Algorithms in Bioinformatics

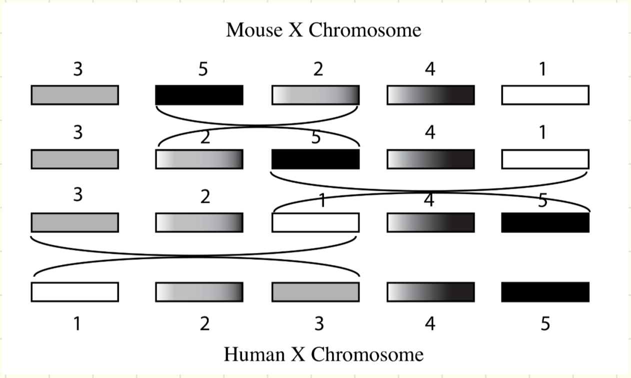
Greedy Algorithms

Recap

- Website should be updated
- HW - due next Tuesday
- Essays - hopefully Thursday?
- On Sep 27 - no class

Today: Ch 5

Problem Motivation : gene flipping



Simple version:

Rearrangement events can be modeled by reversals.

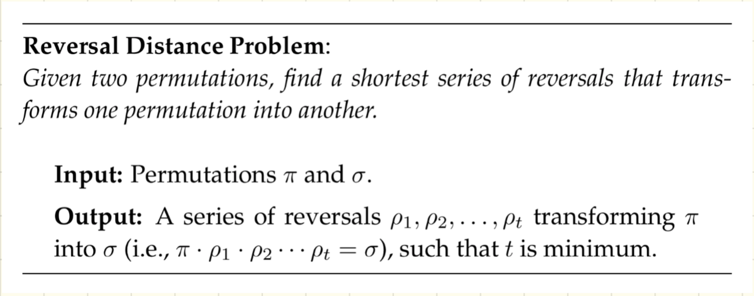
Order of synteny blocks (genes) can be represented by a permutation: a reordering of $1 .. n$.

$$\pi = \pi_1 \pi_2 \cdots \pi_n$$

Reversal: $\pi = \pi_1 \cdots \pi_{i-1} \overrightarrow{\pi_i \cdots \pi_j} \pi_{j+1} \cdots \pi_n$
$= \pi_1 \pi_2 \cdots \pi_i \pi_j \pi_{j-1} \cdots \pi_{i+1} \pi_i \cdots \pi_2$

Ex: $\pi = 1\,2\,\overline{4\,3\,7\,5}\,6 = 1\,2\,\underbrace{5\,7\,3\,4}\,6$

The problem:



Reversal Distance Problem:
Given two permutations, find a shortest series of reversals that transforms one permutation into another.

Input: Permutations π and σ.

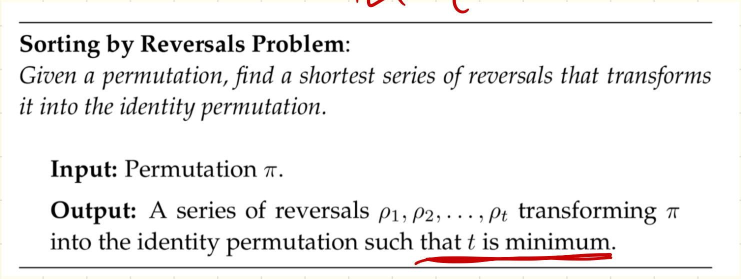
Output: A series of reversals $\rho_1, \rho_2, \ldots, \rho_t$ transforming π into σ (i.e., $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$), such that t is minimum.

Then $t = d(\pi, \sigma)$ is the reversal distance between π & σ.

In practice, one is the main one, & we reset σ to be $1..n$.
(& update π accordingly)

replace σ_i with i (in π)

Then:



Sorting by Reversals Problem:
Given a permutation, find a shortest series of reversals that transforms it into the identity permutation.

Input: Permutation π.

Output: A series of reversals $\rho_1, \rho_2, \ldots, \rho_t$ transforming π into the identity permutation such that t is minimum.

Then $t = d(\pi)$ is the reversal distance of π.

$\text{prefix}(\pi) = 3$

Ex: $\pi = 123\,\underbrace{64}_{\rho(4,5)}5$

What would you do?

