CSE 5311 Notes 1: Mathematical Preliminaries

(Last updated 1/20/18 12:56 PM)

Chapter 1 - Algorithms & Computing

Relationship between complexity classes, e.g. $\log n$, n, $n \log n$, n^2, 2^n, etc.

Chapter 2 - Getting Started

Loop Invariants and Design-by-Contract (especially Table 1 on p. 13):
`http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2578702.2506375`

RAM Model

	Assumed	Reality
Memory Access		
Arithmetic		
Word size		

Chapter 3 - Asymptotic Notation

$f(n) = O(g(n)) \Rightarrow g(n)$ bounds $f(n)$ above (by a constant factor)

$f(n) = \Omega(g(n)) \Rightarrow g(n)$ bounds $f(n)$ below (by a constant factor)

$f(n) = \theta(g(n)) \Rightarrow f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Iterated Logarithms

$\log^k n = (\log n)^k$

$\log^{(k)} n = \log(\log^{(k\text{-}1)} n) = \log \log \ldots \log n \quad (\log^{(1)} n = \log n)$

$\lg^* n$ = Count the times that you can punch a log key on your calculator before value is ≤ 1

Arises for algorithms that run in "practically" or "nearly" constant time.

Appendix A - Summations

Review: Geometric Series (p. 1147) Harmonic Series (p. 1147)

Approximation by integrals (p. 1154)

Chapter 4 - Recurrences

SUBSTITUTION METHOD - Review

1. Guess the bound.
2. Prove using (strong) math. induction.

Exercise:

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

Try $O\left(n^3\right)$ and confirm by math induction

Assume $T(k) \le ck^3$ for $k < n$

$$T\left(\frac{n}{2}\right) \le c\frac{n^3}{8}$$

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + n \\ &\le 4c\frac{n^3}{8} + n \\ &= \frac{c}{2}n^3 + n \\ &= cn^3 - \frac{c}{2}n^3 + n \qquad -\frac{c}{2}n^3 + n \le 0 \text{ if } n \text{ is sufficiently large} \\ &\le cn^3 \end{aligned}$$

Improve bound to $O\left(n^2\right)$ and confirm

Assume $T(k) \le ck^2$ for $k < n$

$$T\left(\frac{n}{2}\right) \le c\frac{n^2}{4}$$

$T(n) = 4T\left(\frac{n}{2}\right) + n \le 4c\frac{n^2}{4} + n = cn^2 + n$ STUCK!

$\Omega\left(n^3\right)$ as lower bound:

Assume $T(k) \ge ck^3$ for $k < n$

$$T\left(\frac{n}{2}\right) \ge c\frac{n^3}{8}$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$$\ge 4c\frac{n^3}{8} + n$$

$$= \frac{c}{2}n^3 + n$$

$$= cn^3 - \frac{c}{2}n^3 + n \qquad -\frac{c}{2}n^3 + n \ge 0\,?$$

$\ge cn^3$ DID NOT PROVE!!!

$\Omega\left(n^2\right)$ as lower bound:

Assume $T(k) \ge ck^2$ for $k < n$

$$T\left(\frac{n}{2}\right) \ge c\frac{n^2}{4}$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n \ge 4c\frac{n^2}{4} + n = cn^2 + n \ge cn^2 \text{ for } 0 < c$$

What's going on . . .

$$T(1) = d$$

$$T(2) = 4T(1) + 2 = 4d + 2$$

$$T(4) = 4T(2) + 4 = 4(4d + 2) + 4 = 16d + 8 + 4 = 16d + 12$$

$$T(8) = 4T(4) + 8 = 4(16d + 12) + 8 = 64d + 48 + 8 = 64d + 56$$

$$T(16) = 4T(8) + 16 = 4(64d + 56) + 16 = 256d + 224 + 16 = 256d + 240$$

$$\text{Hypothesis}: \quad T(n) = dn^2 + n^2 - n = (d+1)n^2 - n = cn^2 - n$$

$O\left(n^2\right)$:

Assume $T(k) \le ck^2 - k$ for $k < n$

$$T\left(\frac{n}{2}\right) \le c\frac{n^2}{4} - \frac{n}{2}$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n \le 4\left(c\frac{n^2}{4} - \frac{n}{2}\right) + n = cn^2 - 2n + n = cn^2 - n$$

$\Omega\left(n^2\right)$: [This was already proven.]

