Lecture 23: QuickSort
QuickSort Step 1: Partition

1. Choose a pivot. √
2. Place that pivot in the right spot.
3. Pivot the rest of the array.

48 41 51 66 84 89 87 68 37 23 96 98 52 14 62 43
QuickSort

<table>
<thead>
<tr>
<th>48</th>
<th>41</th>
<th>51</th>
<th>66</th>
<th>84</th>
<th>89</th>
<th>87</th>
<th>68</th>
<th>37</th>
<th>23</th>
<th>96</th>
<th>98</th>
<th>52</th>
<th>14</th>
<th>62</th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>14</td>
<td>23</td>
<td>37</td>
<td>48</td>
<td>84</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>51</td>
<td>69</td>
<td></td>
</tr>
</tbody>
</table>
Running time of QuickSort

- What is the worst-case running time?
  \[ O(n^2) \]

- Can you produce the worst-case running time?
  \[ \text{Min/Max as pivot each time} \]

- So why is it called “QuickSort” then?
Selecting a pivot

Variations of QuickSort change based on pivot selection:

- Deterministic, single-spot.  
  **Adv:** easy to code
- Uniformly at random
- Median-of-3
Average Case Analysis of QuickSort

Suppose

- All permutations equally likely
- All $n$ values are distinct (for simplicity)
- Define $S_1, S_2, \ldots S_n$ as sorted order.

Let $P_{i,j}$ be probability we compare $S_i$ and $S_j$.

\[
P_{i,j} = \frac{\#\text{yes}}{\#\text{total}} = \frac{2}{j-i+1}
\]
Expected number of comparisons

\[ E\left( \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{i,j} \right) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E(X_{i,j}) \]

\[ = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} \]

\[ = \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{2}{k} \]

\[ < \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \]

is \( \Theta(n \log n) \)
Hybrid: Tim Sort

▶ The fastest sort algorithms are recursive
▶ Problem: recursion has an overhead

▶ Solution: