Genome Rearrangements

Study Chapters 5.3-5.5

Python Tutorial on 9/11

From Last Time

- We developed a SimpleReversalSort algorithm that sorts by extending its prefix on every iteration (n-1) steps.
- On $\pi: \underline{6\ 1}\ 2\ 3\ 4\ 5$

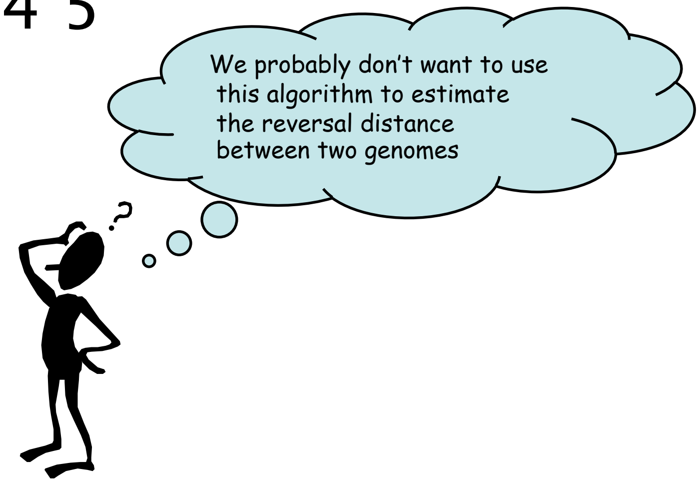
Flip 1: $1\ \underline{6\ 2}\ 3\ 4\ 5$

Flip 2: $1\ 2\ \underline{6\ 3}\ 4\ 5$

Flip 3: $1\ 2\ 3\ \underline{6\ 4}\ 5$

Flip 4: $1\ 2\ 3\ 4\ \underline{6\ 5}$

Flip 5: $1\ 2\ 3\ 4\ 5\ 6$



- But it could have been sorted in two flips:

$\pi: \underline{6\ 1\ 2\ 3\ 4\ 5}$

Flip 1: $\underline{5\ 4\ 3\ 2\ 1}\ 6$

Flip 2: $1\ 2\ 3\ 4\ 5\ 6$

Approximation Algorithms

- Today's algorithms find *approximate* solutions rather than *optimal* solutions
- The approximation ratio of an algorithm $\mathcal{A}$ on input π is:

$$\mathcal{A}(\pi) \;/\; \mathrm{OPT}(\pi)$$

where

$\mathcal{A}(\pi)$ - solution produced by algorithm $\mathcal{A}$

$\mathrm{OPT}(\pi)$ - optimal solution of the problem

Approximation Ratio/Performance Guarantee

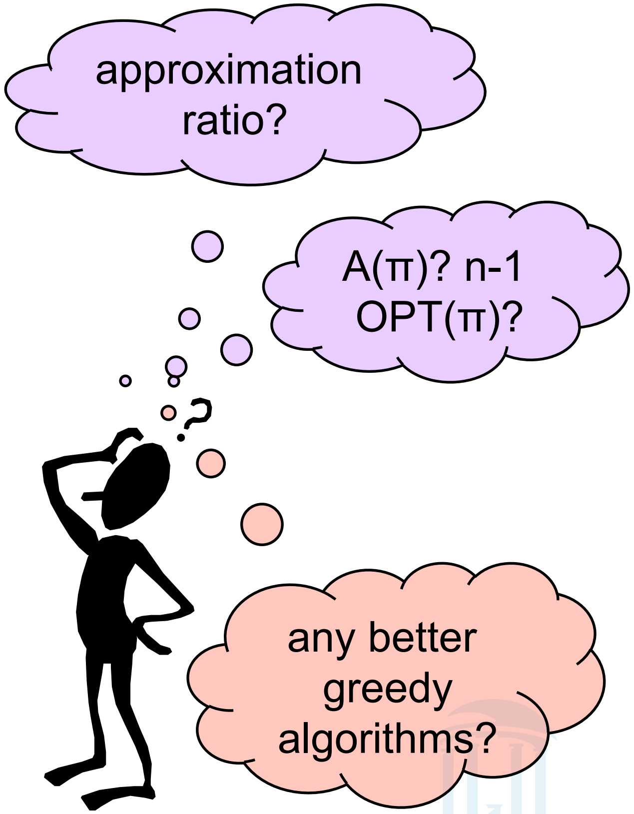
- Approximation ratio (performance guarantee) of algorithm $\mathcal{A}$: max approximation ratio over all inputs of size n
  - For a minimizing algorithm $\mathcal{A}$ (like ours):
    - Approx Ratio = $\max_{|\pi| = n} \mathcal{A}(\pi) / \mathrm{OPT}(\pi) \geq 1.0$
  - For maximization algorithms:
    - Approx Ratio = $\min_{|\pi| = n} \mathcal{A}(\pi) / \mathrm{OPT}(\pi) \leq 1.0$

