Complexity.

What is a rand. alg? What is an alg?

- Turing Machines. RAM with large ints. log-cost RAM as TM.
- language as decision problem (vs optimization problems) “graphs with small min-cut.” algs accept/reject
- complexity class as set of languages
- \( P \). polynomial time in input size
- \( NP \) as \( P \) with good advice string. witnesses
- polytime reductions. hardness, completeness.

Randomized algorithms have advice string, but it is random

- measure probs over space of advice strings
- equivalence to flipping unbiased random bits

\( ZPP \) (zero error probabilistic polytime)

- Polynomial expected time
- \( A(x) \) accepts iff \( x \in L \).
- Las Vegas algorithms

\( RP \) (randomized polytime) (MC with one-sided error).

- polytime (always)
- \( x \notin L \Rightarrow \) rejects (always).
- \( x \in L \Rightarrow \) accepts with probability \( > 1/2 \).
- Monte Carlo algorithm
- one sided error
- precise numbers unimportant: amplification.
- min-cut example
- \( coRP \).

- What if NOT worst case polytime? stop when passes time bound and accept.
- \( ZPP = RP \cap coRP \)
$PP$ (probabilistic polytime) (two-sided MC)
- Worst case polytime (can force)
  - $x \in L \Rightarrow$ accepts prob > $1/2$
  - $x \notin L \Rightarrow$ accepts prob < $1/2$
- weakness: $NP \subseteq PP$

$BPP$ (bounded probabilistic polytime)
- worst case polytime (can force)
  - $x \in L \Rightarrow$ accepts prob > $3/4$
  - $x \notin L \Rightarrow$ accepts prob < $1/4$
- precise numbers unimportant.

Clearly $P \subseteq RP \subseteq NP$. Open questions:
- $RP = coRP$? (equiv $RP = ZPP$)
- $BPP \subseteq NP$?

Tree evaluation.
Moving LOE through a (linear) recurrence.
- define. algo cost is number of leaves. $n = 2^h$
- NOR model
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.

dataet restart/cecking
- $W(0) = L(0) = 1$
- winning position can guess move. $W(h) = L(h - 1)$
- losing must check both. $L(h) = 2W(h - 1)$
- follows $W(h) = 2 \cdot W(h - 2) = 2^{h/2} = n^{1/2}$
  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: $L(h-1) + W(h-1)$.
    * Conjecture: $W(h-1) \leq L(h-1)$
    * Then time $\leq 2L(h-1)$.
  – Each case 1/2 the time. Thus, expected time $\leq (3/2)L(h-1)$.
  – Deduce $W(h) \leq (3/2)L(h-1) \leq (3/2)2W(h-2) = 3W(h-2)$
  – So $W(h) \leq 3^{h/2} = n^{\log_3 3} = n^{0.793}$
  – Go back and confirm assumption that $W(h) \leq L(h)$.

3