Lecture 5: Randomized Algorithms

The Hiring Problem

```
Hire-Assistant(n):
best ← 0
for i ← 1 to n
    interview candidate i
    if candidate i is better than best then
        fire best
        hire candidate i
        best ← i
```

Problem: What's the number of hires?

Worst case: n, which happens when you interview the candidates in the increasing order of quality.

Q: How to avoid the worst case?

A: Interview the candidates in a random order!

A quick review of probability theory

Expectation. Given a discrete random variables X, its expectation $E[X]$ is defined as:

$$E[X] = \sum i \cdot \Pr[X = i]$$

Q: Roll a 6-sided dice. What is the expected value?

A: $E[X] = \sum_{i=1}^{6} i \cdot \frac{1}{6} = 3.5$

Q: Roll two dice. What is the expected maximum value?

A: $E[X] = 1 \cdot \frac{1}{36} + 2 \cdot \frac{3}{36} + 3 \cdot \frac{5}{36} + 4 \cdot \frac{7}{36} + 5 \cdot \frac{9}{36} + 6 \cdot \frac{11}{36} = 4.47$

Q (waiting time for the first success): Coin is heads with probability p and tails with probability $1 - p$. How many flips X until first heads?

A: $$E[X] = \sum_{j=1}^{\infty} j \cdot \Pr[X = j] = \sum_{j=1}^{\infty} j \underbrace{(1-p)^{j-1}}_{\text{j-1 tails}} \underbrace{p}_{\text{1 head}} = \frac{p}{1-p} \sum_{j=1}^{\infty} j(1-p)^{j} = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

Expectation: Two Properties

Indicator random variables. If X only takes 0 or 1, $E[X] = \Pr[X = 1]$.

Linearity of expectation. Given two random variables X and Y (not necessarily independent),

$$E[X+Y] = E[X] + E[Y].$$

Remark: $E[XY] = E[X]E[Y]$ only when X and Y are independent.

Example. Shuffle a deck of n cards; turn them over one at a time; try to guess each card. Suppose you can't remember what's been turned over already, and just guess a card from full deck uniformly at random.

Q. What's the expected number of correct guesses?

A. (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if i^{th} guess is correct and 0 otherwise.
- Let X = number of correct guesses $= X_1 + \cdots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + \cdots + E[X_n] = 1/n + \cdots + 1/n = 1$.

Guessing Cards with Memory

Guessing with memory. Guess a card uniformly at random from cards not yet seen.

Q. What's the expected number of correct guesses?

A.

- Let $X_i = 1$ if i^{th} guess is correct and 0 otherwise.
- Let X = number of correct guesses $= X_1 + \cdots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/(n - i + 1)$.
- $E[X] = E[X_1] + \cdots + E[X_n] = \frac{1}{n} + \cdots + \frac{1}{2} + \frac{1}{1} = \Theta(\log n)$.

The Hiring Problem: Analysis

```
Hire-Assistant(n):
randomly permute all n candiates
best ← 0
for i ← 1 to n
     interview candidate i
     if candidate i is better than best then
          fire best
          hire candidate i
          best ← i
```

Q: What's the expected number of hires?

A:

- Let $X_i = 1$ if you hire candidate i and 0 otherwise.
- Let X = number of hires $= X_1 + \cdots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/i$. (Among the first i candidates, the best has probability $1/i$ to be placed at the last position.)
- $E[X] = E[X_1] + \cdots + E[X_n] = 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} + \frac{1}{n} = \Theta(\log n)$.

How to Generate a Random Permutation

RandomPermute(A):
$n \leftarrow A.length$
for $i \leftarrow 1$ **to** n
    swap $A[i]$ **with** $A[Random(1, i)]$

Note: This algorithm is slightly different from the one in textbook

Generates a random number between 1 and i uniformly.

Analysis

- $O(n)$ time, $O(1)$ working space

Correctness:

- Precise meaning of a "random permutation": Each different permutation is the output with probability $1/n!$
- We will show that after the i-th iteration, $A[1..i]$ has been randomly permuted, by induction on i.
  - Base case $i = 1$: trivial
  - Assume the $A[1..i-1]$ has been randomly permuted after $i-1$ iterations of the algorithm.
  - Consider any permutation $(a_1, \ldots, a_i)$ for $A[1..i]$. What's the probability that $A[1..i] = (a_1, \ldots, a_i)$ after the i-th iteration?