Approximation Ratio

```
SimpleReversalSort($\pi$)
for $i \leftarrow 1$ to $n - 1$
    $j \leftarrow$ position of element $i$ in $\pi$ (i.e., $\pi_j = i$)
    if $j \neq i$
        $\pi \leftarrow \pi \ \ \rho(i, j)$
        output $\pi$
    if $\pi$ is the identity permutation
        return
```

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

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

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

- 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) or (7, 8) and (6,5) are adjacent pairs

Breakpoints

Breakpoints occur between neighboring non-adjacent elements:

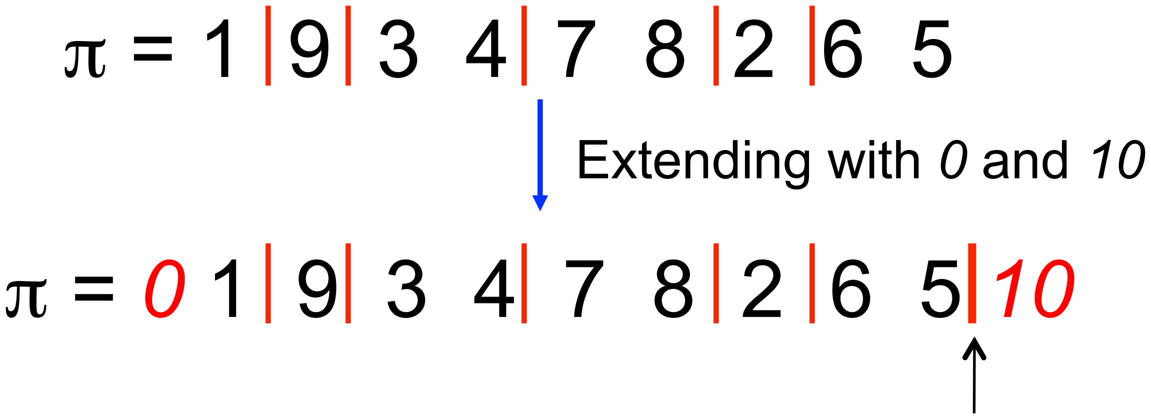
$$\pi = 1\,|\,9\,|\,\underline{3\ \ 4}\,|\,\underline{7\ \ 8}\,|\,2\,|\,\underline{6\ \ 5}$$

- Pairs (1,9), (9,3), (4,7), (8,2) and (2,5) define 5 breakpoints of permutation π

- $b(\pi)$ - # 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



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

Extending with *0* and *10*

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

A new breakpoint was created after extending

An extended permutation of n can have at most (n+1) breakpoints, (n-1 between elements plus 2)

Reversal Distance and Breakpoints

- Breakpoints are the *bottlenecks* 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
  SimpleReversalSort(Extend(π)):

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

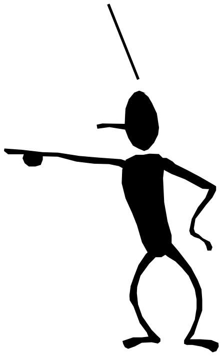
$0 \mid \underline{2\ \ 3 \mid 1} \mid 4 \mid 6\ \ 5 \mid 7 \qquad b(\pi) = 5$

