# A Greedy Approach to Motif Finding

Selected Topics in Computer Intelligence - 2015

***Bioinformatics Programming***

Computer Engineering, Chiang Mai University

## Greedy Algorithms

- Choosing interesting or sensible option and not consider any other
  - Coin changing problem!!! – choosing the largest one first, minimize the number of coins returned
- Sometimes returns incorrect results in some cases
  - Short-sighted notion of “good” option
  - Aka. Suboptimal results
- Takes very little time

# Genome Rearrangement

- Waardenburg Syndrome
  - Described in detail by the Dutch ophthalmology in 1951
  - A rare human genetic disorder resulting in hearing loss and pigmentary abnormality
  - e.g., Two differently colored eyes
  - Gene implication in human chromosome 2
- Mutated mouse with pigmentary abnormalities had been studied and found a "splotch" gene in its chromosomes

## Genome Rearrangement

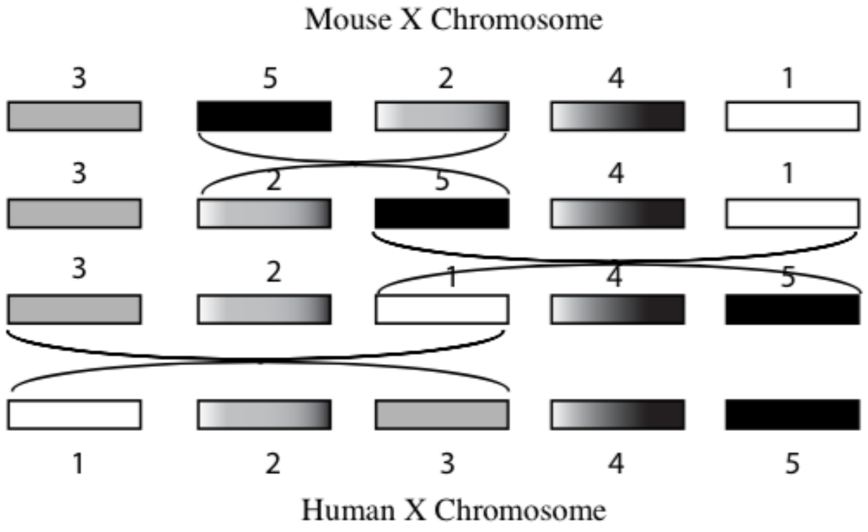
- According to gene mapping:
  - There are groups of genes in mice that appear in the same order as they do in humans
- Human genome is like the mouse genome cut into about 300 large genomic fragments (synteny blocks) that have been pasted together in different order
  - Both sequences are two different shuffling of the ancient mammalian genome
- Synteny
  - The condition of possessing common chromosome sequence
  - The conservation of blocks of order within two sets of chromosomes that are being compared with each other

## Genome Rearrangement

- Find a gene in mice leads to clues about the location of the related gene in humans
- Series of rearrangements can alter the genomic architecture of a species
- Analyzing the rearrangement history of genomes is a challenge problem
  - Solving the combinatorial puzzle of finding a series of rearrangements that transform one genome to another

# Genome Rearrangement

- Rearrangement scenario
  - The elementary rearrangement event is the flipping of genomic segment – a reversal or an inversion



  - Biologists are interested in the scenario involving the smallest number of reversals
    - Lower bound of rearrangement that could occur

## Sorting by Reversals

- Rearrangement events can be modeled by a series of reversals
- The order of genes (synteny blocks) can be represented by a permutation

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

  - E.g. synteny blocks on chromosome X vs. ordering in mice

*(1, 2, 3, 4, 5)  vs. (3, 5, 2, 4, 1)*

## Sorting by Reversals

- A reversal $\rho(i,j)$ transforms

$$\pi = \pi_1 \cdots \pi_{i-1} \underrightarrow{\pi_i \pi_{i+1} \cdots \pi_{j-1} \pi_j} \pi_{j+1} \cdots \pi_n$$

into

$$\pi \cdot \rho(i,j) = \pi_1 \cdots \pi_{i-1} \underleftarrow{\pi_j \pi_{j-1} \cdots \pi_{i+1} \pi_i} \pi_{j+1} \cdots \pi_n$$

