Chapter 11

Randomized Algorithms
11.1 Randomized Algorithms

11.2 Some Probability

11.2.1 Probability - quick review

11.2.1.1 With pictures

(A) $\Omega$: Sample space

(B) $\Omega$: Is a set of elementary event/atomic event/simple event.

(C) Every atomic event $x \in \Omega$ has Probability $\Pr[x]$.

(D) $X \equiv f(x)$: Random variable associate a value with each atomic event $x \in \Omega$.

(E) $E[X]$: **Expectation:**

The average value of the random variable $X \equiv f(x)$.

$$E[X] = \sum_{x \in X} f(x) \ast \Pr[X = x].$$

(F) An event $A \subseteq \Omega$ is a collection of atomic events.

$$\Pr[A] = \sum_{a \in A} \Pr[a].$$

Complement event $A = \Omega \setminus A$. 

Ω
11.2.2 Probability - quick review

11.2.2.1 Definitions

Definition 11.2.1 (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

$$\Pr[X = x \mid Y = y] = \frac{\Pr[(X = x) \cap (Y = y)]}{\Pr[Y = y]}.$$

Equivalent to

$$\Pr[(X = x) \cap (Y = y)] = \Pr[X = x \mid Y = y] \cdot \Pr[Y = y].$$

11.2.3 Probability - quick review

11.2.3.1 Even more definitions

Definition 11.2.2. The events $X = x$ and $Y = y$ are independent, if

$$\Pr[X = x \cap Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

$$\equiv \Pr[X = x \mid Y = y] = \Pr[X = x].$$

Definition 11.2.3. The expectation of a random variable $X$ its average value:

$$E[X] = \sum_x x \cdot \Pr[X = x],$$

11.2.3.2 Linearity of expectations


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

11.2.4 Probability - quick review

11.2.4.1 Conditional Expectation

Definition 11.2.5. $X, Y$: random variables. The conditional expectation of $X$ given $Y$ (i.e., you know $Y = y$):

$$E[X \mid Y] = E[X \mid Y = y] = \sum_x x \cdot \Pr[X = x \mid Y = y].$$

$E[X]$ is a number.

$f(y) = E[X \mid Y = y]$ is a function.
Lemma 11.2.6. \( \forall X, Y \) (not necessarily independent): \( E[X] = E[E[X \mid Y]] \).

\[
E[E[X \mid Y]] = E_y [E[X \mid Y = y]]
\]

*Proof:* Use definitions, and do the math. See class notes. \( \blacksquare \)
Problem 11.3.1 (Sorting Nuts and Bolts). (A) 
Input: Set \( n \) nuts + \( n \) bolts.
(B) Every nut have a matching bolt.
(C) All different sizes.
(D) Task: Match nuts to bolts. (In sorted order).
(E) Restriction: You can only compare a nut to a bolt.
(F) Q: How to match the \( n \) nuts to the \( n \) bolts quickly?
11.3.1 Sorting nuts & bolts...

11.3.1.1 Algorithm

(A) Naive algorithm...
(B) ...better algorithm?

11.3.1.2 Sorting nuts & bolts...

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

QuickSort style...

11.3.2 Running time analysis

11.3.3 What is running time for randomized algorithms?

11.3.3.1 Definitions

Definition 11.3.2. \( \mathcal{RT}(U) \): random variable – running time of the algorithm on input \( U \).

Definition 11.3.3. Expected running time \( E[\mathcal{RT}(U)] \) for input \( U \).

Definition 11.3.4. expected running-time of algorithm for input size \( n \):

\[
T(n) = \max_{U \text{ is an input of size } n} E[\mathcal{RT}(U)].
\]

11.3.4 What is running time for randomized algorithms?

11.3.4.1 More definitions

Definition 11.3.5. rank(\( x \)): rank of element \( x \in S \) = number of elements in \( S \) smaller or equal to \( x \).