Assume $T(k) \ge ck^2 - k$ for $k < n$

$$T\left(\frac{n}{2}\right) \ge c\frac{n^2}{4} - \frac{n}{2}$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n \ge 4\left(c\frac{n^2}{4} - \frac{n}{2}\right) + n = cn^2 - 2n + n = cn^2 - n$$

Exercise:

$$T(n) = T\left(\sqrt{n}\right) + 1$$

$O(\log n)$:

Assume $T(k) \le c \lg k$ for $k < n$

$$T\left(\sqrt{n}\right) \le c \lg \sqrt{n} = c\frac{\lg n}{2}$$

$$T(n) = T\left(\sqrt{n}\right) + 1 \le c\frac{\lg n}{2} + 1 = c \lg n - c\frac{\lg n}{2} + 1$$
$$\le c \lg n \quad \text{if } c \ge 2$$

$O(\log \log n)$

Assume $T(k) \le c \lg \lg k$ for $k < n$. (Note: $\log_a \log_a n \in \Theta(\log_b \log_b n)$)

$$T\left(\sqrt{n}\right) \le c \lg \lg \sqrt{n} = c \lg \frac{\lg n}{2} = c \lg \lg n - c$$

$$T(n) = T\left(\sqrt{n}\right) + 1 \le c \lg \lg n - c + 1$$
$$\le c \lg \lg n \quad \text{if } c \ge 1$$

$\Omega(\log\log n)$

$$\text{Assume } T(k) \ge c \lg\lg k \text{ for } k < n$$

$$T\left(\sqrt{n}\right) \ge c \lg\lg\sqrt{n} = c\lg\frac{\lg n}{2} = c\lg\lg n - c$$

$$T(n) = T\left(\sqrt{n}\right) + 1 \ge c\lg\lg n - c + 1$$

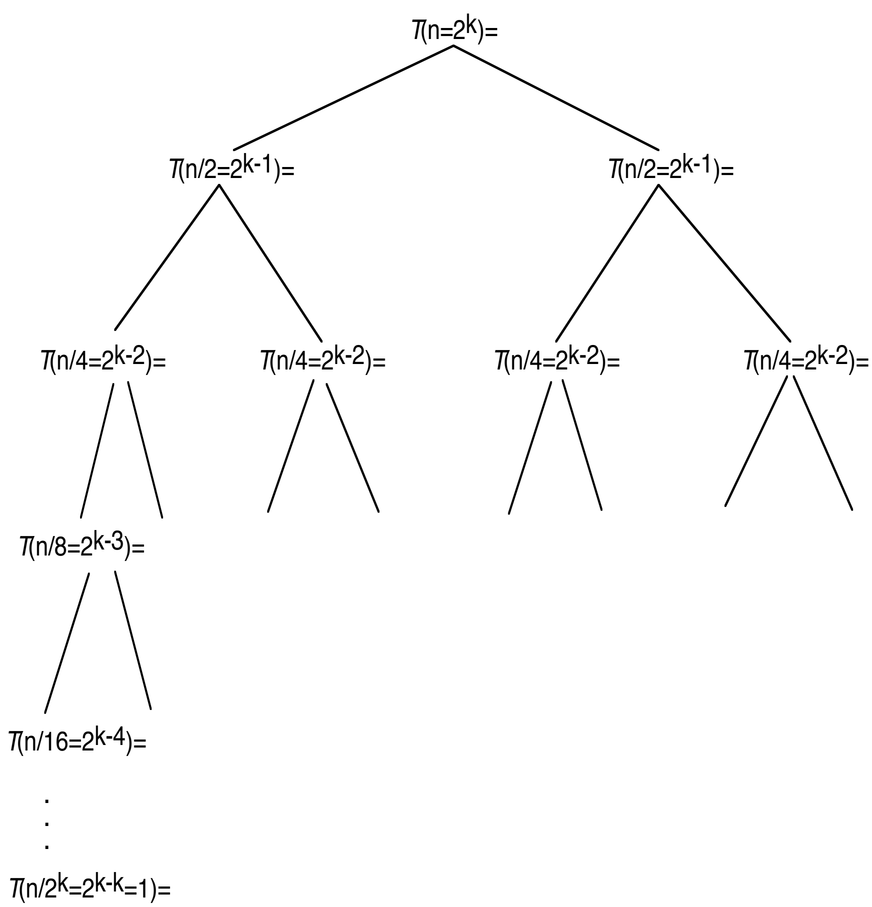
$$\ge c\lg\lg n \;\text{ if } 0 < c \le 1$$

