Probability - quick review
Definitions

Definition (Informal)
Random variable: a function from probability space to \( \mathbb{R} \).
Associates value \( \forall \) atomic events in probability space.

Definition
The conditional probability of \( X \) given \( Y \) is
\[
Pr\left[ X = x \mid Y = y \right] = \frac{Pr\left[ (X = x) \cap (Y = y) \right]}{Pr\left[ Y = y \right]}.
\]
Equivalent to
\[
Pr\left[ (X = x) \cap (Y = y) \right] = Pr\left[ X = x \mid Y = y \right] \cdot Pr\left[ Y = y \right].
\]
Linearity of expectations

Lemma (Linearity of expectation.)
\[ \forall \text{ random variables } X \text{ and } Y : \mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]. \]

Proof.
Use definitions, do the math. See notes for details.

Probability - quick review

Conditional Expectation

Definition
\[ X, Y : \text{ random variables. The conditional expectation of } X \text{ given } Y \text{ (i.e., you know } Y = y): \]
\[ \mathbb{E}[X \mid Y] = \mathbb{E}[X \mid Y = y] = \sum_x x \times \Pr[X = x \mid Y = y]. \]

\[ \mathbb{E}[X] \text{ is a number.} \]
\[ f(y) = \mathbb{E}[X \mid Y = y] \text{ is a function.} \]

Conditional Expectation

Lemma
\[ \forall X, Y \text{ (not necessarily independent):} \]
\[ \mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]. \]
\[ \mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}_y \mathbb{E}[X \mid Y = y] \]

Proof.
Use definitions, and do the math. See class notes.

Sorting Nuts & Bolts

Problem (Sorting Nuts and Bolts)

1. Input: Set \( n \) nuts + \( n \) bolts.
2. Every nut has a matching bolt.
3. All different sizes.
4. Task: Match nuts to bolts.
5. Restriction: You can only compare a nut to a bolt.
6. Q: How to match the \( n \) nuts to the \( n \) bolts quickly?
Matching nuts and bolts...

**Algorithm**

1. Naive algorithm...
2. ...better algorithm?

```plaintext
MatchNutsAndBolts(N: nuts, B: bolts)
    Pick a random nut \( n_{\text{pivot}} \) from \( N \)
    Find its matching bolt \( b_{\text{pivot}} \) in \( B \)
    \( B_L \leftarrow \) All bolts in \( B \) smaller than \( n_{\text{pivot}} \)
    \( N_L \leftarrow \) All nuts in \( N \) smaller than \( b_{\text{pivot}} \)
    \( B_R \leftarrow \) All bolts in \( B \) larger than \( n_{\text{pivot}} \)
    \( N_R \leftarrow \) All nuts in \( N \) larger than \( b_{\text{pivot}} \)
    MatchNutsAndBolts(\( N_R \), \( B_R \))
    MatchNutsAndBolts(\( N_L \), \( B_L \))

QuickSort style...
```

### What is running time for randomized algorithms?

**Definitions**

- **Definition**
  \( \mathcal{R}(U) \): random variable – **running time** of the algorithm on input \( U \).

- **Definition**
  Expected running time \( \mathbb{E}[\mathcal{R}(U)] \) for input \( U \).

- **Definition**
  **expected running-time** of algorithm for input size \( n \):
  \[
  T(n) = \max_{U \text{ is an input of size } n} \mathbb{E}[\mathcal{R}(U)].
  \]

**Definition**

- **rank** \( x \): **rank** of element \( x \in S = \) number of elements in \( S \) smaller or equal to \( x \).
Nuts and bolts running time

Theorem
Expected running time \textit{MatchNutsAndBolts} (QuickSort) is
\[ T(n) = O(n \log n) \]. Worst case is \( O(n^2) \).

