

Genome Rearrangements

Study Chapters 5.3-5.5

Approximation Algorithms



- These 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



Approximation Ratio/Performance Guarantee



- **Approximation ratio (performance guarantee)** of algorithm \mathcal{A} : max approximation ratio over all inputs of size n
 - For algorithm \mathcal{A} that minimizes its objective function (minimization algorithm):
 - $\max_{|\pi| = n} \mathcal{A}(\pi) / \text{OPT}(\pi)$
 - For maximization algorithms:
 - $\min_{|\pi| = n} \mathcal{A}(\pi) / \text{OPT}(\pi)$



Approximation Ratio

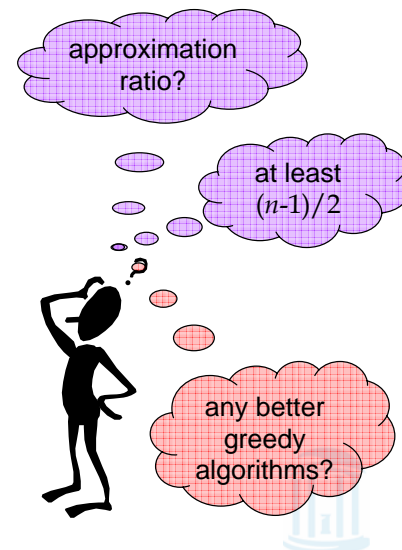


SimpleReversalSort(π)

```
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
```



Adjacencies



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

- A pair of 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



There is a **breakpoint** between any adjacent elements that are non-consecutive:

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

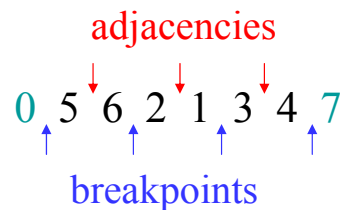
- Pairs (1,9), (9,3), (4,7), (8,2) and (2,5) form breakpoints of permutation π
- $b(\pi)$ - # breakpoints in permutation π



Adjacencies & Breakpoints

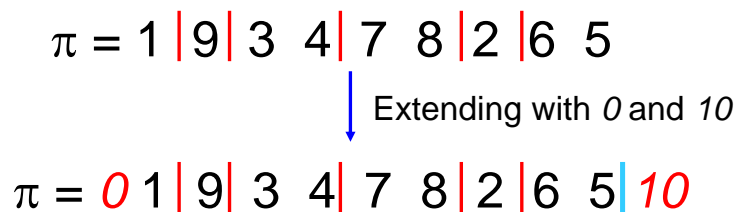
- An **adjacency** - a pair of adjacent elements that are **consecutive**
- A **breakpoint** - a pair of adjacent elements that are **not consecutive**

$\pi = 5 \ 6 \ 2 \ 1 \ 3 \ 4 \longrightarrow$ Extend π with $\pi_0 = 0$ and $\pi_7 = 7$



Extending Permutations

- We put two elements $\pi_0 = 0$ and $\pi_{n+1} = n+1$ at the ends of π



A new breakpoint was created after extending

A permutation of n may have up to $(n+1)$ breakpoints



Reversal Distance and Breakpoints



- Breakpoints are the *bottlenecks* for sorting by reversals.
- Each reversal eliminates at most 2 breakpoints.

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

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 $b(\pi) = 0$

$$d(\pi) \geq \frac{b(\pi)}{2}$$



Sorting By Reversals: A Better Greedy Algorithm



BreakPointReversalSort(π)

- 1 **while** $b(\pi) > 0$
- 2 Among all possible reversals,
choose reversal ρ minimizing $b(\pi \bullet \rho)$
- 3 $\pi \leftarrow \pi \bullet \rho(i, j)$
- 4 **output** π
- 5 **return**



Does it always terminate?

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



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 the strips with 0 and $n+1$

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?

Things to Consider



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



- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$



Things to Consider



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



- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip



Things to Consider



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



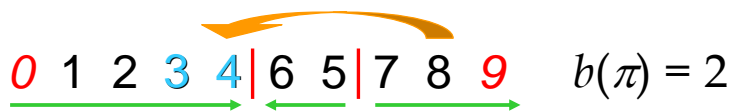
- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip



Things to Consider



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



- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip



Things to Consider



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



- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip



Things to Consider



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

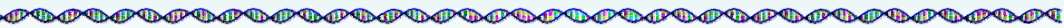
No breakpoint left!



