Divide and Conquer

Divide-and-conquer:

- Break up problem into several parts.
- Solve each part recursively.
- Combine solutions to sub-problems into overall solution.

Most common usage:

- Break up problem of size n into two equal parts of size n/2.
- Solve two parts recursively.
- Combine two solutions into overall solution in linear time.
Mergesort

```c
mergesort(m, low, high) {
    if high == low {
        return m[low]
    }
    else if (high == low + 1) {
        return sort m[low] and m[high];
    }
    else {
        middle = (low + high) / 2
        left = mergesort(m, low, middle-1)
        right = mergesort(m, middle, high)
        return merge(left, right)
    }
}
```

Mergesort Recurrence Relation

- \( T(n) \) = number of comparisons to mergesort an input of size \( n \).

- \( T(n) \leq 2 \cdot T(n/2) + cn \) when \( n > 2 \)

- \( T(2) \leq c \)

- The time to do a merge sort is \( 2^q \) the time to sort subarrays of \( 1/2 \) the size each plus a linear time for the divide & merge

Problem: How do we solve this for \( \theta() \) value?

Generalized Recurrence Problem

- Instead of dividing the problem into 2 subproblems, divide it into \( q \) subproblems.
- Still have linear cost for the divide and merge steps combined.
- Consider case where \( q = 1 \)
Different merge cost

- Consider dividing the problem into 2 subproblems
- What happens if the merge cost is $O(n^2)$ instead of $O(n)$?

<table>
<thead>
<tr>
<th>Divide + Merge size</th>
<th>Subproblem size</th>
<th>Num subproblems</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>n/2</td>
<td>1</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>n/2</td>
<td>2</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>n/2</td>
<td>q</td>
<td>$O(n^{log q})$</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>n/2</td>
<td>2</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>

Recommender Systems

Netflix tries to match your movie preferences with others.
- You rank n movies.
- Netflix consults database to find people with similar tastes.
- Netflix can recommend to you movies that they liked.

Doing this well was worth $1,000,000 to Netflix!!
Counting Inversions

Similarity metric: number of inversions between two rankings.
- My rank: 1, 2, ..., n.
- Your rank: a₁, a₂, ..., aₙ.
- Movies i and j are inverted if i < j, but aᵢ > aⱼ.

Movies

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Me</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>You</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

Inversions

3-2, 4-2

What is the brute force algorithm?

Brute force: check all $\Theta(n^2)$ pairs i and j.