Sanjeev Arora + Boaz Barak, "Computational Complexity: A Modern Approach"
Cambridge Univ. Press. ($48 on amazon)

Introduction to Complexity Theory

Def'n (Recall)

Turing machine model:

Finite Set of States
Finite Set of Symbols.
Finite Set of 1-dimensional tapes
Finite set of transition rules.

Inputs: strings $\in \{0,1\}^*$
Output: Either Accept/Reject or string in $\{0,1\}^*$

Universal model of computation.

→ In two senses:

(a) $\exists$ universal T.M. U s.t.

$\forall$ TM's M, $U(\ulcorner M \urcorner, x) = M(x)$.

and runtime of $U(\ulcorner M \urcorner, x) \le c_M \cdot r \log r$

where r = runtime of $M(x)$.

(b) For any "reasonable" model of computation,

$\forall$ program f, $U(\ulcorner f \urcorner, x) = f(x)$

and runtime of $U(\ulcorner f \urcorner, x) \le c_f \cdot r^c$ ($c=2$, say)

Hence finding runtimes using TM's or using other standard models such as Random Access Machines (RAMs) are equivalent up to polynomial transformations.

RAM, ex. 1.9, p 35.

Also, changing alphabet size / number of tapes makes a difference in runtime of at most a constant factor / logarithmic factor.

E.g. going to an alphabet Γ, $m = |\Gamma|$, only gives a speedup of a factor $\log m$. [Pf. Idea: use m' 0/1's to code one Γ-symbol, Simulate 1 step of the machine that $3 \log|\Gamma|$ steps.]

Slightly harder: Converse holds too: If M has alphabet size $m = |\Gamma|$, then using an alphabet size $m' = |\Gamma'| > m$, gives a speedup of factor $\le \frac{\log|\Gamma'|}{\log|\Gamma|}$.

$\mathbb{N} = \{0,1,2,\ldots\}$

<u>Definition</u>: Let ~~T : N →~~ $T : \mathbb{N} \to \mathbb{N}$.

"accepts L" means same thing

Deterministic Time

$\mathrm{DTIME}(T) = \{L : \text{for some TM } M,\ M \text{ decides } L \text{ and runs in time } O(T(n))\}$.

where: L - a language is any subset of $\{0,1\}^*$.

M <u>decides</u> L iff $\forall \sigma$, M accepts σ if $\sigma \in L$, and M rejects σ if $\sigma \notin L$.

In particular, M halts on all inputs σ. (!)

M runs in time $O(T(n))$ iff $\exists c > 0$ s.t. for all σ, $M(\sigma)$ halts in time $\leq c \cdot T(|\sigma|)$.

<u>Comments</u>:

- M can be a k-tape TM, for any $k \geq 1$.
- $O(T(n))$ is used, as constant factor speedups/slowdowns are irrelevant
- Letting number ~~of~~ k of tapes vary is iffy, but standard.
- We use "languages" or "decision problems" as our central notion. Sometimes we are interested in the DTIME complexity of functions too — this is defined in the obvious fashion.
- RUNTIME - bounded by a function of the <u>length</u> of the input. ⇐!

<u>Def'n</u> $P = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c)$. "Polynomial time"

<u>Remark</u>: This is how it is usually stated, but it really means

$$\bigcup_{c \geq 1} \mathrm{DTIME}(\max\{n^c, 1\}).$$

Small mistakes at lengths $n=0$, $n=1$ should be tolerated without comment.

Remark: By earlier comments, P is the same class whether defined by Turing machines, or a more realistic model such as RAMs.

Remark A function $f: \{0,1\}^* \to \{0,1\}^*$ ~~iff~~ is polynomial time computable iff

(a) f has polynomial growth rate, $\forall \sigma$, $|f(\sigma)| \le |\sigma|^c + c$ for some c

(b) the predicate $\{\langle i, \sigma, s\rangle : i\text{th bit of } f(\sigma) \text{ is symbol } s\}$ is in P.

The class of polynomial time computable functions is denoted FP, or sometimes just P.

History Cobham, 1964 & Edmonds, 1965 - gave original definitions of P/FP and highlighted the importance of P as corresponding to feasible computability.

Other remarks We typically use well-behaved $T(n)$'s to bound runtime.

E.g. $T(n) \geq n$

$T(n)$ is nondecreasing

also, technical condition: $T(n)$ is time-constructible

Space bounded computation:

Intuitively Space = # of tape squares visited during the computation.

A language L is in $SPACE(s(n))$ iff ~~$\forall n$, ...~~,

there is a TM M that uses space $\le O(s(n))$

and decides L.

Small catch, we want to allow $s(n) = \log n$ so as to define LOGSPACE, but the input uses space $n \ne O(\log n)$.

Solution: Ignore the size of the input.

Defn For space-bounded TM's: the input tape is read-only, work tapes (and optionally the output ~~tape~~) are read-write

Only space on read-write tapes is counted in the ~~computation~~. space used.

Definition: Let $S: N \to N$, $S(n) \ge \log n$, $s(n)$ "well-behaved" (non decreasing / space constructible)

$SPACE(S) = \{ L \subseteq \{0,1\}^*$: for some TM M, M decides L and $M(\sigma)$ uses space $O(S(|\sigma|))$ for all inputs $\sigma \}$

Comments
- Page 2 comments all apply again.
- If $s(n) \ge n$, the input tape/work tape distinction is unnecessary.
- If $s(n) < \log n$, weird things can happen.
- Usual definition $f: \{0,1\}^* \to \{0,1\}^*$ being computable in space $s(n)$ is: $\forall \sigma$, $|f(\sigma)| \le 2^{O(s|\sigma|)}$ and $\{ \langle i, \sigma, s \rangle :$ i-th bit of $f(\sigma)$ is $s \}$ is in $SPACE(s(n))$.
  Note here: $|i| = O(s(|\sigma|))$.
- It is usual to use $SPACE(s)$ but $\underline{D}TIME(T)$.

Definition $PSPACE = \bigcup_{c \ge 1} SPACE(n^c)$. $LOGSPACE = SPACE(\log n)$

$PolyLogSpace = \bigcup_{c \ge 1} SPACE((\log n)^c)$.

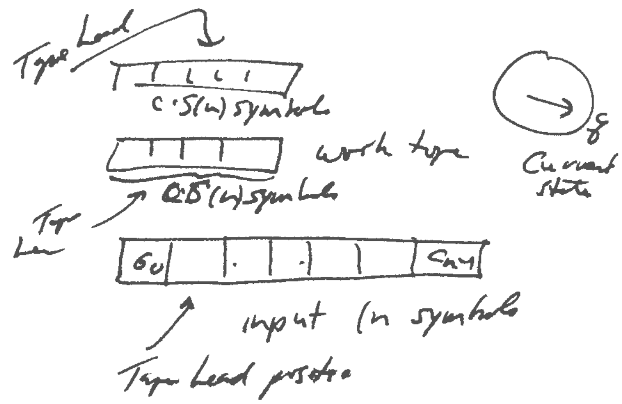
Theorem: DTIME$(T(n)) \subseteq$ SPACE$(T(n))$

Pf: This is obvious.

Theorem: SPACE$(S(n)) \subseteq$ DTIME$\left(2^{O(S(n))}\right)$ $\left(:= \bigcup_{c \geq 1} \text{DTIME}(2^{c \cdot S(n)})\right)$.

Pf: A configuration of a T.M. is ~~the~~ a complete state of its

(a) Tape contents
(b) Tape head positions
(c) Current state.



For a fixed input σ, a configuration can be described with

- Input tape head position
  $O(\log n)$ bits
- Contents of 1st work tape
  $O(c.S(n))$ ~~symbols~~ bits
- Tape head position of 1st work tape: $O(\log(S(n)) + \log c)$ bits

Same for ~~work~~ tapes 2-k (k is fixed for fixed M)

- Current state $O(1)$ bits

Total description of configuration is $O(\log n) + O(\cancel{\log}\, S(n)) + O(\cancel{\log}\, \log S) + O(1)$
$= O(\cancel{\log}\, S)$ many bits.

