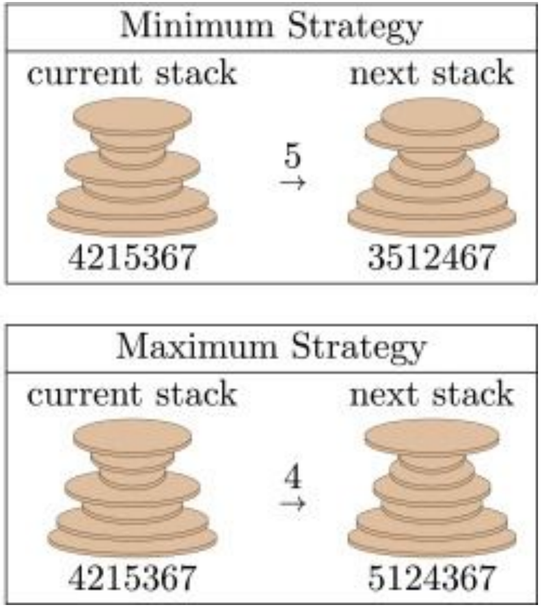
Comp 555 - BioAlgorithms - Spring 2020



- Problem set #5 is due tonight
- Final exam on Friday May 1 (8AM–11AM)
- Study session? 4PM–6PM on Monday 4/27 -or- Tuesday 4/28

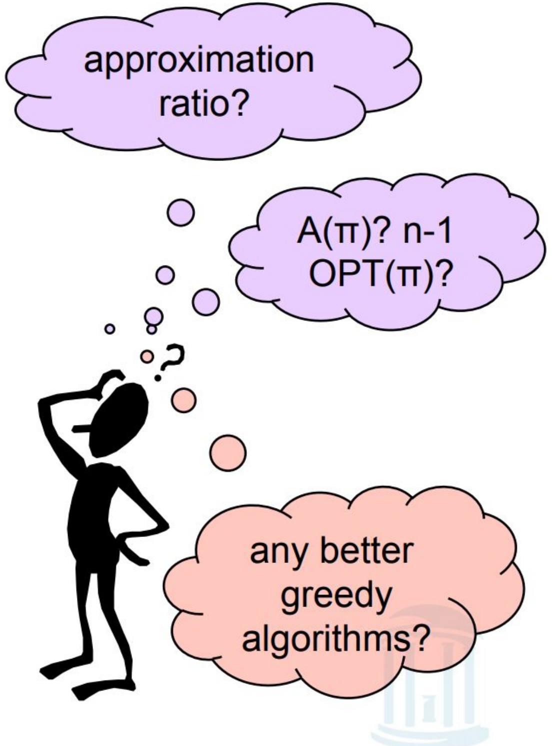
Genome Rearrangements - Continued

In search of Approximation Ratios

```python
def GreedyReversalSort(pi):
    for i in range(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
```

```
        A(Π)
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
Step 5: 1 2 3 4 5 6
```

```
        OPT(Π)?
Step 0: 6 1 2 3 4 5
Step 1: 5 4 3 2 1 6
Step 2: 1 2 3 4 5 6
```



New Idea: Adjacencies

- Adjacencies are locally sorted runs.
- Assume a permutation:

$$\Pi = \pi_1, \pi_2, \pi_3, \ldots \pi_{n-1}, \pi_n$$

- A pair of neighboring elements π_i and π_{i+1} are *adjacent* if:

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

- For example:

$$\Pi = 1, 9, \underline{3, 4}, \underline{7, 8}, 2, \underline{6, 5}$$

- (3,4) and (7,8) and (6,5) are adjacencies.

