CS481: Bioinformatics Algorithms

Can Alkan
EA224
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/
GENOME
REARRANGEMENTS
Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:
Turnip vs Cabbage: Different mtDNA Gene Order

- Gene order comparison:

![Gene order comparison diagram]
Gene order comparison:
Turnip vs Cabbage: Different mtDNA Gene Order

Gene order comparison:
Turnip vs Cabbage: Different mtDNA Gene Order

Gene order comparison:

Before

After

Evolution is manifested as the divergence in gene order
Transforming Cabbage into Turnip

*B. oleracea* (cabbage)

*B. campestris* (turnip)
Types of Rearrangements

**Reversal**

1 2 3 4 5 6 → 1 2 -5 -4 -3 6

**Translocation**

1 2 3 → 1 2 6
4 5 6 → 4 5 3

**Fusion**

1 2 3 4 → 1 2 3 4 5 6
5 6 → 5 6
Reversals: Example

\[ \pi = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \]

\[ \rho(3,5) \]

\[ 1 \ 2 \ 5 \ 4 \ 3 \ 6 \ 7 \ 8 \]
Reversals: Example

\[ \pi = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \]

\[ \rho(3,5) \]

\[ 1 \ 2 \ 5 \ 4 \ 3 \ 6 \ 7 \ 8 \]

\[ \rho(5,6) \]

\[ 1 \ 2 \ 5 \ 4 \ 6 \ 3 \ 7 \ 8 \]
Reversals and Gene Orders

- Gene order is represented by a permutation $\pi$:

$$\pi = \pi_1 \ldots \pi_{i-1} \pi_i \pi_{i+1} \ldots \pi_{j-1} \pi_j \pi_{j+1} \ldots \pi_n$$

- Reversal $\rho(i,j)$ reverses (flips) the elements from $i$ to $j$ in $\pi$:

$$\rho(i,j)$$

$$\pi_1 \ldots \pi_{i-1} \pi_j \pi_{j-1} \ldots \pi_{i+1} \pi_i \pi_{j+1} \ldots \pi_n$$
Reversal Distance Problem

- **Goal**: Given two permutations, find the shortest series of reversals that transforms one into another

- **Input**: Permutations $\pi$ and $\sigma$

- **Output**: A series of reversals $\rho_1, \ldots, \rho_t$ transforming $\pi$ into $\sigma$, such that $t$ is minimum

- $t$ - reversal distance between $\pi$ and $\sigma$

- $d(\pi, \sigma)$ - smallest possible value of $t$, given $\pi$ and $\sigma$
Sorting By Reversals Problem

- **Goal**: Given a permutation, find a shortest series of reversals that transforms it into the identity permutation (1 2 … n)

- **Input**: Permutation \( \pi \)

- **Output**: A series of reversals \( \rho_1, \ldots, \rho_t \) transforming \( \pi \) into the identity permutation such that \( t \) is minimum
Sorting By Reversals: Example

- \( t = d(\pi) \) - reversal distance of \( \pi \)
- Example:

\[
\pi = \begin{array}{cccccccccc}
3 & 4 & 2 & 1 & 5 & 6 & 7 & 10 & 9 & 8 \\
4 & 3 & 2 & 1 & 5 & 6 & 7 & \underline{10} & 9 & 8 \\
4 & 3 & 2 & 1 & 5 & 6 & 7 & 8 & 9 & 10 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10
\end{array}
\]

So \( d(\pi) = 3 \)
Sorting by reversals: 5 steps

Step 0:
2 -4 -3 5 -8 -7 -6 1

Step 1:
2 3 4 5 -8 -7 -6 1

Step 2:
2 3 4 5 6 7 8 1

Step 3:
2 3 4 5 6 7 8 -1

Step 4:
-8 -7 -6 -5 -4 -3 -2 -1

Step 5:
1 2 3 4 5 6 7 8
Sorting by reversals: 4 steps

Step 0: \( \pi \)

\[ \begin{array}{cccccccc}
2 & -4 & -3 & 5 & -8 & -7 & -6 & 1 \\
\end{array} \]

Step 1:

\[ \begin{array}{cccccccc}
2 & 3 & 4 & 5 & -8 & -7 & -6 & 1 \\
\end{array} \]

Step 2:

\[ \begin{array}{cccccccc}
-5 & -4 & -3 & -2 & -8 & -7 & -6 & 1 \\
\end{array} \]

Step 3:

\[ \begin{array}{cccccccc}
-5 & -4 & -3 & -2 & -1 & 6 & 7 & 8 \\
\end{array} \]

Step 4: \( \gamma \)

\[ \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]
Pancake Flipping Problem

- The chef is sloppy; he prepares an unordered stack of pancakes of different sizes.
- The waiter wants to rearrange them (so that the smallest winds up on top, and so on, down to the largest at the bottom).
- He does it by flipping over several from the top, repeating this as many times as necessary.