In particular, there are $< 2^{O(\cancel{\log}\, S)}$ many configurations.

since each is described with only $O(\cancel{\log}\, S)$ many bits.

Consider M that decides L in SPACE(S(n)), and consider a fixed input σ.

Claim M(σ) enters each configuration at most once.

Pf O/w, since M is deterministic, it would be in a loop and never halts.

Hence M(σ) runs for time $\leq$ (# of configurations) $= 2^{O(S(|\sigma|))}$.

q.e.d.

<u>Corollary</u> $\mathrm{LOGSPACE} \subseteq P \subseteq \mathrm{PSPACE} \subseteq \cancel{\mathrm{EXP}}\, E \left(:= \bigcup_c 2^{c \cdot n}\right) \subseteq \mathrm{EXP}\left(:= \bigcup_c 2^{n^c}\right).$

Nondeterministic Computation

Defn of NP #1

Here we view P as the class of feasibly computable problems
+ NP as the class of feasibly verifiable properties.

Defn: $NP = \{ L \subseteq \{0,1\}^* :$ $\exists$ predicate (language) $M \in P$ ~~s.t.~~ and a polynomial $p(n)$, s.t.
$$\forall \sigma,\ \sigma \in L \iff \exists u \in \{0,1\}^{p(|\sigma|)}\ M(x,u) \}$$
i.e. $\langle x,u \rangle \in M$

M can be thought of as a language in P or equivalently as a polynomial time bounded T.M.

When $M(x,u)$, $|u| \le p(|\sigma|)$ holds, u is called a certificate or witness for $x \in L$.

Example Calculus problem w/ known length solution.
$P \leftrightarrow$ computing a solution.
$NP \leftrightarrow$ verifying a solution is correct.

Examples:
(1) Factoring: $L_1 = \{ \cancel{x \in N}, \langle x, L, U \rangle,\ x \ge L \ge U, \in N,$
x has a prime factor $p \in [L,U] \}$
If you could solve this decision problem efficiently, then you could factor integers efficiently by using binary search
Note $L_1 \in NP$.
Open: Is $L_1 \in NP$? (Is Factoring $\in P$.)
Is L_2 NP complete?

(2) ~~ESP~~ Graph Isomorphism: $L_2 = \{ G_1 = \{V_1, E_1\}, G_2 = \{V_2, E_2\} : G_1 \cong G_2 \}$

~~Given graph~~ Inputs G_1, G_2 are coded as binary strings in some straightforward way, e.g., by incidence matrix are $\{0,1\}$. G_1, G_2 - undirected graphs

$L_2 \in NP$ (Take u to be an explicit isomorphism)

Open: is $L_2 \in P$. Is L_2 NP-complete

(3) Travelling Salesman Problem:

$L_3 = \{$ Graph $G = \{V, E\}$, $f: E \to N$, k : $\exists$ a circuit visiting each vertex exactly once with total weight $\le k \}$

$L_3 \in NP$ L_3 - NP-complete.

(4) Theorem of Set Theory ZF: $L_4 = \{ \langle \varphi, 0^k \rangle : \varphi$ is a ZF-formula and has a proof of $\le k$ symbols $\}$

$L_4 \in NP$, L_4 - NP complete

Open if $L_4 \in P$.

<u>Theorem</u> $P \subseteq NP$.

<u>Pf</u> M ignores u, ~~and~~ just solves L.

<u>Theorem</u>: $NP \subseteq PSPACE$.

<u>Pf</u> Algorithm for L:

For each $u \in \{0,1\}^{p(|x|)}$
  if $M(x, u)$ accepts,
    halt + accept
End for
Halt + reject.

Space used: Space to store u, i.e. $O(p(|x|))$
+ Space to run $M(x; u)$, which is $\le$ Time to run $M(x; u)$.

Corollary $NP \subseteq EXPTIME$

Nondeterministic computation

NTM (Non-deterministic TM)

Like a TM <u>but</u> has two transition functions,
i.e., at each configuration, there are two possible moves based on currently read symbols.

$\delta_0(q_0, \sigma_1 \ldots, \sigma_k) = (q_1, a_1 \cdots a_k) =$ q_i - states
$\delta_1(\cdot \cdot \cdot \cdot) = (\cdot \cdot \cdot \cdot)$. σ_i - symbols
$a_i \in T \cup \{L, R\}$

* Comment: • Can allow one option in some states, by setting $\delta_0 = \delta_1$.
• Can simulate any number of options, by repeated choosing among two (any number of local changes to the configuration)

Defn $M(x)$ accepts, M "accepts" x, or $M(x) = 1$,
means, there is some possible computation of $M(x)$ that leads to the accepting state.
If <u>all</u> possible computation paths lead to rejecting state, M rejects x, $M(x) = 0$.
M runs in time $T(n)$ iff $\forall \sigma$, $M(\sigma)$ halts in $\leq T(n)$ steps, $n = |\sigma|$,
on <u>all possible</u> computation paths.

Wlog, $M(\sigma)$ always halts, since we can just cut-off its computation after $T(n)$ steps. (Provided $T(n)$ is time constructible)

Defn: $NTIME(T(n)) = \{ L \subseteq \{0,1\}^* : \exists c$, NTM M, M runs in $c \cdot T(|\sigma|)$ steps on all inputs, and $L = \{\sigma : M \text{ accepts } \sigma\}.\}$

Defn (2nd alternate)
$NP = \bigcup_{c \geq 1} NTIME(n^c)$.

Discussion: NTIME machines never need to have a definition "reject" condition.

Thm The two definitions are equivalent.

Pf: $\Rightarrow$ Suppose L satisfies the first definition

NTM algorithm for L:

Input σ:

"Guess" u by writing out string of $p(|\sigma|)$ many 0/1's on a tape.

Run $M(x,u)$.

Accept iff $M(x,u)$ accepts

$\Leftarrow$ Suppose L satisfies the second definition.

Accepted by M runs in ~~...~~ $p(n)$ steps.

Let $u \in \{0,1\}^{p(n)}$ indicate M's nondeterministic choices

so i-th bit $= \begin{cases} 0 \\ 1 \end{cases}$ means in i-th step $M(\sigma)$ uses $\begin{cases} \delta_0 \\ \delta_1 \end{cases}$.

$M'(x,u)$ runs M deterministically choosing at each step to use δ_0 or δ_1 depending i-th bit of u.

q.e.d.

Thm $NTIME(T(n)) \subseteq SPACE(T(n))$.

Pf: Let $L \in NTIME(T(n))$, run time $< c \cdot T(n)$ on NTM M

$SPACE(T(n))$ algorithm for L:

Input σ, $|\sigma| = n$

Loop; ~~for~~ each $u \in \{0,1\}^{c \cdot T(n)}$,

Run $M(\sigma)$ deterministically for $c \cdot T(n)$ using ~~...~~ bits of u select δ_0/δ_1 transition rule

If $M(\sigma)$ accepts; halt and accept

Else: continue w/ next u.

End loop

Halt & reject.

Picture $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$

<u>Def'n</u> $NSPACE(S(n)) = \{ L \subseteq \{0,1\}^* :$ For some NTM M, M runs in space $S(n)$ for all input σ, $|\sigma|=n$, and $L = L(M)$. $\}$

<u>Def'n</u> $L(M) = \{\sigma : M \text{ accepts } \sigma\} = \{\sigma : M(\sigma) = 1\}$.

<u>Def'n</u> NL : $NSPACE(\log n)$ Nondeterministic log space

$NPSACE = \bigcup_{c \geq 1} NSPACE(n^c)$.

<u>Thm</u> $NSPACE(S(n)) \subseteq DTIME(2^{O(S(n))})$ $\left(= \bigcup_{c\geq 1} DTIME(2^{c \cdot S(n)})\right)$.