Proof.
\[ \Pr[\text{rank}(n_{\text{pivot}}) = k] = \frac{1}{n}. \]
Thus,
\[ T(n) = \mathbb{E}_{k=\text{rank}(n_{\text{pivot}})} \left[ O(n) + T(k - 1) + T(n - k) \right] \]
\[ = O(n) + \mathbb{E}_{k} [T(k - 1) + T(n - k)] \]
\[ = O(n) + \sum_{k=1}^{n} \Pr[\text{Rank(Pivot)} = k] \]
\[ \times (T(k - 1) + T(n - k)) \]
\[ = O(n) + \sum_{k=1}^{n} \frac{1}{n} \times (T(k - 1) + T(n - k)), \]
Solution is \( T(n) = O(n \log n) \).

Worst case vs. average case

Expected running time of a randomized algorithm is
\[ T(n) = \max_{U \text{ is an input of size } n} \mathbb{E}[RT(U)], \]
Worst case running time of deterministic algorithm:
\[ T(n) = \max_{U \text{ is an input of size } n} RT(U), \]

Alternative intuitive analysis...

Which is not formally correct

1. \textit{MatchNutsAndBolts} is \textit{lucky} if \( \frac{n}{4} \leq \text{rank}(n_{\text{pivot}}) \leq \frac{3}{4} n \).
2. \( \Pr[\text{"lucky"}] = 1/2 \).
3. \( T(n) \leq O(n) + \Pr[\text{"lucky"}] \times (T(n/4) + T(3n/4)) + \Pr[\text{"unlucky"}] \times T(n). \)
4. \( T(n) = O(n) + \frac{1}{2} \times (T(\frac{n}{4}) + T(\frac{3n}{4})) + \frac{1}{2} T(n). \)
5. Rewriting: \( T(n) = O(n) + T(n/4) + T((3/4)n). \)
6. ... solution is \( O(n \log n) \).

High Probability running time...

Definition
Running time \( \text{Alg} \) is \( O(f(n)) \) with \textit{high probability} if
\[ \Pr[\text{RT(Alg(n))} \geq c \cdot f(n)] = o(1). \]
\[ \implies \Pr[\text{RT(Alg)} > c \cdot f(n)] \to 0 \text{ as } n \to \infty. \]
Usually use weaker def:
\[ \Pr[\text{RT(Alg(n))} \geq c \cdot f(n)] \leq \frac{1}{n^d}, \]
Technical reasons... also assume that
\[ \mathbb{E}[\text{RT(Alg(n))}] = O(f(n)). \]
A Slick Analysis of \textbf{QuickSort}

Let $Q(A)$ be number of comparisons done on input array $A$:

1. For $1 \leq i < j < n$ let $R_{ij}$ be the event that rank $i$ element is compared with rank $j$ element.

2. $X_{ij}$: indicator random variable for $R_{ij}$. $X_{ij} = 1 \iff$ rank $i$ element compared with rank $j$ element, otherwise 0.

3. $Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$ and hence by linearity of expectation,

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}].$$

As such, probability of comparing 5 to 8 is $Pr[R_{4,7}]$.

1. If pivot too small (say 3 [rank 2]). Partition and call recursively:
   
   \[
   \begin{array}{cccccccc}
   7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
   6 & 4 & 8 & 1 & 2 & 3 & 7 & 5 \\
   \end{array}
   \]

   Decision if to compare 5 to 8 is moved to subproblem.

2. If pivot too large (say 9 [rank 8]):
   
   \[
   \begin{array}{cccccccc}
   7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
   7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
   \end{array}
   \]

   Decision if to compare 5 to 8 moved to subproblem.

Conclusion:

$R_{ij}$ happens if and only if:

- $i$th or $j$th ranked element is the first pivot out of $i$th to $j$th ranked elements.

How to analyze this?

Thinking acrobatics!

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as \textbf{QuickSort} do).
A Slick Analysis of QuickSort

Question: What is \( \Pr[R_{ij}] \)?

How to analyze this?

