Comp 555 - BioAlgorithms - Spring 2021

- Problem set #4 grades are posted
- Problem set #5 due Tonight
- Problem set #6 is posted

Genome Rearrangements

The Pancake Flipping Problem¶

The chef at "Breadman's" is sloppy.

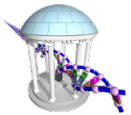
He makes pancakes of nonuniform sizes, and just throws them onto the plate.

Before the waitress delivers them to your table, she rearranges them so that the smaller pancakes are stacked on the larger ones.

Since she has only one free hand to perform this culinary rearrangement, she does it by inserting a spatula into the stack and then she flips all the cakes above the spatula. She repeats the process over and over until the entire stack is sorted.

I was wondering, how many such flips are needed for her to transform a sloppy stack into a sorted one?

Pancake Flipping Problem: Formulation



- **Goal:** Given a stack of *n* pancakes, what is the minimum number of flips to rearrange them into a sorted stack (small on top of large)?
- **Input:** A permutation Π
- **Output:** A series of *prefix reversals*, ρ_1, ρ_2, ρ_t that transforms Π into the identity permutation such that t is minimal.

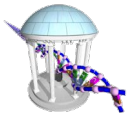
$$\Pi = \underline{\pi_1, \ldots, \pi_{i-1}, \pi_i}, \pi_{i+1} \ldots \pi_n$$

$$\Pi = \pi_i, \pi_{i-1}, \ldots, \pi_1, \pi_{i+1} \ldots \pi_n$$

"Bring-to-Top" Algorithm

1. Flip the biggest pancake to the top of the stack
2. Flip the entire stack (*n*), to place the biggest pancake on the bottom
3. Flip the next largest pancake to the top
4. Flip the n-1 top pancakes, thus, placing the 2nd largest on top of the largest
5. Repeat until sorted

"Bring-to-Top" Algorithm for *n* Pancakes

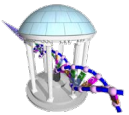


- If *n* = 1, the smallest pancake is already on top - we are done
- Otherwise, flip pancake *n* to the top, and then flip it to position *n*
- Now use:

Analysis:

- Performs 2 flips to put the i^{th} pancake into its correct position.
- There are (n-1) pancakes to sort
- So, no more than 2(n−1) flips are needed to sort the entire stack

What type of algorithm is "Bring-to-Top?



The point of every "pair" of steps is to bring the largest pancake to its proper position without considering the resulting position of any other pancakes. It is ***Greedy***.

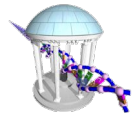
An algorithm where, at each step, one takes what seems to be the immediately best option

- Cons
  - It may return incorrect results
  - It may require more steps than necessary
- Pros
  - It often requires very little time to make a greedy choice
  - Choices are considered independently

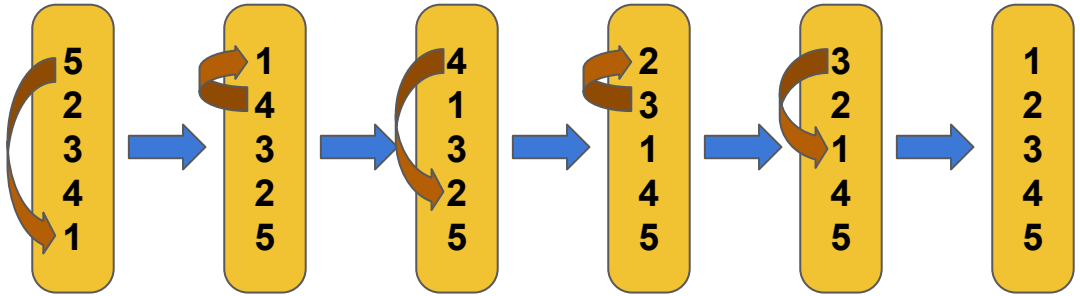
We have seen greedy algorithms in previous lectures.

Coin change problem

Is Bring-to-Top the best we can do?

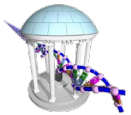


- Our algorithm is correct, but can it be done with fewer flips?
- Let's apply it to the following stack



- The "Bring-to-Top" algorithm sorted the stack in only 5 flips!
- The predicted "8" flips is an upper-bound on the number of flips required by "Bring-to-Top"
- Is there another algorithm that can always rearrange the stack in fewer flips?