$0\ \ 1 \mid \underline{3\ \ 2} \mid 4 \mid 6\ \ 5 \mid 7 \qquad b(\pi) = 4$

$0\ \ 1\ \ 2\ \ 3\ \ 4 \mid \underline{6\ \ 5} \mid 7 \qquad b(\pi) = 2$

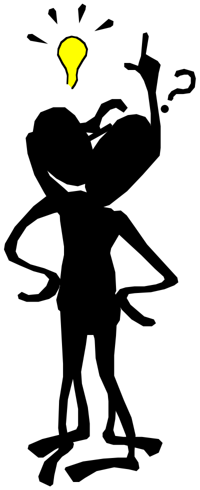
$0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7 \qquad b(\pi) = 0$

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



Sorting By Reversals: A Better Greedy Algorithm

```
BreakPointReversalSort($\pi$)
while $b(\pi) > 0$
    Among all possible reversals, choose reversal $\rho$ minimizing $b(\pi \bullet \rho)$
    $\pi \leftarrow \pi \bullet \rho(i, j)$
    output $\pi$
return
```



The "greedy" concept here is to reduce as many breakpoints as possible

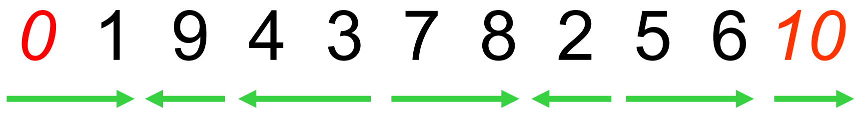
Does it always terminate?

What if no reversal reduces the number of breakpoints?

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

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)

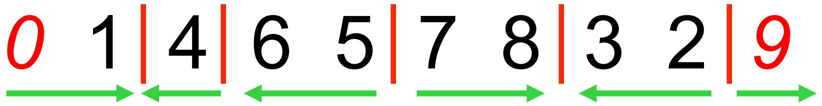


0 1 9 4 3 7 8 2 5 6 *10*

  - A *single-element strip* can be declared either increasing or decreasing. We will choose to declare them as decreasing with exception of extension strips (with *0* and *n+1)*

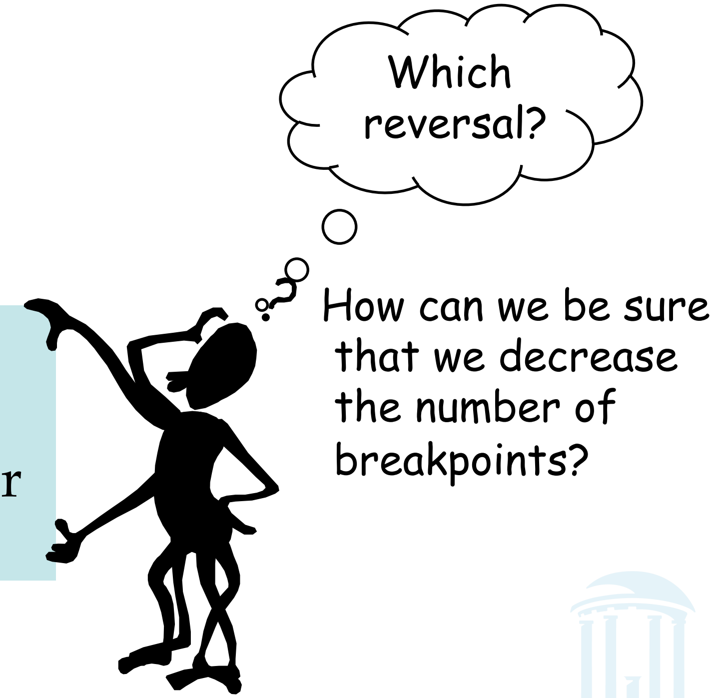
Reducing the Number of Breakpoints

Consider π = 1 4 6 5 7 8 3 2



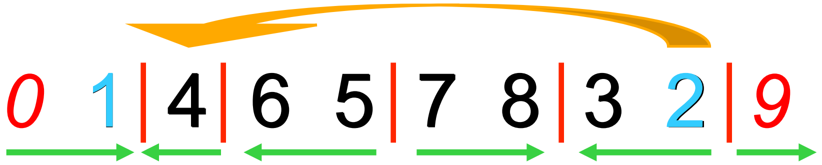
0 1 | 4 | 6 5 | 7 8 | 3 2 | 9 $b(\pi) = 5$

If permutation π contains at least one decreasing strip, then there exists a reversal ρ which decreases the number of breakpoints (i.e. $b(\pi \bullet \rho) < b(\pi)$).



