Greedy Algorithms
and
Genome Rearrangements
Outline

1. Transforming Cabbage into Turnip
2. Genome Rearrangements
3. Sorting By Reversals
4. Pancake Flipping Problem
5. Greedy Algorithm for Sorting by Reversals
6. Approximation Algorithms
7. Breakpoints: a Different Face of Greed
8. Breakpoint Graphs
Motivating Example: Turnip vs. Cabbage

- Although cabbages and turnips share a recent common ancestor, they look and taste differently.
Motivating Example: Turnip vs. Cabbage

- In the 1980s Jeffrey Palmer studied evolution of plant organelles by comparing mitochondrial genomes of cabbage and turnip.

- He found 99% similarity between genes.

- These surprisingly similar gene sequences differed in gene order.

- This study helped pave the way to analyzing genome rearrangements in molecular evolution.
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

- Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:
Motivating Example: Turnip vs. Cabbage

• Gene order comparison:

Before

After

• Evolution is manifested as the divergence in gene order and orientation caused by these inversions of segments of genes
Another Example: Mice and Humans

• The X chromosomes of mice and humans give another example.

• What are the similarity blocks and how do we find them?

• What is the ordering of the blocks?

• What is the scenario for transforming one genome into another?
History of Chromosome X

Rat Consortium, Nature, 2004
Reversals: Example

Break and Invert

5’ ATGCCCTGTA CTA 3’
3’ TACGGACATGAT 5’

5’ ATGTACAGGCTA 3’
3’ TACATGTCCCGAT 5’
Reversals

- Blocks represent conserved genes.

1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Reversals

- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by
Reversals

- Blocks represent conserved genes.
- A reversal introduces two breakpoints, represented by ⚡.
- As a result of the reversal, the gene ordering has changed to 1, 2, 3, 8, 7, 6, 5, 4, 9, 10.
Reversals: A Formal Motivation

• A sequence of $n$ genes is represented by a permutation $\pi$, where a permutation is an ordering of the integers 1 to $n$:

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

$$\rho(i,j)$$

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

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

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

\[ \rho(3,5) \]

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

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

\[ \rho(3,5) \]

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

\[ \rho(5,6) \]

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

• The minimal such \( t \) is called the **reversal distance** between \( \pi \) and \( \sigma \) and is often written as \( d(\pi, \sigma) \).
Sorting By Reversals Problem

• The most natural permutation is the identity, $I = (1\ 2\ 3\ \ldots\ n)$

• **Sorting By Reversals Problem**: Given a permutation, find a shortest series of reversals that transforms it into the identity.

• **Input**: Permutation $\pi$

• **Output**: A series of reversals $\rho_i, \ldots, \rho_t$ transforming $\pi$ into the identity permutation such that $t$ is minimum. We call the minimum such $t$ the “reversal distance” of $\pi$, denoted $d(\pi)$.
Sorting by Reversals: Example

• Say we are given $\pi = (2 \ 4 \ 3 \ 5 \ 8 \ 7 \ 6 \ 1)$

• We can sort $\pi$ in four steps as follows:

  Step 0: $\pi = (2 \ 4 \ 3 \ 5 \ 8 \ 7 \ 6 \ 1)$
  Step 1: $(2 \ 3 \ 4 \ 5 \ 8 \ 7 \ 6 \ 1)$
  Step 2: $(5 \ 4 \ 3 \ 2 \ 8 \ 7 \ 6 \ 1)$
  Step 3: $(5 \ 4 \ 3 \ 2 \ 1 \ 6 \ 7 \ 8)$
  Step 4: $(1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8)$

• But can we sort $\pi$ in three steps? Two? How can we know?
Pancake Flipping Problem

- A chef is sloppy; he prepares an unordered stack of pancakes of different sizes.
- The waiter wants to quickly rearrange them (so that the smallest winds up on top, down to the largest at the bottom).
- He does it by flipping over several from the top of the stack.
Pancake Flipping Problem

- **Goal**: Given a stack of $n$ pancakes, what is the minimum number of flips needed to rearrange them into the perfect stack?