- Choose the decreasing strip with the smallest element k in π
- Find $k - 1$ in the permutation
- Reverse the segment between k and $k-1$
- Repeat until there is no decreasing strip



Things to Consider



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

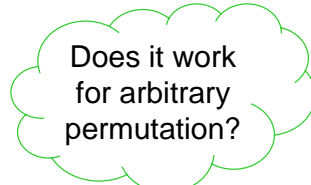
0 1 | 4 | 6 5 | 7 8 | 3 2 | 9 $b(\pi) = 5$

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 $b(\pi) = 0$

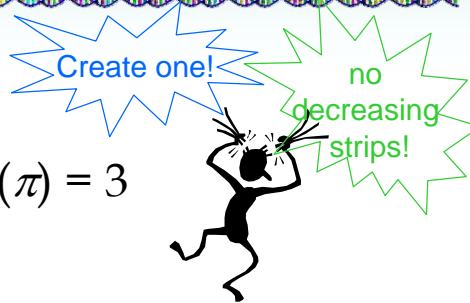
$d(\pi) = 3$



What if there is no Decreasing Strip?



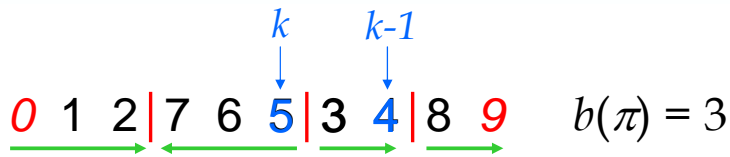
0 1 2 | 5 6 7 | 3 4 | 8 9 $b(\pi) = 3$



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



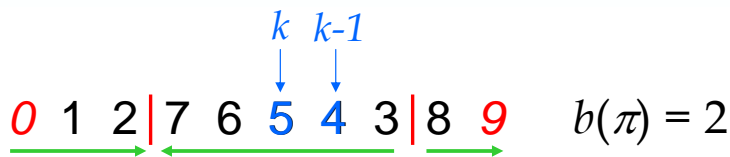
What if there is no Decreasing Strip?



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



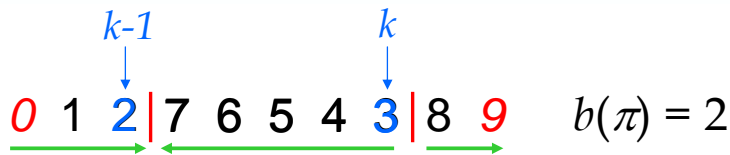
What if there is no Decreasing Strip?



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



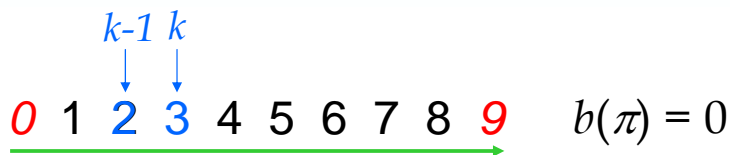
What if there is no Decreasing Strip?



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



What if there is no Decreasing Strip?



- If there is no decreasing strip, there may be **no reversal ρ that reduces the number of breakpoints** (i.e. $b(\pi \cdot \rho) \geq b(\pi)$) for any reversal ρ).
- By reversing an increasing strip (# of breakpoints remains unchanged), we will create a decreasing strip. Then the number of breakpoints will be reduced in the **next** step.



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
```



ImprovedBreakpointReversalSort: Performance Guarantee

- *ImprovedBreakPointReversalSort* is an approximation algorithm with a performance guarantee of at most 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?!
 - Whenever the algorithm creates a permutation without decreasing strip, the previous reversal reduced $b(\pi)$ by two.
- At most $b(\pi)$ reversals are needed.
- Approximation ratio: $\frac{b(\pi)}{d(\pi)} \leq \frac{b(\pi)}{\frac{b(\pi)}{2}} = 2$



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



Try it yourself



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



Next Time



- Dynamic Programming Algorithms