Adjacencies and Breakpoints

- *Breakpoints* occur between neighboring non-adjacent elements

$$\Pi = 1, \mid 9, \mid \underline{3, 4}, \mid \underline{7, 8}, \mid 2, \mid \underline{6, 5}$$

- There are 5 breakpoints in our permuation between pairs (1,9), (9,3), (4,7), (8,2) and (2,5)
- We define b(Π) as the number of breakpoints in permutation Π

Extending Permutations

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

$$1, \mid 9, \mid \underline{3, 4,} \mid \underline{7, 8,} \mid 2, \mid \underline{6, 5}$$

$$\downarrow$$

$$\Pi = 0\ 1, \mid 9, \mid \underline{3, 4,} \mid \underline{7, 8,} \mid 2, \mid \underline{6, 5,} \mid 10$$

- An addtional 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(Π))

$$\Pi = \quad 2, 3, 1, 4, 6, 5$$

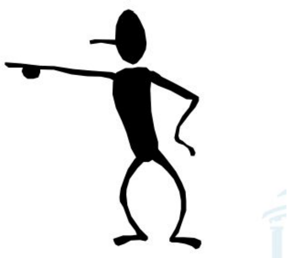
$$0 \mid 2, 3 \mid 1 \mid 4 \mid 6, 5 \mid 7 \qquad b(\Pi) = 5$$

$$0, \overline{1 \mid 3, 2} \mid 4 \mid 6, 5 \mid 7 \qquad b(\Pi) = 4$$

$$0, 1, \overline{2, 3}, 4 \mid 6, 5 \mid 7 \qquad b(\Pi) = 2$$

$$0, 1, 2, 3, 4, \overline{5, 6}, 7 \qquad b(\Pi) = 0$$

$$\textit{required reversals} \geq \frac{b(\pi)}{2}$$



Sorting-by-Reversals: A second Greedy Algorithm

BreakpointReversalSort(π):

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

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?

0 1 2 | 5 6 7 | 3 4 | 8 9

Yet Another New Idea: *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)

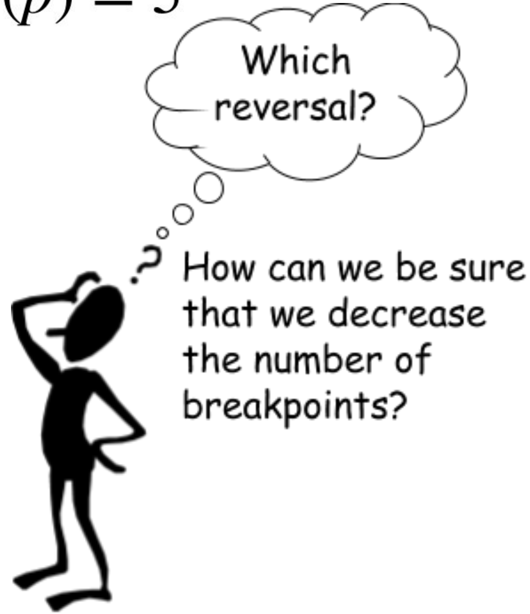
$$\overrightarrow{0, 1},\ \overleftarrow{9},\ \overleftarrow{4, 3},\ \overrightarrow{7, 8},\ \overleftarrow{2},\ \overrightarrow{5, 6},\ \overrightarrow{10}$$

Reducing the Number of Breakpoints

- Consider Π =1,4,6,5,7,8,3,2

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

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



Things to Consider

- Consider Π = 1,4,6,5,7,8,3,2

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

- Choose the decreassing strip with the smallest elment k in Π
  - It'll always be the right-most 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 Π = 1,4,6,5,7,8,3,2

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

- Choose the decreassing strip with the smallest elment k in Π
  - It'll always be the right-most 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 Π = 1,4,6,5,7,8,3,2

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

- Choose the decreassing strip with the smallest elment k in Π
    - It'll always be the right-most 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 Π = 1,4,6,5,7,8,3,2

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

- Choose the decreassing strip with the smallest elment k in Π
    - It'll always be the right-most 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 Π = 1,4,6,5,7,8,3,2