Bring-to-Top in Code

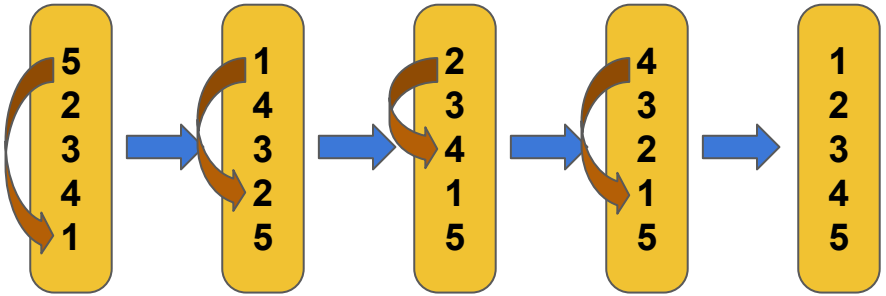


In [7]:
```
def BringToTop(pi):
    t = 0
    for bottom in range(len(pi)-1,0,-1):
        i = pi.index(max(pi[:bottom+1]))
        if (i != bottom):
            if (i > 0):
                pi = [pi[j] for j in range(i,-1,-1)] + pi[i+1:]
                print("   Moved to top:", pi)
                t += 1
            pi = [pi[j] for j in range(bottom,-1,-1)] + pi[bottom+1:]
            print("Moved to bottom:", pi)
            t += 1
    return t

print(BringToTop([5,2,3,4,1]))
```

```
Moved to bottom: [1, 4, 3, 2, 5]
   Moved to top: [4, 1, 3, 2, 5]
Moved to bottom: [2, 3, 1, 4, 5]
   Moved to top: [3, 2, 1, 4, 5]
Moved to bottom: [1, 2, 3, 4, 5]
5
```

You can do it in 4 flips!

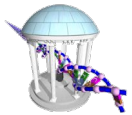


William Gates III (yeah, the Microsoft guy) and Christos Papadimitriou showed in the mid-1970s that this problem can be solved by at least *17/16 n* and at most *5/3 (n+1)* prefix reversals (flips) of *n* pancakes.

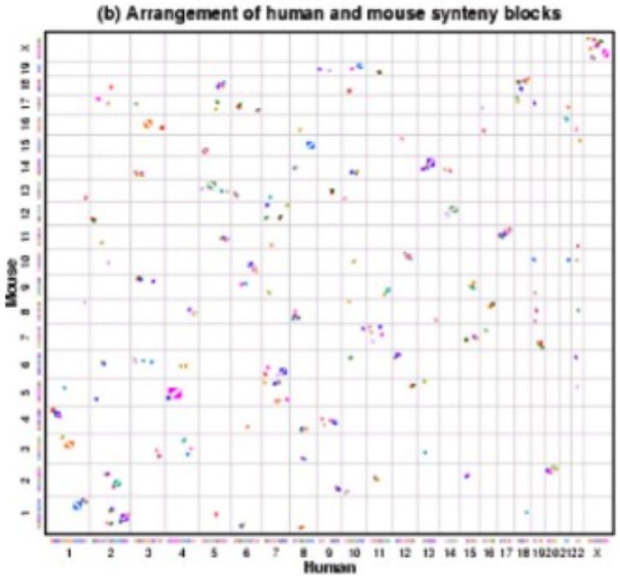
A Serious Scientific Problem

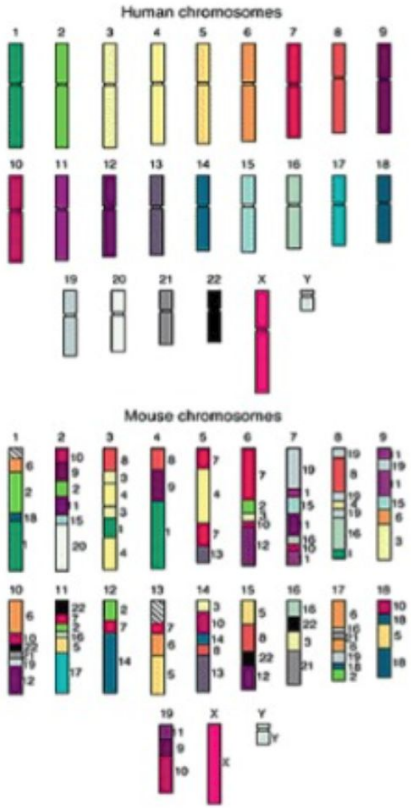
- Some organisms are obviously similar...
- Some organisms are obviously different...
- Some are close calls...
- The differences are all in their genes!
- And the gene order is important!