- **Input**: Stack of $n$ pancakes

- **Output**: A minimal sequence of flips transforming the stack into the perfect stack.

- this formulation isn’t very mathematical…
Pancake Flipping Problem

- Label the smallest pancake as 1, the biggest by \( n \), etc.;
- A stack of pancakes is represented by a permutation.

- A flip of the stack is a special reversal ("prefix reversal")
  - must involve the first element.

- With this mathematical framework, restate the pancake flipping problem as the "Sorting by Prefix Reversals Problem."
Sorting by Prefix Reversals

• **Sorting by Prefix Reversals Problem**: Given a permutation $\pi$, find the shortest sequence of *prefix reversals* transforming $\pi$ into the identity permutation.

• **Input**: Permutation $\pi$

• **Output**: A series of prefix reversals $\rho_1, \ldots, \rho_t$ transforming $\pi$ into the identity permutation and such that $t$ is minimized.
Pancake Flipping Problem: Greedy Algorithm

- Greedy approach: at most 2 prefix reversals at most to place the biggest pancake on the bottom, at most 2 prefix reversals to place the second-biggest pancake in the second position from the bottom…
- This results in an algorithm which requires $2n - 2$ prefix reversals.
- Bill Gates (!) and Christos Papadimitriou showed in the mid-1970s that this problem can actually be solved by at most $5/3 (n + 1)$ prefix reversals.

Note: This means that our greedy method is “close” to optimal.
Sorting By Reversals: A Greedy Algorithm

• If the first three elements in permutation \( \pi = 1 \ 2 \ 3 \ 6 \ 4 \ 5 \) are already in order, it does not make any sense to break them.

• The length of the already sorted prefix of \( \pi \) is denoted \( \text{prefix}(\pi) \)
  • In the example above, \( \text{prefix}(\pi) = 3 \)

• Idea for a greedy algorithm: increase \( \text{prefix}(\pi) \) at every step.
Greedy Algorithm: Example

• Doing so, $\pi$ can be sorted in two steps:

1 2 3 6 4 5

\[ \rightarrow \]

1 2 3 4 6 5

\[ \rightarrow \]

1 2 3 4 5 6

• The number of steps needed to sort a permutation of length $n$ is at most $(n - 1)$.

Note: Why is it $n - 1$ and not $n$? Think about the final step...
Greedy Algorithm: Pseudocode

```
SINGLEREVERSALSORT(\pi)
1   for i ← 1 to n − 1
2     j ← position of element i in \pi (i.e., \pi_j = i)
3     if j ≠ i
4       \pi ← \pi \cdot \rho(i, j)
5     output \pi
6     if \pi is the identity permutation
7     return
```
SimpleReversalSort: Analysis

• SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5$:
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5$:
  - Step 0: 6 1 2 3 4 5
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on \( \pi = 6\ 1\ 2\ 3\ 4\ 5 \):
  - Step 0: 6\ 1\ 2\ 3\ 4\ 5
  - Step 1: 1\ 6\ 2\ 3\ 4\ 5
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $6\ 1\ 2\ 3\ 4\ 5$
  - Step 1: $1\ 6\ 2\ 3\ 4\ 5$
  - Step 2: $1\ 2\ 6\ 3\ 4\ 5$
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6\ 1\ 2\ 3\ 4\ 5$:
  - Step 0: $6\ 1\ 2\ 3\ 4\ 5$
  - Step 1: $1\ 6\ 2\ 3\ 4\ 5$
  - Step 2: $1\ 2\ 6\ 3\ 4\ 5$
  - Step 3: $1\ 2\ 3\ 6\ 4\ 5$
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on \( \pi = 6 \ 1 \ 2 \ 3 \ 4 \ 5 \):
  - Step 0: 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
SimpleReversalSort: Analysis

