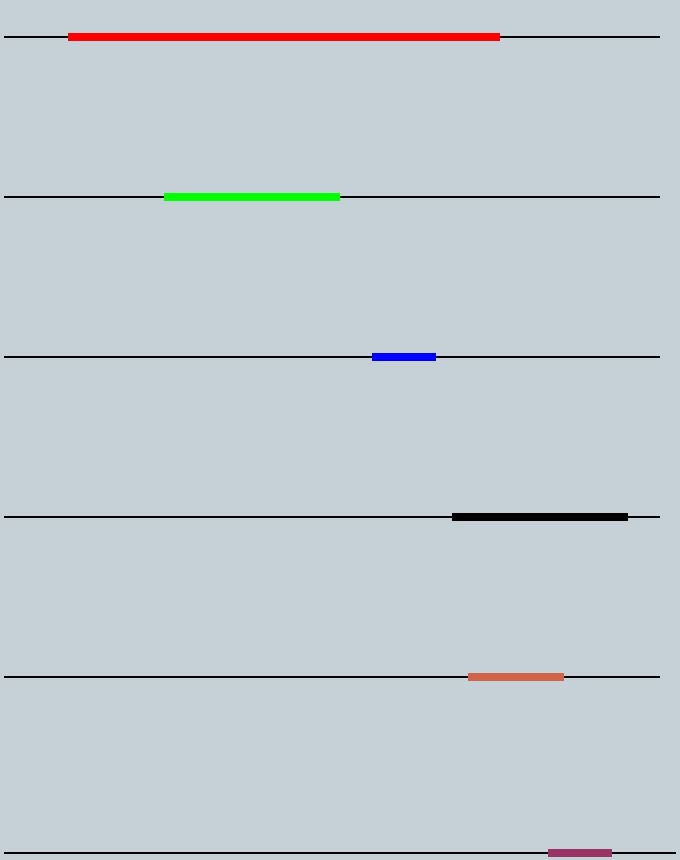
Complementation between pairs of mutant T4 bacteriophages

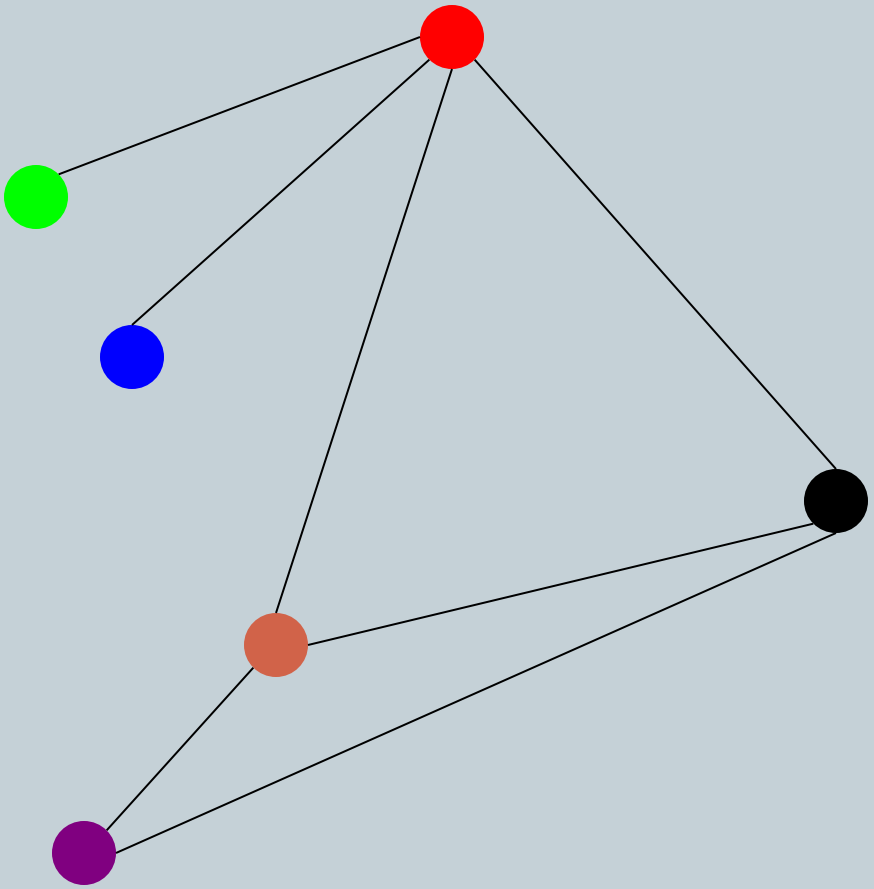


Benzer's Experiment and Graphs

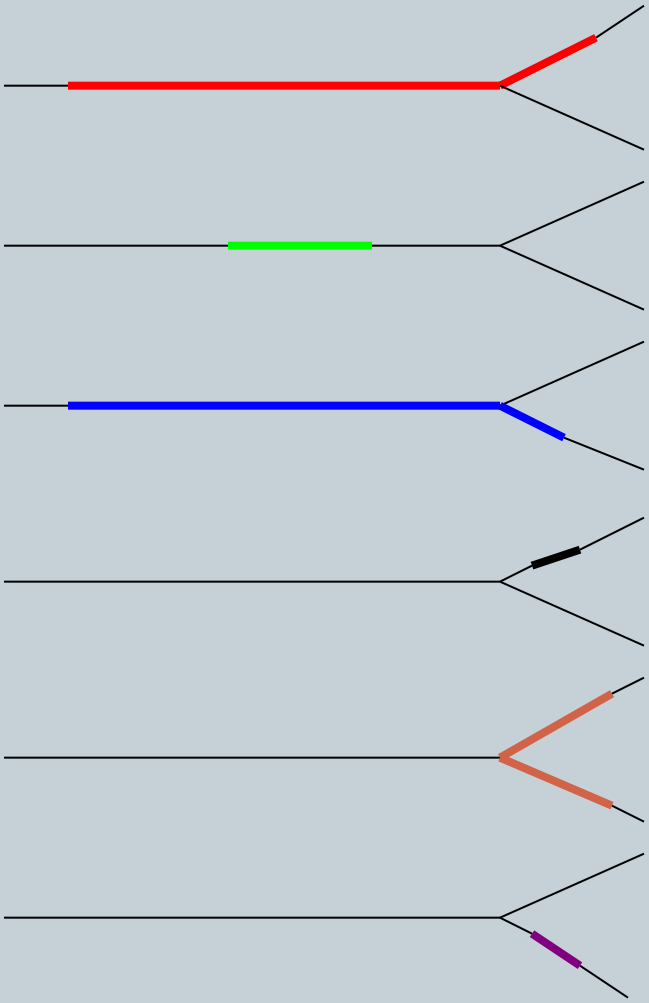
- Construct an **interval graph**: each T4 mutant is a vertex, place an edge between mutant pairs where bacteria survived (i.e., the deleted intervals in the pair of mutants overlap)
- Interval graph structure reveals whether DNA is linear or branched DNA