[Image: Christos Papadimitrou and Bill Gates flip pancakes]
Pancake Flipping Problem: Formulation

- **Goal**: Given a stack of $n$ pancakes, what is the minimum number of flips to rearrange them into perfect stack?
- **Input**: Permutation $\pi$
- **Output**: A series of prefix reversals $\rho_1, \ldots, \rho_t$ transforming $\pi$ into the identity permutation such that $t$ is minimum
Pancake Flipping Problem: Greedy Algorithm

- Greedy approach: 2 prefix reversals at most to place a pancake in its right position, $2n - 2$ steps total at most
- William Gates and Christos Papadimitriou showed in the mid-1970s that this problem can be solved by at most $\frac{5}{3} (n + 1)$ prefix reversals
Sorting By Reversals: A Greedy Algorithm

- If sorting permutation \( \pi = 1 \ 2 \ 3 \ 6 \ 4 \ 5 \), the first three elements are already in order so it does not make any sense to break them.
- The length of the already sorted prefix of \( \pi \) is denoted \( \text{prefix}(\pi) \)
  - \( \text{prefix}(\pi) = 3 \)
- This results in an idea for a greedy algorithm: increase \( \text{prefix}(\pi) \) at every step
Greedy Algorithm: An Example

- Doing so, $\pi$ can be sorted

1 2 3 6 4 5

1 2 3 4 6 5

1 2 3 4 5 6

- Number of steps to sort permutation of length $n$ is at most $(n - 1)$
Greedy Algorithm: Pseudocode

SimpleReversalSort(\(\pi\))
1 for \(i \leftarrow 1\) to \(n - 1\)
2 \(j \leftarrow\) position of element \(i\) in \(\pi\) (i.e., \(\pi_j = i\))
3 if \(j \neq i\)
4 \(\pi \leftarrow \pi \cdot \rho(i, j)\)
5 output \(\pi\)
6 if \(\pi\) is the identity permutation
7 return
Analyzing SimpleReversalSort

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5$:
  
  - Step 1: $1 \ 6 \ 2 \ 3 \ 4 \ 5$
  - Step 2: $1 \ 2 \ 6 \ 3 \ 4 \ 5$
  - Step 3: $1 \ 2 \ 3 \ 6 \ 4 \ 5$
  - Step 4: $1 \ 2 \ 3 \ 4 \ 6 \ 5$
  - Step 5: $1 \ 2 \ 3 \ 4 \ 5 \ 6$
But it can be sorted in two steps:
\[\pi = 6\ 1\ 2\ 3\ 4\ 5\]
- Step 1: 5 4 3 2 1 6
- Step 2: 1 2 3 4 5 6

So, SimpleReversalSort(\(\pi\)) is not optimal

Optimal algorithms are unknown for many problems; approximation algorithms are used
Approximation Algorithms

- These algorithms find approximate solutions rather than optimal solutions.
- The approximation ratio of an algorithm $A$ on input $\pi$ is:
  \[
  \frac{A(\pi)}{OPT(\pi)}
  \]
  where
  - $A(\pi)$ - solution produced by algorithm $A$
  - $OPT(\pi)$ - optimal solution of the problem
Approximation Ratio/Performance Guarantee

- Approximation ratio (performance guarantee) of algorithm A: max approximation ratio of all inputs of size $n$
  - For algorithm A that minimizes objective function (minimization algorithm):
    - $\max |\pi| = \frac{nA(\pi)}{OPT(\pi)}$
Approximation Ratio/Performance Guarantee

- Approximation ratio (performance guarantee) of algorithm A: max approximation ratio of all inputs of size $n$
  - For algorithm A that minimizes objective function (minimization algorithm):
    - $\max_{\pi} = n \frac{A(\pi)}{OPT(\pi)}$
  - For maximization algorithm:
    - $\min_{\pi} = n \frac{A(\pi)}{OPT(\pi)}$
Adjacencies and Breakpoints

\[ \pi = \pi_1 \pi_2 \pi_3 \ldots \pi_{n-1} \pi_n \]