Buggy Proof

<u>Pf</u>: Let $L \in NSPACE(S(n))$, via NTM M.

<u>Algorithm for L</u>: Input σ, $|\sigma| = n$

Loop, for all $u \in \{0,1\}^{c \cdot S(n)}$

~~Simulate~~ Run $M(x, u)$, for $2^{O(S(n))}$ steps. (using alg + bds on page 5) using u to guide nondeterministic choice

If $M(x,u)$ accepts, halt + accept.

Endloop

Halt + reject.

Runtime: $2^{cS(n)} \cdot 2^{O(S(n))} = 2^{O(S(n))}$.

qed.

<u>Thm</u> $SPACE(S(n)) \subseteq NSPACE(S(n))$

<u>Pf</u>: Obvious!

Savitch's Thm: Let $S(n) \geq \log n$ be "well-behaved" (space-constructible).
Then $NSPACE(S(n)) \subseteq SPACE((S(n))^2)$. [Walt Savitch '70]

Pf: Let $L \in NSPACE(S(n))$ be accepted by NTM M in space $c \cdot S(n)$.
For ~~any~~ fixed input σ, $|\sigma| = n$, M has $2^{O(S(n))} \leq 2^{d \cdot S(n)}$ configurations,
each describable with ~~...~~ $d \cdot S(n)$ bits.

We shall show how to compute

$R_M(\sigma, C_0, C_1, t) :=$ $C_0, C_1 \in \{0,1\}^{d S(n)}$ code configurations of M on input σ,
$t \geq 1$, $t \in \mathbb{N}$, and
$\exists$ computation of $M(\sigma)$ that starts at C_0 and reaches C_1 in $\leq 2^t$ steps.

Then $\sigma \in L \iff \exists$ accepting C_1, $R_M(\sigma$, Initial config, $C_1, d S(n))$.

Also, if $t > 0$, $R_M(\sigma, C_0, C_1, t) \iff \exists C_{1/2}$ $R_M(\sigma, C_0, C_{1/2}, C_1, t-1)$.

And $R_M(\sigma, C_0, C_1, 0)$ is easy to check if true or false.

Algorithm: Input σ.
Loop over all $C_1 \in \{0,1\}^{d \cdot S(n)}$.
If C_1 codes an accepting configuration for $M(\sigma)$
Compute $R_M(\sigma$, (Initial Config.), $C_1, d \cdot S(n))$
If it accepts, halt + accept.
End loop
Reject

Algorithm for R_M

If $t = 0$, Return True/False ~~accept or reject~~, based on M's δ_0/δ_1 function.
Else
Loop over all $C_{1/2} \in \{0,1\}^{d \cdot S(n)}$
Compute $R_M(\sigma, C_0, C_{1/2}, t-1)$ +
" $R_M(\sigma, C_{1/2}, C_1, t-1)$.
If both accept, then return True
End loop
Return False.

Space use $2^{O(S(n)^2)}$ – Pf - Recursion depth $d \cdot S(n)$, Local var's use $O(S(n))$ bits.

Defn coNP = ...

Two ~~pre~~ alternate defn:

(1) Thm, version

(2) All computation paths of NTM M lead to acceptance.

Remark $co\text{TIME}(T(n)) = \text{TIME}(T(n))$

$co\text{SPACE}(S(n)) = \text{SPACE}(S(n))$.

Theorem [Immerman - Szelepcsényi ~~87~~ 88].

Let $S(n) \geq \log n$ be space constructible.

Then $\text{NSPACE}(S(n)) = co\text{NSPACE}(S(n))$.

Pf Next time .... or this time.

Theorem $P = NP \Rightarrow coNP = NP$. and $P = coNP$

Pf $coP = P$. ☑

Theorem [Hartmanis-Stearns '64] [Cook '72, ~~[illegible]~~ Zak '83]

If T, T' are time constructible, $T(n), T'(n) \geq n$. and ~~$\lim_{n\to\infty} \frac{T(n)}{T(n)\log T(n)}$~~ $\lim_{n\to\infty} \frac{T(n)\log(T(n))}{T'(n)} = 0$

then $\text{DTIME}(T(n)) \subsetneq \text{DTIME}(T'(n))$.

If $\lim_{n\to\infty} \frac{T(n+1)}{T'(n)} = 0$, then $\text{NTIME}(T(n)) \subsetneq \text{NTIME}(T'(n))$.

Theorem [Stearns - Hartmanis - Lewis] '65]

If S, S' are space-constructible and $\lim_{n\to\infty} \frac{S'(n)}{S(n)}$, then

$\text{SPACE}(S(n)) \subsetneq \text{SPACE}(S'(n))$.

Pfs Next time?

$$\overbrace{L \subseteq NL \subseteq P \subseteq NP \subseteq \text{PSPACE}}^{\neq} = \text{NPSPACE} \subseteq \text{EXP}$$

$NL \subseteq \text{SPACE}(\log^2(n))$, $P \subseteq coNP \subseteq \text{PSPACE}$, $\text{NPSPACE} = co\text{NPSPACE}$

$\text{SPACE}(\log^2(n)) \neq \text{PSPACE}$

Defn coDTIME(T(n)) coNTIME(T(n)) coNSPACE(S(n))

Thm: coSPACE(S(n)) = SPACE(S(n)).

Thm Immerman–Szelepcsényi

$$NSPACE(S(n)) = coNSPACE(S(n)).$$

Corollary ~~NSPACE~~ NL = coNL

Pf Suffices to show coNSPACE(S(n)) ⊆ NSPACE(S(n)).

Let M be NTM that is S(n)-space bounded.

~~By~~ W.l.o.g., M(σ) runs for exactly S(|σ|) steps on all inputs σ.

We want to answer the question "~~Can M(σ) reach~~

(Q) "Is every state M(σ) reaches in exactly S(|σ|) steps accepting?"

with an NSPACE(S(n)) machine M'

s.t. M' has some accepting computation iff (Q) is "Yes".

Algorithm for M' will do essentially the following, on input σ

for i = 0, ..., S(n)

Compute: n_i := # of (distinct) configurations reachable in ~~i~~ exactly i steps by M(σ).

Let R_i be the set of configurations reachable in exactly i steps

Note $n_0 := 1$. (of course!)

Given a correct value for n_i, can compute "C ∈ R_i?" ~~is reachable in~~ by:

~~Loop~~ Init: C, Counter j = 0, i ∈ [0, S(n)], ~~ans = false~~,

Set ans = false

Loop guess configurations $C_{k_0}, C_{k_1}, \ldots, C_{k_{n_i-1}}$,

verify that $C_{k_m} < C_{k_{m+1}}$, ∀m

and that each $C_{k_m} \in R_i$ (by simulating M nondeterministically for i steps)

and set ~~ans = false if~~ ans = true if $C_{k_m} = C$ ☆

End loop If any of above conditions fail, halt + reject.

Return ans.

Alg. for C∉R_i. As above, but return "¬ans" (opposite of ans)

Given correct value for n_i can compute C∈R_{i+1} and C∉R_{i+1} by similar procedure, but replace test ☆ by

"C is reachable from C_{k_m} in one step."

OLD

Algorithm to compute n_{i+1} from n_i

Counter $k = 0$

For $C_k = 0 \ldots, d \cdot S(n)$

If "$C_k \in R_{i+1}$ with $c_i = |R_i|$", k++

If "$C_k \notin R_{i+1}$ with $c_i = |R_i|$", continue

else reject.

Algorithm to compute if all computations are ~~accepting~~ ~~rejecting~~ accepting, given $n_{d \cdot S(n)}$.

~~Loop~~, given config's $C_{k_0} \ldots C_{k_{dS(n)-1}}$

verify $C_{k_{m+1}} > C_{k_m}$, else reject

verify $C_{k_{m+1}} \notin R_i$, else reject

if C_{k_m} not accepting, reject.

End loop

Accept

QED.

Note: Lookup: Hopcroft-Paul-Valiant: $DTIME(T(n)) \subseteq SPACE(T(n)/\log T(n))$.