Things to Consider

Consider π = 1 4 6 5 7 8 3 2

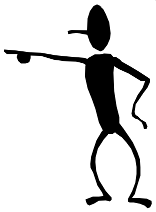


0 1 | 4 | 6 5 | 7 8 | 3 2 | 9

$b(\pi) = 5$

- Choose the decreasing strip with the smallest element *k* in π *(it'll always be the rightmost)*
- Find *k – 1* in the permutation

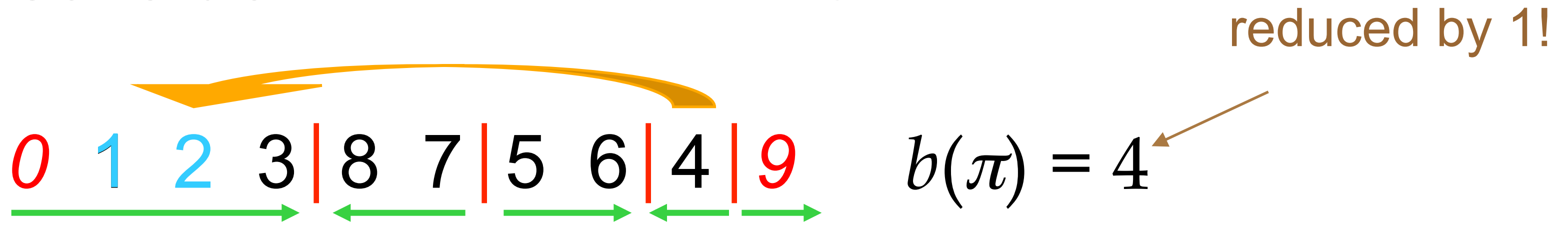
  (it'll always be flanked by a breakpoint)
- Reverse the segment between *k* and *k-1*



Thus, removing the breakpoint flanking k-1

Things to Consider

Consider π = 1 4 6 5 7 8 3 2


reduced by 1!

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

- Choose the decreasing strip with the smallest element k in π *(it'll always be the rightmost)*
- Find $k - 1$ in the permutation
  (*it'll always be flanked by a breakpoint*)
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip

Things to Consider

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

$b(\pi) = 4$

- Choose the decreasing strip with the smallest element k in π *(it'll always be the rightmost)*
- Find $k - 1$ in the permutation *(it'll always be flanked by a breakpoint)*
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip

Things to Consider

Consider π = 1 4 6 5 7 8 3 2

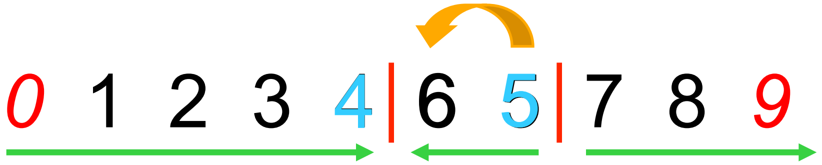
$b(\pi) = 2$

- Choose the decreasing strip with the smallest element *k* in π *(it'll always be the rightmost)*
- Find *k – 1* in the permutation

  (it'll always be flanked by a breakpoint)
- Reverse the segment between *k* and *k-1*
- Repeat until there is no decreasing strip

Things to Consider

Consider π = 1 4 6 5 7 8 3 2



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

$b(\pi) = 2$

- Choose the decreasing strip with the smallest element *k* in π *(it'll always be the rightmost)*
- Find *k – 1* in the permutation *(it'll always be flanked by a breakpoint)*
- Reverse the segment between *k* and *k-1*
- Repeat until there is no decreasing strip

Things to Consider

Consider π = 1 4 6 5 7 8 3 2

No breakpoint left!