Substitution Method is a general technique - See CSE 2320 Notes 08 for quicksort analysis

RECURSION-TREE METHOD - Review and Prelude to Master Method

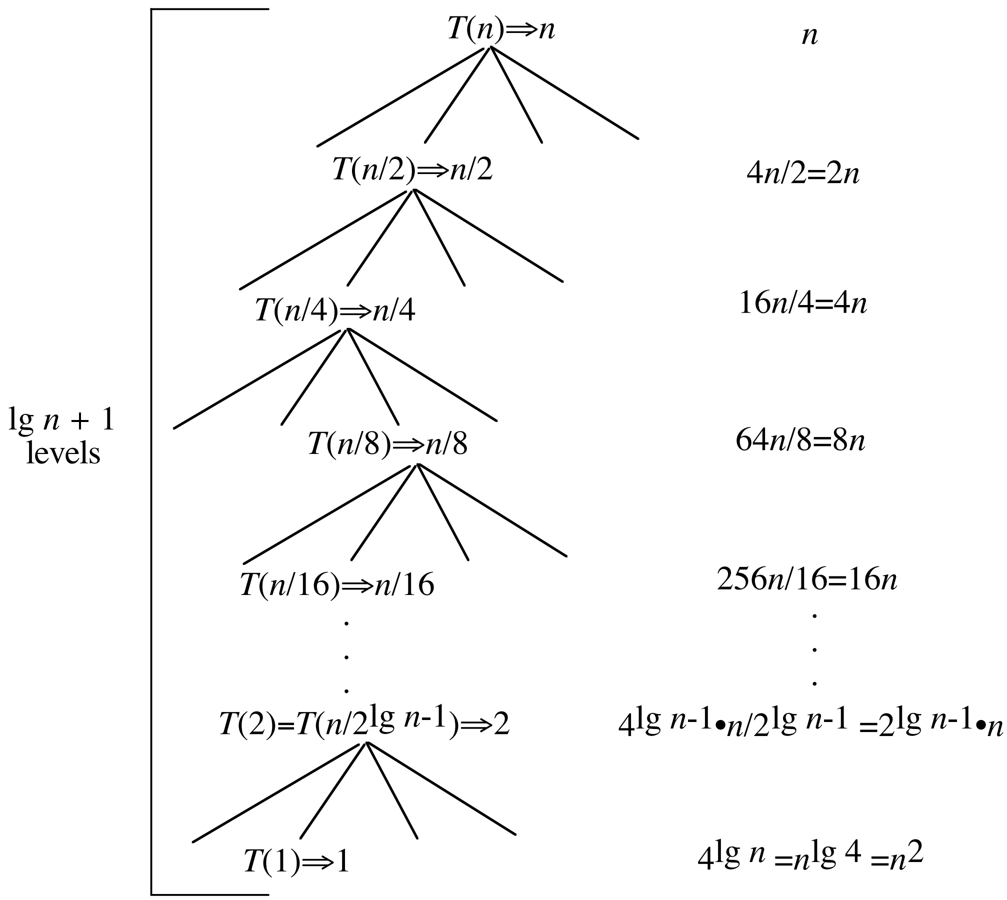
Convert to summation and then evaluate

Example: Mergesort, p. 38 $T(n) = 2T\left(\frac{n}{2}\right) + n$ (case 2 for master method)