- A pair of elements \( \pi_i \) and \( \pi_{i+1} \) are adjacent if

\[ \pi_{i+1} = \pi_i \pm 1 \]

- For example:

\[ \pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \]

- (3, 4) or (7, 8) and (6,5) are adjacent pairs
There is a breakpoint between any adjacent element that are non-consecutive:

\[ \pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \]

- Pairs \((1,9), (9,3), (4,7), (8,2)\) and \((2,6)\) form breakpoints of permutation \(\pi\)
- \(b(\pi)\) - # breakpoints in permutation \(\pi\)
Adjacency & Breakpoints

• An **adjacency** - a pair of adjacent elements that are **consecutive**

• A **breakpoint** - a pair of adjacent elements that are **not** consecutive

\[ \pi = 5 \ 6 \ 2 \ 1 \ 3 \ 4 \]

Extend \( \pi \) with \( \pi_0 = 0 \) and \( \pi_7 = 7 \)

---

\[ 0 \ 5 \ 6 \ 2 \ 1 \ 3 \ 4 \ 7 \]

### Adjacencies

\[ \uparrow \ \uparrow \ \uparrow \ \uparrow \]

### Breakpoints

\[ \downarrow \ \downarrow \ \downarrow \ \downarrow \]
Extending Permutations

- We put two elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$ at the ends of $\pi$

Example:

$$\pi = 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5$$

Extending with 0 and 10

$$\pi = 0 \ 1 \ 9 \ 3 \ 4 \ 7 \ 8 \ 2 \ 6 \ 5 \ 10$$

Note: A new breakpoint was created after extending
Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.

\[ \pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5 \]

\[
\begin{array}{c|c|c|c|c|c|c|c}
0 & 2 & 3 & 1 & 4 & 6 & 5 & 7 \\
\hline
0 & 1 & 3 & 2 & 4 & 6 & 5 & 7 \\
\hline
0 & 1 & 2 & 3 & 4 & 6 & 5 & 7 \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

- \( b(\pi) = 5 \)
- \( b(\pi) = 4 \)
- \( b(\pi) = 2 \)
- \( b(\pi) = 0 \)
Reversal Distance and Breakpoints

- Each reversal eliminates at most 2 breakpoints.
- This implies:

\[
\text{reversal distance} \geq \frac{\text{#breakpoints}}{2}
\]

\[
\pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5
\]

\[
\begin{array}{c|cccccc}
0 & 2 & 3 & 1 & 4 & 6 & 5 & 7 \\
\end{array}
\]

\[
b(\pi) = 5
\]

\[
\begin{array}{c|cccccc}
0 & 1 & 3 & 2 & 4 & 6 & 5 & 7 \\
\end{array}
\]

\[
b(\pi) = 4
\]

\[
\begin{array}{c|cccccc}
0 & 1 & 2 & 3 & 4 & 6 & 5 & 7 \\
\end{array}
\]

\[
b(\pi) = 2
\]

\[
\begin{array}{c|cccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array}
\]

\[
b(\pi) = 0
\]
Sorting By Reversals: A Better Greedy Algorithm

BreakPointReversalSort(\(\pi\))
1  while \(b(\pi) > 0\)
2  Among all possible reversals, choose reversal \(\rho\) minimizing \(b(\pi \cdot \rho)\)
3  \(\pi \leftarrow \pi \cdot \rho(i, j)\)
4  output \(\pi\)
5  return
Sorting By Reversals: A Better Greedy Algorithm

**BreakPointReversalSort**($\pi$)

1. while $b(\pi) > 0$
2. Among all possible reversals, choose reversal $\rho$ minimizing $b(\pi \cdot \rho)$
3. $\pi \leftarrow \pi \cdot \rho(i, j)$
4. output $\pi$
5. return

Problem: this algorithm may work forever
Strips

- **Strip**: an interval between two consecutive breakpoints in a permutation
  - **Decreasing strip**: strip of elements in decreasing order (e.g. 6 5 and 3 2).
  - **Increasing strip**: strip of elements in increasing order (e.g. 7 8)

- A single-element strip can be declared either increasing or decreasing. We will choose to declare them as decreasing with exception of the strips with 0 and $n+1$
Reducing the Number of Breakpoints

Theorem 1:

If permutation $\pi$ contains at least one decreasing strip, then there exists a reversal $\rho$ which decreases the number of breakpoints (i.e. $b(\pi \cdot \rho) < b(\pi)$)
Things To Consider