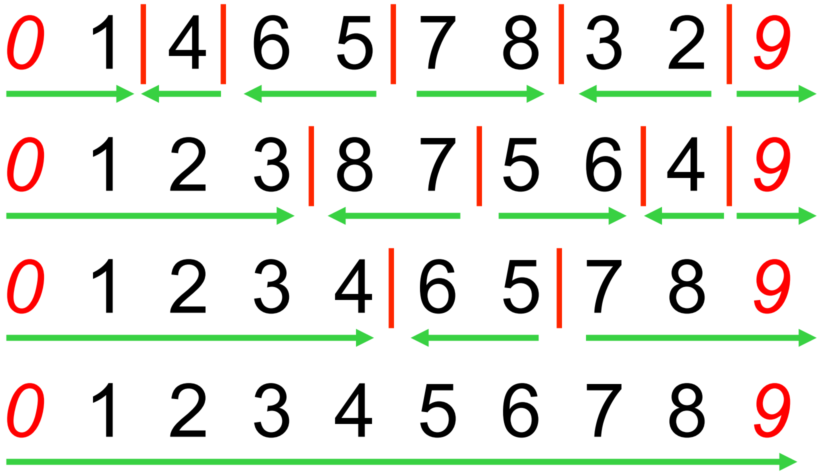
0 1 2 3 4 5 6 7 8 *9*

$b(\pi) = 0$

- Choose the decreasing strip with the smallest element *k* in π *(it'll always be the rightmost)*
- Find *k – 1* in the permutation
  (it'll always be flanked by a breakpoint)
- Reverse the segment between *k* and *k-1*
- Repeat until there is no decreasing strip

Things to Consider

Consider π = 1 4 6 5 7 8 3 2



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

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

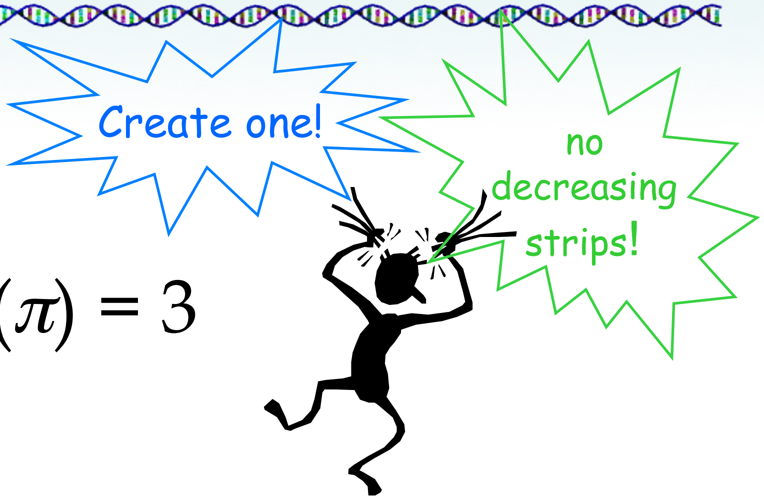
0 1 2 3 4 | 6 5 | 7 8 9 $b(\pi) = 2$

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

$d(\pi) = 3$

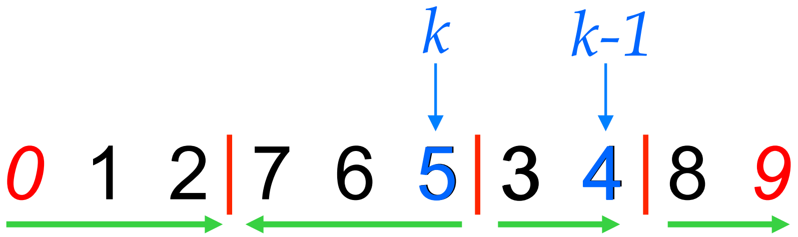
Potential Gotcha

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



- If there is no decreasing strip, there may be no strip-reversal ρ that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ 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