Exercise:

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$



Using definite geometric sum formula:

$$cn\sum_{k=0}^{\lg n-1} 2^k + cn^2 = cn\frac{2^{\lg n}-1}{2-1} + \mathrm{c}n^2 \qquad \text{Using } \sum_{k=0}^{t} x^k = \frac{x^{t+1}-1}{x-1} \qquad x \neq 1$$

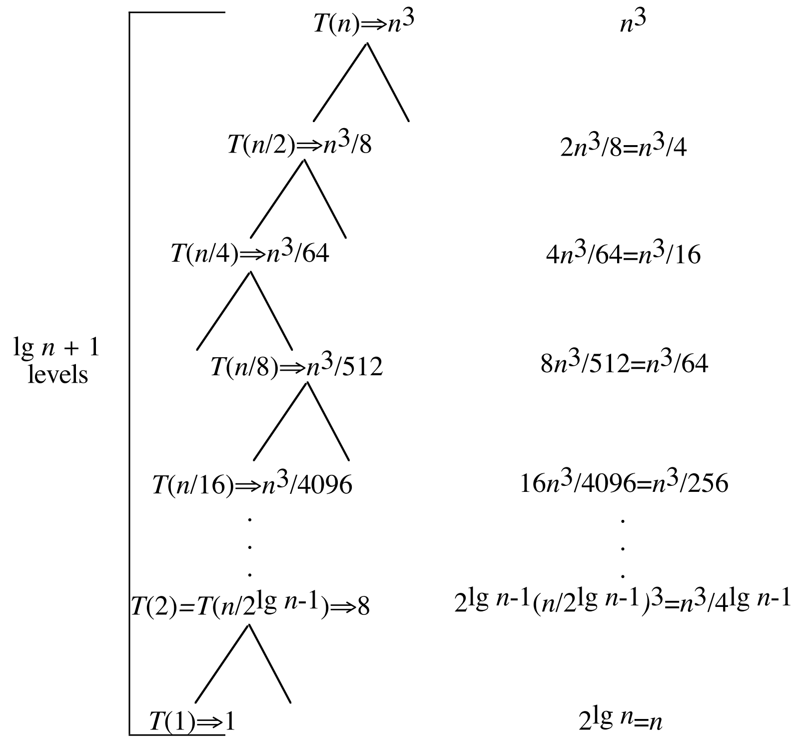
$$= cn(n-1) + \mathrm{c}n^2$$

$$= cn^2 - cn + \mathrm{c}n^2 = \Theta\left(n^2\right)$$

(Case 1 of master method)

Exercise: (case 3 of master method)

$$T(n) = 2T\left(\frac{n}{2}\right) + n^3$$



Using indefinite geometric sum formula:

$$cn^3 \sum_{k=0}^{\lg n-1} \frac{1}{4^k} + cn \le cn^3 \sum_{k=0}^{\infty} \frac{1}{4^k} + cn$$

$$= cn^3 \frac{1}{1-\frac{1}{4}} + cn \qquad \text{From } \sum_{k=0}^{\infty} x^k = \frac{1}{1-x} \qquad 0 < x < 1$$

$$= \frac{4}{3} cn^3 + cn = O(n^3)$$

Using definite geometric sum formula:

$$cn^3 \sum_{k=0}^{\lg n-1} \frac{1}{4^k} + cn = cn^3 \frac{\left(\frac{1}{4}\right)^{\lg n} - 1}{\frac{1}{4} - 1} + cn \qquad \text{Using } \sum_{k=0}^{t} x^k = \frac{x^{t+1} - 1}{x - 1} \qquad x \ne 1$$

$$= cn^3 \frac{n^{-2} - 1}{-\frac{3}{4}} + cn = cn^3 \frac{1 - n^{-2}}{\frac{3}{4}} + cn = \frac{4}{3} cn^3 \left(1 - \frac{1}{n^2}\right) + cn$$

$$= \Theta\left(n^3\right)$$

MASTER METHOD/THEOREM ("new") - CLRS 4.5

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

$a \geq 1 \quad b > 1$

Three mutually exclusive cases (proof sketched in 4.6.1):