$\pi \circ \rho(4,5)$

$\cancel{4}$

$1\;2\;3\;4\;\underbrace{6\;5}$

$1\;2\;3\;4\;5\;6$

$d(\pi) \leq 2$

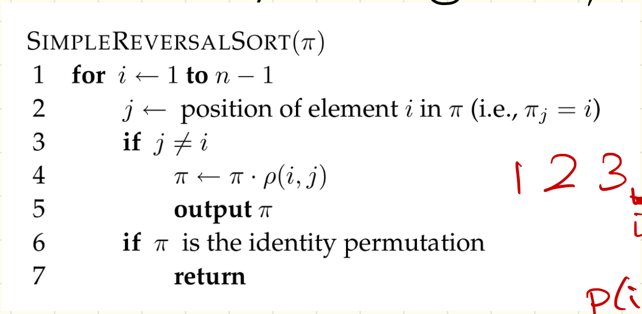
Can it be $= 1$?

(lower bnd)

exhaustively, no

Define prefix(π) = # of already sorted elements in π.

Try to increase this by 1 iteratively (and <u>greedily</u>):



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

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

$i=4$

$j=5$

$p(i,j) =$ swap from i to j

Does this always work?

use induction:
After round i of loop, i is in correct spot.

How many flips?

worst case: $n-1$

(<u>not</u> optimal)

Optimal?

$\underbrace{6\,2\,1}\,3\,4\,5$

Ex: $\overbrace{6\,1}\,2\,3\,4\,5$

greedy:

$1\,6\,2\,3\,4\,5$

$1\,2\,\overbrace{6\,3}\,4\,5$

5 swaps

$1\,2\,3\,\overbrace{6\,4}\,5$

$1\,2\,3\,4\,\overbrace{6\,5}$

$1\,2\,3\,4\,5\,6$

best?

$\underbrace{6\,1\,2\,3\,4\,5}$

no 2

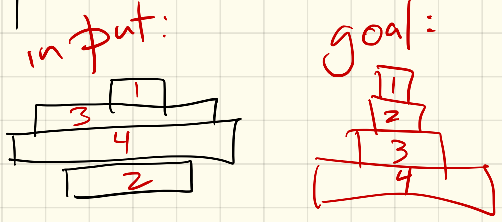
$\underbrace{5\,4\,3\,2\,1}\,6$

Worse: $\pi = n\;1\;2\;\dots\;(n-2)\;(n-1)$

Similar CS problem:

Pancake flipping

Stack of pancakes (out of order)



All you have is a spatula.

(Basically, permutations will all be prefixes.)

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

↳ 6 3 2 1 4 5

(The minimum # of needed flips is unknown.)

OK, so greedy doesn't always work!

Let's talk approximation:

Performance guarantee:

$A(\pi)$ = solution produced by algorithm A on input π

$OPT(\pi)$ = best solution on input π

Approximation ratio (for minimization problem) =

$$\max_{|\pi|=n} \frac{A(\pi)}{OPT(\pi)}$$

(Worst case scenerio)

Ex: We showed approx. ratio of SIMPLEREVERSESORT is $\geq \frac{n-1}{2}$

If $n=1001$, alg could return $500 \cdot OPT$ permutations.

Better: a different notion of greed

Breakpoints:

To make it easier, extend π:

$$\pi = \pi_0 \, \pi_1 \pi_2 \cdots \pi_n \, \pi_{n+1}$$

For any $i \leq n$,

- if $\pi_i \pi_{i+1}$ are consecutive, call it an adjacency.
- if not, a <u>break point</u> (& let $b(\pi) = \#$ breakpoints)

Ex: $\pi = 0, 2\ 1, \underline{3\ 4\ 5}, \underline{8\ 7\ 6}, 9$

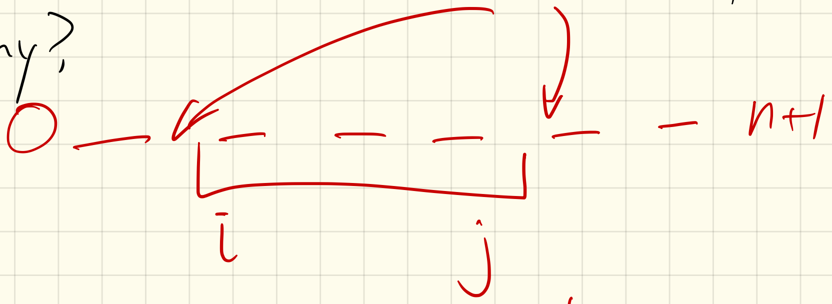
$$b(\pi) = 4$$

$$0 \ \overbrace{n \ \ n{-}1 \ \cdots \ 1} \ n{+}1$$

Note: only 1 permutation has 0 break points.

