Genome Rearrangements - Continued
Lessons from last time

1. With each reversal, one can remove at most 2 breakpoints
2. If there is any decreasing strip there exists a reversal that will remove at least one breakpoint
3. If breakpoints remain and there is no decreasing strip one can be created by reserving any remaining strip

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

\[
\begin{array}{c}
b(p) = 3 \\
b(p) = 3 \\
b(p) = 2 \\
b(p) = 0 \\
\end{array}
\]

\[
\begin{array}{c}
\rho(3, 5) \\
\rho(6, 7) \\
\rho(3, 7) \\
\text{Done!} \\
\end{array}
\]

An optimal algorithm would remove 2 breakpoints at every step. The last reversal always removes 2 breakpoints, thus if the number of breakpoints is odd, even the optimal algorithm must make at least one reversal that removes only 1 breakpoint.
An Improved Breakpoint Reversal Sort

ImprovedBreakpointReversalSort(π)

1. while b(π) > 0
2.     if π has a decreasing strip
3.         Among all possible reversals, choose reversal ρ that minimizes b(π • ρ)
4.     else
5.         Choose a reversal ρ that flips an increasing strip in π
6.     π ← π • ρ
7. output π
8. return
def improvedBreakpointReversalSort(seq, verbose=True):
    seq = [0] + seq + [max(seq)+1] # Extend sequence
    N = 0
    while hasBreakpoints(seq):
        increasing, decreasing = getStrips(seq)
        if len(decreasing) > 0:
            removed, reversal = pickReversal(seq, decreasing)
        else:
            removed, reversal = 0, increasing[0] # No breakpoints can be removed
        if verbose:
            print "Strips: ", increasing, decreasing
            print "%d: %s  rho%s" % (removed, seq, reversal)
            raw_input("Press Enter: ")
        seq = doReversal(seq, reversal)
        N += 1
    if verbose:
        print seq, "Sorted"
    return N

# Also try: [1,9,3,4,7,8,2,6,5]
print improvedBreakpointReversalSort([3,4,1,2,5,6,7,10,9,8], verbose=True)
Performance

- *ImprovedBreakPointReversalSort* is a greedy algorithm with a performance guarantee of no worse than 4 when compared to an optimal algorithm
  - It eliminates at least one breakpoint in every two steps (flip an increasing then remove 1)
  - That’s at most: $2b(\Pi)$ steps
  - An optimal algorithm could *at most* remove 2 breakpoints in every step, thus requiring $\frac{b(\Pi)}{2}$ steps
  - The approximation ratio is:
    \[
    \frac{A(\Pi)}{OPT(\Pi)} = \frac{2b(\Pi)}{\frac{b(\Pi)}{2}} = 4
    \]
- But there is a solution with far fewer flips
A Better Approximation Ratio

- If there is a decreasing strip, the next reversal reduces \( b(\pi) \) by at least one.
- The only bad case is when there is no decreasing strip. Then we do a reversal that does not reduce \( b(\pi) \).
- If we always choose a reversal reducing \( b(\pi) \) and, at the same time, select a permutation such that the result has at least one decreasing strip, the bad case would never occur.
- If all possible reversals that reduce \( b(\pi) \) create a permutation without decreasing strips, then there exists a reversal that reduces \( b(\pi) \) by 2 (Proof not given)!
- When the algorithm creates a permutation without a decreasing strip, the previous reversal must have reduced \( b(\pi) \) by two.
- At most \( b(\pi) \) reversals are needed.
- The improved Approximation ratio:

\[
\frac{A_{new}(\Pi)}{OPT(\Pi)} = \frac{b(\Pi)}{\frac{b(\Pi)}{2}} = 2
\]
Comparing Greedy Algorithms

**SimpleReversalSort**

- Attempts to extend the prefix($\pi$) at each step
- Approximation ratio $\frac{n-1}{b(\Pi)/2}$ steps

**ImprovedBreakpointReversalSort**

- Attempts to reduce the number of breakpoints at each step
- Approximation ratio $\frac{b(\Pi)}{b(\Pi)/2} = 2$ steps
Problem Set Time!