"On Time versus Space", J. ACM 24(2) 332-337, 1977

prelim version in FOCS '75.

Def'n: coTIME($T(n)$) coSPACE($T(n)$)
coNTIME($T(n)$). coNSPACE($T(n)$).
$coNP = \bigcup_c$ coNTIME(n^c).

Thm: coTIME($T(n)$) = TIME($T(n)$). and coSPACE($T(n)$) = SPACE($T(n)$).

Pf easy.

Corollary to Savitch: coNSPACE($S(n)$) $\subseteq$ SPACE($(S(n))^2$).

Open Problem $NP = coNP$?

Thm Immermann-Szelepcsény
NSPACE($S(n)$) = coNSPACE ($S(n)$).

Corollary: $NL = coNL$.

Pf: Suffices to show coNSPACE($S(n)$) $\subseteq$ NSPACE($S(n)$).

Let M be a NTM that is $S(n)$-space bounded.

W.l.o.g. M runs for exactly $S(n)$ steps for all inputs σ, $|\sigma|=n$.

Let $L = \{\sigma : M(\sigma)$ accepts σ on every computation path$\}$.

We want recognise L in an NTM M', i.e. an M' that runs in SPACE($S(n)$), and s.t. $\sigma \in L \iff M'(\sigma)$ accepts on some computation path.

All $M'(\sigma)$ needs is one accepting path, so it is OK if some paths reject.

~~Claim: There are uniform~~

General idea: Let n_i := # of ~~any~~ configurations that can be reached by some computation path of $M(\sigma)$ in exactly i steps

$R_i := \{C : C$ reachable in exactly i steps$\}$

$n_i := |R_i|$.

$M'(\sigma)$ will compute $n_0, n_1, n_2, \ldots$, ~~$n_{S(n)}$~~ , $n_{2^{dS(n)}}$
~~and then~~ using n_i to help compute n_{i+1}.
Then, using $n_{2^{dS(n)}}$ will decide if every state reachable in $|\sigma|$ steps is accepting.

Claim: M_1 can do any of the following tasks, in the sense that ~~Some comp~~ there is some nonrejecting path that succeeds, + every " " " succeeds
(~~but~~, paths that fail will reject ← e.g. "bad choices")

Note $n_i := |R_i|$ → Let $R_i = \{C : C$ is a configuration reachable by $M(\sigma)$ in exactly i steps$\}$.

Task 1: ~~Determine if~~ Recognize when a given configuration $C \in R_i$.

Alg: "guess" the computation

Task 2: Given a correct value for n_i, determine if $C \notin R_i$.

Alg: Input σ, n_i, C, i.

Initialize FoundC = false

Loop, ~~successively~~ guessing configs $C_0, C_1, \ldots, C_{n_i - 1}$
and checking that $C_j < C_{j+1}$ <- lex order
and $C_j \in R_i$ - if not reject ~~(...)~~
if ~~any~~ $C_j = C$, set FoundC = true

Endloop

~~Output~~

Accept iff FoundC = false.

Memory usage: index j, n_i, i - all $O(S(n))$ bits
C_j, C_{j+1}, C - each $O(S(n))$ bits

Task #3. Given a correct value for n_i, determine if $C \notin R_{i+1}$

Alg Same as above, but replace test "$C_j = C$" with "C is reachable in 1 step from C_j".

Task #4: Given n_i, compute n_{i+1}

Alg: ~~log~~ For all configurations $C_0, \ldots C_{2^{O(S(n))}-1}$ (taken in lex order)
~~find that~~
nondeterministically verify $C_j \in R_i$ or verify $C_j \notin R_i$.
Keep count of how many in R_i.

Task #5: Given $n_{O(S(n))}$ determine if all $C \in R_{O(S(n))}$ are accepting. [Alg as above, but test "C_j is rejecting"]

Alternating Turing Machines:

Two examples: An NTM has *existential* states, i.e.
two transition functions δ_0, δ_1:
Accepts iff $\exists$ a series of choices of allowable moves that leads to acceptance

A coNTIME machine is defined exactly the same, but the states are "universal". It accepts iff
$\forall$ ~~sequences~~ of choices of moves, the machine enters an accepting state.

Formally an Alternating TM. has:
finite list of states ~~parti~~tioned into
~~Accepting~~ Existential ($\exists$)
Universal ($\forall$)
Accepting } Halting
Rejecting }
– one "initial" state.
· Finite # of tapes, ~~start~~ alphabet symbols
Two transition functions δ_0, δ_1

Defn Let ATM M have input σ. Define the set of configurations as before. Define $\delta_i(C)$ = config reached by δ_i's rule

A configuration C of $M(\sigma)$ is ~~[illegible]~~ accepting iff
if C is a $\forall$-state, both $\delta_0(C)$ and $\delta_1(C)$ are accepting
if C " " $\exists$-state, one of " or " is accepting
if C is in an accepting state q_{accept}

$M(\sigma)$ is accepting iff its initial configuration accepting.

__Defn__ ATIME(T(n)) ASPACE(S(n)) [CKS]

Alternating Polynomial Time $= \bigcup_c$ ATIME(n^c).

__Thm__ NSPACE(S(n)) $\subseteq$ ATIME($S(n)^2$) $S(n) \geq n$

__Pf__ Use Savitch's Theorem's Proof. $\boxed{\text{Actually ATISP}(S(n)^2, S(n))}$

sum

__Thm__ ATIME(T(n)) $\subseteq$ SPACE(T(n)).

__Pf__ __Obvious?__ Try all possible choices in lex order.
Need to save current choice, ~~for~~
for each node along current computation path,
if taking second choice (δ_1) whether the ~~state~~ configuration
reached by the first choice (δ_0) was accepting.

__Comments:__ Wlog. the ~~graph~~ directed graph with nodes the set of configurations, and edges as defined by δ_0, δ_1 is ~~an~~ __acyclic__. Reason: With a time or space bound, can just run a "clock" and make ~~any configuration~~ the ATM enter a rejecting state if the clock runs over the allotted time

__Corollary__ Alternating PTIME = PSPACE.

Open: Can $S(n)^2$ be replaced by $S(n)$ in the above, or in Savitch's Theorem.

__Thm__ ASPACE(S(n)) $\subseteq$ DTIME($2^{O(S(n))}$)

__Pf__ Essentially same as proof that NSPACE(S(n)) $\subseteq$ DTIME($2^{O(S(n))}$).

__Thm__ DTIME($2^{O(S(n))}$) $\subseteq$ ASPACE(S(n)).

__Pf:__ ~~Configs are labelled accept/reject by checking all predecessors~~ immediate

Pairs of <time t, pos, symbol> ~~for accepting/rejecting states~~ are shown to be correct/incorrect.

Bounded Alternation / Polynomial Time Hierarchy

Def^n A ~~Σ_i-TIME~~ Σ_i-alternation TM is an alternation TM which switches from existential – universal … a total of i times.

Π_i-alternation TM defined dually.

Def^n $\Sigma_i\text{-TIME}(T(n)) = \{ L \subseteq \{0,1\}^* : L$ is accepted by a Σ_i-ATM with runtime $\le T(n)$ on all computation paths $\}$

$\Pi_i\text{-TIME}(T(n))$ defined dually.

$$\left.\begin{aligned} \Sigma_i^P &= \bigcup_{c\ge 1} \Sigma_i\text{-TIME}(n^c) \\ \Pi_i^P &= \bigcup_{c\ge 1} \Pi_i\text{-TIME}(n^c) \end{aligned}\right\} \text{ Collectively called the polynomial time hierarchy}$$

Thm $NP = \Sigma_1^P$ ~~TIME~~ $coN = \Pi_1^P$ ~~TIME~~. $\Sigma_i^P = co\Pi_i^P$

$$\begin{array}{ccc} \Pi_3^P & & \Sigma_3^P \\ \text{IU} & & \text{UI} \\ \Pi_2^P & \subseteq \supseteq & \Sigma_2^P \\ \text{IU} & \subseteq \supseteq & \text{IU} \\ \Pi_1^P = coNP & & NP = \Sigma_1^P \\ \supseteq & & \subseteq \\ & P & \end{array}$$

? of the polynomial time hierarchy.