Observation: Every reversal can eliminate at most 2 breakpoints.

Why?



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

invent some parameter to measure hardness for lower bnds

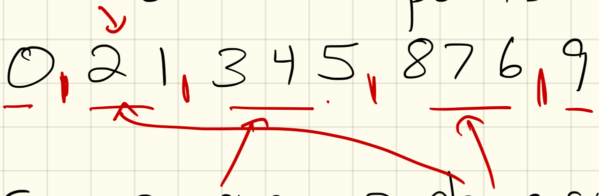
Goal: try to choose flips that lower $b(\pi)$

But - can removing a breakpoint increase new ones later?

Or could we get stuck?

Dfn: Strips: an interval between 2 breakpoints

Ex: $0 \mid 2\ 1 \mid 3\ 4\ 5 \mid 8\ 7\ 6 \mid 9$



Either increasing or decreasing (or size 1, in which case both)

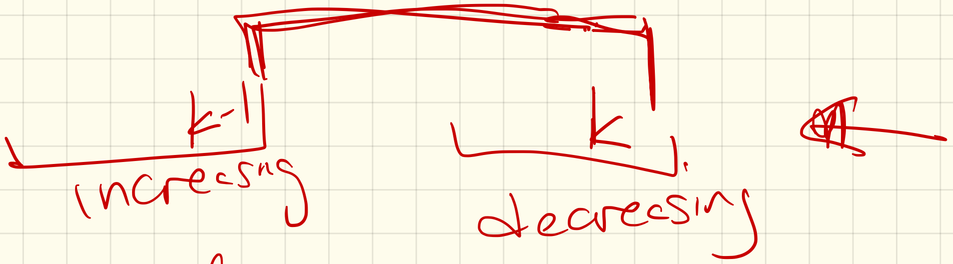
Thm: If a permutation π contains a decreasing strip or a strip of size 1, then there is a flip that decreases the # of breakpoints.

Pf: Choose decreasing (or single) strip with smallest k.

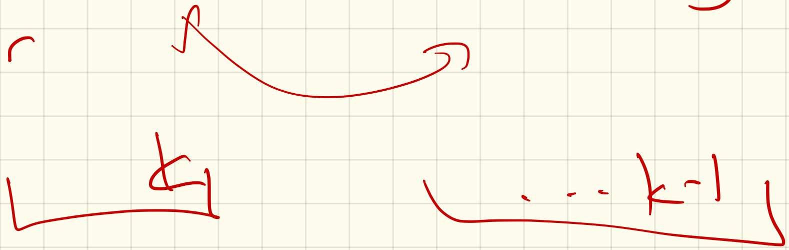
$\underline{0\,1\,2}\ \ \underline{7\,6\,5}\ \ \underline{8}\ \ \underline{4\,3}\ \ \underline{9}$

Note k-1 can't be in a decreasing strip, & it in fact ends some increasing strip

So



or



reverse between
k + k-1

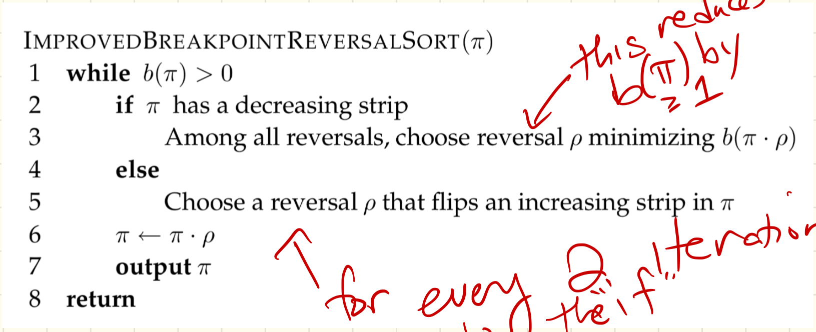
So - breakpts
decrease by 1
(maybe more)
k-1 + k are now adjacent

So: if have a decreasing strip,
can reduce.