- SimpleReversalSort does not guarantee the smallest number of reversals and takes five steps on $\pi = 6 1 2 3 4 5$:
  - Step 0: $6 \underline{1} 2 3 4 5$
  - Step 1: $1 \underline{6} 2 3 4 5$
  - Step 2: $1 2 \underline{6} 3 4 5$
  - Step 3: $1 2 3 \underline{6} 4 5$
  - Step 4: $1 2 3 4 \underline{6} 5$
  - Step 5: $1 2 3 4 5 \underline{6}$
SimpleReversalSort: Analysis

• But $\pi$ can be sorted in two steps:
SimpleReversalSort: Analysis

• But $\pi$ can be sorted in two steps:
  • Step 0: 6 1 2 3 4 5
SimpleReversalSort: Analysis

• But $\pi$ can be sorted in two steps:
  • Step 0: 6 1 2 3 4 5
  • Step 1: 5 4 3 2 1 6
SimpleReversalSort: Analysis

• But $\pi$ can be sorted in two steps:
  • Step 0: $6 \ 1 \ 2 \ 3 \ 4 \ 5$
  • Step 1: $5 \ 4 \ 3 \ 2 \ 1 \ 6$
  • Step 2: $1 \ 2 \ 3 \ 4 \ 5 \ 6$
Approximation Algorithms

• SimpleReversalSort is not optimal for sorting by reversals.

• When there is no optimal algorithm, use approximation algorithms instead
  – find good approximate solutions rather than optimal ones.

• Define the Approximation Ratio of an algorithm A on input π as:
  \[
  \frac{A(\pi)}{OPT(\pi)}
  \]
  where \( A(\pi) = \) solution produced by algorithm A
  \( OPT(\pi) = \) optimal solution of the problem
Approximation Ratio/Performance Guarantee

- The approximation ratio (performance guarantee) of a minimization algorithm $A$ is the maximum approximation ratio of all inputs of size $n$.

- Formally, for a minimization algorithm $A$, the approximation ratio of $A$ is as follows:

$$R = \max_{\pi = n} \frac{A(\pi)}{OPT(\pi)}$$

- **Big Question**: What is the best approximation ratio we can find for sorting by reversals?

*Note*: What is the approximation ratio for our greedy algorithm for pancake flipping?
Adjacencies and Breakpoints

• Let $\pi = \pi_1 \pi_2 \pi_3 \ldots \pi_{n-1} \pi_n$ be a permutation. A pair of elements $\pi_i$ and $\pi_{i+1}$ is called an adjacency if

$$\pi_{i+1} = \pi_i \pm 1$$

The remaining pairs are called breakpoints.

• Example:

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

• (3, 4), (7, 8) and (6,5) are adjacent pairs
• (1,9), (9,3), (4,7), (8,2) and (2,6) are breakpoints.
Extending Permutations

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

- Example:

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

Extending with 0 and 10

\[
\pi = 0 \mid 1 \mid 9 \mid 3 \mid 4 \mid 7 \mid 8 \mid 2 \mid 6 \mid 5 \mid 10
\]

- Note that after extension, both a new breakpoint (5, 10) and a new adjacency (0, 1) have been created.
Sorting by Reversals = Breakpoint Elimination

• Define $b(\pi)$ to be the number of breakpoints in the extension of $\pi$.

• Note that sorting by reversals appears to correspond to removal of breakpoints.
  
  • Example:
    
    $\pi = 2\ 3\ 1\ 4\ 6\ 5$
    
    $0 | 2\ 3\ 1\ | 4\ 6\ 5\ | 7$  \quad b(\pi) = 5
    
    $0 | 1\ 3\ 2\ | 4\ 6\ 5\ | 7$  \quad b(\pi) = 4
    
    $0 | 1\ 2\ 3\ 4\ | 6\ 5\ | 7$  \quad b(\pi) = 2
    
    $0 | 1\ 2\ 3\ 4\ 5\ 6\ 7$  \quad b(\pi) = 0
## Sorting by Reversals = Breakpoint Elimination

- How many breakpoints can each reversal eliminate?