Genome Rearrangements

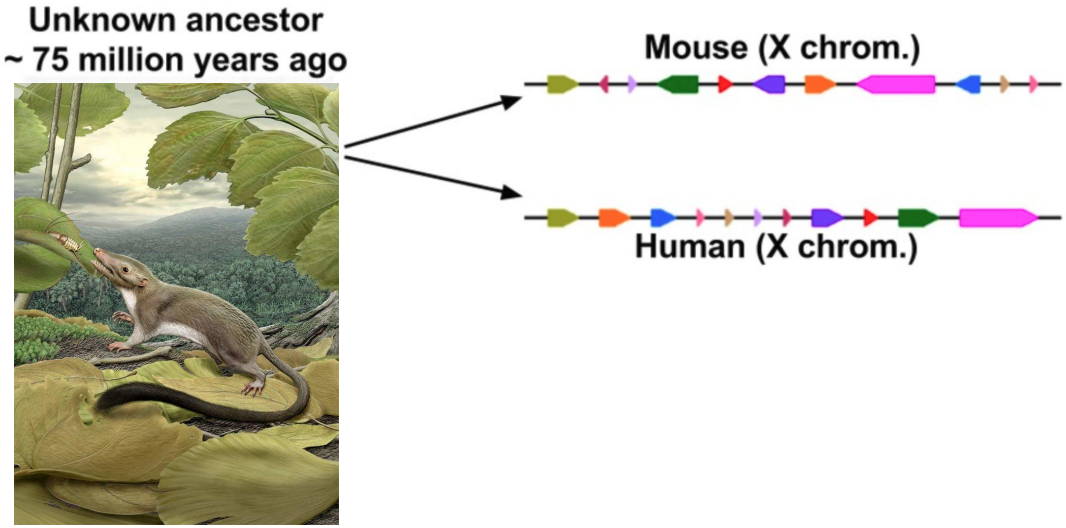


- Humans and mice have similar genomes, but their genes are ordered differently
- ≈ 245 rearrangements
- ≈ 300 large synteny blocks



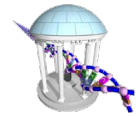


Genomes are Connected



- What are the syntenty blocks, and how do we find them?
- What is the architecture of the common ancestral genome?
- What is the evolutionary scenario for transforming one genome into the other?

