Genome Rearrangements - Continued

- How many pancakes should I flip to make the next new stack?
- Can I always ignore some of the possible flips?

<table>
<thead>
<tr>
<th>Minimum Strategy</th>
<th>current stack</th>
<th>next stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4215367</td>
<td>3512467</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Strategy</th>
<th>current stack</th>
<th>next stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4215367</td>
<td>5124367</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
A Greedy Algorithm for Sorting by Reversals

- When sorting the permutation, $\Pi = 1, 2, 3, 6, 4, 5$, one notices that the first three elements are already in order.
- So it does not make sense to break them apart.
- The length of the already sorted prefix of $\Pi$ is denoted as $\text{prefix}(\Pi) = 3$
- This inspires the following simple greedy algorithm

  while $\text{prefix}(\Pi) < \text{len}(\Pi)$:
    perform a reversal $\rho(\text{prefix}(\Pi) + 1, k)$ such that $\text{prefix}(\Pi)$ increases by at least 1.

- Such a reversal must always exist
- Finding, $k$, is as simple as finding the index of the minimum value of the remaining unsorted part
Geedy Reversal Sort: Example

Step 1: $\Pi_1 = 1, 2, 3, 6, 4, 5 \quad \rho(4, 5)$

Step 2: $\Pi_2 = 1, 2, 3, 4, \overline{6}, 5 \quad \rho(5, 6)$

Done: $\Pi_3 = 1, 2, 3, 4, 5, 6$

- The number of steps to sort any permutation of length $n$ is at most $(n - 1)$
Greedy Reversal Sort as code

```python
def GreedyReversalSort(pi):
    t = 0
    for i in xrange(len(pi)-1):
        j = pi.index(min(pi[i:]喁)
        if (j != i):
            pi = pi[:i] + [v for v in reversed(pi[i:j+1])] + pi[j+1:
            print "rho(%2d,%2d) = %s" % (i+1,j+1,pi)
            t += 1
    return t

print GreedyReversalSort([3,4,2,1,5,6,7,10,9,8])
```

rho( 1, 4) = [1, 2, 4, 3, 5, 6, 7, 10, 9, 8]
rho( 3, 4) = [1, 2, 3, 4, 5, 6, 7, 10, 9, 8]
rho( 8,10) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3
Analyzing GreedyReversalSort()

- GreedyReversalSort requires at most $n - 1$ steps
- For example, on $\Pi = 6, 1, 2, 3, 4, 5, t = 5$

$$\Pi_1 : 6, 1, 2, 3, 4, 5$$

$$\rho(1, 2) : 1, 6, 2, 3, 4, 5$$

$$\rho(2, 3) : 1, 2, 6, 3, 4, 5$$

$$\rho(3, 4) : 1, 2, 3, 6, 4, 5$$

$$\rho(4, 5) : 1, 2, 3, 4, 6, 5$$

$$\rho(5, 6) : 1, 2, 3, 4, 5, 6$$

- But there is a solution with far fewer flips
Greed Gone Wrong

- The same sequence sorted with two reversals
  \[ \Pi : 6, 1, 2, 3, 4, 5 \]
  \[ \rho(1, 6) : 5, 4, 3, 2, 1, 6 \]
  \[ \rho(1, 5) : 1, 2, 3, 4, 5, 6 \]

- Note, this solution makes no progress (no elements of the permutation are placed in their correct order) after its first move
- Yet it beats a greedy approach handily.
- So SimpleReversalSort(\pi) is correct (as a sorting routine), but non-optimal
- For many problems there is no known optimal algorithm, in such cases approximation algorithms are often used.
Approximation Algorithms

Today’s algorithms find approximate solutions rather than optimal solutions.