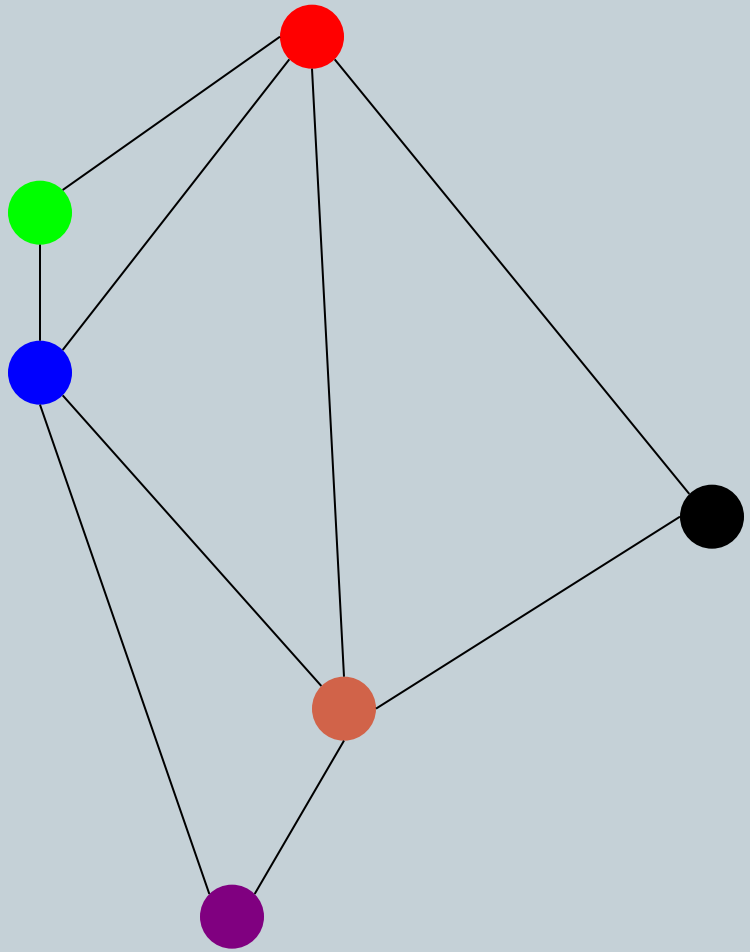
Interval Graph: Linear Genes



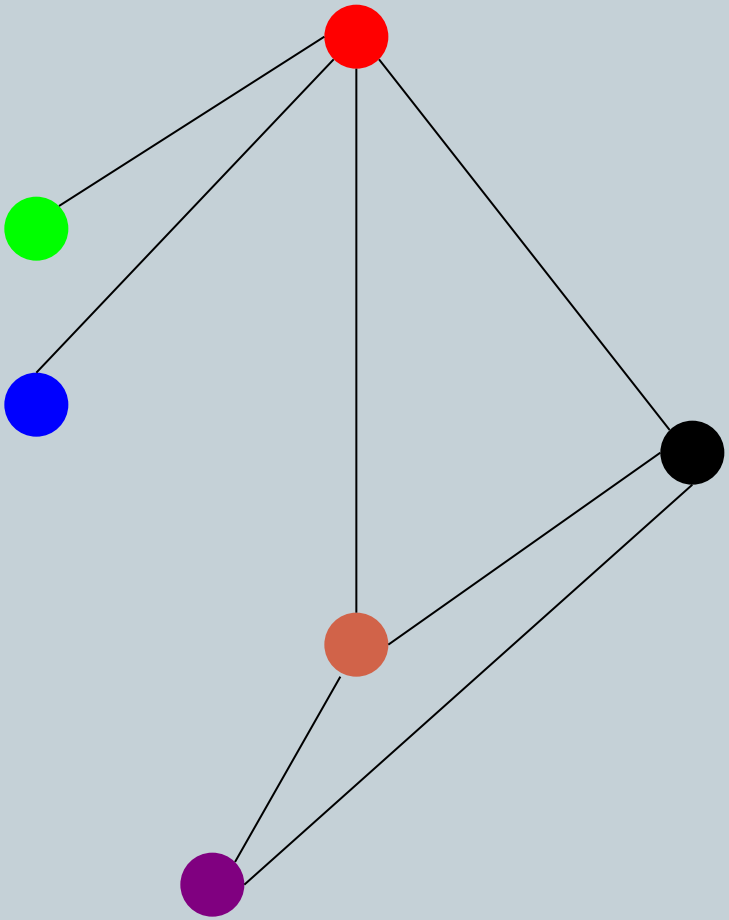


Interval Graph: Branched Genes

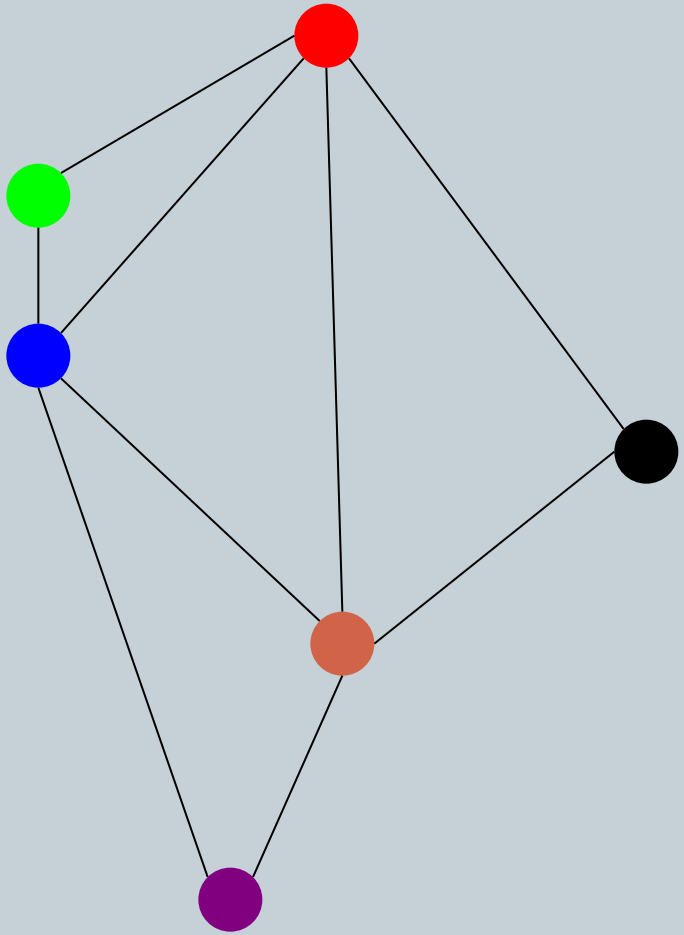




Interval Graph: Comparison



Linear genome



Branched genome

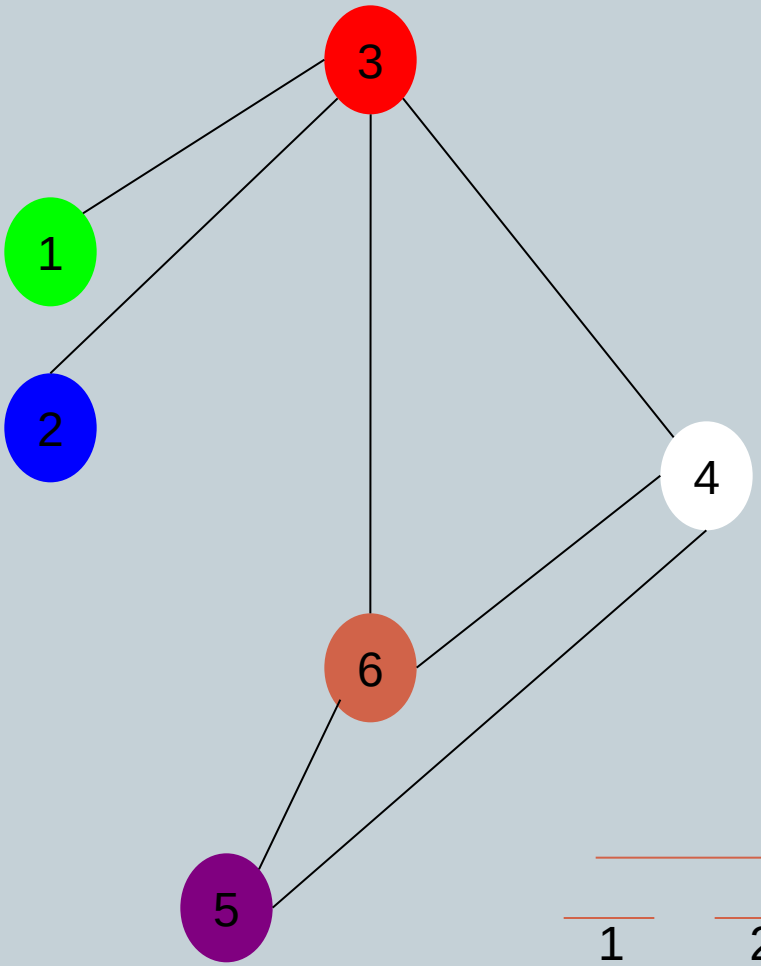
Interval graph recognition in linear time

- Michel Habib, Ross McConnell, Christophe Paul, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, Volume 234, Issues 1-2, 6 March 2000, Pages 59-84.
  - Simple linear time algorithm.
    - No complicated data structures required, easy to implement.
    - Requires some graph theory to understand why and how it works (details out of the scope of this course).