- For $\pi = 1 \, 4 \, 6 \, 5 \, 7 \, 8 \, 3 \, 2$
  
  $\begin{array}{cccccc}
  0 & 1 & 4 & 6 & 5 & 7 \hline
  8 & 3 & 2 & 9
  \end{array}$

  \[ b(\pi) = 5 \]

- Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)
Things To Consider (cont’d)

- For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$
  
    \[
    0\ 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2\ 9 \quad b(\pi) = 5
    \]

- Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)
Things To Consider (cont’d)

For \( \pi = 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \)

\[
\begin{array}{cccccccc}
0 & 1 & 4 & 6 & 5 & 7 & 8 & 3 & 2 & 9 \\
\end{array}
\]

\( b(\pi) = 5 \)

- Choose decreasing strip with the smallest element \( k \) in \( \pi \) (\( k = 2 \) in this case)
- Find \( k - 1 \) in the permutation
Things To Consider (cont’d)

- For \( \pi = 1 \ 4 \ 6 \ 5 \ 7 \ 8 \ 3 \ 2 \)
  
  \[
  \begin{array}{cccccccc}
  0 & 1 & 4 & 6 & 5 & 7 & 8 & 3 & 2 & 9 \\
  \end{array}
  \]

  \( b(\pi) = 5 \)

  - Choose decreasing strip with the smallest element \( k \) in \( \pi \) (\( k = 2 \) in this case)
  - Find \( k - 1 \) in the permutation
  - Reverse the segment between \( k \) and \( k - 1 \):

    \[
    \begin{array}{cccccccc}
    0 & 1 & \underline{2} & 3 & 8 & 7 & 5 & 6 & 4 & 9 \\
    \end{array}
    \]

    \( b(\pi) = 4 \)
Reducing the Number of Breakpoints Again

- If there is no decreasing strip, there may be no reversal $\rho$ that reduces the number of breakpoints (i.e. $b(\pi \cdot \rho) \geq b(\pi)$ for any reversal $\rho$).

- By reversing an increasing strip ( # of breakpoints stay unchanged ), we will create a decreasing strip at the next step. Then the number of breakpoints will be reduced in the next step (theorem 1).
Things To Consider (cont’d)

- There are no decreasing strips in $\pi$, for:

$$
\pi = 0 \ 1 \ 2 \ 5 \ 6 \ 7 \ 3 \ 4 \ 8 \quad b(\pi) = 3
$$

$$
\pi \cdot \rho(6,7) = 0 \ 1 \ 2 \ 5 \ 6 \ 7 \ 4 \ 3 \ 8 \quad b(\pi) = 3
$$

✓ $\rho(6,7)$ does not change the # of breakpoints
✓ $\rho(6,7)$ creates a decreasing strip thus guaranteeing that the next step will decrease the # of breakpoints.
ImprovedBreakpointReversalSort

ImprovedBreakpointReversalSort(\pi)
1 \textbf{while} \ b(\pi) > 0
2 \textbf{if} \ \pi \ \text{has a decreasing strip}
3 \quad \text{Among all possible reversals, choose reversal} \ \rho
4 \quad \text{that minimizes} \ b(\pi \cdot \rho)
5 \textbf{else}
6 \quad \text{Choose a reversal} \ \rho \ \text{that flips an increasing strip in} \ \pi
7 \quad \pi \leftarrow \pi \cdot \rho
8 \text{output} \ \pi
9 \text{return}
ImprovedBreakpointReversalSort: Performance Guarantee

- *ImprovedBreakPointReversalSort* is an approximation algorithm with a performance guarantee of at most 4
  - It eliminates at least one breakpoint in every two steps; at most $2b(\pi)$ steps
  - Approximation ratio: $2b(\pi) / d(\pi)$
  - Optimal algorithm eliminates at most 2 breakpoints in every step: $d(\pi) \geq b(\pi) / 2$
  - Performance guarantee:
    - $(2b(\pi) / d(\pi)) \geq [2b(\pi) / (b(\pi) / 2)] = 4$
GRAPHS
1) Represent the elements of the permutation $\pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5$ as vertices in a graph (ordered along a line)

2) Connect vertices in order given by $\pi$ with black edges (black path)

3) Connect vertices in order given by $1 \ 2 \ 3 \ 4 \ 5 \ 6$ with grey edges (grey path)

4) Superimpose black and grey paths
Two Equivalent Representations of the Breakpoint Graph

- Consider the following Breakpoint Graph
- If we line up the gray path (instead of black path) on a horizontal line, then we would get the following graph
- Although they may look different, these two graphs are the same
What is the Effect of the Reversal?