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

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

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

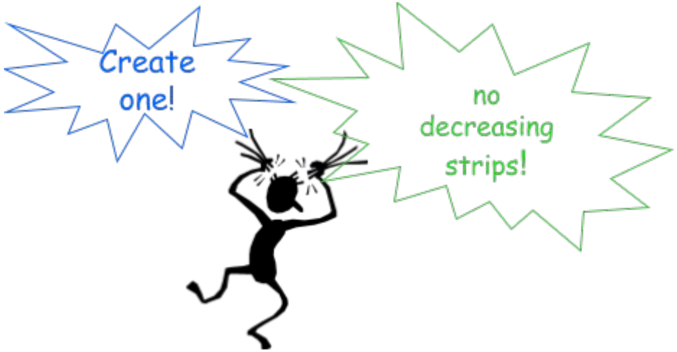
$$\overrightarrow{0,1,2,3,4,5,6,7,8,9} \qquad b(p) = 0$$

$$d(\Pi) = 3$$

Potential Gotcha

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

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



Potential Gotcha

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

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

- If there is no decreasing strip, there may be *no strip-reversal* ρρ *that reduces the number of breakpoints* (i.e. b(Π˙ρ(i,j)) ≥ b(Π) for any reversal ρ).
- 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.

Putting it all together

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

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

```
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
```

Breakpoints and Strips

In [11]:
```
def hasBreakpoints(seq):
    """ returns True if sequences is not strictly increasing by 1 """
    for i in range(1, len(seq)):
        if (seq[i] != seq[i-1] + 1):
            return True
    return False

def getStrips(seq):
    """ find contained intervals where sequence is ordered, and return intervals
    in as lists, increasing and decreasing. Single elements are considered
    decreasing. "Contained" excludes the first and last interval. """
    deltas = [seq[i+1] - seq[i] for i in range(len(seq)-1)]
    increasing = list()
    decreasing = list()
    start = 0
    for i, diff in enumerate(deltas):
        if (abs(diff) == 1) and (diff == deltas[start]):
            continue
        if (start > 0):
            if deltas[start] == 1:
                increasing.append((start, i+1))
            else:
                decreasing.append((start, i+1))
        start = i+1
    return increasing, decreasing
```

Handle Reversals

In [15]:
```
def pickReversal(seq, strips):
    """ test each decreasing interval to see if it leads to a reversal that
    removes two breakpoints, otherwise, return a reversal that removes only one """
    for i, j in strips:
        k = seq.index(seq[j-1]-1)
        if (seq[k+1] + 1 == seq[j]):
            # removes 2 breakpoints
            return 2, (min(k+1, j), max(k+1, j))
    # In the worst case we remove only one, but avoid the length "1" strips
    for i, j in strips:
        k = seq.index(seq[j-1]-1)
        if (j - i > 1):
            break
    return 1, (min(k+1, j), max(k+1, j))

def doReversal(seq,reversal):
    i, j = reversal
    return seq[:i] + [element for element in reversed(seq[i:j])] + seq[j:]
```

Let’s do it!

In [13]:
```
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))
            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 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(Π) steps
  - An optimal algorithm could *at most* remove 2 breakpoints in every step, thus requiring b(Π)/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:

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

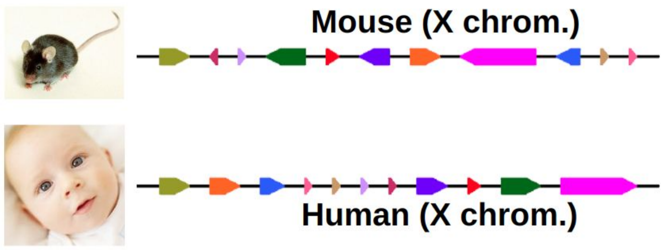
Comparing Greedy Algorithms

SimpleReversalSort

- Attempts to extend the prefix(π) at each step
- Approximation ratio (n−1)/(b(Π)/2) can be large

ImprovedBreakpointReversalSort

- Attempts to reduce the number of breakpoints at each step
- Approximation ratio b(Π)/(b(Π)/2) = 2x



Next Time

- **A little randomness**
- **Study session?**

  Monday 4/27 or Tuesday 4/28 4pm-6pm?
- **Need to resolve all outstanding grading issues**