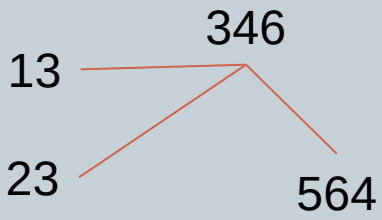
Interval graph recognition in linear time (outline)

- Number the vertices in the *lexicographic breadth-first order*.
- Construct a tree of *maximal cliques* (in a suitable evaluation order).
- Find a chain of cliques by a recursive *clique partitioning refinement with pivots* algorithm (generalization of quicksort to graphs).
- If each vertex appears only in consecutive cliques in the chain, then the graph is interval graph, otherwise not.

Simulation: yes



Tree of max cliques



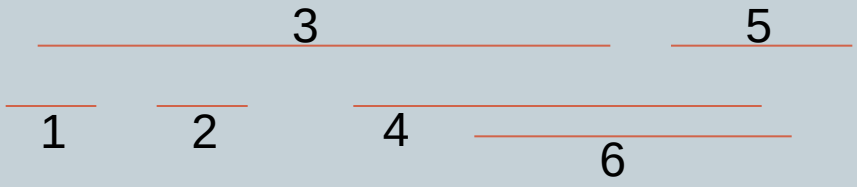
{564, 346, 23}, {13}

3

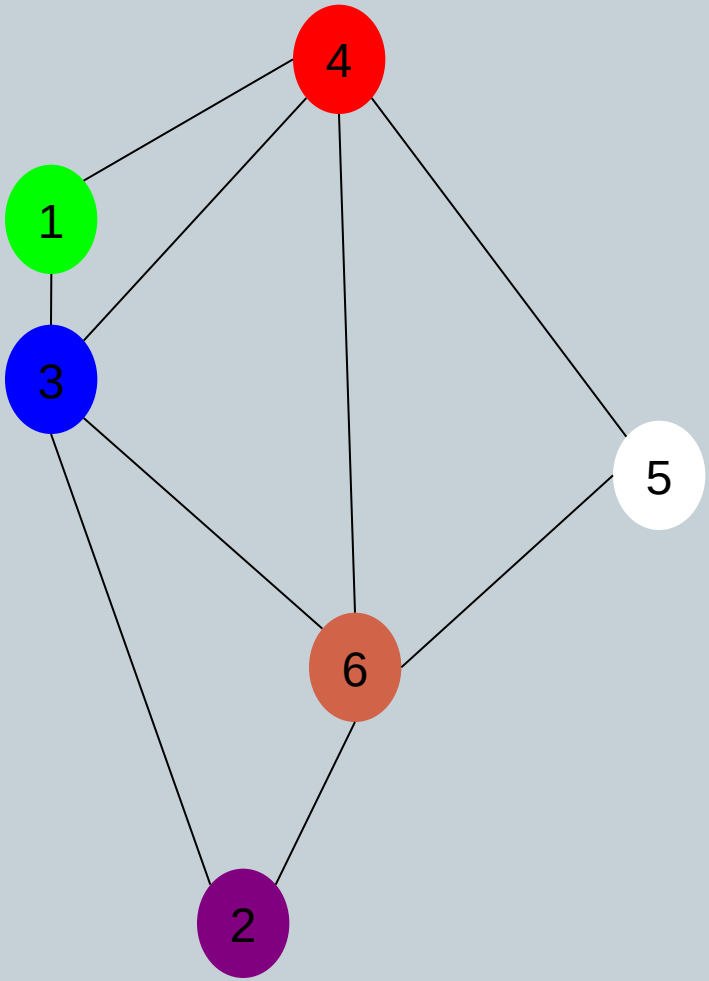
{564}, {346, 23}, {13}

4

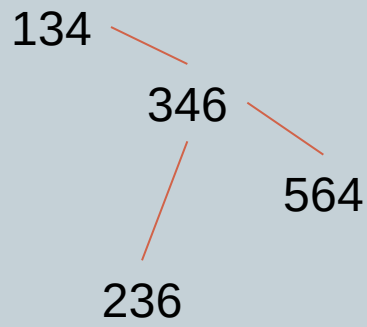
{564}, {346}, {23}, {13}



Simulation: no



Tree of max cliques



{564, 346, 236}, {134}

3

{564}, {346, 236}, {134}

{564}, {346}, {236}, {134}

Not an interval graph

Part II

SHORTEST SUPERSTRING

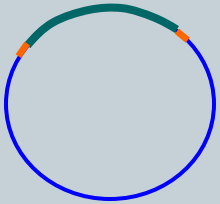
DNA Sequencing: History

Sanger method (1977): labeled ddNTPs terminate DNA copying at random points.

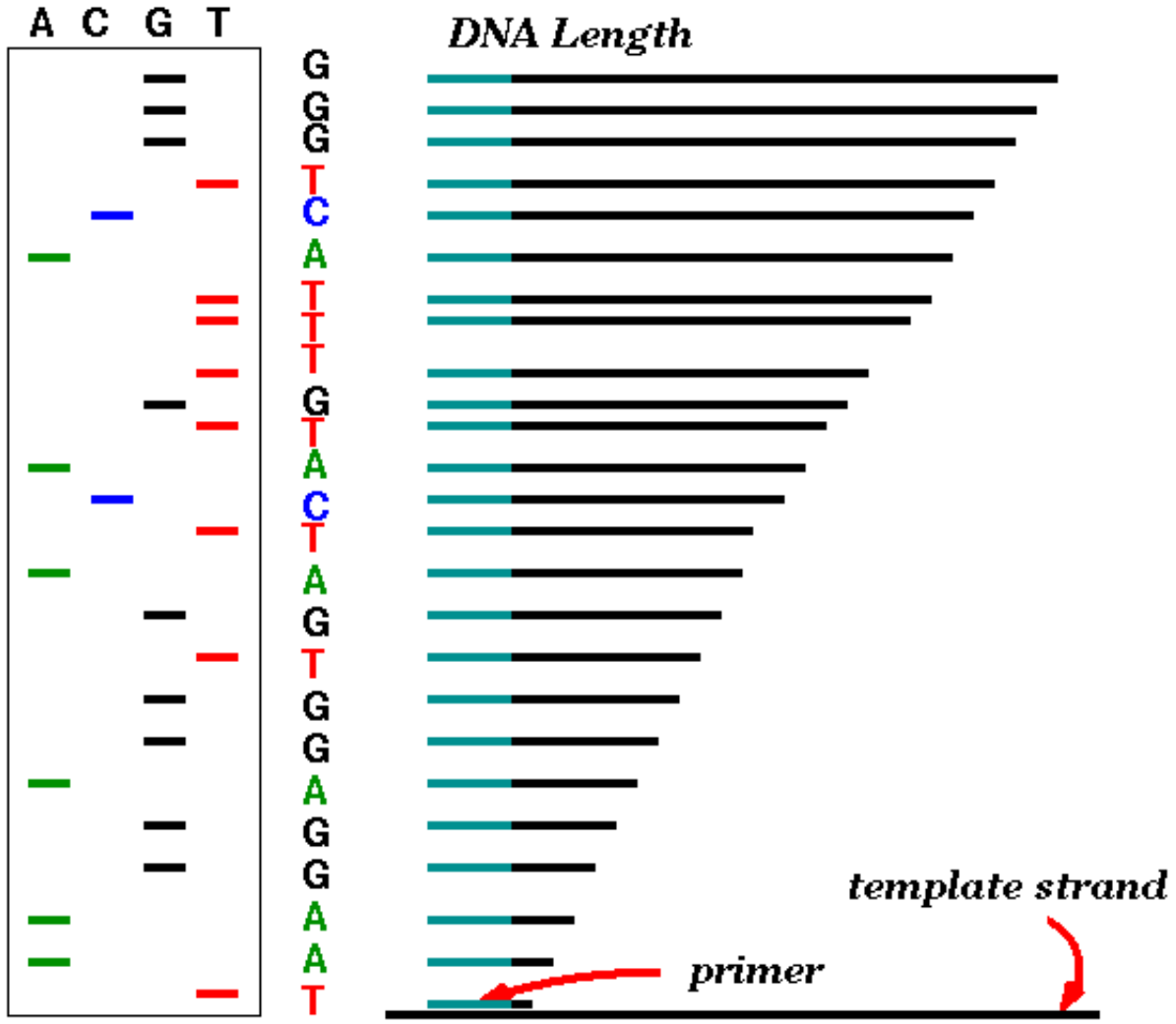
Gilbert method (1977): chemical method to cleave DNA at specific points (G, G+A, T+C, C).

