Presentation for use with the textbook, Algorithm Design and Applications, by M. T. Goodrich and R. Tamassia, Wiley, 2015

# Randomized Algorithms

Trees with snow on branches, "Half Dome, Apple Orchard, Yosemite," 1933. Ansel Adams. U.S. government image. U.S. National Archives and Records Administration.

# Applications: Simple Algorithms and Card Games

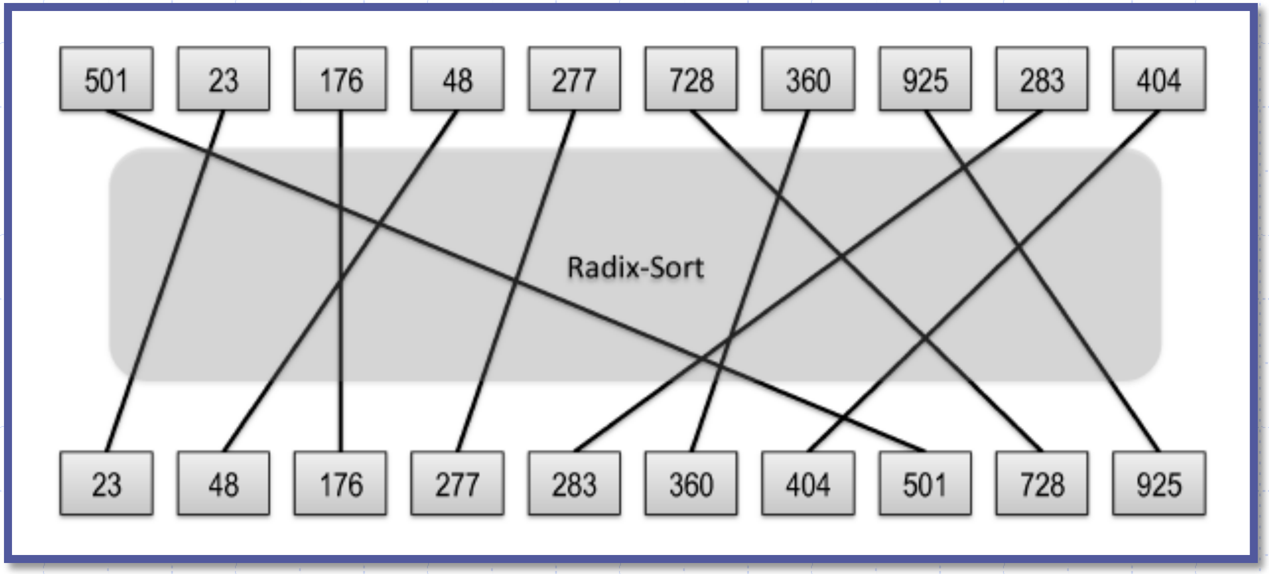
- A **randomized algorithm** is an algorithm whose behavior depends, in part, on the outcomes of random choices or the values of random bits.
- The main advantage of using randomization in algorithm design is that the results are often simple and efficient.
- In addition, there are some problems that need randomization for them to work effectively.
- For instance, consider the problem common in computer games involving playing cards—that of randomly shuffling a deck of cards so that all possible orderings are equally likely.

# Generating Random Permutations

- The input to the random permutation problem is a list, $X = (x_1, x_2, \ldots, x_n)$, of n elements, which could stand for playing cards or any other objects we want to randomly permute.
- The output is a reordering of the elements of X, done in a way so that all permutations of X are equally likely.
- We can use a function, **random**(k), which returns an integer in the range $[0, k - 1]$ chosen uniformly and independently at random.

# Algorithm 1: Random Sort

- This algorithm simply chooses a random number for each element in X and sorts the elements using these values as keys.



# Basic Probability (Sec. 1.2.4)

- In order to analyze this, and other randomized algorithms, we need to use probability.
- A **probability space** is a sample space S together with a probability function, Pr, that maps subsets of S to real numbers between 0 and 1, inclusive.
- Formally, each subset A of S is an event, and we have the following:

1. $\Pr(\emptyset) = 0$.
2. $\Pr(S) = 1$.
3. $0 \leq \Pr(A) \leq 1$, for any $A \subseteq S$.
4. If $A, B \subseteq S$ and $A \cap B = \emptyset$, then $\Pr(A \cup B) = \Pr(A) + \Pr(B)$.

# Independence and Conditional Probability

Two events $A$ and $B$ are ***independent*** if

$$\Pr(A \cap B) = \Pr(A) \cdot \Pr(B).$$

A collection of events $\{A_1, A_2, \ldots, A_n\}$ is ***mutually independent*** if

$$\Pr(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}) = \Pr(A_{i_1}) \Pr(A_{i_2}) \cdots \Pr(A_{i_k}),$$

for any subset $\{A_{i_1}, A_{i_2}, \ldots, A_{i_k}\}$.

The ***conditional probability*** that an event $A$ occurs, given an event $B$, is denoted as $\Pr(A|B)$, and is defined as

$$\Pr(A|B) = \frac{\Pr(A \cap B)}{\Pr(B)},$$

assuming that $\Pr(B) > 0$.