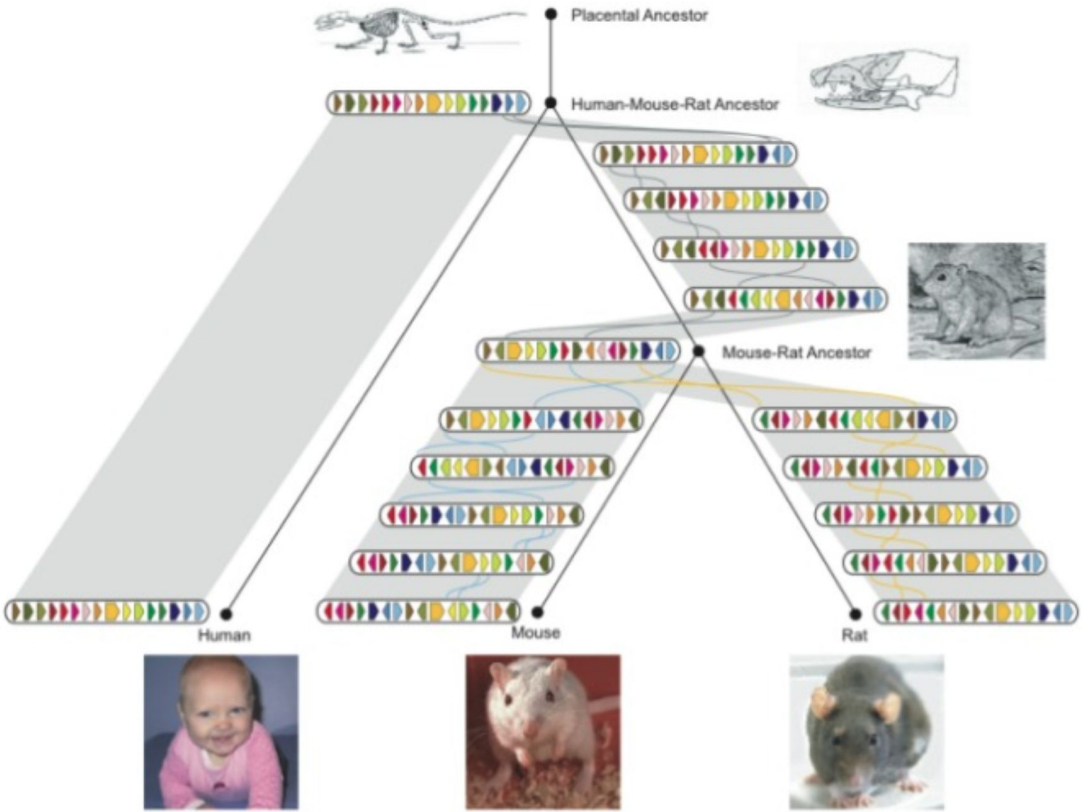
The History of X



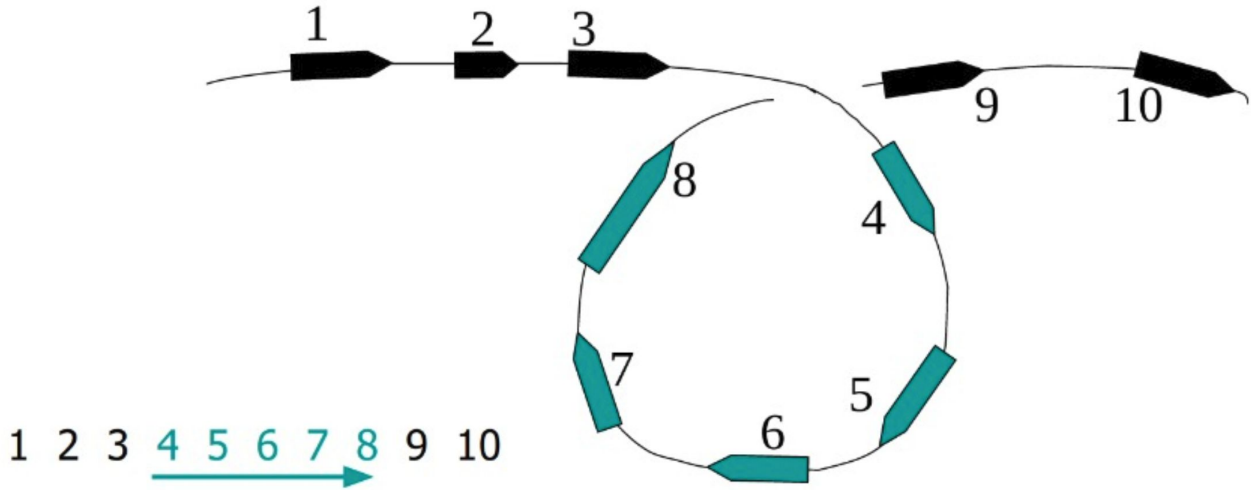
Rat Consortium, Nature 2004

Rearrangement Events:

- Reversals
- Fusions
- Fissions
- Translocations

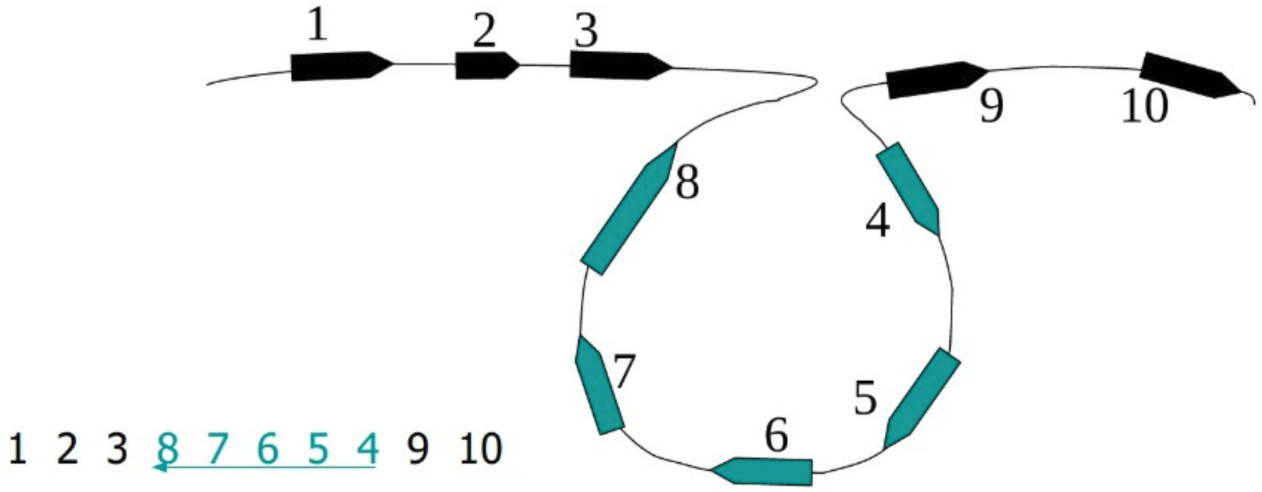


Let’s consider Reversals



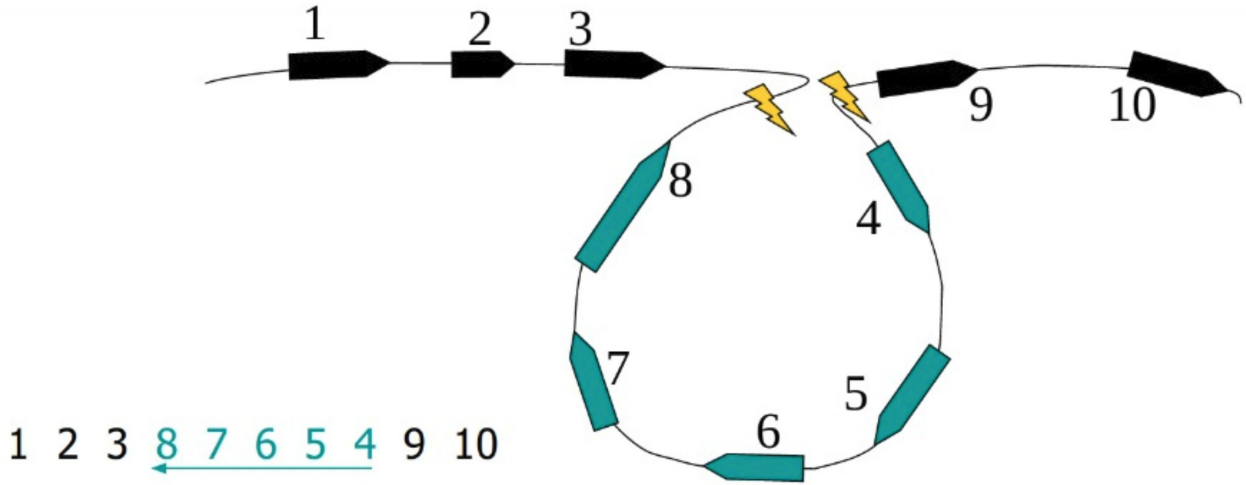
- Blocks represent conserved genes
- Reversals, or ***inversions***, are particularly relevant to speciation. Recombinations cannot occur between reversed and normally ordered segments. Must recombine to reproduce.

Let’s consider Reversals



- Blocks represent conserved genes
- In the course of evolution, blocks 1 ... 10 could be reordered as 1 2 3 8 7 6 5 4 9 10

