Average Case Analysis

Worst-case analysis.
- Analyze running time as function of worst input of a given size.

Average case analysis.
- Analyze average running time over some distribution of inputs.
  - Ex: quicksort.

Amortized analysis.
- Worst-case bound on sequence of operations.
  - Ex: splay trees, union-find.

Competitive analysis.
- Make quantitative statements about online algorithms.
  - Ex: paging, load balancing.

Average Case Analysis

- Analyze average running time over some distribution of inputs.
- Ex: quicksort.
  - O(N log N) if input is assumed to be in random order.
  - leads to randomized algorithm with O(N log N) expected running time, independent of input
- Major disadvantage: hard to quantify what input distributions will look like in practice.

Quicksort

- Assume all elements are unique.
- Assume input is a random permutation of inputs.
- Denote ith largest element by i.

```c
void quicksort(Item a[], int left, int right) {
    int m;
    if (right > left) {
        m = partition(a, left, right);
        quicksort(a, left, m - 1);
        quicksort(a, m + 1, right);
    }
}
```
QuickSort: BST Representation of Pivots

Probability that i = 2 and j = 7 get compared.

Let x be pivot element that separates i and j.

Case 1: x ∈ \{3, 4, 5, 6\} ⇒ i and j not compared.

Case 2: x ∈ \{2, 7\} ⇒ i and j are compared.

QuickSort: Average Case Analysis

\[
\Pr[i \text{ and } j \text{ compared } (i < j)] = \frac{2}{j - i + 1}
\]
Quicksort: Average Case Analysis

- Expected # of comparisons = \[ \sum_{1 \leq i < j \leq N} \frac{2}{j-i+1} \]

\[ = 2 \sum_{i=1}^{N} \sum_{j=2}^{i} \frac{1}{j} \]
\[ \leq 2N \sum_{j=1}^{N} \frac{1}{j} \]
\[ = 2N \ln N \]

Remark: same analysis if we choose partition element uniformly at random.

- Running time independent of input.

Comparison Based Sorting Lower Bound

Lower bound = \( \Omega(N \log_2 N) \): applies to comparison-based algorithms.

- Tree height \( h \) determines worst case running time.
- \( N! \) different input orderings.
- One (or more) leaves corresponds to each ordering.
- Binary tree with \( N! \) leaves has height:

\[ h \geq \log_2 (N!) \]
\[ \geq \log_2 (\frac{N}{e})^N \]
\[ = N \log_2 N - N \log_2 e \]
\[ = \Omega(N \log_2 N) \]

Decision Tree of Program