Open: Is $P = NP \cap coNP$.

Alternate Def^n of Σ_i^P / Π_i^P

Thm $L \in \Sigma_i^P$ iff $\exists$ polys $p_1, \ldots, p_i$ and a polynomial time $R(\ )$

s.t. $\forall \sigma\ (\sigma \in L \iff \exists u_1, |u_1| \le p_1(|\sigma|)\ \forall u_2, |u_2| \le p_2(|\sigma|) \ldots Q_i u_i, |u_i| \le p_i(|\sigma|)$ $R(\langle \sigma, u_1, \ldots u_i \rangle)$.

+ dually for Π_i^P.

Pf Like before, for $NP = \bigcup_{c\ge 1} \Sigma_1\text{-TIME}(n^c)$.

Oracle TM's:

Recall a language L is a subset of $\{0,1\}^*$.

Def'n A oracle Ω is a subset of $\{0,1\}^*$.

An oracle Turing Machine, written M^{Ω}, (Ω may be just a placeholder or might be a particular oracle)

is a TM augmented with a

special "query" state q_{Ω} and a designated "oracle query tape" and two designated oracle answer states, q_{Yes}, q_{No}.

Whenever M enters state q_{Ω} the oracle tape ~~head~~ tapehead is

over some string $\sigma \in \{0,1\}^*$ (delimited by non-0,1's).

The next step of M places M in either

q_{Yes} if $\sigma \in L$

or q_{No} if $\sigma \notin L$

and M's tape contents + tape-head positions unchanged.

Meta-def'n All our earlier def'ns ~~now~~ extend to oracle Turing machines, as do all proved theorems so far!

The notations extend to complexity classes too:

For example

$$P^{NP} = \{L : \text{for some OTM } M^{\Omega} \in \bigcup_{c \geq 1} DTIME(n^c), \text{ and some } L' \in NP, \; L = M^{L'} = \{\sigma \in \{0,1\}^* : M^{L'}(\sigma) \text{ accepts}\}\}.$$

Thm $NP \subseteq P^{NP}$, $coNP \subseteq P^{NP}$

Thm $P^{NP} = P^{coNP}$

Thm $P^{NP} \subseteq \Sigma_2^P \cap \Pi_2^P$

Pf It suffices to prove $P^{NP} \subseteq \Sigma_2^P$ (since P^{NP} is closed under complementation and $\Sigma_2^P = co\Pi_2^P$).

Let $L \in P^{NP}$, say accepted by Polytime M^{Ω} with $\Omega \in NP$.

Here $\Omega = \{\sigma : \exists u, |u| \le |\sigma|^c + c, N(\langle\sigma, u\rangle)\}$ for some $N \in$ Ptime.

Consider σ input to M^{Ω} (to decide if $\sigma \in L$).

The computation of $M^{\Omega}(\sigma)$ is a series of configurations,

$$C_0, C_1 \ldots \quad C_i \xrightarrow{\text{oracle } \Omega} C_{i+1} \qquad C_{p(n)}$$

where p is a polynomial in $n = |\sigma|$.

At certain steps, there is an oracle call, i.e. C_i string σ_i on the oracle query tape and C_{i+1} is in state q_Y, q_N depending whether $\sigma_i \in \Omega$, i.e whether there is a u_i s.t. $|u_i| < |\sigma_i|^c + c$ and $N(\langle\sigma, u\rangle)$.

Σ_2^P-algorithm for L:

Input σ. Goal: decide if $\sigma \in L$.

Existentially guess full computation $C_0, \ldots C_{p(n)}$,
and for each oracle query u_i in config C_i
that has C_{i+1} in state q_{Yes}, guess ~~the~~
a value u_i; $|u_i| \le |u_i|^c + c$.

Deterministically verify that these u_i's satisfy $N(\langle\sigma_i, u_i\rangle)$
reject if any one fails.

Universally pick u_i's for the remaining oracle queries,
i.e., for each C_i, C_{i+1} query with C_{i+1} in q_{No},
verify that $N(\langle\sigma_i, u_i\rangle)$ is false.
If any fails to be false, reject

Deterministically verify $C_0 \ldots C_{p(n)}$ is a correct computation
based on the oracle query answers

Remark: Total runtime is $\le p'(n)$ for some polynomial p'.

$$
\begin{array}{ccccc}
 & & \rotatebox{45}{$\subseteq$} & & \\
 & & P^{\Sigma_2^p} = P^{\Pi_2^p} & & \\
 & \rotatebox{45}{$\subseteq$} & & \rotatebox{-45}{$\supseteq$} & \\
\Pi_2^p & & & & \Sigma_2^p \\
 & \rotatebox{-45}{$\supseteq$} & & \rotatebox{45}{$\subseteq$} & \\
 & & P^{NP} & & \\
 & \rotatebox{45}{$\subseteq$} & & \rotatebox{-45}{$\supseteq$} & \\
\Pi_1^p = coNP & & & & NP = \Sigma_1^p \\
 & \rotatebox{-45}{$\supseteq$} & & \rotatebox{45}{$\subseteq$} & \\
 & & P & &
\end{array}
$$

<u>Thm</u> If $P = NP$, then $\forall i$, $\Sigma_i^P = \Pi_i^P = P$, i.e. PH$\downarrow$

<u>Thm</u> If $NP = coNP$, then PH$\downarrow$, $PH = NP$.

<u>Pf</u>: Any predicate in PH is in some Σ_i^p and can be written as

$$\sigma \in L \iff \Big\{ \exists u_1, |u_1| \le p_1(|\sigma|) \ldots (\forall u_{k-1}, |u_{k-1}| \le \ldots)(\exists u_k\, |u_k| \le p_k) N(\sigma, \vec{u}) \Big\}$$

for some PTIME N.

If $NP = coNP$, the NP predicate

$M(\sigma, u_1 \ldots u_{k-1}) := (\exists u_k, |u_k| \le p_k(|\sigma|)) N(\sigma, u_1 \ldots u_{k-1}, u_k)$ can be rewritten as a coNP predicate

$$\forall v_k, |v_k| \le p'_k(|\sigma| + \textstyle\sum |u_1| \ldots |u_{k-1}|)\ N'(\sigma, u_1 \ldots u_{k-1}, v_k).$$

Then

$$\sigma \in L \iff \exists u_1 \ldots \quad (\forall \langle u_{k-1}, v_k \rangle, |\langle u_{k-1}, v_k \rangle| \le p^*_{k-1}(|\sigma|)) N'(\sigma, \vec{u}, v_k)$$

Then can use induction on i.

Complete Problems:

ⓐ Canonical example:

SAT is NP-complete.

Def'n SAT is the following decision problem:

Input: $r \in \{0,1\}^*$ encoding a set of clauses

A literal is x_i or $\overline{x_i}$

A clause is a set of literals

A truth assignment $\tau: \{x_i\}_i \to \{T, F\}$ $\tau(\overline{x_i}) = \neg\tau(x_i)$

$\tau \models C$, τ makes C true or satisfies C if $\tau(\ell) = T$ for some $\ell \in C$.

$\tau \models \Gamma$, Γ a set of clauses, iff $\tau \models C$, $\forall C \in \Gamma$.

I.e. Γ is a CNF formula (an $\wedge$ of $\vee$'s of literals)

Thm: For all $L \in NP$, $\exists$ a ptime computable f,

s.t. $\forall x (x \in L \Leftrightarrow f(x) \in SAT)$.

Pf: See Garey-Johnson. For input of size n to DTM, runtime + space bdd by functions of n

Salient ideas: Use variables $X_{i,p,t,\sigma}$

to mean "Tape #i, position p, contains symbol σ at time t."