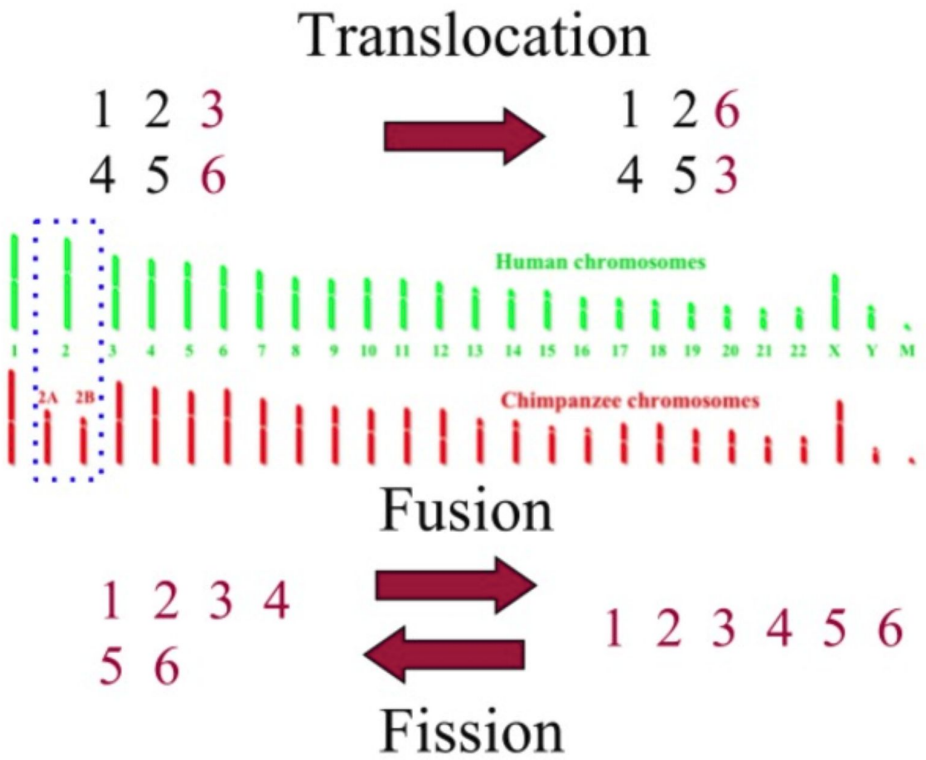
Reversal Breakpoints



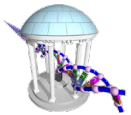
- A inversion introduces two breakpoints (disruptions in order).

Other Types of Rearrangements

- Translocations - genes move from one contig to another
- Fusion - two contigs join into one
- Fission - a single contig splits into two
- **Inversions - A segment of DNA is reversed and replaced in the genome (Inversons create barriers to future recombinations)**



Reversals and Gene Orders



- Gene order can be represented by a permuation Π:

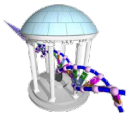
$$\pi = \pi_1 \ldots \pi_{i-1} \pi_i \pi_{i+1} \ldots \pi_{j-1} \pi_j \pi_{j+1} \ldots \pi_n$$

$$\rho(i,j)$$

$$\pi_1 \ldots \pi_{i-1} \pi_j \pi_{j-1} \ldots \pi_{i+1} \pi_i \pi_{j+1} \ldots \pi_n$$

- Reversal $\rho(i,j)$ reverses (flips) the elements from i to j in Π
- Like a two-spatula “flip”

Reversal Examples



- Reversal $\rho(i,j)$ reverses (flips) the elements from i to j in Π
- How many reversals are required to transform one permutation to another?

$$\Pi = 1, 2, \underline{3, 4, 5}, 6, 7, 8$$

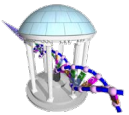
$$\rho(3, 5) \downarrow$$

$$\Pi = 1, 2, 5, 4, \underline{3, 6}, 7, 8$$

$$\rho(5, 6) \downarrow$$

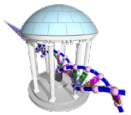
$$\Pi = 1, 2, 5, 4, 6, 3, 7, 8$$

"Reversal Distance" Problem



- **Goal:** Given two permutations over *n* elements,
  find the shortest series of reversals that transforms one into another.
- **Input:** Permutations Π and Σ
- **Output:** A series of reversals $\rho_1(s_1,e_1),\rho_2(s_2,e_2),...\rho_t(s_t,e_t)$ that transforms Π to Σ such that t is minimal
- t - reversal distance between Π and Σ

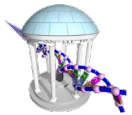
"Sorting By Reversals" Problem



A simplified restatement of the same problem...