If not: all strips are increasing.
What to do?

flip some increasing strip

Algorithm:



```
IMPROVEDBREAKPOINTREVERSALSORT($\pi$)
1  while $b(\pi) > 0$
2      if $\pi$ has a decreasing strip
3          Among all reversals, choose reversal $\rho$ minimizing $b(\pi \cdot \rho)$
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
```

this reduces $b(\pi)$ by ≥ 1

for every 2 iterations, do the if

Thm: This algorithm is a 4-approximation.

(ie it's # of swaps is $\leq 4 \cdot OPT$)

pf: # steps in alg $\leq 2\,b(\pi)$

since $b(\pi)$ goes down by (at least) 1 for every 2 iterations

and observation 3? slides ago:

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

$\hookrightarrow$ rearrange $b(\pi) \leq 2d(\pi)$

$$\frac{A(\pi)}{OPT(\pi)}$$

steps in alg $\leq 2b(\pi) \leq 2(2d(\pi))$

$\leq 4d(\pi)$

Another example: Motifs again
(Sec. 5.5)

Motif finding: $O(ln^t)$ or $O(4^l nt)$

Greedy approch:

- Scan each DNA sequence once
- Decide which l-mers have best contribution as we go

More detail:

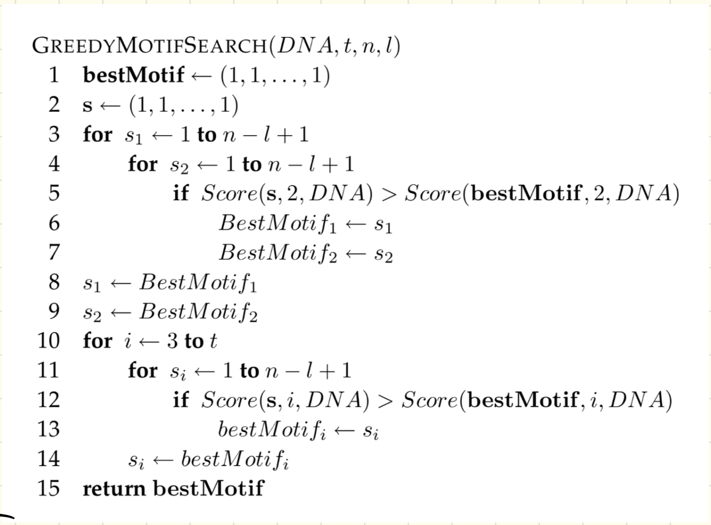
- Scan sequence 1 + 2, + find 2 closest l-mers (in Hamming distance)

  Time: $l(n-l+1)$

- Then as $j \leftarrow 3$ to t scan j^{th} sequence + select the best l-mer (max Score(s, i))

  Time: $l(n-l+))$ each time

Pseudocode:



```
GreedyMotifSearch($DNA, t, n, l$)
bestMotif $\leftarrow (1, 1, \dots, 1)$
s $\leftarrow (1, 1, \dots, 1)$
for $s_1 \leftarrow 1$ to $n - l + 1$
    for $s_2 \leftarrow 1$ to $n - l + 1$
        if $Score(\mathbf{s}, 2, DNA) > Score(\mathbf{bestMotif}, 2, DNA)$
            $BestMotif_1 \leftarrow s_1$
            $BestMotif_2 \leftarrow s_2$
$s_1 \leftarrow BestMotif_1$
$s_2 \leftarrow BestMotif_2$
for $i \leftarrow 3$ to $t$
    for $s_i \leftarrow 1$ to $n - l + 1$
        if $Score(\mathbf{s}, i, DNA) > Score(\mathbf{bestMotif}, i, DNA)$
            $bestMotif_i \leftarrow s_i$
    $s_i \leftarrow bestMotif_i$
return bestMotif
```

Time:

This tool is called CONSENSUS.

No approximation ratio!
So optimal can be missed entirely, or can be bad.

<u>Note:</u> - usually do more than 2 rows

- usually run several times in random order

Next time:

Greedy approach for Shortest common Superstring.

mention in 8.4

(my notes are based on another source - I'll post a link)