1. $f(n) = O\left(\frac{n^{\log_b a}}{n^{\varepsilon}}\right),\ \varepsilon > 0 \Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$ (leaves dominate)

2. $f(n) = \Theta\left(n^{\log_b a}\right) \Rightarrow T(n) = \Theta\left(n^{\log_b a} \log n\right)$ (each level contributes equally)

3. $f(n) = \Omega\left(n^{\log_b a} n^{\varepsilon}\right),\ \varepsilon > 0$, and $af\left(\frac{n}{b}\right) \leq cf(n)$, $c < 1$ for all sufficiently large $n \Rightarrow T(n) = \Theta(f(n))$ (root dominates)

(Problem 4.6-3 shows that $\varepsilon > 0$ in 3. follows from the existence of c and n_0. By taking $n = b^k n_0$ and the condition on f, $\left(\frac{a}{c}\right)^{\log_b n - \log_b n_0} f(n_0) \leq f(n)$ may be established. Last step is $\varepsilon \leq \log_b\left(\frac{1}{c}\right)$.)

Example:

$$T(n) = 10T\left(\frac{n}{10}\right) + \sqrt{n}$$

$a = 10 \quad b = 10 \quad f(n) = n^{.5} \quad n^{\log_{10} 10} = n$

Case 1: $n^{.5} = O(n^{1-\varepsilon}),\ 0 < \varepsilon \leq 0.5$

$$\Rightarrow T(n) = \Theta(n)$$

Example: (recursion tree given earlier - leaves dominate)

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$a = 4 \quad b = 2 \quad f(n) = n \quad n^{\log_2 4} = n^2$

Case 1: $n = O(n^{2-\varepsilon}),\ 0 < \varepsilon \le 1$

$$\Rightarrow T(n) = \Theta\left(n^2\right)$$

Example:

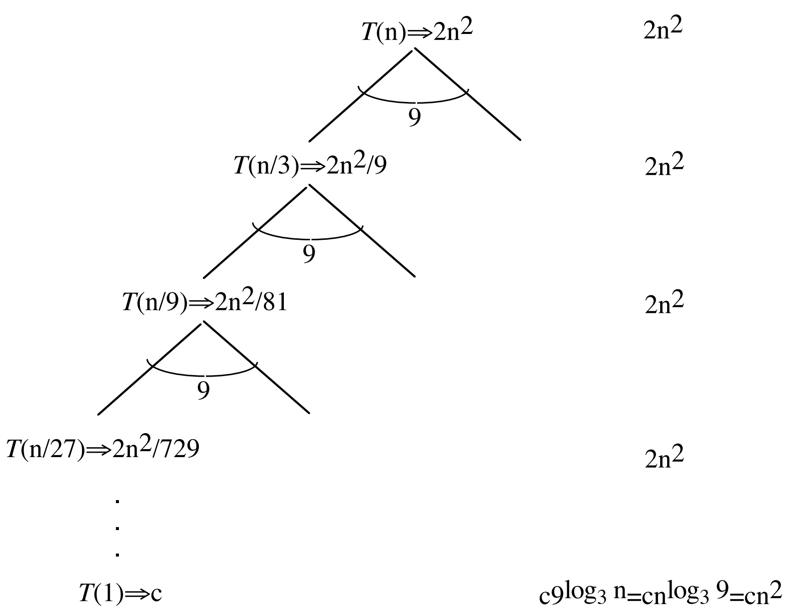
$$T(n) = 9T\left(\frac{n}{3}\right) + 2n^2$$