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>6</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b(\pi)$</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 3 1 4 6 5 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(\pi)$</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 3 2 4 6 5 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(\pi)$</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 6 5 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b(\pi)$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sorting by Reversals = Breakpoint Elimination

• How many breakpoints can each reversal eliminate?
• Each reversal eliminates at most 2 breakpoints.

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

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

\[ b(\pi) = 5 \quad b(\pi) = 4 \quad b(\pi) = 2 \quad b(\pi) = 0 \]
Sort by Reversals = Breakpoint Elimination

• How many breakpoints can each reversal eliminate?
• Each reversal eliminates at most 2 breakpoints.
• This implies:
  • \( d(\pi) \geq b(\pi) / 2 \)
  • We will use this idea to create a new greedy algorithm.

\[
\pi = 2 \ 3 \ 1 \ 4 \ 6 \ 5 \\
0 \underline{2} \underline{3} \ 1 \ 4 \ 6 \ 5 \ 7 \quad b(\pi) = 5 \\
0 \ 1 \underline{3} \underline{2} \ 4 \ 6 \ 5 \ 7 \quad b(\pi) = 4 \\
0 \ 1 \ 2 \ 3 \ 4 \underline{6} \underline{5} \ 7 \quad b(\pi) = 2 \\
0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \quad b(\pi) = 0
\]
A Better Greedy Algorithm

**BREAKPOINTREVERSALSORT**(\(\pi\))

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

- **Problem**: this algorithm may never terminate.
Strips

- **Strip**: an interval between two consecutive breakpoints in a permutation.
- **Decreasing strip**: a strip of elements in decreasing order (e.g. 6 5 4 and 3 2).
- **Increasing strip**: a strip of elements in increasing order (e.g. 6 7 8 9)

```
0 1 9 4 3 7 8 2 5 6 10
```

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

• **Theorem 1**: If a permutation $\pi$ contains at least one decreasing strip, then there exists a reversal $\rho$ which decreases the number of breakpoints (i.e. $b(\pi \cdot \rho) < b(\pi)$).

• We will use this result to help adapt our algorithm to one which is guaranteed to terminate.
Idea for New Algorithm

• For $\pi = 1 4 6 5 7 8 3 2$
  
  $\begin{array}{l}
    \hline \\
    0 & 1 & 4 & 6 & 5 & 7 & 8 & 3 & 2 & 9 \\
    \hline 
  \end{array}$  
  $b(\pi) = 5$

• Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)
Idea for New Algorithm

- For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$
  
  $\begin{array}{c|c|c|c|c|c|c|c|c|c}
  0 & 1 & 4 & 6 & 5 & 7 & 8 & 3 & 2 & 9 \\
  \end{array}$
  $b(\pi) = 5$

- Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)

- Find $k - 1$ in $\pi$
Idea for New Algorithm

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

• Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)

• Find $k - 1$ in $\pi$

• Reverse the segment between $k - 1$ and $k$:
  \[ 0 \ 1 \ 2 \ 3 \ 8 \ 7 \ 5 \ 6 \ 4 \ 9 \]  \[ b(\pi) = 4 \]
Idea for New Algorithm

• For $\pi = 1\ 4\ 6\ 5\ 7\ 8\ 3\ 2$
  $b(\pi) = 5$
  
• Choose decreasing strip with the smallest element $k$ in $\pi$ ($k = 2$ in this case)

• Find $k - 1$ in $\pi$

• Reverse the segment between $k - 1$ and $k$:
  $b(\pi) = 5$

  0 1 4 6 5 7 8 3 2 9

  0 1 2 3 8 7 5 6 4 9
  $b(\pi) = 4$

• This gives us a way of decreasing the number of breakpoints, but what if there are no decreasing strips?
The Case of No Decreasing Strips