The approximation ratio of an algorithm, $A$, on input $\Pi$ is:

$$r = \frac{A(\Pi)}{OPT(\Pi)}$$

where:
- $A(\Pi)$ is the number of steps using the given algorithm.
- $OPT(\Pi)$ is the number of steps required using, a possibly unknown, optimal algorithm.
Performance Guarantees

- On an occasional input our approximation algorithm may give an optimal result, however we want to consider the value of $r$ for the worst case input
- When our objective is to minimize something (like reversals in our case)

$$r = \max_{i=0}^{\text{len}(\Pi)!} \frac{A(\Pi_i)}{OPT(\Pi_i)} \geq 1.0$$

- Or when our objective is to maximize something (like money)

$$r = \max_{i=0}^{\text{len}(\Pi)!} \frac{A(\Pi_i)}{OPT(\Pi_i)} \leq 1.0$$

- Sounds cool in theory, but there are lots of open ends here
  - if we don't know $OPT(\Pi_i)$ how are we supposed to know how many steps it requires?
  - do we really need to test for all $\text{len}(\Pi)!$ possible inputs?

8
How do we get Approximation Ratios?

```python
def GreedyReversalSort(pi):
    for i in xrange(len(pi)-1):
        j = pi.index(min(pi[i:] if j != i:
            pi = pi[:i]
            + [v for v in reversed(pi[i:j+1])]
            + pi[j+1:]
    return pi
```

<table>
<thead>
<tr>
<th>A((\pi))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 0: 6 1 2 3 4 5</td>
</tr>
<tr>
<td>Step 1: 1 6 2 3 4 5</td>
</tr>
<tr>
<td>Step 2: 1 2 6 3 4 5</td>
</tr>
<tr>
<td>Step 3: 1 2 3 6 4 5</td>
</tr>
<tr>
<td>Step 4: 1 2 3 4 6 5</td>
</tr>
<tr>
<td>Step 5: 1 2 3 4 5 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPT((\pi))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 0: 6 1 2 3 4 5</td>
</tr>
<tr>
<td>Step 1: 5 4 3 2 1 6</td>
</tr>
<tr>
<td>Step 2: 1 2 3 4 5 6</td>
</tr>
</tbody>
</table>
New Idea: Adjacencies

- Recall breakpoints from last lecture. Adjacencies are the opposite.
- Assume a permutation:
  \[ \Pi = \pi_1, \pi_2, \pi_3, \ldots \pi_{n-1}, \pi_n, \]
- A pair of neighboring elements \( \pi_i \) and \( \pi_{i+1} \) are adjacent if:
  \[ \pi_{i+1} = \pi_i + 1 \]
- For example:
  \[ \Pi = 1, 9, \underline{3}, 4, 7, \underline{8}, 2, 6, 5 \]
- (3,4) and (7,8) and (6,5) are adjacencies.
Adjacencies and Breakpoints

- **Breakpoints** occur between neighboring non-adjacent elements

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

- There are 5 breakpoints in our permutation between pairs (1,9), (9,3), (4,7), (8,2) and (2,5)

- We define \( b(\Pi) \) as the number of breakpoints in permutation \( \Pi \)
Extending Permutations

- One can place two elements, $\pi_0 = 0$ and $\pi_{n+1} = n + 1$ at the beginning and end of $\Pi$ respectively.

\[
1, 9, 3, 4, 7, 8, 12, 16, 5
\]

\[
\Pi = 0 1, 9, 3, 4, 7, 8, 12, 16, 5, 10
\]

- An additional breakpoint was created after extending.
- An extended permutation of length $n$ can have at most $(n + 1)$ breakpoints.
- $(n - 1)$ between the original elements plus 2 for the extending elements.
How Reversals Effect Breakpoints

- Breakpoints are the *targets* for sorting by reversals.
- Once they are removed, the permutation is sorted.
- Each "useful" reversal eliminates at least 1, and at most 2 breakpoints.
- Consider the following application of GreedyReversalSort(Extend(\(\Pi\)))

\[
\Pi = \begin{array}{cccccc}
2 & 3 & 1 & 4 & 6 & 5 \\
0 & 2 & 3 & 1 & 4 & 6, 5, 7 \\
0, 1 & 3, 2 & 4 & 6, 5, 7 \\
0, 1, 2 & 3, 4 & 6, 5, 7 \\
0, 1, 2, 3, 4, 5, 6, 7
\end{array}
\]

\[b(\Pi) = 5, 4, 2, 0\]
Sorting By Reversals:  
A second Greedy Algorithm

**BreakpointReversalSort(π):**

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

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

The "greedy" concept here is to reduce as many breakpoints as possible at each step.