$Z_{q,t}$ - "M is in state q at time t"

$Y_{i,p,t}$, "Tape i's head is at posit p at time t"

$u_i \Leftrightarrow$ i-th bit of input is a 0/1.

To discuss deterministic computation

Each $X_{i,p,t+1,\sigma}$, $Z_{q,t+1}$, $Y_{i,p,t+1}$ depends on only finitely many $X_{i,p,t,\sigma}$'s, $Z_{q,t}$, $Y_{i,p,t}$'s.

$X_{i,p,0,\sigma}$, $Y_{i,p,0}$, $Z_{q,0}$ have definite values depend on only finitely many bits of input u to M.

Thus uniquely describe by finite size formula, Convert to CNF form.

W_t - M accept at time t. $W_t \Leftrightarrow$ accept at t

Then $u \in L \Leftrightarrow \underbrace{\exists v, |v| \le p(|u|)}_{\text{encode with polysize formula}} M(u,v)$.

$u \in L \Leftrightarrow f(u) \in SAT$

$f(u)$ encodes formula expressing all of the above.

Q.E.D. (handwavy)

Comments f can actually be logspace computable.

(Explain what this means).

This known as "many one complete" or "Karp-complete".

Comment: Also 3-SAT

Defn CVP, Circuit Value Problem. - Inputs 0/1 (Not variables)
- Built from $\wedge, \vee, \neg, 0, 1$.

Thm CVP $\in$ P. $\boxed{\text{Arora-Barak call it "Circuit Eval"}}$

Thm $\forall L \in P\ \exists f$, computable in logspace such that

$$\forall u\,(u \in L \Leftrightarrow f(u) \in CVP).$$

Pf Like above, but $X_{i,p,t+1}$, $Z_{q,t+1}$, $Y_{i,p,t+1}$ are given by fixed size circuits of finitely many of the values at time t. Put these together to compute Z_{accept}, T_{max}.

qed

With care, f is logspace computable

Defn L is $\mathcal{C}$-complete (many-one $\mathcal{C}$ complete) under logspace reductions iff $L \in \mathcal{C}$ and $\forall L' \in \mathcal{C}\ \exists f \in \text{logspace}\ \forall u\,(u \in L' \Leftrightarrow f(u) \in L)$.

Def'n $L \subseteq \{0,1\}^*$ has circuits of size $S(n)$, provided,

for all $n \geq 1$, $\exists$ Boolean circuit C_n

a) with variables $x_1 \ldots x_n$; gates $\wedge, \vee, \neg$;

b) a single output signal

c) For all $x_1 \ldots x_n$, $x_1 \ldots x_n \in L \Leftrightarrow C_n(x_1, \ldots, x_n) = 1$

d) For all n, $\mathrm{size}(C_n) \leq S(n)$.

Thm ~~Def'n~~ P has polynomial size circuits.

Pf ~~I~~ Immediate from the above construction

Def'n P/poly = ~~class~~ of languages L with polynomial size circuits.

A complete problem for NL

Path = { directed graph $G = (V, E)$ and designated nodes s, t :
there is a path in G from s to t }.

Thm Path $\in$ NL

NL algorithm: Start at s, nondeterministically choose a neighbor s';
set $s := s'$
Repeat, until $s = t$, then halt & accept.

Thm Path is NL-complete.

Pf If $L \in NL$, accepted by machine M for input σ,

let G = configuration graph of $M(\sigma)$, ← size $= 2^{O(\log |\sigma|)}$
s = initial configuration
t = final accepting configuration

Note $(G, s, t) \in$ PATH iff $M(\sigma)$ accepts.

Define $f_L : \sigma \mapsto (G, s, t)$.

Remark: G depends on σ in this construction. (How could it not?)

QED.

A complete problem for PSPACE

Def'n QBF = Quantified Boolean Formulas ← propositional formula, no quantifiers

$= \{$ formulas $\psi := Q_1 x_1\, Q_2 x_2 \ldots Q_k x_k\, \varphi(x_1 \ldots x_k)$
built from $\forall, \exists, x_i, \vee, \wedge, \neg, T, F$ (optional) :
ψ is true $\}$.

Thm QBF $\in$ PSPACE.

Thm: If $L \in$ (N)PSPACE, then $\exists$ logspace f, $\forall \sigma\ (\sigma \in L \Leftrightarrow f(\sigma) \in QBF)$.

Pf Mimic proof of Savitch's Thm. Note quadratic blowup in size.

Randomized Computation / Probabilistic Complexity Classes

Replace nondeterminism/conondeterminism

A probabilistic TM has two transition functions δ_0, δ_1.
At each step, it chooses to use δ_0 or δ_1 with probability $\frac{1}{2}$ each.
The machine (wlog) always halts and ~~outputs~~ Accept or Rejects (outputs 1 or 0).
Runtime $T(n)$, space $S(n)$, etc, defined as before.

Def'n $PP = \{L : \forall\sigma,\ \sigma \in L \Leftrightarrow Pr[M(\sigma)=1] \geq \frac{1}{2}\}$
for some M that runs in polynomial time

Equivalent defn:

$L \in PP \Leftrightarrow \exists$ poly time probabilistic TM M and a polynomial $p(n)$ s.t.

$$\forall\sigma\left[\sigma \in L \Leftrightarrow \left|\{u \in \{0,1\}^{p(|\sigma|)} : M(u,\sigma) \text{ accepts}\}\right| \geq 2^{p(|\sigma|)-1}\right\}$$

Pf: Easy.

Def'n: $BPTIME(T(n))$ is the set of languages L s.t. there is a probabilistic TM that runs in time $O(T(n))$ s.t.

$\forall\sigma$: $\sigma \in L \Rightarrow Pr[M(\sigma)=1] \geq 2/3$.

$\sigma \notin L \Rightarrow Pr[M(\sigma)=0] \geq 2/3$.

R'k: For PP, it is $\geq 1/2$ and $< \frac{1}{2}$.

Defn $BPP = \bigcup_c BPTIME(n^c)$

Def'n $RTIME(T(n))$ = set of $\{$languages L s.t. $\exists$ prob. TM M, runtime $O(T(n))$ s.t.

$\forall\sigma$: $\sigma \in L \Rightarrow Pr[M(\sigma)=1] \geq 2/3$

$\sigma \notin L \Rightarrow Pr[M(\sigma)=1] = 0$ (so $Pr[M(\sigma)=0]=1$.)

$RP = \bigcup_{n^c} RTIME(n^c)$.

"two-sided error"

Intuitions for BPP - Good probability of getting right answer

"one-sided error" RP - " " " and if answer $=1$, you are sure it's correct

"zero-sided error" $RP \cap coRP$ - " " " getting answer, and when you get it, you are sure it is correct.

PP - too delicately poised to get useful information (as far as we know!).

Thm ~~BPP~~ $RP \cap coRP \subseteq BPP \subseteq PP$

Pf Easy.

Z: "Zero"

Defn $ZPP(T(n))$ set of languages L s.t. there is a probabilistic Turing machine M s.t. ① $\forall \sigma$, $M(\sigma)$ halts with probability 1 and $M(\sigma) = L(\sigma)$

② For all σ, expected runtime of $M(\sigma)$ is $O(T(n))$.

Theorem: $ZPP = RP \cap coRP$.

Pf $\Leftarrow$ Run $RP \cap coRP$ repeatedly until answer is obtained.