- **Goal:** Given a permutation find the shortest series of reversals that transforms it into the identity permutation $1,2,3,...n$.
- **Input:** Permutation Π
- **Output:** A series of reversals $\rho_1(s_1,e_1),\rho_2(s_2,e_2),...\rho_t(s_t,e_t)$ transforming Π to the identity permutation such that *t* is minimal
- t = d(Π) - reversal distance of Π

Sorting By Reversals: Example



$$\Pi_0 = \underline{3,4},2,1,5,6,7,10,9,8 \qquad \rho(1,2)$$

$$\Pi_1 = \overline{4,3,2,1},5,6,7,\underline{10,9,8} \qquad \rho(8,10)$$

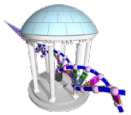
$$\Pi_2 = \underline{4,3,2,1},5,6,7,\overline{8,9,10} \qquad \rho(1,4)$$

$$\Pi_3 = \overline{1,2,3,4},5,6,7,8,9,10$$

$$d(\Pi) = 3$$

- Three reversals sort this permutation

Sorting By Reversals: Another example



$$\Pi_0 = 2, \underline{4, 3}, 5, 8, 7, 6, 1 \qquad \rho(2, 3)$$

$$\Pi_1 = 2, 3, 4, 5, \underline{8, 7, 6}, 1 \qquad \rho(5, 7)$$

$$\Pi_2 = \underline{2, 3, 4, 5, 6, 7, 8}, 1 \qquad \rho(1, 7)$$

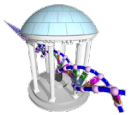
$$\Pi_3 = \underline{8, 7, 6, 5, 4, 3, 2, 1} \qquad \rho(1, 8)$$

$$\Pi_4 = 1, 2, 3, 4, 5, 6, 7, 8$$

$$d(\Pi) = 4$$

- It took 4 reversals to sort this permutation. Can it be done in 3? What is the minimum number required?

A Greedy Algorithm for Sorting by Reversals



Inspired by our pancake flipping algorithm:

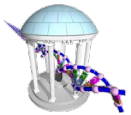
- When sorting the permutation, $\Pi=1,2,3,6,4,5$, one notices that the first three elements $(1,2,3)$ are already in order.
- So it does not make sense to break them apart.
- The length of the already sorted prefix of Π is denoted as prefix(Π)=3
- This inspires the following simple ***greedy*** algorithm

  while prefix(Π) < len(Π):

  perform a reversal ρ(prefix(Π)+1,k) such that prefix(Π) increases by at least 1.

- Such a reversal must always exist
- Finding, k, is as simple as finding the index of the minimum value of the remaining unsorted part

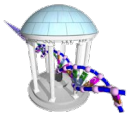
Geedy Reversal Sort: Example



$$Step1 : \Pi_1 = 1, 2, 3, \overline{6, 4}, 5 \qquad \rho(4, 5)$$
$$Step2 : \Pi_2 = 1, 2, 3, 4, \overline{6, 5} \qquad \rho(5, 6)$$
$$Done : \Pi_3 = 1, 2, 3, 4, 5, 6$$

- The number of steps to sort any permutation of length n is at most (n−1)

Greedy Reversal Sort as code



In [4]:

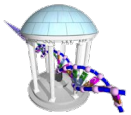
```
def GreedyReversalSort(pi):
    t = 0
    for i in range(len(pi)-1):
        j = pi.index(min(pi[i:]))
        if (j != i):
            pi = pi[:i] + [v for v in reversed(pi[i:j+1])] + pi[j+1:]
            print("rho(%2d,%2d) = %s" % (i+1,j+1,pi))
            t += 1
    return t

print(GreedyReversalSort([3,4,2,1,5,6,7,10,9,8]))
```

```
rho( 1, 4) = [1, 2, 4, 3, 5, 6, 7, 10, 9, 8]
rho( 3, 4) = [1, 2, 3, 4, 5, 6, 7, 10, 9, 8]
rho( 8,10) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3
```

- How fast is GreedyReversalSort()?
- Can it be improved upon?