Does it always terminate?

What if no reversal reduces the number of breakpoints?
New Concept: *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 extension strips (with 0 and n+1)

\[ \overrightarrow{0, 1, 9, 4, 3, 7, 8, 2, 5, 6, 10} \]
Reducing the Number of Breakpoints

- Consider $\Pi = 1, 4, 6, 5, 7, 8, 3, 2$

$\begin{array}{cccccc}
0 & 1 & 4 & 6 & 5 & 7 & 8 & 3 & 2 & 9
\end{array}$  

$b(p) = 5$

If permutation $p$ contains at least one decreasing strip, then there exists a reversal $r$ which decreases the number of breakpoints (i.e. $b(p \circ r) < b(p)$).

How can we be sure that we decrease the number of breakpoints?

Which reversal?
Things to Consider

- Consider \( \Pi = 1, 4, 6, 5, 7, 8, 3, 2 \)

\[
\begin{align*}
0, 1, & 4, 6, 5, 7, 8, 3, 2, & 9 & b(p) = 5
\end{align*}
\]

- Choose the decreasing strip with the smallest element \( k \) in \( \Pi \)
  - It'll always be the rightmost element of that strip
- Find \( k - 1 \) in the permutation
  - it'll always be flanked by a breakpoint
- Reverse the segment between \( k \) and \( k - 1 \)
Things to Consider

- Consider $\Pi = 1, 4, 6, 5, 7, 8, 3, 2$

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

$ b(p) = 4$

- Choose the decreasing strip with the smallest element $k$ in $\Pi$
  - It'll always be the rightmost element of that strip
- Find $k - 1$ in the permutation
  - it'll always be flanked by a breakpoint
- Reverse the segment between $k$ and $k - 1$
Reversal Examples

- Consider $\Pi = 1, 4, 6, 5, 7, 8, 3, 2$

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

- Choose the decreasing strip with the smallest element $k$ in $\Pi$
  - It'll always be the rightmost element of that strip
- Find $k - 1$ in the permutation
  - it'll always be flanked by a breakpoint
- Reverse the segment between $k$ and $k - 1$

$b(p) = 2$
Reversal Examples

- Consider $\Pi = 1, 4, 6, 5, 7, 8, 3, 2$
  
  \[
  b(p) = 0, \quad 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  \]

- Choose the decreasing strip with the smallest element $k$ in $\Pi$
  - It'll always be the rightmost element of that strip
- Find $k - 1$ in the permutation
  - It'll always be flanked by a breakpoint
- Reverse the segment between $k$ and $k - 1$
Things to Consider

- Consider \( \Pi = 1, 4, 6, 5, 7, 8, 3, 2 \)

\[
\begin{array}{c}
0, 1, \longrightarrow 4, \longrightarrow 6, 5, \longrightarrow 7, 8, \longrightarrow 3, 2, \longrightarrow 9 \\
\longrightarrow 0, 1, 2, 3, \longrightarrow 8, 7, \longrightarrow 5, 6, \longrightarrow 4, \longrightarrow 9 \\
\longrightarrow 0, 1, 2, 3, 4, \longrightarrow 6, 5, \longrightarrow 7, 8, \longrightarrow 9 \\
\longrightarrow 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
\end{array}
\]

\( b(p) = 5 \)

\( b(p) = 4 \)

\( b(p) = 2 \)

\( b(p) = 0 \)

\( d(\Pi) = 3 \)

Does it work for any permutation?
Potential Gotcha

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

However, reversing an increasing strip creates a decreasing strip, and the number of breakpoints remains unchanged.

Then the number of breakpoints will be reduced in the following steps.

$0, 1, 2, 5, 6, 7, 3, 4, 8, 9 \quad b(p) = 3$
Potential Gotcha

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

However, reversing an increasing strip creates a decreasing strip, and the number of breakpoints remains unchanged.

Then the number of breakpoints will be reduced in the following steps.
Next Time

- Look at the Code
- How about performance?