• If there is no decreasing strip, there may be no reversal that reduces the number of breakpoints
  • \( b(\pi \cdot \rho) \geq b(\pi) \) for any reversal \( \rho \)
• Reverse an increasing strip (# of breakpoints stay unchanged)
• Example:
  \[
  \pi = 0 \ 1 \ 2 \ | \ 5 \ 6 \ 7 \ | \ 3 \ 4 \ | \ 8 \quad b(\pi) = 3
  \]
  \[
  \pi \cdot \rho(6,7) = 0 \ 1 \ 2 \ | \ 5 \ 6 \ 7 \ | \ 4 \ 3 \ | \ 8 \quad b(\pi) = 3
  \]
• \( \rho(6,7) \) creates a decreasing strip thus guaranteeing that the next step will decrease the # of breakpoints.
**ImprovedBreakpointReversalSort**

```plaintext
IMPROVEDBREAKPOINTREVERSALSORT(\(\pi\))
1   while \(b(\pi) > 0\)
2      if \(\pi\) has a decreasing strip
3         Among all reversals, choose reversal \(\rho\) minimizing \(b(\pi \cdot \rho)\)
4      else
5         Choose a reversal \(\rho\) that flips an increasing strip in \(\pi\)
6         \(\pi \leftarrow \pi \cdot \rho\)
7   output \(\pi\)
8   return
```
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: $\frac{2b(\pi)}{d(\pi)}$

- Optimal algorithm eliminates at most 2 breakpoints in every step:
  $$d(\pi) \geq \frac{b(\pi)}{2}$$

- Performance guarantee:
  $$\frac{2 \cdot b(\pi)}{d(\pi)} \leq \frac{2 \cdot b(\pi)}{b(\pi)} \cdot \frac{1}{2} = 4$$
Additional Application: Breakpoint 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 solid edges.

\[ \pi = 0 \ 2 \ 3 \ 1 \ 4 \ 6 \ 5 \ 7 \]
**Additional Application: Breakpoint 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 solid edges.
3) Connect vertices in order given by $1\ 2\ 3\ 4\ 5\ 6$ with dotted edges.
4) Superimpose solid and dotted paths.

\[ \pi = 0\ 2\ 3\ 1\ 4\ 6\ 5\ 7 \]
Additional Application: Breakpoint Graphs

• If we line up the dotted path (instead of the solid path) on a horizontal line, then we would get the following graph.

• Although they look different, these graphs are the same.

\[ \pi = 0\ 2\ 3\ 1\ 4\ 6\ 5\ 7 \]
Additional Application: Breakpoint Graphs

- How does a reversal change the breakpoint graph?
  - The dotted paths stay the same for both graphs.
  - However, the red (solid) edges are replaced with blue ones.

Before: 0 2 3 1 4 6 5 7

After: 0 2 3 5 6 4 1 7
Additional Application: Breakpoint Graphs

- A reversal removes 2 edges (red) and replaces them with 2 new edges (blue).
Maximum Cycle Decomposition

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

Alternating Cycle: Every two consecutive edges are of distinct color

Balanced vertex in $G(\pi)$: Equal number of solid and dashed edges incident on the vertex
Maximum Cycle Decomposition

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

Every edge in \( G(\pi) \) belongs to exactly one cycle.

Edge disjoint, alternating cycles

Maximum Cycle Decomposition, \( c(\pi) = 4 \)
Additional Application: Breakpoint Graphs

- A reversal removes 2 edges (red) and replaces them with 2 new edges (blue).
Additional Application: Breakpoint Graphs

Case 1:

Both edges belong to the same cycle

- Remove the center red edges and replace them with new blue edges (there are two ways to replace them)
- (a) After this replacement, there now exists 2 cycles instead of 1 cycle
  \[ c(\pi \rho) - c(\pi) = 1 \]
- (b) Or after this replacement, there still exists 1 cycle
  \[ c(\pi \rho) - c(\pi) = 0 \]
Additional Application: Breakpoint Graphs