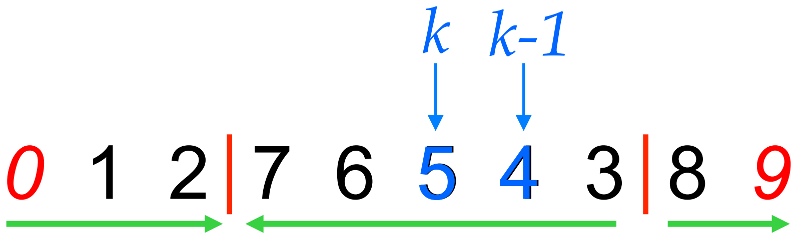


$0 \; 1 \; 2 \,|\, 7 \; 6 \; 5 \,|\, 3 \; 4 \,|\, 8 \; 9$

$b(\pi) = 3$

- If there is no decreasing strip, there may be no strip-reversal ρ that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ for any reversal ρ).
- However, reversing an <u>increasing</u> 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

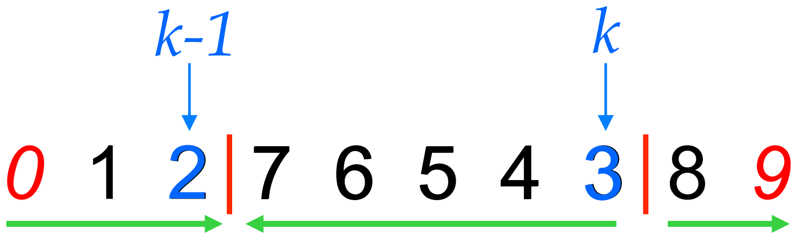


0 1 2 | 7 6 5 4 3 | 8 9

$b(\pi) = 2$

- If there is no decreasing strip, there may be no strip-reversal ρ that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ 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.

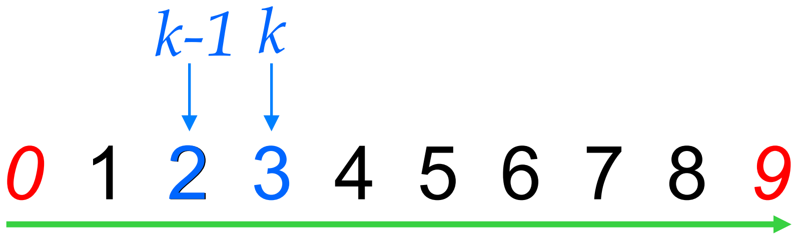
Potential Gotcha



$b(\pi) = 2$

- If there is no decreasing strip, there may be no strip-reversal ρ that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ 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



$b(\pi) = 0$

- If there is no decreasing strip, there may be no strip-reversal ρ that reduces the number of breakpoints (i.e. $b(\pi \bullet \rho) \geq b(\pi)$ 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.

ImprovedBreakpointReversalSort

```
ImprovedBreakpointReversalSort($\pi$)
while $b(\pi) > 0$
    if $\pi$ has a decreasing strip
        Among all possible reversals, choose reversal $\rho$
                that minimizes $b(\pi \bullet \rho)$
    else
        Choose a reversal $\rho$ that flips an increasing strip in $\pi$
    $\pi \leftarrow \pi \bullet \rho$
    output $\pi$
return
```

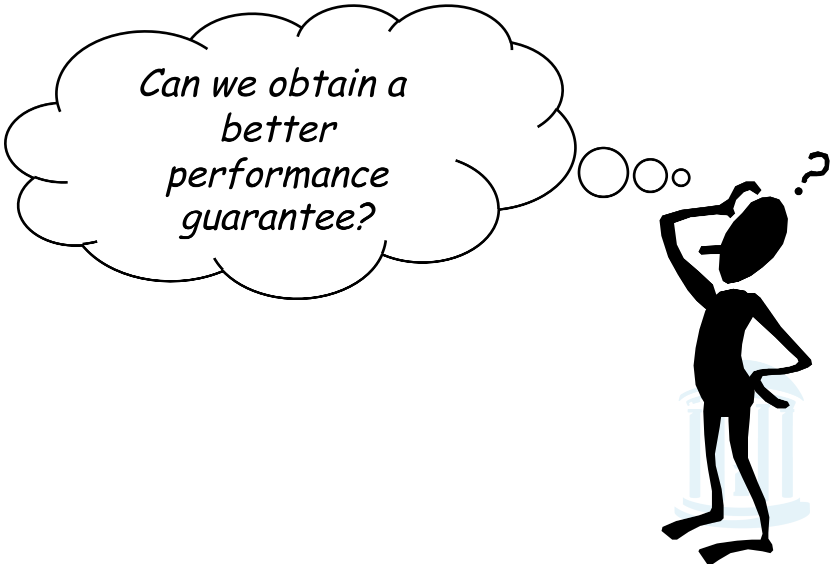
In Python

```
def improvedBreakpointReversalSort(seq):
    while hasBreakpoints(seq):
        increasing, decreasing = getStrips(seq)
        if len(decreasing) > 0:
            reversal = pickReversal(seq, decreasing)
        else:
          reversal = increasing[0]
        print seq, "reversal", reversal
        seq = doReversal(seq,reversal)
    print seq, "Sorted”
    return
```