$a = 9 \quad b = 3 \quad f(n) = 2n^2 \quad n^{\log_3 9} = n^2$

Case 2: $2n^2 = \Theta(n^2)$

$$\Rightarrow T(n) = \Theta\left(n^2 \log n\right)$$

Recursion Tree



$$T(n) = 2n^2 \log_3 n + cn^2 = \Theta\left(n^2 \log n\right)$$

Example:

$$T(n) = 2T\left(\frac{n}{2}\right) + n^3$$

$a = 2 \quad b = 2 \quad f(n) = n^3 \quad n^{\lg 2} = n$

Case 3: $n^3 = \Omega(n^{1+\varepsilon})$, $0 < \varepsilon \le 2$

$$af\left(\frac{n}{b}\right) = 2\left(\frac{n^3}{8}\right) \le cn^3 = cf(n),\ \frac{1}{4} \le c < 1$$

$$\Rightarrow T(n) = \Theta\left(n^3\right)$$

Example:

$$T(n) = T\left(\frac{n}{2}\right) + \frac{n}{2}$$

$a = 1 \quad b = 2 \quad f(n) = n/2 \quad n^{\lg 1} = n^0 = 1$

Case 3: $n/2 = \Omega(n^0 n^\varepsilon)$, $0 < \varepsilon \le 1$

$$af\left(\frac{n}{b}\right) = 1\left(\frac{\frac{n}{2}}{2}\right) = \frac{n}{4} \le c\,\frac{n}{2} = cf(n),\ \frac{1}{2} \le c < 1$$

$$\Rightarrow T(n) = \Theta(n)$$

From CLRS, p. 95:

$$T(n) = 2T\left(\frac{n}{2}\right) + n\lg n$$

$a = 2 \quad b = 2 \quad f(n) = n \lg n \quad n^{\lg 2} = n^1 = n$

$f(n) = \omega(n)$, but $f(n) \notin \Omega\left(nn^\varepsilon\right)$ for any $\varepsilon > 0$ . . .

$$af\left(\frac{n}{b}\right) = 2f\left(\frac{n}{2}\right) = 2\frac{n}{2}\lg\left(\frac{n}{2}\right) = n\lg n - n = cn\lg n + (1-c)n\lg n - n$$
$$\le cf(n) = cn\lg n \text{ for } c \ge 1$$

So master method (case 3) does not support $T(n) = \Theta(n\lg n)$

Using exercise 4.6-2 as a hint . . . (or a simple recursion tree with $n = 2^k$)

Substitution method to show $T(n) = \Theta\left(n\lg^2 n\right)$

O:

Assume $T(k) \le ck\lg^2 k$ for $k < n$

$T\left(\frac{n}{2}\right) \le c\frac{n}{2}\lg^2\left(\frac{n}{2}\right) = c\frac{n}{2}(\lg n - 1)^2$

$T(n) = 2T\left(\frac{n}{2}\right) + n\lg n \le 2c\frac{n}{2}(\lg n - 1)^2 + n\lg n = cn\lg^2 n - 2cn\lg n + cn + n\lg n$

$\le cn\lg^2 n$ for $c \ge \frac{1}{2} + \varepsilon$

Ω:

Assume $T(k) \ge ck\lg^2 k$ for $k < n$

$T\left(\frac{n}{2}\right) \ge c\frac{n}{2}\lg^2\left(\frac{n}{2}\right) = c\frac{n}{2}(\lg n - 1)^2$

$T(n) = 2T\left(\frac{n}{2}\right) + n\lg n \ge 2c\frac{n}{2}(\lg n - 1)^2 + n\lg n = cn\lg^2 n - 2cn\lg n + cn + n\lg n$

$\ge cn\lg^2 n$ for $0 < c \le \frac{1}{2}$

PROBABILISTIC ANALYSIS

Hiring Problem - Interview potential assistants, always hiring the best available.

Input: Permutation of 1 . . . n

Output: The number of sequence values that are *larger* than all previous sequence elements.

3 1 2 **4**   **1 2 3 4**   **4** 3 2 1

Worst-case = n

Observation: For any k-prefix of sequence, the last element is the largest with probability ______.

Summing for all k-prefixes:

$$\ln n < \sum_{k=1}^{n} \frac{1}{k} = H_n \le \ln n + 1$$

Hiring m-of-n Problem

Observation: For any k-prefix of sequence, the last element is one of the m largest with probability ______.

