Lecture 5: Randomized Algorithms

Version of February 20, 2019

Outline

1. A Quick Review of Probability

2. The Hiring Problem

3. Generating a Random Permutation

4. Various Other Items
   - Shuffling Cards
   - The Birthday Paradox
   - Coupon Collectors
   - Generating Random Numbers

A quick review of probability theory

Expectation. The *expectation*, $E[X]$, of a discrete random variable 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 TOTAL value?

A: $E[X] = 2 \cdot \frac{1}{36} + 3 \cdot \frac{2}{36} + 4 \cdot \frac{3}{36} + 5 \cdot \frac{4}{36} + 6 \cdot \frac{5}{36} + 7 \cdot \frac{6}{36} + 8 \cdot \frac{5}{36} + 9 \cdot \frac{4}{36} + 10 \cdot \frac{3}{36} + 11 \cdot \frac{2}{36} + 12 \cdot \frac{1}{36} = 7$

Q: Roll two dice. What is the expected MAXIMUM value seen on a die?

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

A quick review of probability theory [II]

Expectation. The *expectation*, $E[X]$, of a discrete random variable X, is defined as:

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

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

A:

$$E[X] = \sum_{j=1}^{\infty} j \cdot Pr[X = j] = \sum_{j=1}^{\infty} j \cdot (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=1}^{\infty} j \cdot (1-p)^{j} = \frac{p}{1-p} \cdot \frac{1-p}{p^2} = \frac{1}{p}$$

j-1 tails 1 head

Expectation: Two Properties

(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: Roll two dice. What is the expected TOTAL value X?

Already saw that can calculate

$$E[X] = 2\cdot\frac{1}{36} + 3\cdot\frac{2}{36} + 4\cdot\frac{3}{36} + 5\cdot\frac{4}{36} + 6\cdot\frac{5}{36} + 7\cdot\frac{6}{36} + 8\cdot\frac{5}{36} + 9\cdot\frac{4}{36} + 10\cdot\frac{3}{36} + 11\cdot\frac{2}{36} + 12\cdot\frac{1}{36} = 7$$

Easier way is to let X_1, X_2 be the random variables that are values of first and second die. Then $E[X_1] = 3.5, E[X_2] = 3.5$ and by *Linearity of Expectation*

$$E[X_1+X_2] = E[X_1] + E[X_2] = 3.5 + 3.5 = 7.$$

Expectation: Two Properties [II]

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

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

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

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

Q. What is 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. Same setup as previous problem but now you guess value of the card uniformly at random from the set of cards not yet seen. *(At i^{th} step, n-i+1 cards have not yet been 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 $= X_1 + \cdots + X_n$.

- $E[X_i] = \Pr[X_i = 1] = 1/(n - i + 1)$.

- $E[X] = E[X_1] + \cdots + E[X_n] = \frac{1}{n} + \cdots + \frac{1}{2} + \frac{1}{1} = \Theta(\log n)$.

Will now use this to analyze a simple RANDOMIZED algorithm

Outline

1. A Quick Review of Probability

2. The Hiring Problem

3. Generating a Random Permutation

4. Various Other Items
   - Shuffling Cards
   - The Birthday Paradox
   - Coupon Collectors
   - Generating Random Numbers

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: How many people are hired?

(Hiring too many is bad. Costs money to fire old hires.)

Worst case: n;

occurs when candidates interviewed in increasing order of quality.

Q: How to avoid the worst case?

A: Interview the candidates in a random order!

(Mix up the files before starting the interviews.)

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 is the expected number of hires?

A: Similar to *Guessing Cards with Memory*

- Let $X_i = 1$ if you hire candidate i and 0 otherwise.
- Set $X =$ 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.)

- $E[X] = E[X_1] + \cdots + E[X_n] = 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} + \frac{1}{n} = \Theta(\log n)$.

Outline

1. A Quick Review of Probability

2. The Hiring Problem

3. Generating a Random Permutation

4. Various Other Items
   - Shuffling Cards
   - The Birthday Paradox
   - Coupon Collectors
   - Generating Random Numbers

Generating a Random Permutation

- Our solution to Hiring problem required randomly ordering the interview order of the applicants
- Mathematically, we wanted to find a random permutation (ordering of the applicants. How can we do this?
- There are $n!$ different permutations of n items. An algorithm that generates a random permutation would generate each one with probability $1/n!$
- Computers normally only allow you to choose a random integer in a range. How can we use a procedure that generates a random integer to generate a random permutation?
- On next page we assume that our computer has a procedure *Random(i,j)* that generates a random uniform integer between i and j.
  *(**uniform** means that each integer has same probability of occurring)*