How does a reversal change the breakpoint graph?

- The gray paths stayed the same for both graphs
- There is a change in the graph at this point
- There is another change at this point
- The black edges are unaffected by the reversal so they remain the same for both graphs

Before: 0 2 3 1 4 6 5 7

After: 0 2 3 5 6 4 1 7
A reversal affects 4 edges in the breakpoint graph

- A reversal removes 2 edges (red) and replaces them with 2 new edges (blue)
Effects of Reversals

Case 1:

Both edges belong to the same cycle

- Remove the center black edges and replace them with new black edges (there are two ways to replace them)
- (a) After this replacement, there now exists 2 cycles instead of 1 cycle
- (b) Or after this replacement, there still exists 1 cycle

\[ c(\pi \rho) - c(\pi) = 0 \text{ or } 1 \]

Therefore, after the reversal

This is called a proper reversal since there’s a cycle increase after the reversal.
Effects of Reversals (Continued)

Case 2:
Both edges belong to different cycles
• Remove the center black edges and replace them with new black edges
• After the replacement, there now exists 1 cycle instead of 2 cycles

Therefore, for every permutation $\pi$ and reversal $\rho$, $c(\pi \rho) - c(\pi) \leq 1$
Identity permutation \((n=6)\)
Reversal Distance and Maximum Cycle Decomposition

• Since the identity permutation of size $n$ contains the maximum cycle decomposition of $n+1$, $c(\text{identity}) = n+1$

• $c(\text{identity}) - c(\pi)$ equals the number of cycles that need to be “added” to $c(\pi)$ while transforming $\pi$ into the identity

• Based on the previous theorem, at best after each reversal, the cycle decomposition could be increased by one, then:
  
  \[ d(\pi) = c(\text{identity}) - c(\pi) = n+1 - c(\pi) \]

• Yet, not every reversal can increase the cycle decomposition

Therefore, $d(\pi) \geq n+1 - c(\pi)$
Signed Permutations

- Up to this point, all permutations to sort were unsigned
- But genes have directions… so we should consider signed permutations

\[ \pi = 1 \quad -2 \quad -3 \quad 4 \quad -5 \]
Signed Permutation

• Genes are *directed* fragments of DNA and we represent a genome by a signed permutation

• If genes are in the same position but there orientations are different, they do not have the equivalent gene order

• For example, these two permutations have the same order, but each gene’s orientation is the reverse; therefore, they are not equivalent gene sequences
From Signed to Unsigned Permutation

- Begin by constructing a normal signed breakpoint graph
- Redefine each vertex $x$ with the following rules:
  - If vertex $x$ is positive, replace vertex $x$ with vertex $2x-1$ and vertex $2x$ in that order
  - If vertex $x$ is negative, replace vertex $x$ with vertex $2x$ and vertex $2x-1$ in that order
  - The extension vertices $x = 0$ and $x = n+1$ are kept as it was before
• Construct the breakpoint graph as usual
• Notice the alternating cycles in the graph between every other vertex pair
• Since these cycles came from the same signed vertex, we will not be performing any reversal on both pairs at the same time; therefore, these cycles can be removed from the graph
Interleaving Edges

- Interleaving edges are grey edges that cross each other

*Example: Edges (0,1) and (18, 19) are interleaving*

- Cycles are interleaving if they have an interleaving edge
An Interleaving Graph is defined on the set of cycles in the Breakpoint graph and are connected by edges where cycles are interleaved.
• Oriented cycles are cycles that have the following form
• Mark them on the interleave graph
• Unoriented cycles are cycles that have the following form
• In our example, A, B, D, E are unoriented cycles while C, F are oriented cycles
Hurdles

- Remove the oriented components from the interleaving graph
- The following is the breakpoint graph with these oriented components removed
- Hurdles are connected components that do not contain any other connected components within it
Reversal Distance with Hurdles

• Hurdles are obstacles in the genome rearrangement problem

• They cause a higher number of required reversals for a permutation to transform into the identity permutation

• Let $h(\pi)$ be the number of hurdles in permutation $\pi$

• Taking into account of hurdles, the following formula gives a tighter bound on reversal distance:

\[ d(\pi) \geq n + 1 - c(\pi) + h(\pi) \]