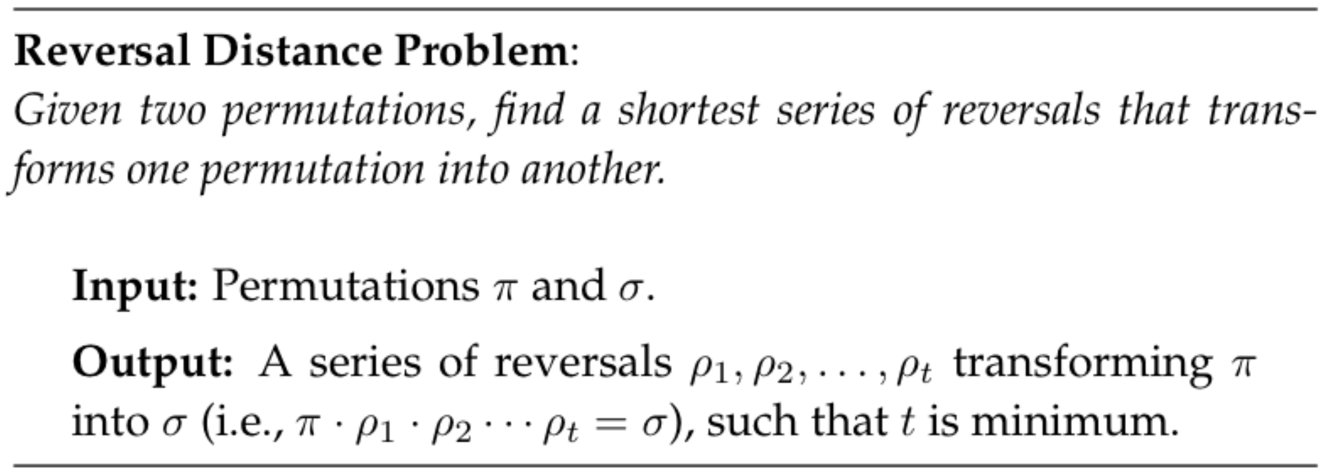
- e.g.,

$$\pi = 1\,2\,\underrightarrow{4\,3\,7\,5}\,6$$

$$\pi \cdot \rho(3,6) = 1\,2\,\underleftarrow{5\,7\,3\,4}\,6$$

## Sorting by Reversals

- Biological rearrangement process



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

**Input:** Permutations $\pi$ and $\sigma$.

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

$d(\pi, \sigma)$ – reversal distance between $\pi$ and $\sigma$

in practice, we use $\sigma$ to be the *identity permutation* $1,2,\ldots,n$

# Sorting by Reversals

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

**Input:** Permutation $\pi$.

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

$d(\pi)$ – reversal distance between $\pi$ and the *identify permutation*

- We do not need to move the already-sorted elements

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

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

$prefix(\pi) = 123$

## Sorting by Reversals

- Moving its $i^{th}$ element to the $i^{th}$ position

```
SIMPLEREVERSALSORT($\pi$)
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          $\pi \leftarrow \pi \cdot \rho(i, j)$
5          output $\pi$
6      if $\pi$ is the identity permutation
7          return
```

- A greedy algorithm that chooses the "best" reversals at every step but increasing *prefix($\pi$)* does NOT guarantee the optimal

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

$$\underline{6\,1\,2\,3\,4\,5} \rightarrow \underline{5\,4\,3\,2\,1}\,6 \rightarrow 1\,2\,3\,4\,5\,6 \qquad \textit{optimal!!!}$$

# Sorting by Reversals

- SIMPLEREVERSALSORT is not a correct algorithm!
  - Takes $n - 1$ steps to sort $\pi = n\ 1\ 2 \ldots (n - 1)$
  - Even though the minimum reversal distance, $d(\pi) = 2$
- Pancake Flipping Problem:
  - Finding $d(\pi)$ of the form $\rho(1, i)$ sorting $\pi$
  - Rearranging pancake
    - Goal: Smaller on on top of the larger one
    - Reversing several from the top
    - Repeating as many times as necessary
  - What is the maximum number of flips?
    - Given that there are $n$ pancakes

$$(\underline{1\,2\,3\,6}\,4\,5 \rightarrow \underline{6\,3\,2\,1\,4\,5} \rightarrow \underline{5\,4\,1\,2\,3}\,6 \rightarrow \underline{3\,2\,1}\,4\,5\,6 \rightarrow 1\,2\,3\,4\,5\,6)$$