Both methods generate labeled fragments of varying lengths that are further electrophoresed.

Sanger Method: Generating Read

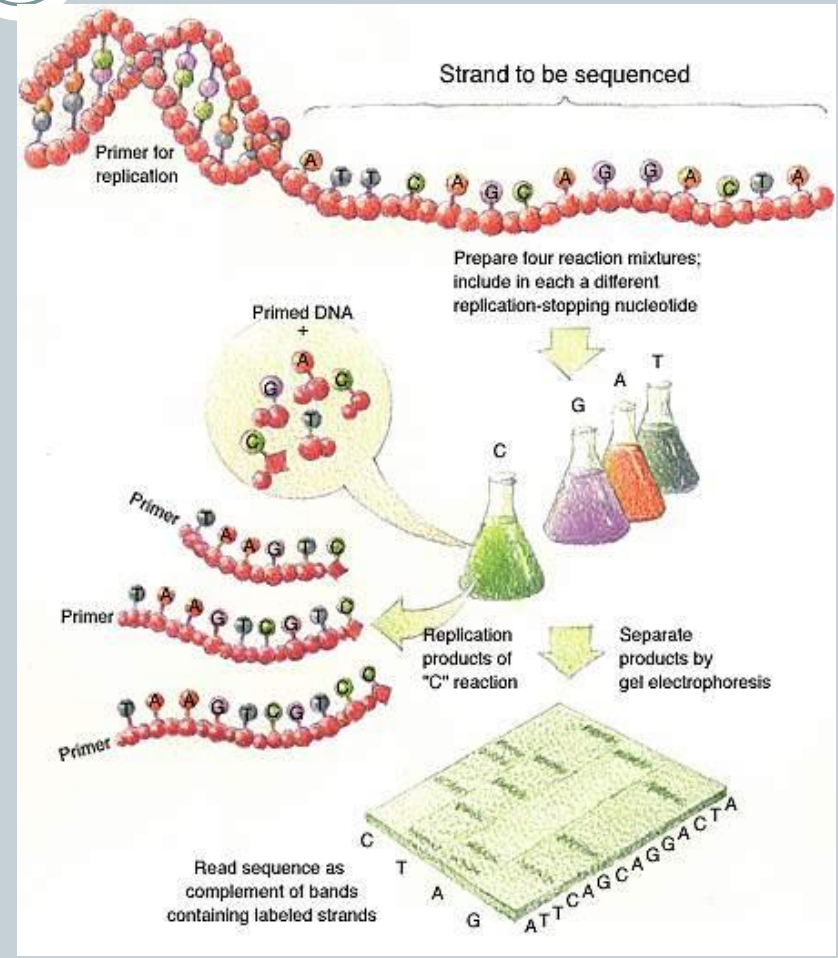


1. Start at primer (restriction site)
2. Grow DNA chain
3. Include ddNTPs
4. Stops reaction at all possible points
5. Separate products by length, using gel electrophoresis



DNA Sequencing

- Shear DNA into millions of small fragments
- Read 500 – 700 nucleotides at a time from the small fragments (Sanger method)



Fragment Assembly

- **<u>Computational Challenge</u>:** assemble individual short fragments (reads) into a single genomic sequence ("superstring")
- Until late 1990s the shotgun fragment assembly of human genome was viewed as intractable problem
  - Now there exists "complete" sequences of human genomes of several individuals
- For small and "easy" genomes, such as bacterial genomes, fragment assembly is tractable with many software tools
- Remains to be difficult problem for more complex genomes

Shortest Superstring Problem

- Problem: Given a set of strings, find a shortest string that contains all of them
- Input: Strings $S=\{s_1, s_2, \ldots, s_n\}$
- Output: A string s that contains all strings $s_1, s_2, \ldots, s_n$ as substrings, such that the length of s is minimized

- **Complexity:** NP-hard
- Recall:
  - Greedy approximation algorithm at the study group
  - Extension to approximate case in the exercises

Reducing SSP to TSP

- Define *overlap(s_i, s_j)* as the longest prefix of s_j that matches a suffix of s_i, e.g.:

  aaaggcatcaaatctaaaggcatcaaa

  aaaggcatcaaatctaaaggcatcaaa
- Define *prefix(s_i, s_j)* as the part of s_i after its longest overlap with s_j is removed.
- Construct a *prefix graph* with
  - n vertices representing the n strings $s_1, s_2, \ldots, s_n$; and
  - edges of length $|prefix(s_i, s_j)|$ between vertices s_i and s_j.
- Add a dummy vertex d to prefix graph with edges of length $|s_i|$ between each s_i and d.
- Find the shortest path which visits every vertex exactly once.
- This is the *Asymmetric Traveling Salesman Problem (ATSP)*, which is also NP-complete.