Expected runtime $\leq$ ~~$\frac{2}{3}T(n) + \frac{1}{3}\left[T(n) + \frac{2}{3}T(n) + \frac{1}{3}\left[\frac{2}{3}T(n) + \cdots\right.\right.$~~

~~$= \frac{2}{3}T(n)\left(1 + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} + \cdots\right)$~~

$$\leq \frac{2}{3}T(n) + \frac{1}{3}\left(T(n) + \left[\frac{2}{3}T(n) + \frac{1}{3}\left[T(n) + \left[\frac{2}{3}T(n) + \frac{1}{3}\left[\cdots\right.\right.\right.\right.\right.$$

$$= \frac{2}{3}T(n) + \frac{1}{3}\left(\frac{5}{3}T(n) + \frac{1}{3}\left(\frac{5}{3}T(n) + \cdots\right.\right.$$

$$= \frac{5}{3}T(n)\left(1 + \frac{1}{3} + \frac{1}{3^2} + \frac{1}{3^3} + \cdots\right) - T(n)$$

$$= \frac{3}{2}T(n).$$

$\frac{2}{3}\cdot 0 + \frac{1}{3}\cdot R =$

$\Rightarrow$ If a ZPP has expected time $\leq T(n)$, it halts in time $\leq 3(T(n))$ with probability $\geq \frac{1}{3}$. Thus it is in $RP(3T(n)) = coRP(3T(n))$

$\geq 1 - \frac{1}{3} = \frac{2}{3}$ $\qquad = RP(T(n)) \cap coRP(T(n))$.

Markov inequality
For $X \geq 0$ $Pr[X \leq k\mu] \geq \frac{1}{k}$

Examples of Probabilistic Algorithms (see Arora-Barak pp 126-129

(1) Find the median element; or Finding the k-th element of a list

Algorithm Input array $A = a_0 \dots a_{n-1}$, and $k \in [0, n-1]$.

Find k-th(A, k):

Select $i \in [0, n-1]$ at random.

Scan A, count # N of a_j's with $a_j \le a_i$.

If $N \ge k$,

Scan again, creating new list B of the N elements $\le a_i$

Return Find k-th(B, k).

Else

Scan again, creating new list B of $n-N$ elements $> a_i$.

Return Find k-th$(B, k-N)$.

Note small fix needed if elements are not distinct

This is the most efficient algorithm known.

Claim: Expected running time is $\Theta(n)$.

Pf Intuition is that $|B| \le \frac{3}{4}|A|$ with probability $\ge \frac{1}{2}$ (say)

If this happens ~~exactly~~ every other time, recursion halts after

~~two~~ $2 \log_{4/3} |A|$,

Running time is $O(|A|) +$ time for recursive calls

i.e., $O(|A| + \frac{3}{4}|A| + (\frac{3}{4})^2|A| + \dots) = O(A)$.

More generally if you do $4 \log_{4/3} |A|$ iterations, you have $> \log_{4/3} |A|$ successes w/ high probability (close to 1).

So running time is

$$\le \left[\left(4 \log_{4/3} |A|\right)(1-\varepsilon) + \varepsilon\left[4 \log_{4/3} |A| (1-\varepsilon) + \varepsilon[\cdots]\right]\right]$$

$$= O(|A|).$$

Not good enough! Faulty analysis!

$4 \log_{4/3} |A| \dots \log |A| \dots$ (log's are base 2).

Probabilistic Primality Testing (Solovay–Strassen)

Problem

Input: N

Output: Accept if N is a composite (non-prime), with high probability.

Define $QR_N(A) = \begin{cases} 0 & \text{if } \gcd(A,N) \neq 1 \quad \rightarrow \text{implies } N \text{ is composite for } A \in [2, N-1] \\ +1 & \text{if } A = B^2 \pmod N \text{ for some } B \\ -1 & \text{otherwise.} \end{cases}$

Fact: $QR_N(A) = A^{(N-1)/2}$ for odd prime N, and hence is computable in polynomial time.

Defn Jacobi symbol $\left(\frac{N}{A}\right) := \prod_{P_i} QR_{P_i}(A)$ where P_i ranges over prime factors (with repetitions) of N.

Fact $\left(\frac{N}{A}\right)$ is computable in polynomial time (by simple recurrence)

Fact: If N is prime, $\left(\frac{N}{A}\right) = QR_N(A)$ for all $A \in [2, N]$

If N is not prime, at most half of A's satisfy this.

Algorithm A_1

Input N.

Pick $A \in [2, \ldots N-1]$ at random

If $\left(\frac{N}{A}\right) = QR_N(A)$, reject

If $\left(\frac{N}{A}\right) \neq QR_N(A)$, accept

A_1 - ~~Pr[A]~~ If N composite, $Pr[A_1] = 1 > 1/2$

If N prime, $Pr[A_1] = 1 = 0$

Repeat test k times, probability rises from $\frac{1}{2}$ to $1 - \left(\frac{1}{2}\right)^k$

Polynomial algorithm can repeat the test $k = p(n)$ times, $n = |N|$.

So probability of error shrinks to $\left(\frac{1}{2}\right)^{p(n)}$ i.e. 2^{-n^c} for any desired fixed c.

Agarwal-Kayal-Saxena

— Ptime algorithm for primality testing

Polynomial Identity Testing (Has no known efficient p-time algorithm!)

Input $p(\vec{x}) \in \mathbb{Z}[\vec{x}]$ - integer coefficients.

$\Rightarrow$ Specified by an algebraic circuit or straight line program.

with inputs $x_1 .. x_n$, ~~outp~~ and a single output

Goal Decide if $p(\vec{x}) = 0$ for all $\vec{x} \in \mathbb{Z}$. [Some coeff is nonzero]

Schwartz-Zippel Lemma: Let $p(\vec{x})$ be a non-zero polynomial (in n variables) have total degree $\le d$,

Let ~~$S \subseteq N$~~ $S \subseteq \mathbb{Z}$ be finite. Then

$$\Pr_{a_1..a_n \in S}\left[p(\vec{a}) = 0\right] \le \frac{d}{|S|}.$$

Rk: Independent of n!

Pf: By induction on n.

Base case: n=1. A degree d polynomial has $\le d$ roots (in $\mathbb{Q}$ or $\mathbb{R}$ or $\mathbb{C}$).

Induction Step: Let ~~total degree be d+1~~ number of variables be n+1.

Write $p(\vec{x}) = \sum_{k=0}^{d} x_{n+1}^k \, p_k(x_1..x_n)$, each p_k of degree $\le d-k$.

Choose max k such $p_k(x_1..x_n)$ is non-zero.

By ind. hyp $\Pr_{a_1..a_n \in S}\left[p_k(a_1..a_n) = 0\right] \le \frac{d-k}{|S|}$

For each fixed $a_1..a_n$, with $p_k(\vec{a}) \ne 0$, we get a fixed polynomial $P_{a_1..a_n}(x_{n+1})$ of degree k in x_{n+1}. If $a_{n+1} \in S$ is chosen at random, $\Pr\left[P_{a_1..a_n}(a_{n+1}) = 0\right] \le \frac{k}{|S|}$

Thus $\Pr\left[p(a_1..a_{n+1}) = 0\right] \le \frac{d-k}{|S|} + \left(1 - \frac{d-k}{|S|}\right)\left(\frac{k}{|S|}\right) \le \frac{d}{|S|}.$

Schwartz-Zippel also holds over $GF(q)$, with $S \subseteq GF(q)$, $(d < q)$

$\Rightarrow$ Same proof works.

Input Polynomial p, given by circuit/straightline program of size m, inputs $x_1 \ldots x_n$. constants 0, 1. $n \le m$
Goal Is $p(\vec{x}) = 0 \ \forall z \in \mathbb{Z}$?

Algorithm: If p computed by a circuit of size m, then it has total

degree $< 2^m$ (pf degree at most doubles at each gate.)

$4 \cdot 2^m$ Pick $S = \{1, \ldots 2^{m+2}\}$. Idea: Choose $a_1 \ldots a_n \in S$ at random accept if value $= 0$.

So $p(\vec{a}) = 0$ $\le \frac{1}{4}$.

For values $a_1 \ldots a_n \in S$, $p(a_1 \ldots a_n)$ can be bounded by $(2^{m+2})^{2^m} \le 2^{m 2^m} = 2^{2^{m \log m}}$

These values unfortunately require $2^{m \log m}$ bits just to recognize and so cannot be written out in polynomial time.