# Approximation Algorithms

- Computer scientists often find a compromise in approximation algorithms – approx. solution
  - Better than an optimal on (with trade-off)

- Approx. ratio of algorithm $A$ on input $\pi$ is defined as

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

Soln. produced by algorithm $A$

The optimal soln. of the problem

- Performance guarantee of algorithm $A$ (approx. ratio)

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

$A$ is minimization algorithm

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

$A$ is maximization algorithm

*Worst-case scenario*

# Approximation Algorithms

- SIMPLEREVERSALSORT has approx. ratio = $(n - 1) / 2$
  - If n is 1001, the algorithm could have reversals of 500 times (of the optimal)
- Goal: design approx. algorithms with better performance guarantee

## Breakpoints

- SIMPLEREVERSALSORT is a naive measure of progress toward the identity permutation
- Better algorithm can be derived based on breakpoints rather than *prefix($\pi$)*
- Extending the permutation $\pi$ by adding $\pi_0 = 0$ and $\pi_{n+1} = n+1$ on both ends
  - We do NOT move $\pi_0$ and $\pi_{n+1}$
- Adjacency – if $\pi_i$ and $\pi_{i+1}$ are consecutive numbers
- Breakpoint – if $\pi_i$ and $\pi_{i+1}$ are NOT consecutive numbers

## Breakpoints

- Example:

  $$0\ \underleftarrow{2\ 1}\ \underrightarrow{3\ 4\ 5}\ \underleftarrow{8\ 7\ 6}\ 9$$

  - Extended by 0 and 9 on the ends
  - Decreasing strip $(\leftarrow)$
  - Increasing strip $(\rightarrow)$
  - Adjacencies: 21, 34, 45, 87, 76
  - Breakpoints: 02, 13, 58, 69
- Maximum # breakpoints = $(n + 1)$

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

## Breakpoints

- The identity permutation is the only one without breakpoints

  $$0\ 1\ 2\ 3\ 4\ 5\ \ldots\ n\ (n+1)$$

  - A breakpoint must be separated in process of transforming $\pi$ to the identity permutation

- Every reversal can eliminate at most two breakpoints

  $$d(\pi) \geq \frac{b(\pi)}{2}, \quad b(\pi) \text{ is the number of breakpoints}$$

## Breakpoints

- Algorithm with breakpoint

```
BreakpointReversalSort($\pi$)
1 while $b(\pi) > 0$
2     Among all reversals, choose reversal $\rho$ minimizing $b(\pi \cdot \rho)$
3     $\pi \leftarrow \pi \cdot \rho$
4     output $\pi$
5 return
```

  - Why is this algorithm better than SimpleReversalSort?
  - Does removing breakpoints does not introduce other?

- Strip – interval between two consecutive breakpoints
  - *Increasing* and *decreasing* strips
  - Single-element → decreasing strip
  - Element $0$ and $(n+1)$ → increasing strip

# Approximation Algorithms

- *Theorem 5.1*
  - If a permutation $\pi$ contains a decreasing strip
  - There is a reversal $\rho$ that decreases # breakpoints in $\pi$

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

- Example:

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

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

  - If a permutation $\pi$ does NOT contain a decreasing strip
  - Find any increasing strip (exclude $\pi_0$ and $\pi_{n+1}$) and flip it

## Breakpoints

- Improved Breakpoint algorithm

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

## Breakpoints

- *Theorem 5.2*
  - The algorithm is an approx. algorithm with a performance guarantee of at most 4
- The algorithm will eliminates at least one breakpoint in every two steps
  - Firstly flip the increasing strip, then remove breakpoints
  - Worst-case requires $2b(\pi)$ steps
  - The approx. ratio = $2b(\pi)/d(\pi)$, where $d(\pi) \geq \frac{b(\pi)}{2}$
  - The performance guarantee bounded above by:

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

# A Greedy Motif Finding

- Based on *Gary Stormo* and *Gerald Hertz*

```
GREEDYMOTIFSEARCH($DNA, t, n, l$)
bestMotif $\leftarrow (1, 1, \ldots, 1)$
s $\leftarrow (1, 1, \ldots, 1)$
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
```