______ if $k \le m$

______ otherwise

Sum for all k-prefixes (expected number that are hired)

$$m + \sum_{k=m+1}^{n} \frac{m}{k} = m + mH_n - mH_m$$

What data structure supports this application?

Coupon Collecting (Knuth)

n types of coupon. One coupon per cereal box.

How many boxes of cereal must be bought (expected) to get at least one of each coupon type?

Collecting the n coupons is decomposed into n steps:

Step 0 = get first coupon

Step 1 = get second coupon

Step m = get m+1st coupon

Step n - 1 = get last coupon

Number of boxes for step m

Let p_i = probability of needing *exactly* i boxes (difficult)

$p_i = \left(\frac{m}{n}\right)^{i-1} \frac{n-m}{n}$, so the expected number of boxes for coupon m + 1 is $\sum_{i=1}^{\infty} ip_i$

Let q_i = probability of needing *at least* i boxes = probability that *previous* i - 1 boxes are failures (much easier to use)

So, $p_i = q_i - q_{i+1}$

$$\sum_{i=1}^{\infty} ip_i = \sum_{i=1}^{\infty} i(q_i - q_{i+1})$$

$$= \sum_{i=1}^{\infty} iq_i - \sum_{i=1}^{\infty} iq_{i+1}$$

$$= \sum_{i=1}^{\infty} iq_i - \sum_{i=2}^{\infty} (i-1)q_i$$

$$= q_1 + \sum_{i=2}^{\infty} iq_i - \sum_{i=2}^{\infty} iq_i + \sum_{i=2}^{\infty} q_i$$

$$= \sum_{i=1}^{\infty} q_i$$

$$q_1 = 1$$

$$q_2 = \frac{m}{n}$$

$$q_3 = \left(\frac{m}{n}\right)^2$$

$$q_k = \left(\frac{m}{n}\right)^{k-1}$$

$$\sum_{i=1}^{\infty} q_i = \sum_{i=0}^{\infty} \left(\frac{m}{n}\right)^i = \frac{1}{1-\frac{m}{n}} = \frac{n}{n-m} = \text{Expected number of boxes for coupon } m + 1$$

Summing over all steps gives

$$\sum_{i=0}^{n-1} \frac{n}{n-i} = n\left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right) \le n(\ln n + 1)$$

Analyze the following game:

n sides on each die, numbered 1 through n.

Roll dice one at a time.

Keep a die only if its number > numbers on dice you already have.

a. What is the expected number of *rolls* (including those not kept) to get a die numbered n?

$$\sum_{i=0}^{\infty} \left(\frac{n-1}{n}\right)^i = \frac{1}{1-\frac{n-1}{n}} = \frac{n}{n-(n-1)} = n$$

b. What number of dice do you expect (mathematically) to *keep*?

Keep going until an n is encountered.

Repeats of a number are discarded.

Results in hiring problem $\left(H_n \le \ln n + 1\right)$

Suppose a gambling game involves a sequence of rolls from a standard six-sided die. A player wins $1 when the value rolled is the ***same*** as the previous roll. If a sequence has 1201 rolls, what is the expected amount paid out?

Probability of a roll paying $1 = 1/6. 1200/6 = $200

Suppose a gambling game involves a sequence of rolls from a standard six-sided die. A player wins $1 when the value rolled is ***larger*** than the previous roll. If a sequence has 1201 rolls, what is the expected amount paid out?

$$\text{Payout for next roll} = \frac{1}{6}\sum_{i=1}^{6} \text{Expected payout after roll of } i$$

$$= \frac{1}{6}\left(\frac{5}{6} + \frac{4}{6} + \frac{3}{6} + \frac{2}{6} + \frac{1}{6} + 0\right)$$

$$= \frac{15}{36} = \frac{5}{12}$$

1200*5/12 = $500

Suppose a gambling game involves a sequence of rolls from a standard six-sided die. A player wins k dollars when the value k rolled is ***smaller*** than the previous roll. If a sequence has 601 rolls, what is the expected amount paid out?