How to Generate a Random Permutation

```
RandomPermute(A):
n ← A.length
for i ← 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:

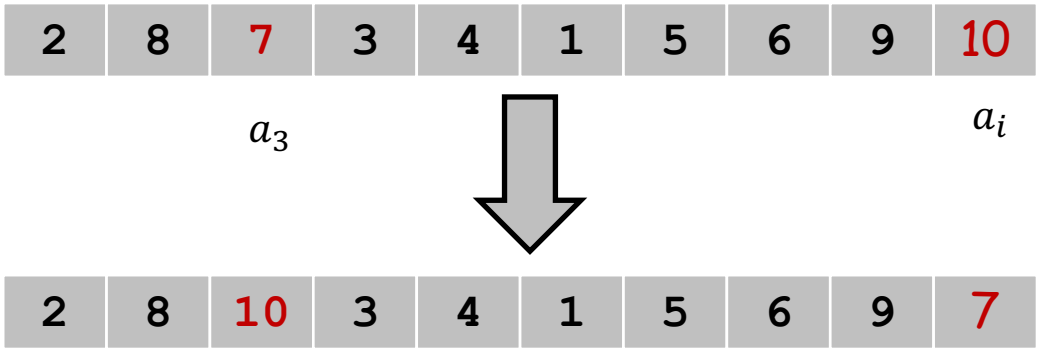
- Precise meaning of a "random permutation":
  Each different permutation is output with probability $1/n!$

- We will show by induction on i that,
  after the i-th iteration, $A[1..i]$ has been randomly permuted,
  - Base case $i = 1$: trivial
  - Assume $A[1..i-1]$ 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 $1, 2, \ldots, i$.
- Suppose after $(i-1)$-st step $A[1..i-1] = (2,8,7,3,4,1,5,6,9)$
  and after i-th step $A[1..i] = (2,8,10,3,4,1,5,6,9,7)$
- What was swapped in i-th step?



After the i-th step,

$A[1..i] =(2,8,10,3,4,1,5,6,9,7)$

if and only if

$Random(1,i)$ returned 3 (because 10 was swapped with 7)

$A[1..i-1] =(2,8,7,3,4,1,5,6,9)$

Random Permutation: Correctness

- We just saw that, after the i-th step, (i=10),
  $A[1..i]$ =(2,8,10,3,4,1,5,6,9,7) if and only if both
  - (a) $Random(1,i)$ returned 3
  - (b) $A[1..i-1]$ =(2,8,7,3,4,1,5,6,9)

- But
  - (a) occurs with probability $1/i$
  - (b) by induction, occurs with probability $1/(i-1)!$
  - So $A[1..i]$ =(2,8,10,3,4,1,5,6,9,7) with probability $\frac{1}{(i-1)!} \cdot \frac{1}{i} = \frac{1}{i!}$

- There was nothing special about $A[1..i]$=(2,8,10,3,4,1,5,6,9,7). The exact same analysis says that $A[1..i] = (a_1, a_2, \ldots, a_i)$ iff
  - $Random(1,i)$ returns a specific number and
  - $A[1..i\text{-}1]$ is a specific $(a_1, a_2, \ldots, a_{i-1})$

  so the probability that $A[1..i] = (a_1, a_2, \ldots, a_i)$ is $\frac{1}{(i-1)!} \cdot \frac{1}{i} = \frac{1}{i!}$
  which means that it's a random permutation.

Outline

1. A Quick Review of Probability

2. The Hiring Problem

3. Generating a Random Permutation

4. Various Other Items
   - Shuffling Cards
   - The Birthday Paradox
   - Coupon Collectors
   - Generating Random Numbers

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 every person's birthday falls on one of the n days with equal probability.
There are k people in a room. How large should k be for us to expect two people in the room to have 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.

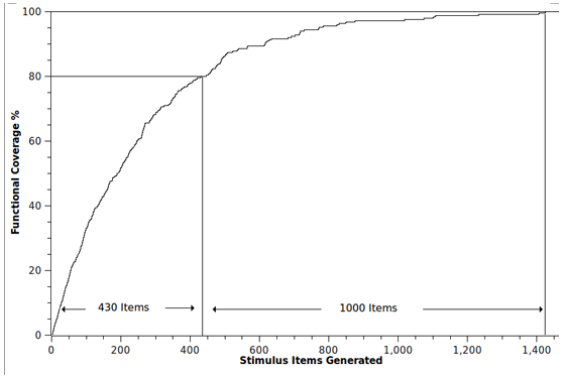
Coupon Collector

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 $= p = \frac{n-i}{n}$

$\Rightarrow$ expected *waiting time* $= \frac{1}{p} = \frac{n}{n-i}$

saw this w.t. formula at start of this set of slides

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`