11.3.4.2 Nuts and bolts running time

Theorem 11.3.6. Expected running time \( \text{MatchNutsAndBolts} \) (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,

\[
T(n) = \mathbb{E}_{k=\text{rank}(n_{\text{pivot}})} \left[ O(n) + T(k-1) + T(n-k) \right]
\]

\[
= O(n) + \mathbb{E}_k [T(k-1) + T(n-k)]
\]

\[
= O(n) + \sum_{k=1}^{n} \Pr[\text{Rank(Pivot)} = k]
\]

\[
\times (T(k-1) + T(n-k))
\]

\[
= O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k)),
\]

Solution is \( T(n) = O(n \log n) \).

11.3.4.3 Alternative incorrect solution

11.3.5 Alternative intuitive analysis...

11.3.5.1 Which is not formally correct

(A) MatchNutsAndBolts is \textit{lucky} if \( \frac{n}{4} \leq \text{rank}(n_{\text{pivot}}) \leq \frac{3}{4}n \).

(B) \( \Pr[\text{“lucky”}] = 1/2 \).

(C) \( T(n) \leq O(n) + \Pr[\text{“lucky”}] \times (T(n/4) + T(3n/4)) + \Pr[\text{“unlucky”}] \times T(n) \).

(D) \( T(n) = O(n) + \frac{1}{2} \times (T(\frac{n}{4}) + T(\frac{3n}{4})) + \frac{1}{2}T(n) \).

(E) Rewriting: \( T(n) = O(n) + T(n/4) + T((3/4)n) \).

(F) ... solution is \( O(n \log n) \).

11.3.6 What are randomized algorithms?

11.3.6.1 Worst case vs. average case

Expected running time of a randomized algorithm is

\[
T(n) = \max_U \mathbb{E}_{U \text{ is an input of size } n} \mathbb{E} [RT(U)],
\]

Worst case running time of deterministic algorithm:

\[
T(n) = \max_U \mathbb{E}_{U \text{ is an input of size } n} RT(U),
\]

11.3.6.2 High Probability running time...

Definition 11.3.7. Running time \textbf{Alg} is \( O(f(n)) \) with \textit{high probability} if

\[
\Pr[RT(\text{Alg}(n)) \geq c \cdot f(n)] = o(1).
\]

\[\implies \Pr[RT(\text{Alg}) > c \cdot f(n)] \rightarrow 0 \text{ as } n \rightarrow \infty.\]

Usually use weaker def:

\[
\Pr[RT(\text{Alg}(n)) \geq c \cdot f(n)] \leq \frac{1}{n^d}.
\]

Technical reasons... also assume that \( \mathbb{E}[RT(\text{Alg}(n))] = O(f(n)) \).
11.4 Slick analysis of QuickSort

11.4.0.1 A Slick Analysis of QuickSort

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

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

(B) $X_{ij}$: indicator random variable for $R_{ij}$.

$X_{ij} = 1 \iff$ rank $i$ element compared with rank $j$ element, otherwise 0.

$$Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$$

and hence by linearity of expectation,

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

11.4.0.2 A Slick Analysis of QuickSort

$R_{ij}$ = rank $i$ element is compared with rank $j$ element.

**Question:** What is $\Pr[R_{ij}]$?

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

(A) If pivot too small (say 3 [rank 2]). Partition and call recursively:

Decision if to compare 5 to 8 moved to subproblem.

(B) If pivot too large (say 9 [rank 8]):

Decision if to compare 5 to 8 moved to subproblem.

11.4.1 A Slick Analysis of QuickSort

**Question:** What is $\Pr[R_{i,j}]$?

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

(A) If pivot is 5 (rank 4). Bingo!

(B) If pivot is 8 (rank 7). Bingo!

(C) If pivot in between the two numbers (say 6 [rank 5]):

5 and 8 will never be compared to each other.
11.4.2 A Slick Analysis of QuickSort

11.4.2.1 Question: What is Pr[R_{i,j}]?

Conclusion:

\[ R_{i,j} \text{ happens if and only if:} \]

\[ \begin{align*}
& \text{i} \text{th or} \text{j} \text{th ranked element is the first pivot out of} \\
& \text{i} \text{th to} \text{j} \text{th ranked elements.}
\end{align*} \]

How to analyze this?

Thinking acrobatics!