$$\frac{1}{6}\left(1 \bullet \frac{5}{6} + 2 \bullet \frac{4}{6} + 3 \bullet \frac{3}{6} + 4 \bullet \frac{2}{6} + 5 \bullet \frac{1}{6} + 6 \bullet \frac{0}{6}\right) = \frac{5+8+9+8+5}{36} = \frac{35}{36}$$

600*35/36 = $583.33

Suppose a gambling game involves a sequence of rolls from a standard six-sided die. What is the expected number of rolls until the first pair of consecutive sixes appears?

$$\text{Expected rolls to get first six} = 6 = \left(\sum_{i=0}^{\infty}\left(\frac{6-1}{6}\right)^{i} = \frac{1}{1-\frac{6-1}{6}} = \frac{6}{6-(6-1)}\right)$$

$$\text{Probability that next roll is not a six} = \frac{5}{6}$$

Expected rolls for consecutive sixes = $(6+1)\sum_{k=0}^{\infty}\left(\frac{5}{6}\right)^k = (6+1)\frac{1}{1-\frac{5}{6}} = 42$

Also, see: `http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2408776.2408800` and `http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2428556.2428578`

along with the book by Grinstead & Snell (`https://math.dartmouth.edu/~prob/prob/prob.pdf`), especially the first four chapters (and Example 4.6 on p. 136, `http://marilynvossavant.com/game-show-problem/`).

GENERATING RANDOM PERMUTATIONS

PERMUTE-BY-SORTING (p. 125, skim)

Generates randoms in 1 . . . n^3 and then sorts to get permutation in $\Theta(n \log n)$ time.

Can use radix/counting sort (CSE 2320 Notes 8) to perform in $\Theta(n)$ time.

RANDOMIZE-IN-PLACE

Array A must initially contain a permutation. Could simply be identity permutation: A[i] = i.

for i=1 to n
    swap A[i] and A[RANDOM(i,n)]

Code is equivalent to reaching in a bag and choosing a number to "lock" into each slot.

Uniform - all n! permutations are equally likely to occur.

Problem 5.3-3 PERMUTE-WITH-ALL

for i=1 to n
    swap A[i] and A[RANDOM(1,n)]

Produces n^n outcomes, but n! does not divide into n^n evenly.

$\therefore$ Not uniform - some permutations are produced more often than others.

Assume n=3 and A initially contains identity permutation. RANDOM choices that give each permutation.

```
1 2 3:  1 2 3  1 3 2  2 1 3  3 2 1
1 3 2:  1 2 2  1 3 3  2 1 2  2 3 1  3 1 1
2 1 3:  1 1 3  2 2 3  2 3 2  3 1 2  3 3 1
2 3 1:  1 1 2  1 3 1  2 2 2  2 3 3  3 1 3
3 1 2:  1 1 1  2 2 1  3 2 2  3 3 3
3 2 1:  1 2 1  2 1 1  3 2 3  3 3 2
```

RANDOMIZED ALGORITHMS

Las Vegas

Output is always correct.
Time varies depending on random choices (or randomness in input).
Challenge: Determine expected time.

Classic Examples:

Randomized (pivot) quicksort takes expected time in $\Theta(n \log n)$.
Universal hashing

This Semester: Treaps Perfect Hashing Smallest Enclosing Disk Rabin-Karp Text Search

Asides:

Ethernet: `http://en.wikipedia.org/wiki/Exponential_backoff`

List Ranking - Shared Memory and Distributed Versions
(`http://ranger.uta.edu/~weems/NOTES4351/09notes.pdf`)

Monte Carlo

Correct solution with some (large) probability. Use repetition to improve odds.

Usually has one-sided error - one of the outcomes can be ***wrong***

Example - testing primality without factoring

To check N for being prime:

Randomly generate some a with $1 < a < \sqrt{N}$)

If $N \bmod a = 0$, then report *composite* else report *prime*

One-sided error - *prime* could be wrong.

(Several observations from number theory are needed to make this robust for avoiding Carmichael numbers)

Concept: Repeated application of Monte Carlo technique can lead to:

Improved reliability

Las Vegas algorithm that gives correct result "with high probability"