Thinking acrobatics!
1. Assign every element in the array a random priority (say in \([0, 1])
2. Choose pivot to be the element with lowest priority in subproblem.

\[ R_{ij} \text{ happens if either } i \text{ or } j \text{ have lowest priority out of elements rank } i \text{ to } j. \]

There are \( k = j - i + 1 \) relevant elements.

\[ \Pr[R_{ij}] = \frac{2}{k} = \frac{2}{j - i + 1}. \]

Lemma \( \Pr[R_{ij}] = \frac{2}{j - i + 1}. \)

Proof.
Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be elements of \( A \) in sorted order. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)

Observation: \( a_i \) is compared with \( a_j \) if and only if either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation...

\[ \sum_{1 \leq i < j \leq n} \Pr[R_{ij}] = \sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1} \]

\[ = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} \leq 2 \sum_{i=1}^{n} (H_n - H_{i+1} - 1) \leq 2 \sum_{1 \leq i < n} H_n \]

\[ \leq 2nH_n = O(n \log n) \]
Randomized Quick Selection

**Input**  Unsorted array $A$ of $n$ integers, an integer $j$.
**Goal**  Find the $j$th smallest number in $A$ (rank $j$ number)

Randomized Quick Selection

1. Pick a pivot element uniformly at random from the array.
2. Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
3. Return pivot if rank of pivot is $j$.
4. Otherwise recurse on one of the arrays depending on $j$ and their sizes.

Algorithm for Randomized Selection

**Assume** for simplicity that $A$ has distinct elements.

QuickSelect($A$, $j$):
Pick pivot $x$ uniformly at random from $A$
Partition $A$ into $A_{\text{less}}$, $x$, and $A_{\text{greater}}$ using $x$ as pivot
if $(|A_{\text{less}}| = j - 1)$ then
  return $x$
if $(|A_{\text{less}}| \geq j)$ then
  return QuickSelect($A_{\text{less}}$, $j$)
else
  return QuickSelect($A_{\text{greater}}$, $j - |A_{\text{less}}| - 1$)

QuickSelect analysis

1. $S_1, S_2, \ldots, S_k$ be the subproblems considered by the algorithm.
   Here $|S_1| = n$.
2. $S_i$ would be successful if $|S_i| \leq (3/4)|S_{i-1}|$
3. $Y_1 =$ number of recursive calls till first successful iteration.
   Clearly, total work till this happens is $O(Y_1 n)$.
4. $n_i =$ size of the subproblem immediately after the $(i - 1)$th successful iteration.
5. $Y_i =$ number of recursive calls after the $(i - 1)$th successful call, till the $i$th successful iteration.
6. Running time is $O(\sum_i n_i Y_i)$.

QuickSelect analysis

**Example**

$S_i =$ subarray used in $i$th recursive call
$|S_i| =$ size of this subarray
Red indicates successful iteration.

<table>
<thead>
<tr>
<th>Inst</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
<th>$S_8$</th>
<th>$S_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Succ’</td>
<td>$Y_1 = 2$</td>
<td>$Y_2 = 4$</td>
<td>$Y_3 = 2$</td>
<td>$Y_4 = 1$</td>
<td>$n_1 = n_1 = 100$</td>
<td>$n_2 = 60$</td>
<td>$n_3 = 25$</td>
<td>$n_4 = 2$</td>
<td></td>
</tr>
</tbody>
</table>

1. All the subproblems after $(i - 1)$th successful iteration till $i$th successful iteration have size $\leq n_i$.
2. Total work: $O(\sum_i n_i Y_i)$. 
QuickSelect analysis

Total work: $O(\sum_i n_i Y_i)$.

We have:

1. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1} n$.
2. $Y_i$ is a random variable with geometric distribution
   Probability of $Y_i = k$ is $1/2^i$.

As such, expected work is proportional to

$$ E\left[ \sum_i n_i Y_i \right] = \sum_i E[n_i Y_i] \leq \sum_i E[(3/4)^{i-1} n Y_i] $$

$$ = n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1} E[Y_i] \leq 8n. $$