(A) Assign every element in the array a random priority (say in \([0, 1]\)).

(B) Choose pivot to be the element with lowest priority in subproblem.

(C) Equivalent to picking pivot uniformly at random

\( \text{(as QuickSort do).} \)

11.4.3 A Slick Analysis of QuickSort

11.4.3.1 Question: What is Pr[R_{i,j}]?

How to analyze this?

Thinking acrobatics!

(A) Assign every element in the array a random priority (say in \([0, 1]\)).

(B) Choose pivot to be the element with lowest priority in subproblem.

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

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

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

11.4.3.2 A Slick Analysis of QuickSort

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

Lemma 11.4.1. \( \Pr[R_{ij}] = \frac{2}{j - i + 1}. \)

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: If pivot is chosen outside \( S \) then all of \( S \) either in left array or right array.

Observation: \( a_i \) and \( a_j \) separated when a pivot is chosen from \( S \) for the first time. Once separated no comparison.

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...
11.4.4 A Slick Analysis of QuickSort

11.4.4.1 Continued...

Lemma 11.4.2. \( \Pr[R_{ij}] = \frac{2}{j-i+1} \).

Proof: Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \). 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.

Observation: Given that pivot is chosen from \( S \) the probability that it is \( a_i \) or \( a_j \) is exactly \( \frac{2}{|S|} = \frac{2}{(j-i+1)} \) since the pivot is chosen uniformly at random from the array.

11.4.5 A Slick Analysis of QuickSort

11.4.5.1 Continued...

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

Lemma 11.4.3. \( \Pr[R_{ij}] = \frac{2}{j-i+1} \).

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

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

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

\[ \leq 2 \sum_{1 \leq i < n} H_n \]

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

11.5 Quick Select

11.6 Randomized Selection

11.6.0.1 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

(A) Pick a pivot element \textit{uniformly at random} from the array.
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Return pivot if rank of pivot is \( j \).
(D) Otherwise recurse on one of the arrays depending on \( j \) and their sizes.
11.6.0.2 Algorithm for Randomized Selection

Assume for simplicity that \( A \) has distinct elements.

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

11.6.0.3 QuickSelect analysis

(A) \( S_1, S_2, \ldots, S_k \) be the subproblems considered by the algorithm.
   Here \( |S_1| = n \).
(B) \( S_i \) would be \textit{successful} if \( |S_i| \leq (3/4)|S_{i-1}| \)
(C) \( Y_1 = \) number of recursive calls till first successful iteration.
   Clearly, total work till this happens is \( O(Y_1n) \).
(D) \( n_i = \) size of the subproblem immediately after the \( (i-1) \)th successful iteration.
(E) \( Y_i = \) number of recursive calls after the \( (i-1) \)th successful call, till the \( i \)th successful iteration.
(F) Running time is \( O(\sum_i n_i Y_i) \).

11.6.0.4 QuickSelect analysis

Example

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

\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline
Inst & S_1 & S_2 & S_3 & S_4 & S_5 & S_6 & S_7 & S_8 & S_9 \\
\hline
|S_1| & 100 & 70 & 60 & 50 & 40 & 30 & 25 & 5 & 2 \\
Succ & Y_1 = 2 & Y_2 = 4 & Y_3 = 2 & Y_4 = 1 \\
n_i & n_1 = 100 & n_2 = 60 & n_3 = 25 & n_4 = 2 \\
\hline
\end{tabular}

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

11.6.0.5 QuickSelect analysis

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

We have:
(A) \( n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n \).
(B) \( Y_i \) is a random variable with geometric distribution
   Probability of \( Y_i = k \) is \( 1/2^i \).
(C) \( E[Y_i] = 2 \).
   As such, expected work is proportional to

\[
E\left[\sum_i n_i Y_i\right] = \sum_i E[n_i Y_i] \leq \sum_i E\left[(3/4)^{i-1}n Y_i\right] \\
= n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1}^\infty (3/4)^{i-1} 2 \leq 8n.
\]
11.6.0.6 QuickSelect analysis

Theorem 11.6.1. The expected running time of QuickSelect is $O(n)$. 