Graphs and DNA sequencing

CS 466
Saurabh Sinha
Three problems in graph theory
Eulerian Cycle Problem

- Find a cycle that visits every edge exactly once
- Linear time
Hamiltonian Cycle Problem

• Find a cycle that visits every <i>vertex</i> exactly once

• NP – complete

Game invented by Sir William Hamilton in 1857
Travelling Salesman Problem

- Find the cheapest tour of a given set of cities
- “Cost” associated with going from any city to any other city
- Must visit every city exactly once
- NP-complete
DNA Sequencing
DNA Sequencing

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

• **Computational Challenge**: assemble individual short fragments (reads) into a single genomic sequence ("superstring")
Shortest Superstring Problem

• **Problem:** Given a set of strings, find a shortest string that contains all of them

• **Input:** Strings \( s_1, s_2, \ldots, s_n \)

• **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 – complete
Shortest Superstring Problem: Example

The Shortest Superstring problem

Set of strings: \{000, 001, 010, 011, 100, 101, 110, 111\}

Concatenation

Superstring

000 001 010 011 100 101 110 111

Shortest superstring

0 0 0 1 1 1 0 1 0 0
Shortest Superstring Problem

• Can be framed as Travelling Salesman Problem (TSP):
• Overlap(s_i, s_j) = largest overlap between s_i and s_j
• Complete directed graph with vertices for substrings (s_i) and edge weights being -overlap(s_i, s_j)
Shortest Superstring Problem

• Doesn’t help to cast this as TSP
  – TSP is NP-complete

• Early sequencing algorithms used a greedy approach: merge a pair of strings with maximum overlap first
  – Conjectured to have performance guarantee of 2.
Generating the fragments

Vector
Circular genome (bacterium, plasmid)

DNA fragments

Known location (restriction site)

Cloning (many many copies)
Read Coverage

Length of genomic segment: \( L \)
Number of reads: \( n \)
Length of each read: \( l \)

Coverage \( C = \frac{nI}{L} \)

How much coverage is enough?

**Lander-Waterman model:**
Assuming uniform distribution of reads, \( C=10 \) results in 1 gapped region per 1,000,000 nucleotides
Lander-Waterman Model

• Major Assumptions
  – Reads are randomly distributed in the genome
  – The number of times a base is sequenced follows a Poisson distribution
    \[ p(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \]  
    \( \lambda = LN/G \) (coverage)

• Implications
  – \( G \) = genome length, \( L \) = read length, \( N \) = \# reads
  – Mean of Poisson: \( \lambda = LN/G \) (coverage)
  – % bases not sequenced: \( p(X=0) = 0.0009 = 0.09\% \)
  – Total gap length: \( p(X=0) \times G \)

This model was used to plan the Human Genome Project…
Assemblers

• The first large eukaryotic genomes (fruitfly, human) originally assembled using assembly programs such as the “Celera assembler” and “Arachne”.

• We will see one of these assemblers later in the course
Challenges in Fragment Assembly

- Repeats: A **major** problem for fragment assembly
- > 50% of human genome are repeats
- over 1 million *Alu* repeats (about 300 bp)
- about 200,000 LINE repeats (1000 bp and longer)
Figure 1.5: Misassembly of repeated sequences. Gray arrows indicate repeat copies in the target sequence, gray bars indicate sequence reads sampling the repeat region, black bars indicate reads sampling unique parts. I. Sequence reads sampling different repeat copies appear to overlap. II. The resulting assembly is erroneous, piling reads from different repeat copies. III. The consensus sequence is erroneously computed, with repeat copies merged.

Dealing with repeats

• Use “mate-pair” reads

• Fragments of length ~ L are selected, and both ends are sequenced
  – L >> length of typical repeat

• Reads are now in pairs, separated by approximately known distance (L)

• Both reads of a mate-pair are unlikely to lie in repeat regions

• Using their approximate separation, we can resolve assembly problems
Shotgun Sequencing

cut many times at random (Shotgun)

Get one or two reads from each segment

~500 bp

~500 bp
A completely different sequencing method: Sequencing by Hybridization

- 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 array.

- 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.
How SBH Works (cont’d)

• Using a spectroscopic detector, determine which probes hybridize to the DNA fragment to obtain the $l$–mer composition of the target DNA fragment.

• Apply a combinatorial algorithm to reconstruct the sequence of the target DNA fragment from the $l – $ mer composition.
$l$-mer composition

- **Spectrum** ($s, l$) - *unordered* multiset of all possible $(n - l + 1)$ $l$-mers in a string $s$ of length $n$
- The order of individual elements in Spectrum ($s, l$) does not matter
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 $Spectrum(s,l) = S$

Different from the Shortest Superstring Problem
SBH: Hamiltonian Path Approach

\[ S = \{ \text{ATG, AGG, TGC, TCC, GTC, GGT, GCA, CAG} \} \]

\( H \)

ATG  AGG  TGC  TCC  GTC  GGT  GCA  CAG

Path visited every VERTEX once
SBH: Eulerian Path Approach

\[ S = \{ \text{ATG, TGC, GTG, GGC, GCA, GCG, CGT} \} \]

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
Euler Theorem

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

\[ \text{in}(v) = \text{out}(v) \]

• **Theorem:** A connected graph is Eulerian (has an Eulerian cycle) if and only if each of its vertices is balanced.
Euler Theorem: Proof

• Eulerian $\rightarrow$ balanced
  for every edge entering $v$ (incoming edge) there exists an edge leaving $v$ (outgoing edge). Therefore
  $$in(v)=out(v)$$

• Balanced $\rightarrow$ Eulerian
  ???
Algorithm for Constructing an Eulerian Cycle

a. Start with an arbitrary vertex \( v \) and form an arbitrary cycle with unused edges until a dead end is reached. Since the graph is Eulerian this dead end is necessarily the starting point, i.e., vertex \( v \).
Algorithm for Constructing an Eulerian Cycle (cont’d)

b. If cycle from (a) is not an Eulerian cycle, it must contain a vertex \( w \), which has untraversed edges. Perform step (a) again, using vertex \( w \) as the starting point. Once again, we will end up in the starting vertex \( w \).
Algorithm for Constructing an Eulerian Cycle (cont’d)

c. Combine the cycles from (a) and (b) into a single cycle and iterate step (b).
SBH as Eulerian Path Problem

- A vertex $v$ is “semibalanced” if $|\text{in-degree}(v) - \text{out-degree}(v)| = 1$

- If a graph has an Eulerian path starting from $s$ and ending at $t$, then all its vertices are balanced *with the possible exception of $s$ and $t*

- Add an edge between two semibalanced vertices: now all vertices should be balanced (assuming there was an Eulerian path to begin with). Find the Eulerian cycle, and remove the edge you had added. You now have the Eulerian path you wanted.