SSP to TSP: An Example

S = { ATC, CCA, CAG, TCC, AGT }

SSP

AGT

CCA

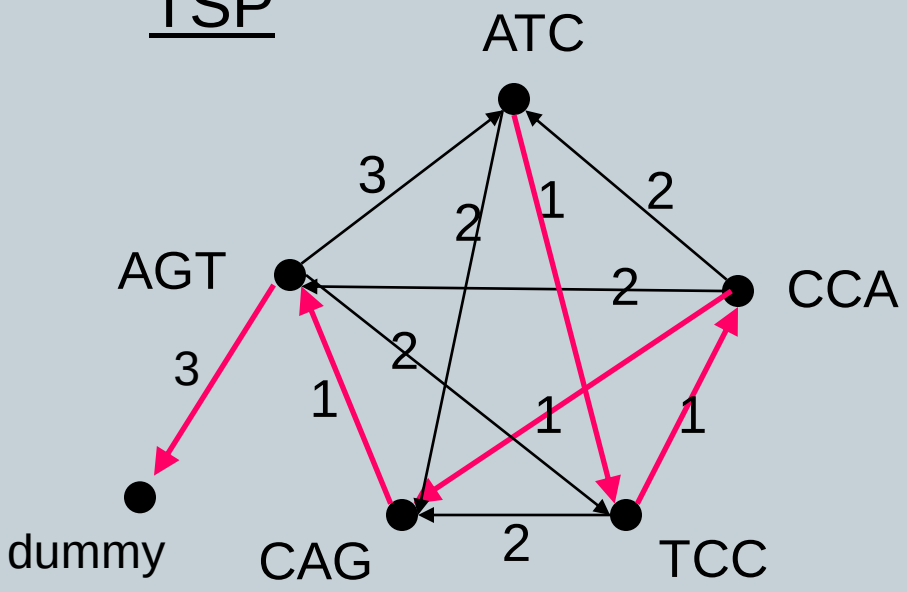
ATC

ATCCAGT

TCC

CAG

TSP



ATCCAGT

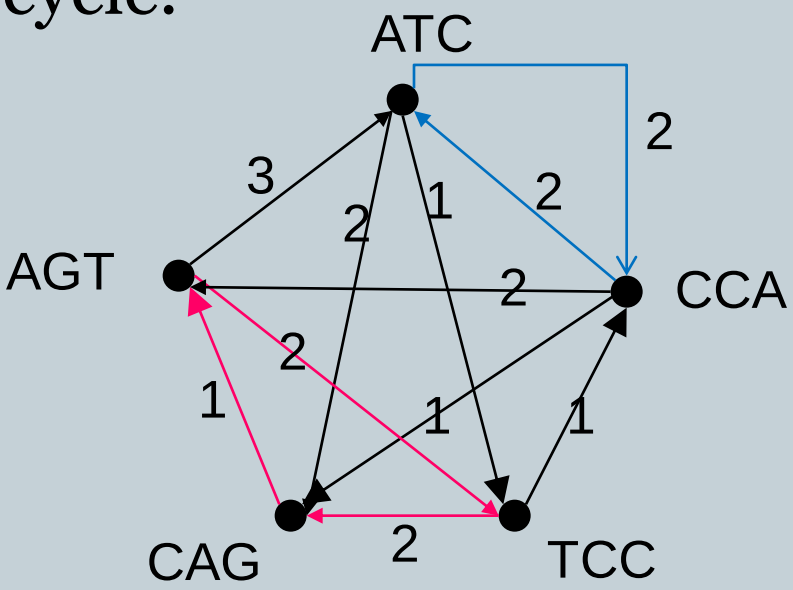
(note: only subset of edges shown)

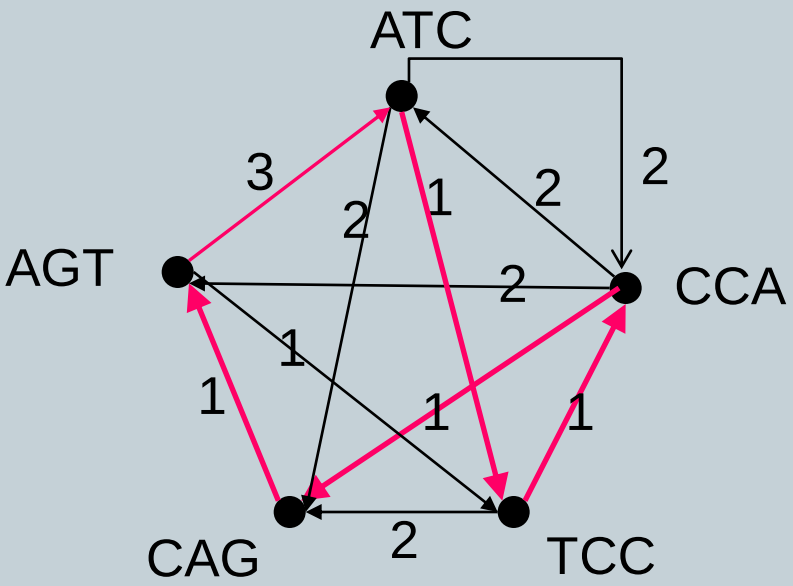
Shortest superstring: 4-approximation

- There are logarithm-factor approximation algorithms for ATSP, but the prefix graph instances admit constant factor approximation algorithms:
  - Resulting superstring is at most **c** times longer than the optimal *OPT*, for some constant **c**.
- 4-approximation algorithm:
  1. Construct the prefix graph corresponding to strings in *S*.
  2. Find a *minimum weight cycle cover* on the prefix graph, $C=\{c_1,\ldots,c_k\}$.
  3. Read the superstring defined by the cycle cover.

Cycle cover

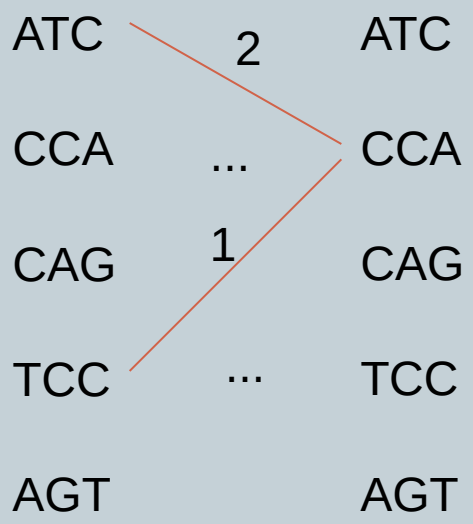
- A cycle cover is a set of disjoint cycles covering all vertices.
- ATSP *tour* is a special case: cycle cover with one cycle.





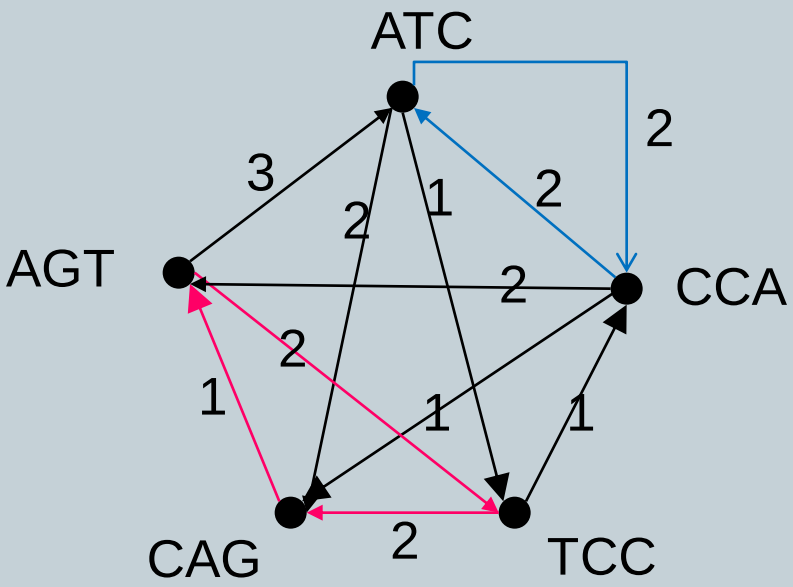
Minimum weight cycle cover

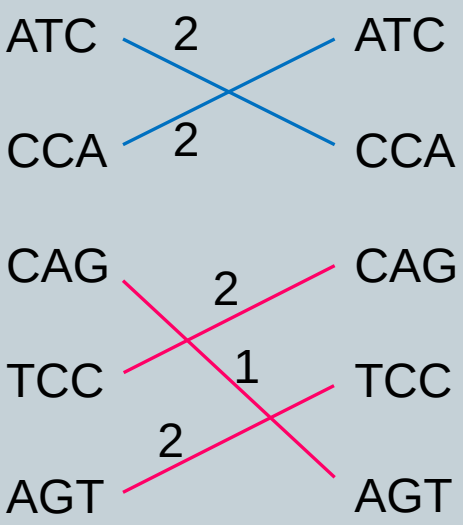
- Minimum weight cycle cover is polynomial time solvable!
- Reduction to *minimum weight perfect matching on bipartite graph*:
  - Create two vertices u_i and v_i from each s_i to a graph H.
  - Add edge (u_i,v_j) with weight $|prefix(s_i,s_j)|$, for $i \neq j$.
  - Each cycle cover in prefix graph corresponds to a minimum weight perfect matching on H and vice versa.



Minimum weight perfect matching

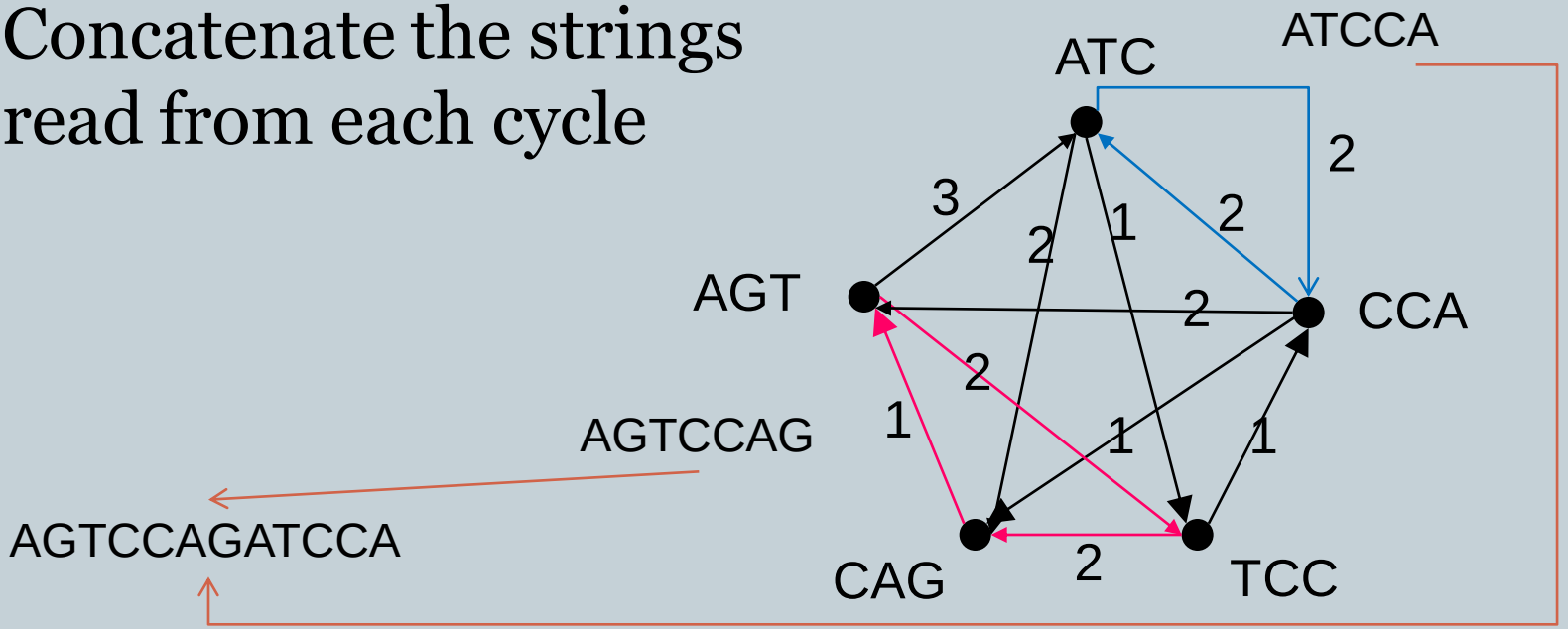
- Classical non-trivial graph problem with polynomial time solutions.





Reading superstring from cycle cover

- For each cycle:
  - concatenate prefixes corresponding to weights starting from any vertex
  - append the overlap of last and start vertex
- Concatenate the strings read from each cycle



Analysis

- The length of the superstring read from the cycle cover is $wt(C)+\sum_{c\in C}|overlap(e(c),b(c))|$, where
  - $wt(C)$ is the length/weight of the cycle cover, and
  - $b(c)$ and $e(c)$ are the strings corresponding to the first and last vertices of cycle c.
- **Observation**: $wt(C)$ is smaller or equal to OPT.
- In the worst case, $|overlap(e(c),b(c))|=\min(|e(c)|,|b(c)|)$.
  - For pessimistic estimate, it is enough to assume that the overlap is as long as the length of any *representative* string r corresponding to a vertex in c.

Analysis…

- **Lemma**: Let c and c' be two cycles in C, and let r, r' be representative strings from these cycles. Then |overlap(r,r')|<wt(c)+wt(c').
  - Proof. Study group work, see Vazirani, Approximation algorithms, page 63.
- Let $r_1, r_2, \ldots, r_k$ be the representatives of cycles in the optimal cycle cover C, numbered in order of their leftmost occurrence in the shortest superstring:
  - $OPT \geq \sum_i |r_i| - \sum_i |overlap(r_i, r_{i+1})| > \sum_i |r_i| - 2wt(C)$
- Hence,
  - $wt(C) + \sum_{c \in C} |overlap(e(c), b(c))| \leq wt(C) + \sum_i |r_i| \leq 4OPT$

Part III

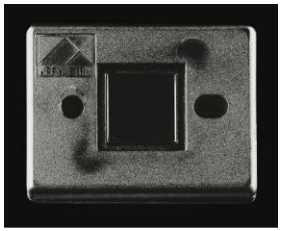
SEQUENCING BY HYBRIDIZATION

Sequencing by Hybridization (SBH): History

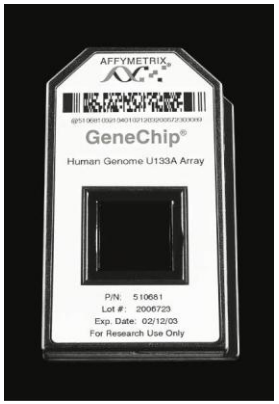
- 1988: SBH suggested as an an alternative sequencing method. Nobody believed it will ever work
- 1991: Light directed polymer synthesis developed by Steve Fodor and colleagues.
- 1994: Affymetrix develops first 64-kb DNA microarray

First microarray prototype (1989)

First commercial DNA microarray prototype w/16,000 features (1994)



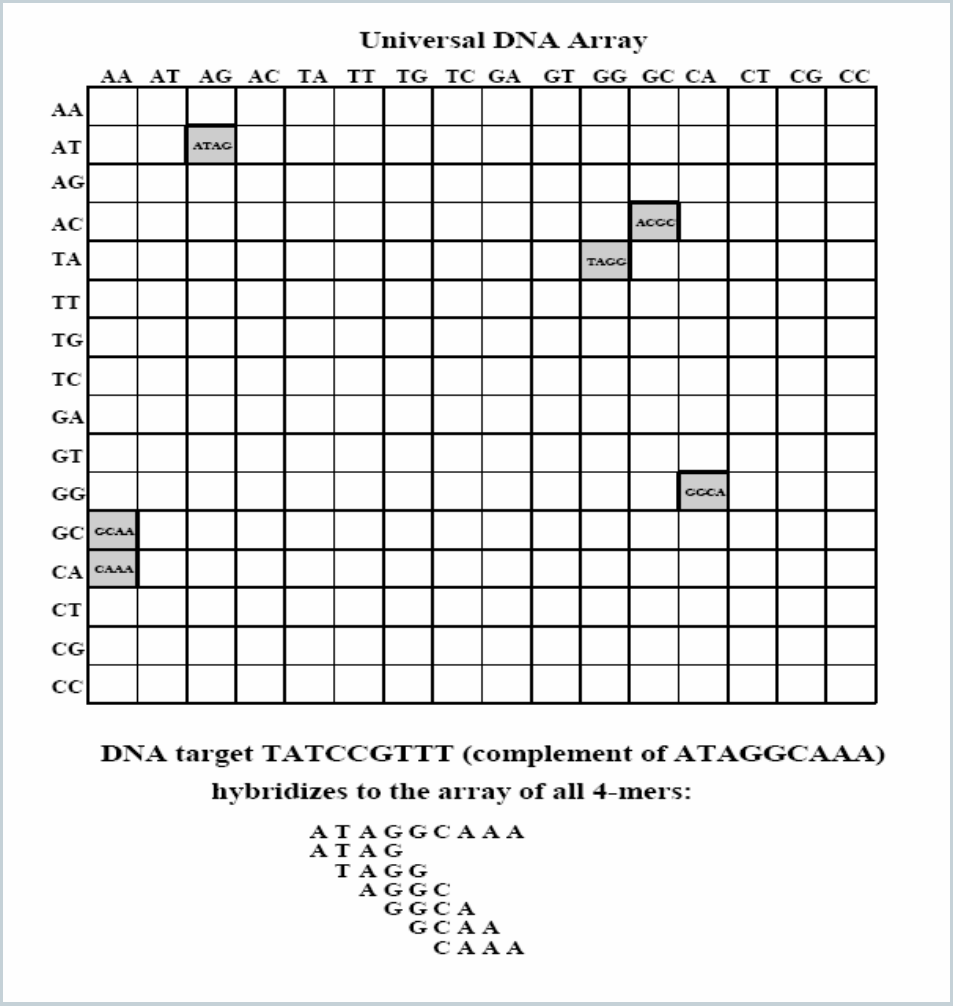
500,000 features per chip (2002)



How SBH Works

- Attach all possible DNA probes of length *l* to a flat surface, each probe at a distinct and known location. This set of probes is called the *DNA microarray*.
- Apply a solution containing fluorescently labeled DNA fragment to the array.
- The DNA fragment hybridizes with those probes that are complementary to substrings of length *l* of the fragment.
- Using a spectroscopic detector, determine which probes hybridize to the DNA fragment to obtain the *l*-mer composition of the target DNA fragment.
- Reconstruct the sequence of the target DNA fragment from the *l*-mer composition.

Hybridization on DNA Array



l-mer composition

- *Spectrum (s, l)* is a multiset of all possible *(n – l + 1) l*-mers in a string *s* of length *n*
- For *s* = TATGGTGC, *Spectrum (s, 3)*:

  {TAT, ATG, TGG, GGT, GTG, TGC}
- Different sequences may have the same spectrum:

  Spectrum(GTATCT,2)=

  Spectrum(GTCTAT,2)=

  {AT, CT, GT, TA, TC}

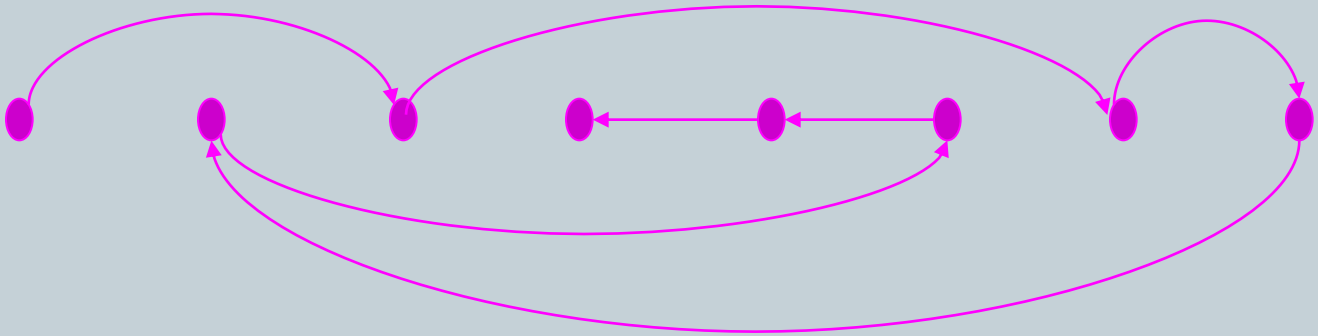
The SBH Problem

- Goal: Reconstruct a string from its *l*-mer composition
- Input: A set *S*, representing all *l*-mers from an (unknown) string *s*
- Output: String *s* such that *Spectrum(s,l) = S*

SBH: Hamiltonian Path Approach

$S = \{ ATG\ \ AGG\ \ TGC\ \ TCC\ \ GTC\ \ GGT\ \ GCA\ \ CAG \}$

H ATG AGG TGC TCC GTC GGT GCA CAG

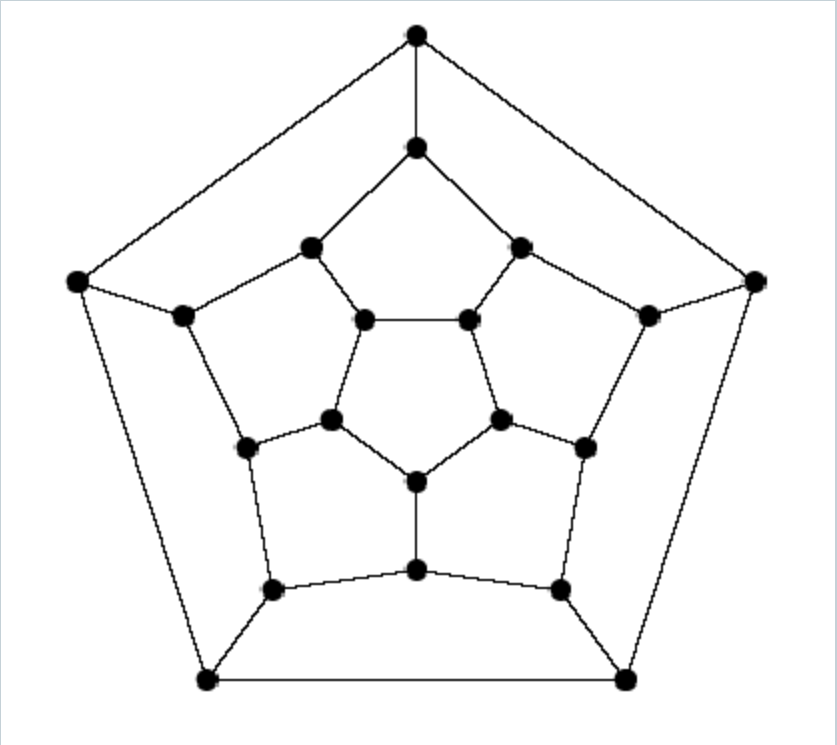


ATGCAGGTCC

Path visited every VERTEX once

Hamiltonian Cycle Problem

- Find a cycle that visits every ***vertex*** exactly once
- NP-complete



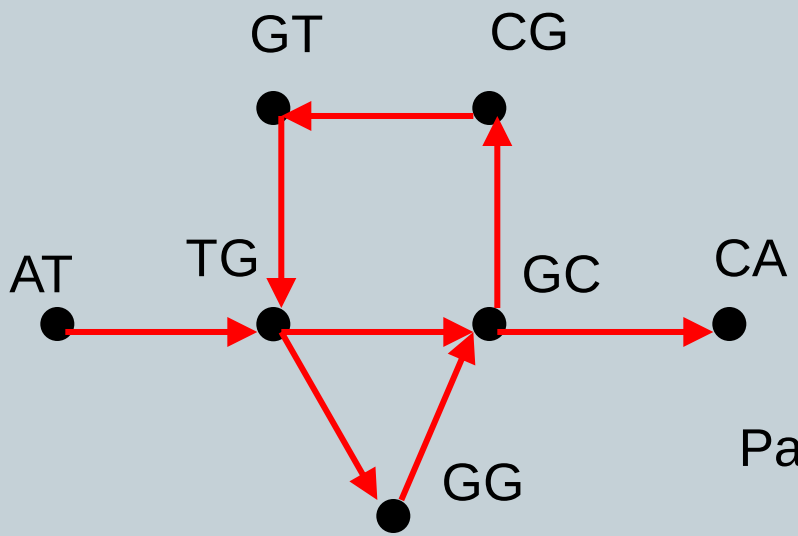
Game invented by Sir William Hamilton in 1857