Random Permutation: Correctness

Proof of correctness (continued):

- Label the initial elements as $1, 2, \dots, i$.
- If after the i-th step, $A[1..i] = (a_1, a_2, \dots, a_i)$, then the i-th step must be like this ($i = 10$ in the example):

2	8	7	3	4	1	5	6	9	10
		a_i							i

$$\Downarrow$$

2	8	10	3	4	1	5	6	9	7

- So after the $(i-1)$-the step, $A[1..i-1]$ =(2,8,7,3,4,1,5,6,9), which happens with probability $1/(i-1)!$ (by induction hypothesis)
- The last swap must choose a_i to swap, which happens with probability $1/i$
- Thus, after the i-th step, $\Pr[A[1..i] = (a_1, \dots, a_i)] = \frac{1}{(i-1)!} \cdot \frac{1}{i} = \frac{1}{i!}$

How Humans Do Shuffling

Riffle shuffle

Analysis:

- $\frac{3}{2}\log n$ riffle shuffles can shuffle a deck of n cards to produce a distribution that is close to uniform [Bayer & Diaconis, 1992].
- For $n = 52$, 8 shuffles are good, 7 also OK.

The Birthday Paradox

Problem: Suppose there are $n = 365$ days in a year, and in a room of k people, each person's birthday falls in any one of the n days with equal probability. How large should k be for us to expect two people with the same birthday?

Analysis:

- Define $X_{ij} = 1$ if person i and person j have the same birthday, and 0 otherwise.
- We know $E[X_{ij}] = \Pr[X_{ij} = 1] = 1/n$.
- Let $X = \sum_{1 \le i < j \le k} X_{ij}$ be the number of pairs of people having the same birthday.
- We have
$$E[X] = E\left[\sum_{1 \le i < j \le k} X_{ij}\right] = \binom{k}{2}\frac{1}{n} = \frac{k(k-1)}{2n}$$
- So, when $\frac{k(k-1)}{2n} \ge \frac{(k-1)^2}{2n} \ge 1$, or $k \ge \sqrt{2n} + 1 \approx 28$, we expect to see at least one pair of people having the same birthday.

Coupon Collector

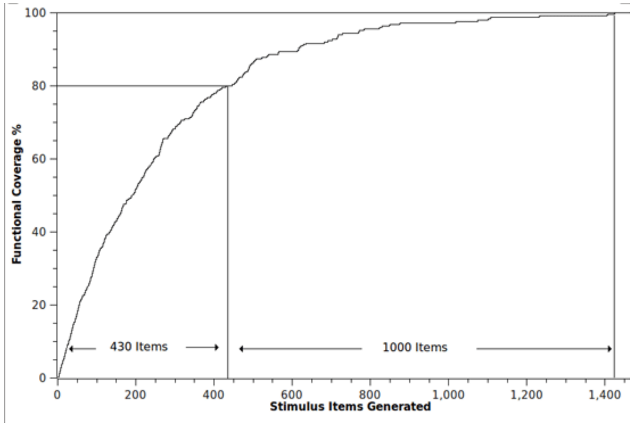
Coupon collector. Each box of cereal contains a coupon. There are n different types of coupons. Assuming a box contains each type of coupon equally likely, how many boxes do you need to open to have at least one coupon of each type?

Solution.

- Stage i = time between i and $i+1$ distinct coupons.
- Let X_i = number of steps you spend in stage i.
- Let X = number of steps in total = $X_0 + X_1 + \cdots + X_{n-1}$.

$$E[X] = \sum_{i=0}^{n-1} E[X_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=1}^{n} \frac{1}{i} = \Theta(n \log n)$$

prob of success = $(n-i)/n$
$\Rightarrow$ expected waiting time = $n/(n-i)$



Epilogue: How does a computer generate a random number?

Pseudorandom numbers:

- Computed by a deterministic algorithm from a "seed".
- If the "seed" is unknown, then it's difficult to predict the next number to be generated.
  - Often use current machine time as the seed.
- Higher difficulty needs more complicated algorithms.
  - `rand`: "linear generator" $x_n = (214013x_{n-1}+2531011)\ mod\ 2^{32}$
  - `ranlux48`
  - `knuth_b`
  - http://en.cppreference.com/w/cpp/numeric/random

True random numbers:

- Electronic noise, thermal noise, atmospheric noise, etc.
- Expensive and slow
- http://www.random.org