Data Structures and Algorithms
Probability in Computer Science

CS 225
G Carl Evans

April 14, 2023
Randomization in Algorithms

1. Assume input data is random to estimate average-case performance

2. Use randomness inside algorithm to estimate expected running time

3. Use randomness inside algorithm to approximate solution in fixed time
Average-Case Analysis: BST

Let $S(n)$ be the average total internal path length over all BSTs that can be constructed by uniform random insertion of $n$ objects.

**Claim:** $S(n)$ is $O(n \log n)$

N=0: 

N=1:
Average-Case Analysis: BST

Let $S(n)$ be the average total internal path length over all BSTs that can be constructed by uniform random insertion of $n$ objects.

N=3:
Average-Case Analysis: BST

Let \( S(n) \) be the average \textbf{total internal path length} over all BSTs that can be constructed by uniform random insertion of \( n \) objects. Let \( 0 \leq i \leq n - 1 \) be the number of nodes in the left subtree.

Then for a fixed \( i \), \( S(n) = (n - 1) + S(i) + S(n - i - 1) \)
Average-Case Analysis: BST

Let $S(n)$ be the \textbf{average} total internal path length over all BSTs that can be constructed by uniform random insertion of $n$ objects.

$$S(n) = (n - 1) + \frac{1}{n} \sum_{i=0}^{n-1} S(i) + S(n - i - 1)$$
Average-Case Analysis: BST

\[ S(n) = (n - 1) + \frac{2}{n} \sum_{i=1}^{n-1} S(i) \]

\[ S(n) = (n - 1) + \frac{2}{n} \sum_{i=1}^{n-1} (ci \ln i) \]

\[ S(n) \leq (n - 1) + \frac{2}{n} \int_{1}^{n} (cx \ln x) dx \]

\[ S(n) \leq (n - 1) + \frac{2}{n} \left( \frac{cn^2}{2} \ln n - \frac{cn^2}{4} + \frac{c}{4} \right) \approx cn \ln n \]
Average-Case Analysis: BST

**Summary**: All operations are on average $O(\log n)$

**Randomness**: 

**Assumptions**: 
Expectation Analysis: Randomized Quicksort
Expectation Analysis: Randomized Quicksort

```
6 1 0 3 7 9 2 4
1 0 3 2 4 9 6 7
1 0 3 2 4 9 6 7
1 0 2 3 4 6 7 9
1 0 2 3 4 6 7 9
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
...```
Expectation Analysis: Randomized Quicksort

In randomized quicksort, the selection of the pivot is random.

**Claim:** The expected comparisons is $O(n \log n \cdot n)$ for any input!

Let $X$ be the total comparisons and $X_{ij}$ be an indicator variable:

$$X_{ij} = \begin{cases} 
1 & \text{if } i\text{th object compared to } j\text{th} \\
0 & \text{if } i\text{th object not compared to } j\text{th}
\end{cases}$$

Then...
Key Ideas

1. Never compare $X_i$ with $X_i$

2. Never compare $X_i$ and $X_j$ more than once
Expectation Analysis: Randomized Quicksort

Claim: \( E[X_{ij}] = \frac{2}{j-i+1} \)

Base Case: (N=2)
Expectation Analysis: Randomized Quicksort

Claim: $E[X_{i,j}] = \frac{2}{j-i+1}$

Induction: Assume true for all inputs of $< n$
Expectation Analysis: Randomized Quicksort

\[
E[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}] \\
E[X_{ij}] = \frac{2}{j - i + 1}
\]
Expectation Analysis: Randomized Quicksort

\[ E[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{ij}] \quad E[X_{ij}] = \frac{2}{j - i + 1} \]

\[ E[X] = \sum_{i=1}^{n} 2\left(\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n - i + 1}\right) \]

\[ E[X] = \sum_{i=1}^{n} 2(H_{n-1} - 1) \leq 2n \cdot H_n \leq 2n \ln n \]
Expectation Analysis: Randomized Quicksort

**Summary:** Randomized quick sort is $O(n \log n)$ regardless of input.

**Randomness:**

**Assumptions:**
Probabilistic Accuracy: Fermat primality test

Pick a random \( a \) in the range \([2, p - 2]\)

If \( p \) is prime and \( a \) is not divisible by \( p \), then \( a^{p-1} \equiv 1 \pmod{p} \)

But… *sometimes* if \( n \) is composite and \( a^{n-1} \equiv 1 \pmod{n} \)
Probabilistic Accuracy: Fermat primality test

<table>
<thead>
<tr>
<th></th>
<th>$a^{p-1} \equiv 1 \pmod{p}$</th>
<th>$a^{p-1} \not\equiv 1 \pmod{p}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ is prime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p$ is not prime</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Probabilistic Accuracy: Fermat primality test

Let’s assume $\alpha = 0.5$

First trial: $a = a_0$ and prime test returns ‘prime!’

Second trial: $a = a_1$ and prime test returns ‘prime!’

Third trial: $a = a_2$ and prime test returns ‘not prime!’

Is our number prime?

What is our false positive probability? Our false negative probability?
Probabilistic Accuracy: Fermat primality test

**Summary:** Randomized algorithms can also have fixed (or bounded) runtimes at the cost of probabilistic accuracy.

**Randomness:**

**Assumptions:**
Types of randomized algorithms

A **Las Vegas** algorithm is a randomized algorithm which will always give correct answer if run enough times but has no fixed runtime.

A **Monte Carlo** algorithm is a randomized algorithm which will run a fixed number of iterations and may give the correct answer.
Next Class: Randomized Data Structures

Sometimes a data structure can be too ordered / too structured

Randomized data structures rely on expected performance

Randomized data structures ‘cheat’ tradeoffs!