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

$$\overrightarrow{0, 1, 2,}\ |\ \underleftarrow{\overrightarrow{5, 6, 7,}}\ |\ \overrightarrow{3, 4,}\ |\ \overrightarrow{8, 9} \qquad b(p) = 3 \qquad \rho(3, 5)$$

$$\overrightarrow{0, 1, 2,}\ |\ \overleftarrow{7, 6, 5,}\ |\ \overrightarrow{3, 4,}\ |\ \overrightarrow{8, 9} \qquad b(p) = 3 \qquad \rho(6, 7)$$

$$\overrightarrow{0, 1, 2,}\ |\ \overleftarrow{7, 6, 5, 4, 3,}\ |\ \overrightarrow{8, 9} \qquad b(p) = 2 \qquad \rho(3, 7)$$

$$\overrightarrow{0, 1, 2, 3, 4, 5, 6, 7, 8, 9} \qquad b(p) = 0 \qquad Done!$$

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 reersal that removes only 1 breakpoint.

An Improved Breakpoint Reversal Sort

ImprovedBreakpointReversalSort(π)

```
while b(π) > 0
    if π has a decreasing strip
        Among all possible reversals, choose reversal ρ that minimizes b(π • ρ)
    else
        Choose a reversal ρ that flips an increasing strip in π
    π ← π • ρ
output π
return
```

Improved Breakpoint Reversal Sort in Python

```
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:                             # pick a reversal that removes a decreasing strip
            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)
```

```
Strips: [(1, 3), (3, 5), (5, 8)] [(8, 11)]
2: [0, 3, 4, 1, 2, 5, 6, 7, 10, 9, 8, 11]  rho(8, 11)
Press Enter:
Strips: [(1, 3), (3, 5)] []
0: [0, 3, 4, 1, 2, 5, 6, 7, 8, 9, 10, 11]  rho(1, 3)
Press Enter:
Strips: [(3, 5)] [(1, 3)]
1: [0, 4, 3, 1, 2, 5, 6, 7, 8, 9, 10, 11]  rho(3, 5)
Press Enter:
Strips: [] [(1, 5)]
2: [0, 4, 3, 2, 1, 5, 6, 7, 8, 9, 10, 11]  rho(1, 5)
Press Enter:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11] Sorted
4
```

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:

$$\frac{\mathcal{A}(\Pi)}{OPT(\Pi)} = \frac{2b(\Pi)}{\frac{b(\Pi)}{2}} = 4$$

- But there is a solution with far fewer flips

A Better Approximation Ratio

- If there is a decreasing strip, the next reversal reduces b(π) 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(π).
- If we always choose a reversal reducing b(π) 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(π) create a permutation without decreasing strips, then there exists a reversal that reduces b(π) by 2 (Proof not given)!
- When the algorithm creates a permutation without a decreasing strip, the previous reversal must have reduced b(π) by two.
- At most b(π) reversals are needed.
- The improved Approximation ratio:

$$\frac{\mathcal{A}_{new}(\Pi)}{OPT(\Pi)} = \frac{b(\Pi)}{\frac{b(\Pi)}{2}} = 2$$

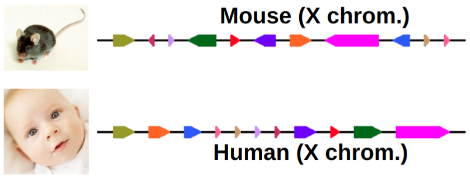
Comparing Greedy Algorithms

SimpleReversalSort

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

ImprovedBreakpointReversalSort

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



Problem Set Time!