Analyzing GreedyReversalSort()

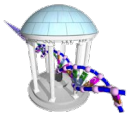


- GreedyReversalSort requires at most $n-1$ steps
- For example, on $\Pi=6,1,2,3,4,5$, $t=5$

$$\begin{aligned}
\Pi_1 &: 6, 1, 2, 3, 4, 5 \\
\rho(1,2) &: 1, 6, 2, 3, 4, 5 \\
\rho(2,3) &: 1, 2, 6, 3, 4, 5 \\
\rho(3,4) &: 1, 2, 3, 6, 4, 5 \\
\rho(4,5) &: 1, 2, 3, 4, 6, 5 \\
\rho(5,6) &: 1, 2, 3, 4, 5, 6
\end{aligned}$$

- But there is a solution with far fewer flips

Greed Gone Wrong

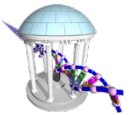


- The same sequence sorted with two reversals

$$\Pi : 6, 1, 2, 3, 4, 5$$
$$\rho(2, 6) : 6, 5, 4, 3, 2, 1$$
$$\rho(1, 6) : 1, 2, 3, 4, 5, 6$$

- Note, this solution makes no progress
  (no elements of the permutation are placed in their correct order) after its first move
- Yet it beats a greedy approach handily.
- So SimpleReversalSort(π) is correct (as a sorting routine), but non-optimal
- For many problems there is no known optimal algorithm, in such cases approximation algorithms are often used.

Approximation Algorithms

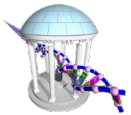


- Algorithms that find solutions that are within some bounded ratio of the optimal solution
- The *approximation ratio* of an algorthim, A on input Π is:

$$r = \frac{\mathcal{A}(\Pi)}{OPT(\Pi)}$$

- Where:
  - $A(\Pi)$ is the number of steps using the given algorithm
  - $OPT(\Pi)$ is the number of steps required using, a possibly unknown, optimal algorithm

Performance Guarantees



- On an occasional input our approximation algorithm may give an optimal result, however we want to consider the value of *r* for the worst case input
- When our objective is to minimize something (like reversals in our case)

$$r = \max_{i=0}^{len(\Pi)!} \frac{\mathcal{A}(\Pi_i)}{OPT(\Pi_i)} \geq 1.0$$

- Or, when our ojective is to maximize something (like money)

$$r = \max_{i=0}^{len(\Pi)!} \frac{\mathcal{A}(\Pi_i)}{OPT(\Pi_i)} \leq 1.0$$

- Sounds cool in theory, but there are lots of open ends here
  - If we don't know $OPT(\Pi_i)$ how are we supposed to know how many steps it requires?
  - Do we really need to test for all $len(\Pi)!$ possible inputs?

How do we get Approximation Ratios?

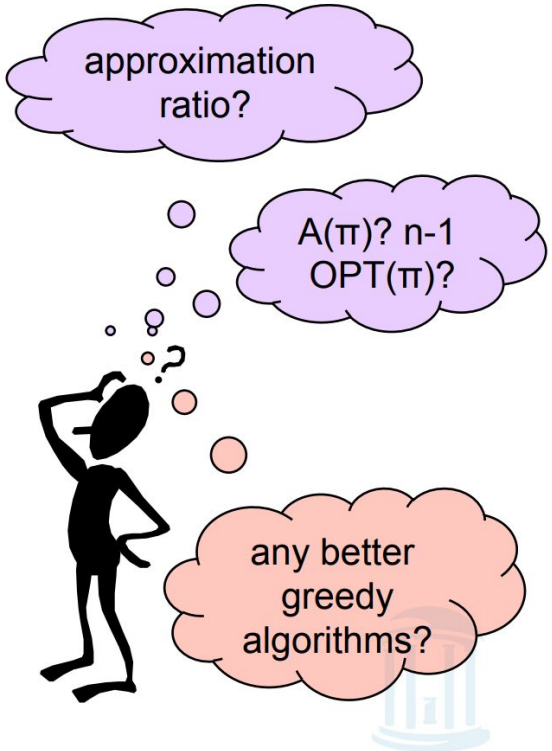
```
def GreedyReversalSort(pi):
    for i in range(len(pi)-1):
        j = pi.index(min(pi[i:]))
        if (j != i):
            pi = pi[:i]
               + [v for v in reversed(pi[i:j+1])]
               + pi[j+1:]
    return pi
```

A(Π)

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

OPT(Π)?

Step 0: 6 1 2 3 4 5
Step 1: 5 4 3 2 1 6
Step 2: 1 2 3 4 5 6



Next Time

Bounding optimal:

- Is there a limit to how few reversals are needed?
- Approximation ratio for greedy?
- Can we beat greedy?