• Homework 8 due Friday **at 5pm**
• Final Exam:
  • Thursday, May 14 from 4pm to 5:50pm in WWH 101
  • Must bring a Laptop! (Email me **asap** if you do not have one!)
Exercise: You are given an n-sided die where side $i$ has probability $p_i$ of being rolled. What is the most efficient data structure for simulating rolls of the die?
Suppose we want to randomly one of four elements with probabilities: $\frac{1}{2}, \frac{1}{3}, \frac{1}{12}, \frac{1}{12}$

First: consider the average probability $\frac{1}{4}$
  - Some of our probabilities will be less, others will be equal or more

Imagine four buckets, each corresponding to an element with capacity $\frac{1}{4}$
  - Goal: fill each bucket such that it represents no more than two elements
• Start filling the buckets corresponding to the smallest probabilities

\[
\begin{array}{cccc}
1/2 & 1/3 & 1/12 & 1/12 \\
\end{array}
\]

• Fill the remainder of the partially filled bucket by taking away from the larger probabilities (arbitrary)

\[
\begin{array}{cccc}
1/2 & 1/6 & 1/12 & 2/3 \\
1/3 & &
\end{array}
\]
• Repeat the previous step with the next largest probability
• How does this help us sample more efficiently?
How do you simulate a biased coin?
- Heads with probability p, tails with probability 1-p

Problem is now reduced to randomly picking a bucket (uniformly) then flipping a biased coin

General approach is called **alias method**
- Specific implementations, e.g., Vose’s algorithm
- Running time: $O(n \log n)$ initialization, $O(1)$ generation
1. Create arrays Alias and Prob, each of size $n$
2. Create a balanced binary search tree $T$
3. Insert $n \cdot p_i$ into $T$ for each probability $i$
4. For $j=1$ to $n-1$:
   1. Find and remove the smallest value in $T$; call it $p_l$
   2. Find and remove the largest value in $T$; call it $p_g$
   3. Set $\text{Prob}[l]=p_l$
   4. Set $\text{Alias}[l]=g$
   5. Set $p_g:=p_g-(1-p_l)$
   6. Add $p_g$ to $T$
5. Let $i$ be the last probability remaining, which must have weight 1
6. Set $\text{Prob}[i]=1$
Exercise: There are 100 shoelaces in a box. At each stage, you pick two random ends and tie them together. Either this results in a longer shoelace (if the two ends came from different pieces), or it results in a loop (if the two ends came from the same piece). What are the expected number of steps until everything is in loops, and the expected number of loops after everything is in loops?
• Indicator random variables: 
  \[ I = \begin{cases} 
  0 & \text{with probability } 1 - p \\
  1 & \text{with probability } p 
\end{cases} \]

• Initially 200 free ends
• The number of ends decrease by 2 each time, so exactly 100 steps
  • Either by creating a loop, or joining 2 laces
• At each step, create an indicator R.V. representing if a loop is created
• Probability of forming a loop at each step with n strings is?
• Use linearity of expected values to find solution
**Exercise**: A group of 50 people are comparing their birthdays. What is the expected number of pairs who share a birthday? (Assume no leap years)

**Exercise**: I have a sack with 12 quarters: 4 are normal, 4 are double-headed, 4 are double-tailed. I randomly pick a quarter and randomly show you a side. If you see a head, what is the probability that I also see a head?

**Exercise**: What is the expected number of 6-sided dice rolls before you get two 4’s in a row?
• A finite **Markov Chain** is essentially a finite state machine where edges are labeled with probabilities
• Conditional on being on a given state, the edges give the probability that you will be on each of the adjacent nodes on the next step
• Mathematically:
  • A sequence of random variables $X_0, X_1, \ldots$ is called a Markov chain if:

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, x_0 = i_0) = P(X_{n+1} = j \mid X_n = i)$$

• Intuitively:
  • The only previous information that matters is the most recent step
  • All other past history provides no further information
• Graph representation:
  • Note that all outgoing edges have probabilities that sum to 1

• Matrix representation:
  • Note that each row sums to 1
Exercise: You have 60 health and your opponent has 80 health. Each turn you roll a 10-sided die, and he rolls a 6-sided die to determine damage inflicted (i.e., how much to decrease your opponent’s health by). The battle ends when at least one player has lost all health (if your health goes negative, it just becomes 0). Give an algorithm to compute the probability you win with at least 5 health remaining. Show how to compute your expected remaining health.
• **Expectation maximization**: determine the expected value of the optimal move

• **Minimax**: assume two players taking turns
  - Compute using recursion/memoization on game state assuming each player makes a move which maximizes their score / chances of winning

• **Working backwards**: solve a game for a simple end case and work backwards using induction

• **Nim games**: class of games with two players taking turns removing objects from distinct heaps
  - Player who can no longer take anything loses
Exercise: Two players take turns placing identical coins on a round table. The coins cannot overlap. A player loses if they can’t make a move. Who can guarantee to win?
Competitive Programming Chapter 5

Alias Method: http://www.keithschwarz.com/darts-dice-coins/

www.projecteuler.net