Performance

- *ImprovedBreakPointReversalSort* is an approximation algorithm with a performance guarantee of no worse than 4
  - It eliminates at least one breakpoint in every two steps; at most $2b(\pi)$ steps
  - Optimal algorithm eliminates *at most 2 breakpoints* in every step: $d(\pi) \geq b(\pi) / 2$
  - Approximation ratio:

$$\frac{2b(\pi)}{d(\pi)} \leq \frac{2b(\pi)}{\frac{b(\pi)}{2}} = 4$$



A Better Approximation Ratio

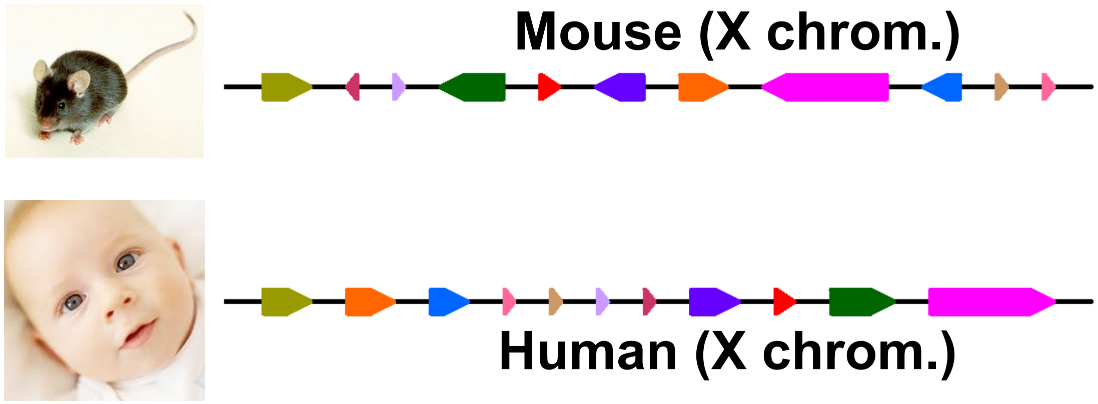
- If there is a decreasing strip, the next reversal reduces $b(\pi)$ by at least one.
- The only bad case is when there is no decreasing strip, as then we need a reversal that does not reduce $b(\pi)$.
  - If we could always choose a reversal reducing $b(\pi)$ and, at the same time, yielding a permutation that again has at least one decreasing strip, the bad case would never occur.
  - If all reversals that reduce $b(\pi)$ create a permutation without decreasing strips, then there exists a reversal that reduces $b(\pi)$ by two?!
  - *When the algorithm creates a permutation without decreasing strip, the previous reversal must have reduced $b(\pi)$ by two.*
- At most $b(\pi)$ reversals are needed.
- Approximation ratio: $\frac{b(\pi)}{d(\pi)} \le \frac{b(\pi)}{\frac{b(\pi)}{2}} = 2$

correct?

Both are Greedy Algorithms

- SimpleReversalSort
  - Attempts to maximize *prefix(π)* at each step
  - Performance guarantee: $\frac{n-1}{2}$

- ImprovedBreakPointReversalSort
  - Attempts to reduce the number of breakpoints at each step
  - Performance guarantee: 2



Try it yourself

0 1 | 3 | 8 7 6 | 2 | 4 5 | 9 *10*

Next Time

- Dynamic Programming Algorithms