Coupon collecting.

Practice with linearity of expectation.

- n coupon types. Get a random one each round. How long to get all coupons?
- general example of waiting for combinations of events to happen.
- expected case analysis:
  - after get k coupons, each sample has $1 - k/n$ chance for new coupon
  - so wait for $(k+1)^{st}$ coupon has geometric distribution.
  - expected value of geo dist w/param p is $1/p$
  - so get harmonic sum
  - what standard tools did we use? using **conditional expectation** to study on phase; used **linearity of expectation** to add
  - expected time for all coupons: $n \ln n + O(n)$.

Stable Marriage

Problem:

- complete preference lists
- stable if no two unmarried (to each other) people prefer each other.
- med school
- always exists.

Proof by proposal algorithm:

- rank men arbitrarily
- lowest unmarried man proposes in order of preference
- woman accepts if unmarried or prefers new proposal to current mate.

Time Analysis:

- woman's state only improves with time
- only n improvements per woman
- while unattached man, proposals continue
- (some woman available, since every woman he proposed to is married now)
- must eventually all be attached

Stability of final solution

- suppose X and y are dissatisfied with final pairing X-x, Y-y.
- X proposed to y first
- y prefers current Y to X.

Average case analysis

- nonstandard for our course

- random preference lists
- how many proposals?
- **principle of deferred decisions**
  - used intuitively already
  - random choices all made in advance
  - same as random choices made when algorithm needs them.
- use for discussing autopartition, quicksort
- Proposal algorithm:
  - defered decision: each proposal is random among unchosen women
  - still hard
  - Each proposal among all women
  - **stochastic domination**: X s.d. Y when $\Pr[X > z] \geq \Pr[Y > z]$ for all z.
  - **Result:** $E[X] \geq E[Y]$.
  - done when all women get a proposal.
  - at each step $1/n$ chance women gets proposal
  - This is just coupon collection: $O(n \log n)$

Game Trees

Tree evaluation.

Moving LOE through a (linear) recurrence.

- define. algo cost is number of leaves. $n = 2^h$
- NOR model
- Difference from book

deterministic model: must examine all leaves. time $2^h = 4^{h/2} = n$

- by induction: on any tree of height h, as questions are asked, can answer such that root is not determined until all leaves checked.
- Note: bad instance being constructed on the fly as algorithm runs.
- But, since algorithm deterministic, bad instance can be built in advance by simulating algorithm.

nondeterministic/checking

- $W(0) = L(0) = 1$
- winning position can guess move. $W(h) = L(h-1)$
- losing must check both. $L(h) = 2W(h-1)$
- follows $W(h) = 2 * W(h-2) = 2^{h/2} = n^{1/2}$

randomized–guess which leaf wins.

- $W(0) = 1$
- $W(T)$ is a random variable

  - If T is winning time it takes to verify T is a win. Undefined if T is losing.
  - Ditto $L(T)$.
  - Expectation is over random choices of algorithm; NOT over trees.
  - Different trees have different expectations
- $W(h)$ = max over all height-h winning trees of $E[W(T)]$
- $L(h)$ = same for losing trees.
- Consider any losing height-h tree
  - both children are winning
  - must eval both.
  - each takes at most $W(h-1)$ in expectation
  - Thus (by linearity of expectation) we take at most $2W(h-1)$
  - Deduce $L(h) \leq 2W(h-1)$.
- Consider any winning height-h tree
  - Possibly both children are losing. If so, we stop after evaling the first child we pick. Total time $L(h-1)$.
  - If exactly one child losing, two cases:
    * if first choice is winning, eval it and stop: time at most $L(h-1)$.
    * if first choice is losing, eval both children: $L(h-1) + W(h-1)$.
    * So, $W(h) \leq \frac{1}{2}L(h-1) + \frac{1}{2}(W(h-1) + L(h-1))$
    * That is, $W(h) \leq L(h-1) + \frac{1}{2}W(h-1)$
    * This is worse bound than $L(h-1)$
  - Loose approx: $W(h-1) \leq L(h-1)$ and $W(h) \leq (3/2)L(h-1) \leq 3W(h-2)$.
  - conclude $W(h) \leq 3^{h/2}$
  - So $W(h) \leq 3^{h/2} = n^{\log_4 3} = n^{0.793}$
  - Tighter analysis
    * $W(h) \leq 2W(h-2) + \frac{1}{2}W(h-1)$
    * Characteristic equation: $x^2 - x/2 + 2 = 0$
    * Conclude $W(h) = O(\alpha^h)$ where $\alpha = \frac{1}{4}(1 + \sqrt{33})$
    * Recall $h = \log_2 n$
    * So $W(n) \leq \alpha^{\log_2 n} = n^{\log_2 \alpha} = n^{0.753...}$

Note:

- Changed presentation from book.
- We used "game tree" with win/loss
- So if win denoted by 0, loss by 1, then function at each node is NOR
- MR uses "MIN/MAX tree" with d "rounds" (1 move per player)
- corresponds to Win/Loss tree of height $2d$ (role of 0/1 in MIN/MAX gets alternately flipped on W/L

Yao's Minimax Principle

How do we know our randomized algorithm is best possible?
Review tree evaluation.
Lower Bound
Game Theory

- Zero sum games. Scissors Paper Stone. Roberta, Charles.
- Payoff Matrix M. Entries are (large) strategies. chess.

Optimal strategies

- row wants to maximize, column to minimze
- suppose Roberta picks i. Guarantees $\min_j M_{ij}$.
- (Pessimistic) R-optimal strategy: choose i to $\max_i \min_j M_{ij}$.
- (Pessimistic) C-optimal strategy: choose j to $\min_j \max_i M_{ij}$.

When C-optimal and R optimal strategies match, gives **solution** of game.

- if solution exists, knowing opponents strategy useless.
- Sometimes, no solution using these **pure** strategies

Randomization:

- **mixed** strategy: distribution over pure ones
- R uses dist p, C uses dist q, expected payoff $p^T Mq$
- Von Neumann:
$$\max_p \min_q p^T Mq = \min_q \max_p p^T Mq$$
that is, always exists solution in mixed strategies.
- Once p fixed, exists optimal pure q, and vice versa
- Why? Because Mq is a vector with a maximum in one coordinate.

Yao's minimax method:

- Column strategies algorithms, row strategies inputs
- payoff is running time
- randomized algorithm is mixed strategy
- optimum algorithm is optimum randomized strategy
- worst case input is corresponding optimum pure strategy
- Thus:
  - worst case expected runtime of optimum rand. algorithm
  - is payoff of game
  - instead, consider randomized inputs
  - payoff of game via optimum pure strategy
  - which is detemrinistic algorithm!

- Worst case expected runtime of randomized algorithm for any input equals best case running time of a deterministic algorithm for worst distribution of inputs.
- Thus, for lower bound on runtime, show an input distribution with no good deterministic algorithm

Game tree evaluation lower bound.

- Recall Yao's minimax principle.
- lemma: any deterministic alg should finish evaluating one child of a node before doing other: *depth first pruning algorithm.* proof by induction.
- input distribution: each leaf 1 with probability $p = \frac{1}{2}(3 - \sqrt{5})$.
- every node is 1 with probability p
- let $T(h)$ be expected number of leaves evaluated from height h.
- with probablity p, eval one child. else eval 2.
- So
$$T(h) = pT(h-1) + 2(1-p)T(h-1) = (2-p)^h = n^{0.694}$$
- Better bound:
  - build recursively from root,
  - where each winning node has (at random) one winning and one losing child.
  - shows our upper bound of $\frac{1}{4}(1 + \sqrt{33})^h$ is tight.