# Random Variables

- A **random variable** is a function X that maps outcomes from some sample space S to real numbers.
- An **indicator random variable** is a random variable that maps outcomes to the set {0, 1}.
- The **expected value** of a discrete random variable X is defined as

$$E(X) = \sum_x x \Pr(X = x),$$

  where the sum is taken of the range of X.
- Two random variables X and Y are **independent** if

$$\Pr(X = x | Y = y) = \Pr(X = x),$$

  for all real numbers x and y.
- If two random variables X and Y are independent, then we have E(XY) = E(X)E(Y).

# Linearity of Expectation

**Theorem 1.25 (The Linearity of Expectation):** *Let $X$ and $Y$ be two arbitrary random variables. Then $E(X+Y) = E(X) + E(Y)$.*

**Proof:**

$$\begin{aligned}
E(X+Y) &= \sum_x \sum_y (x+y) \Pr(X = x \cap Y = y) \\
&= \sum_x \sum_y x \Pr(X = x \cap Y = y) + \sum_x \sum_y y \Pr(X = x \cap Y = y) \\
&= \sum_x \sum_y x \Pr(X = x \cap Y = y) + \sum_y \sum_x y \Pr(Y = y \cap X = x) \\
&= \sum_x x \Pr(X = x) + \sum_y y \Pr(Y = y) \\
&= E(X) + E(Y).
\end{aligned}$$

# Chernoff Bounds

It is often necessary in the analysis of randomized algorithms to bound the sum of a set of random variables. One set of inequalities that makes this tractable is the set of Chernoff Bounds. Let $X_1, X_2, \ldots, X_n$ be a set of mutually independent indicator random variables, such that each $X_i$ is 1 with some probability $p_i > 0$ and 0 otherwise. Let $X = \sum_{i=1}^{n} X_i$ be the sum of these random variables, and let $\mu$ denote the mean of $X$, that is, $\mu = E(X) = \sum_{i=1}^{n} p_i$. We prove the following later in this book (Section 19.5).

**Theorem 1.29:** *Let $X$ be as above. Then, for $\delta > 0$,*

$$\Pr(X > (1+\delta)\mu) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu},$$

*and, for $0 < \delta \leq 1$,*

$$\Pr(X < (1-\delta)\mu) < e^{-\mu\delta^2/2}.$$

# Analysis of Random-Sort

- To see that every permutation is equally likely to be output by the random-sort method, note that each element, $x_i$, in X has an equal probability, 1/n, of having its random $r_i$ value be the smallest.
- Thus, each element in X has equal probability of 1/n of being the first element in the permutation.
- Applying this reasoning recursively, implies that the permutation that is output has the following probability of being chosen:

$$\left(\frac{1}{n}\right) \cdot \left(\frac{1}{n-1}\right) \cdots \left(\frac{1}{2}\right) \cdot \left(\frac{1}{1}\right) = \frac{1}{n!}$$

- That is, each permutation is equally likely to be output.
- There is a small probability that this algorithm will fail, however, if the random values are not unique.

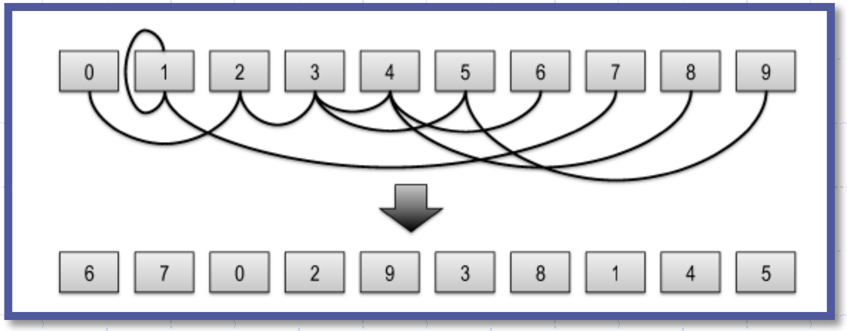
# Fisher-Yates Shuffling

- There is a different algorithm, known as the Fisher-Yates algorithm, which always succeeds.

**Algorithm** FisherYates($X$):

*Input:* An array, $X$, of $n$ elements, indexed from position 0 to $n-1$
*Output:* A permutation of $X$ so that all permutations are equally likely

**for** $k = n-1$ **downto** 1 **do**
  Let $j \leftarrow$ random$(k+1)$  // $j$ is a random integer in $[0, k]$
  Swap $X[k]$ and $X[j]$  // This may "swap" $X[k]$ with itself, if $j = k$
**return** $X$



# Analysis of Fisher-Yates

- This algorithm considers the items in the array one at time from the end and swaps each element with an element in the array from that point to the beginning.
- Notice that each element has an equal probability, of 1/n, of being chosen as the last element in the array X (including the element that starts out in that position).
- Applying this analysis recursively, we see that the output permutation has probability

$$\left(\frac{1}{n}\right) \cdot \left(\frac{1}{n-1}\right) \cdots \left(\frac{1}{2}\right) \cdot \left(\frac{1}{1}\right) = \frac{1}{n!}$$

- That is, each permutation is equally likely.