Case 2:
The edges belong to different cycles

- Remove the center red edges and replace them with new blue edges
- After the replacement, there now exists 1 cycle instead of 2 cycles

\[ c(\pi \rho) - c(\pi) = -1 \]

For every permutation \( \pi \) and reversal \( \rho \), \( c(\pi \rho) - c(\pi) \leq 1 \)
Additional Application: Breakpoint Graphs

• Identity permutation of size n contains the maximum cycle decomposition of $n + 1$, $c(\text{identity}) = n + 1$
Additional Application: Breakpoint Graphs

• $c(\text{identity}) = n+1$

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

• At best after each reversal, the cycle decomposition can be increased by one:

$$d(\pi) = c(\text{identity}) - c(\pi) = n+1 - c(\pi)$$

• Yet, not every reversal can increase the cycle decomposition

$$d(\pi) \geq n+1 - c(\pi)$$
EXTRA
Additional Genome Rearrangements

• Besides reversals, we also have:
  1. Fusion and fission of chromosomes
Additional Genome Rearrangements

- Besides reversals, we also have:
  1. Fusion and fission of chromosomes
Rearrangements: Mathematical Representation

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

Fission

Each number represents a conserved region; +/- represents orientation.
Mice vs. Humans Revisited

- Humans and mice have similar genomes, but their genes are ordered differently.

- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocations
Mice vs. Humans Revisited

- Humans and mice have similar genomes, but their genes are ordered differently.
- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocations
Mice vs. Humans Revisited

• Humans and mice have similar genomes, but their genes are ordered differently.

• ~245 rearrangements
  • Reversals
  • Fusions
  • Fissions
  • Translocations
Mice vs. Humans Revisited

- Humans and mice have similar genomes, but their genes are ordered differently.

- ~245 rearrangements
  - Reversals
  - Fusions
  - Fissions
  - Translocations
Additional Application: Cancer

- Normal cells will have a certain makeup of chromosomes, as revealed by “chromosome painting.”
Additional Application: Cancer

- Normal cells will have a certain makeup of chromosomes, as revealed by “chromosome painting.”
Additional Application: Cancer

- Normal cells will have a certain makeup of chromosomes, as revealed by “chromosome painting.”

*PAINTING THE HUMAN CHROMOSOME*
(Spectral Karyotyping)
NORMAL CELLS
GRIMM Web Server

- Real genome architectures are represented by signed permutations.

- Efficient algorithms to sort signed permutations have been developed.

- GRIMM web server computes the reversal distances between signed permutations.
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 Reversals

- Blocks represent conserved genes.

1, 2, 3, 4, 5, 6, 7, 8, 9, 10
SIGNED Reversals

- Blocks represent conserved genes.
**SIGNED Reversals**

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- Blocks represent conserved genes.
From Signed to Unsigned Permutations

- Genes are *directed* fragments of DNA
- Genome is represented by a signed permutation
- If genes are in the same position but there orientations are different, they do not have the equivalent gene order
- 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 Permutations

- Genes are *directed* fragments of DNA
- Genome is represented by a signed permutation
- If genes are in the same position but have different orientations, they do not have the equivalent gene order
- 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 Permutations

- Genes are directed fragments of DNA
- Genome is represented by a signed permutation
- If genes are in the same position but have different orientations, they do not have the equivalent gene order
- 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 Permutations

• Genes are *directed* fragments of DNA

• Genome is represented by a signed permutation

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

• These two permutations have the same order, but each gene’s orientation is the reverse; therefore, they are not equivalent gene sequences

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \]

\[ -1 \quad 2 \quad -3 \quad -4 \quad -5 \]
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.
An Introduction to Bioinformatics Algorithms

Interleaving Graphs

- An Interleaving Graph is defined on the set of cycles in the Breakpoint graph and are connected by edges where cycles are interleaved.
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 them
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 hurdles, the following formula gives a tighter bound on reversal distance:

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