

Genome Rearrangements

Study Chapters 5.3-5.5

1

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 6 2 3 4 5

Flip 2: 1 2 6 3 4 5

Flip 3: 1 2 3 6 4 5

Flip 4: 1 2 3 4 6 5

Flip 5: 1 2 3 4 5 6



We probably don't want to use this algorithm to estimate the reversal distance between two genomes

- But it could have been sorted in two flips:

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

Flip 1: 5 4 3 2 1 6

Flip 2: 1 2 3 4 5 6

2

Approximation Algorithms

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

$$\mathcal{A}(\pi) / \text{OPT}(\pi)$$

where

$\mathcal{A}(\pi)$ - solution produced by algorithm \mathcal{A}
 $\text{OPT}(\pi)$ - optimal solution of the problem

3

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 = $\max_{|\pi|=n} \mathcal{A}(\pi) / \text{OPT}(\pi) \geq 1.0$
 - For maximization algorithms:
 - Approx Ratio = $\min_{|\pi|=n} \mathcal{A}(\pi) / \text{OPT}(\pi) \leq 1.0$

4

Approximation Ratio

SimpleReversalSort(π)

```

1 for  $i \leftarrow 1$  to  $n-1$ 
2    $j \leftarrow$  position of element  $i$  in  $\pi$  (i.e.,  $\pi_j = i$ )
3   if  $j \neq i$ 
4      $\pi \leftarrow \pi \rho(i, j)$ 
5   output  $\pi$ 
6 if  $\pi$  is the identity permutation
7   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

approximation
ratio?

at least
 $(n-1)/2$



any better
greedy
algorithms?

5

New Idea: Adjacencies

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

- 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

6

Breakpoints

Breakpoints occur between neighboring non-adjacent elements:

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

- Pairs (1,9), (9,3), (4,7), (8,2) and (2,5) define 5 breakpoints of permutation π
- $b(\pi)$ - # breakpoints in permutation π

7

Extending Permutations

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

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

↓ Extending with 0 and 10

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

↑
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)

8

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

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

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

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

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



9

Sorting By Reversals: A Better Greedy Algorithm

BreakPointReversalSort(π)

- 1 while $b(\pi) > 0$
- 2 Among all possible reversals,
choose reversal ρ minimizing $b(\pi \cdot \rho)$
- 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

Does it always terminate?

How can we be sure that removing some breakpoints does not introduce others?



10

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 $n+1$)

11

Reducing the Number of Breakpoints

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

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. $b(\pi \cdot \rho) < b(\pi)$).



Which reversal?

How can we be sure that we don't introduce new breakpoints?

12

Things to Consider

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

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

- 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 $k-1$



13

Things to Consider

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

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

reduced by 1!

- 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

14

Things to Consider

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

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

- 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

15

Things to Consider

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

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

- 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

16

Things to Consider

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

$b(\pi) = 2$

- 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

17

Things to Consider

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

$b(\pi) = 0$

- 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

18

Things to Consider

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

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

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

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

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

$$d(\pi) = 3$$

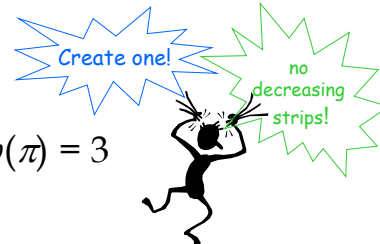
Does it work
for any
permutation?



19

Potential Gotcha

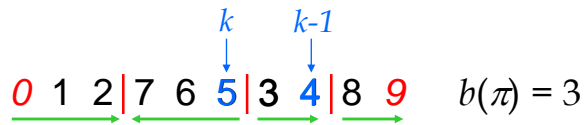
$$0\ 1\ 2\ |5\ 6\ 7|\ 3\ 4|\ 8\ 9 \quad b(\pi) = 3$$



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

20

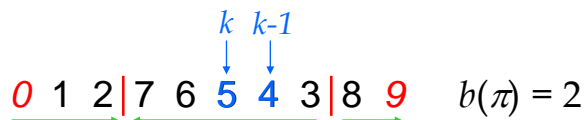
Potential Gotcha



- If there is no decreasing strip, there may be **no strip-reversal ρ that reduces the number of breakpoints** (i.e. $b(\pi \cdot \rho) \geq b(\pi)$ for any reversal ρ).
- However, reversing an increasing strip creates a decreasing strip, and the number of breakpoints remains unchanged (or decreases).
- Then the number of breakpoints will be reduced in the following steps.

21

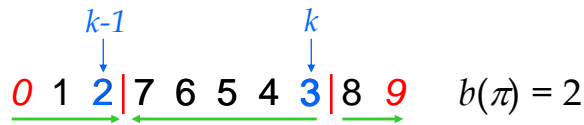
Potential Gotcha



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

22

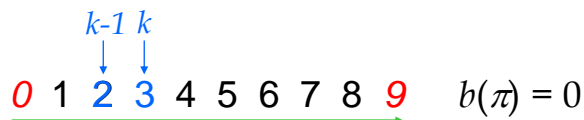
Potential Gotcha



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

23

Potential Gotcha



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

24

ImprovedBreakpointReversalSort

ImprovedBreakpointReversalSort(π)

```

1 while  $b(\pi) > 0$ 
2   if  $\pi$  has a decreasing strip
3     Among all possible reversals, choose reversal  $\rho$ 
        that minimizes  $b(\pi \cdot \rho)$ 
4   else
5     Choose a reversal  $\rho$  that flips an increasing strip in  $\pi$ 
6      $\pi \leftarrow \pi \cdot \rho$ 
7   output  $\pi$ 
8 return

```

25

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

```

26

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:

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

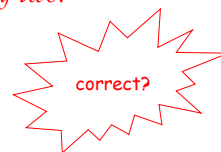


27

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:

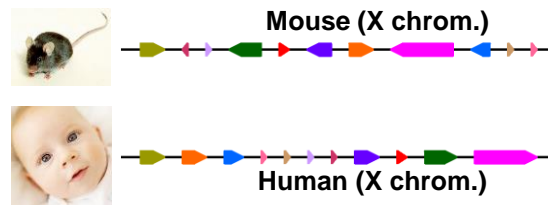
$$\frac{b(\pi)}{d(\pi)} \leq \frac{b(\pi)}{\frac{b(\pi)}{2}} = 2$$



28

Both are Greedy Algorithms

- SimpleReversalSort
 - Attempts to maximize $prefix(\pi)$ at each step
 - Performance guarantee: $\frac{n-1}{2}$
- ImprovedBreakPointReversalSort
 - Attempts to reduce the number of breakpoints at each step
 - Performance guarantee: 2



29

Try it yourself

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

30