SBH: Eulerian Path Approach

S = { ATG, TGC, GTG, GGC, GCA, GCG, CGT }

Vertices correspond to $(l-1)$-mers : { AT, TG, GC, GG, GT, CA, CG }

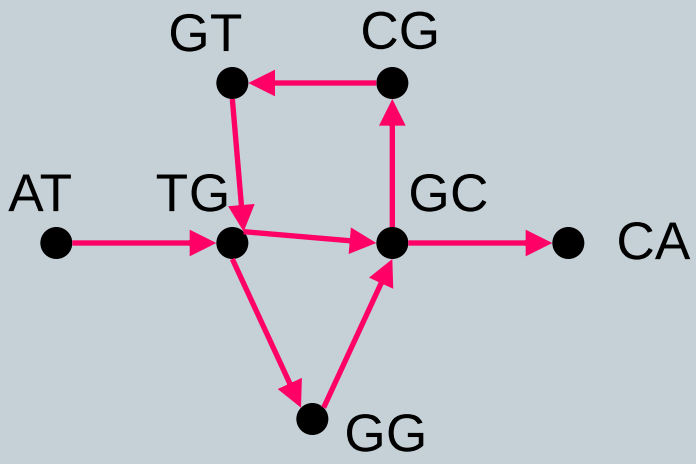
Edges correspond to l–mers from S



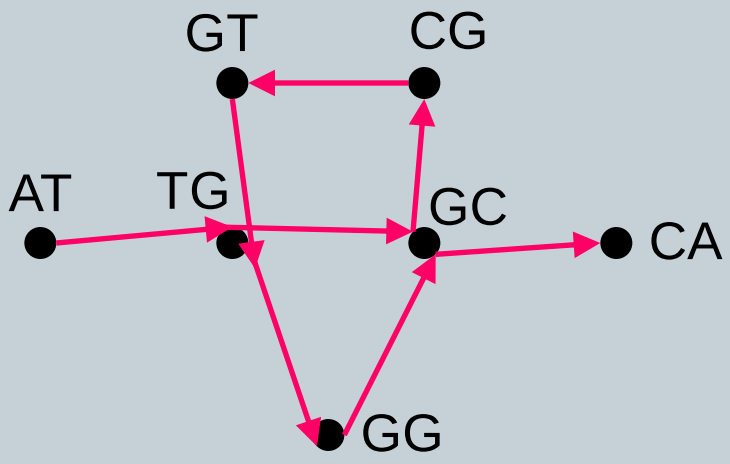
Path visited every EDGE once

SBH: Eulerian Path Approach

S = { AT, TG, GC, GG, GT, CA, CG } corresponds to two different paths:



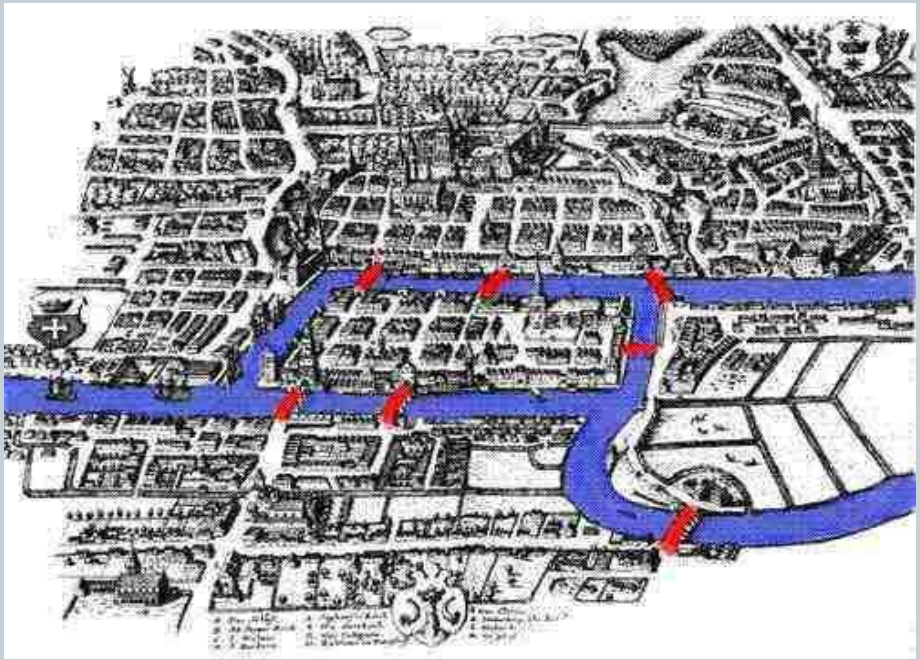
ATGGCGTGCA



ATGCGTGGCA

The Bridge Obsession Problem

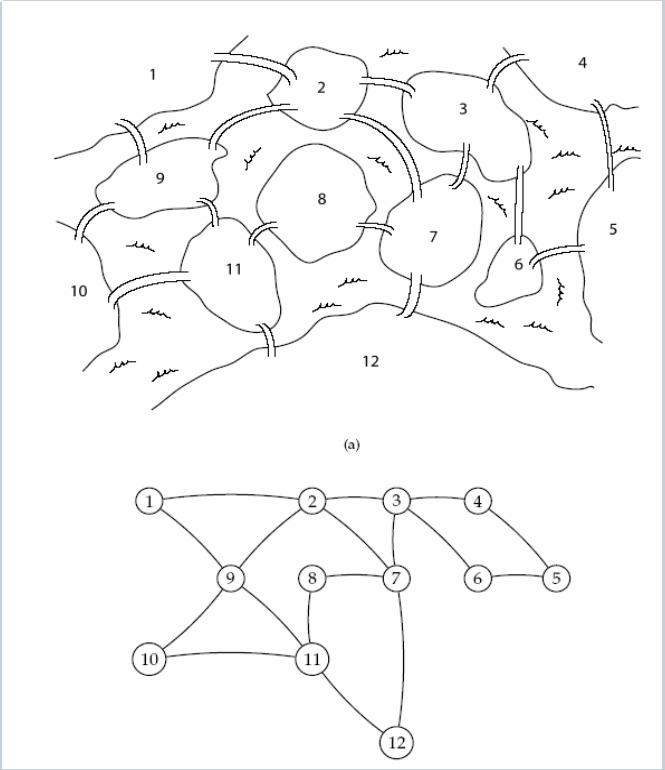
Find a tour crossing every bridge just once
Leonhard Euler, 1735



Bridges of Königsberg

Eulerian Cycle Problem

- Find a cycle that visits every ***edge*** exactly once
- Linear time



More complicated Königsberg

Euler Theorems

- A graph is *balanced* if for every vertex the number of incoming edges equals to the number of outgoing edges:

  $in(v)=out(v)$.

- **Theorem**: A connected graph has an *Eulerian cycle* if and only if each of its vertices is balanced.
- A vertex is *semi-balanced* if $in(v)=out(v)+1$ *or* $in(v)=out(v)-1$
- **Theorem**: A connected graph has an *Eulerian path* if and only if it contains vertex v with $in(v)=out(v)-1$, vertex w with $in(w)=out(w)+1$, and all other vertices are balanced.

Some Difficulties with SBH

- In practice, *l*-mer composition can never be measured with 100% accuracy.
  - With inaccurate data, the computational problem is again NP-hard.
    - Find minimum completion (insertion/deletions edges and nodes) of the graph so that it becomes Eulerian (see exercises)
    - Jacek Błazewicz and Marta Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290(3):1459-1473, 2003
- Microarray technology has found other uses:
  - Widely used in expression analysis and SNP analysis.
- Virtual *l*-mer distributions are used in many fragment assembly tools, leading to heuristics exploiting Eulerian path approach.