$2^{(m+2)2^m} = 2^{2^{m \log m}}$

To fix this, we choose $a_1 \ldots a_n \le 2^{m+2}$ at random, and a q at random $\in [2, 2^{2m}]$. ← (Important: exponent is $2m$, not $m+1$).

Evaluate $p(\vec{a}) \bmod q$: if non-zero, reject. If zero accept.
If $y = p(\vec{a})$ is nonzero and
q is prime and $q \nmid y = p(\vec{a})$, then $p(\vec{a}) \bmod q \ne 0$.

What is the probability q is prime? $\Omega\left(\frac{1}{m}\right)$ by the prime number theorem ($\#$ of primes $\le 2^{2m}$ is $\Omega(\ln(2^{2m})) = \Omega(m)$.

$\frac{2^{m \log m}}{2^{2m}}$

What is the probability $q \mid y$? Ans: $\le \frac{2^{m \log m}}{2^{2m}}$ since $y \le 2^{2^{m \log m}}$.

So Prob. q is not prime or $q \mid y \le \frac{1}{m} + \frac{1}{2^{m \log m}} = \Theta(1)$.

So if ~~$p \ne 0$~~, $p(x)$ is not identically zero, then for $\vec{a} \in [4 \cdot 2^m]$, $q < 2^{2m}$, $\operatorname{Prob}(q \text{ is prime } \& \ q \nmid y) = \Omega\left(\frac{1}{m}\right) - \frac{2^{m \log m}}{2^{2m}} = \Omega(1/m)$

Now Repeat $\Theta(m)$ times!!

$$\Pr[p(\vec{a}) \bmod q \ne 0] = \Omega\left(\frac{1}{m}\right)\frac{3}{4} = \Omega(1/m)$$

Thus the $\{p(x) : p(x)$ is identically zero, p given by a circuit/straightline program$\}$ is in coRP.

I.e. ZEROP $\in$ [redacted] coRP. (1) If $p \equiv 0$ $\Pr[M(p) = \text{accept}] = 1$
If $p \not\equiv$ identically zero $P[M(p) = \text{accept}] < \frac{1}{4} + o(1)$.

Probability amplification

Obviously the "$\frac{4}{4}+o(1)$" above was not optimal,

e.g. run the same test m times (say), reduces the probability of an error from $<\frac{1}{3}$ to $<\left(\frac{1}{3}\right)^m$, i.e. exponentially small probability of error.

Basic Probability Results

$\alpha = k\mu$

$\frac{1}{k} = \frac{\mu}{\alpha}$

(1) Markov's inequality.

a) For X a random variable, taking ~~posi~~ non-negative values only,

$$\Pr[X \geq k\mu] \leq \frac{1}{k}$$, when $\mu =$ mean(X). So $\Pr[X \geq \alpha] \leq \frac{\mu}{\alpha}$

(2) Chebyshev Inequality: Let X have mean μ, std dev. σ

$$\Pr[\,|X - E(X)| \geq k\sigma\,] \leq \frac{1}{k^2}.$$

So $\Pr[X - E(X) \geq \alpha] \leq \left(\frac{\sigma}{\alpha}\right)^2$

Pf Recall

Variance $Var(X) = E((X-\mu)^2)$

$\alpha = k\sigma$

$k = \alpha/\sigma$

~~where~~

Std Dev. $= \sqrt{Var(X)}$. So $\sigma^2 = Var(X) = E((X-\mu)^2)$.

By Markov's Inequality, $\Pr[\,(X-\mu)^2 \geq k\sigma^2\,] \leq \frac{1}{k}$

So $\Pr[\,|X-\mu| > \sqrt{k}\,\sigma\,] \leq \frac{1}{k}$

Now do a change of variables, replacing $\sqrt{k}$ with k.

(3) **Chernoff bounds:** If $X_1, X_2, \ldots X_n$ are mutually independent, random variables, taking values in $\{0,1\}$, then, for $\delta > 0$,

$$\Pr\left[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq \left(e^{-\delta^2/4}\right)^{\mu} \text{ for } \delta < \tfrac{1}{2}.$$

$$\Pr\left[\sum_{i=1}^{n} X_i \leq (1-\delta)\mu\right] \leq \left(\frac{e^{\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} \leq \left(e^{-\delta^2/4}\right)^{\mu}.$$

where $\mu = \sum \mu_i$, $\mu_i = E(X_i)$.

pairwise ~~independence~~

Probability Amplification

Done this already

Let $RP[p] = \{L :$ for some ptime TM M,
$\sigma \in L \Rightarrow Pr[M(\sigma) \text{ accepts}] \geq p.$
$\sigma \notin L \Rightarrow Pr[M(\sigma) \text{ accepts}] = 0 \}$

Thm $RP[1 - \frac{1}{2^{n^c}}] = RP.$ for all $c > 0.$

Pf ~~Let~~ $\subseteq$ is obvious.

$\supseteq$: Let $L \in RP$ + $M(\sigma)$ accept with prob. $> 2/3$ if $\sigma \in L$.

Run $M(\sigma)$ k times, accept if any one iteration accepts

Probability, for $\sigma \in L$, that it does not accept $\leq \cancel{\tfrac{1}{3}k}\ (\frac{1}{3})^k$

Take $k = \lceil \log_3 (2^{n^c}) \rceil$ $\cancel{\leq \log_2(n^c)}$ $\leq n^c.$

~~Thm~~ Fed. $\forall c \in [0,1]$, $RP[c] = RP$. ~~Pf~~ By the same pf.

Thm: $RP(\frac{1}{n^c}) = RP$ for all $c > 0.$

$\supseteq$ is obvious

$\subseteq$: Take $L \in RP(\frac{1}{n^c})$ accepted by RP machine M w/ probability $1/n^c$.

Iterate $M(\sigma)$ n^c times.

Prob of (erroneous) non acceptance is
$$\leq \left(1 - \frac{1}{n^c}\right)^{n^c} \leq \frac{1}{e}$$

Lemma $(1 - \frac{1}{a})^a < e^{-1}$

Pf $a \ln(1 - \frac{1}{a}) \overset{?}{<} -1$

$a \ln(1 - \frac{1}{a}) \leq a(-\frac{1}{a}) = -1.$ ✓

Probability Amplification for BPP.

Define $BPP[p]$ similarly, now $p > 1/2$ is required.

Thm 1 $BPP[\frac{1}{2}+\frac{1}{n^c}] = BPP$.

Pf $\supseteq$ obvious

$\subseteq$ ~~easy~~ easy Chebyshev bounds is good enough.

Thm 2 $BPP[1-\frac{1}{2^{n^c}}] = BPP$

$\subseteq$ obvious.

$\supseteq$: Need Chernoff bounds. (!)

Pf of Thm 1: Given $L \in BPP[\frac{1}{2}+\frac{1}{n^c}]$ with M s.t. $\sigma \in L \Rightarrow Pr[M(u,\sigma)=1] \geq \frac{1}{2}+\frac{1}{n^c}$
$\sigma \notin L \Rightarrow Pr[M(u,\sigma)=0] \leq \frac{1}{2}+\frac{1}{n^c}$.

Algorithm: Run M k-times, take majority answer.

We want to determine k:

For $\sigma \in L$, Consider random variable $X_i = \begin{cases} 1 & \text{if } M(u,\sigma)=1 \\ 0 & \text{if } M(u,\sigma)=0 \end{cases}$

So $Pr[X_i]=1 = p_i \geq \frac{1}{2}+\frac{1}{n^c}$, $Pr[X_i]=1 = 1-p \leq \frac{1}{2}-\frac{1}{n^c}$.

!

$\mu_X := E[X_i] = p$ $Var(X) = E(X_i-\mu) = pq$

Let $X = \sum_{i=1}^{k} X_i$.. $E[X] = k\cdot p$ $Var(X) = k\cdot pq$ (by pairwise independence)

$\sigma_X := \text{Std Dev}(X) = \sqrt{kpq} \leq \sqrt{k(1+\frac{1}{n^c})(1-\frac{1}{n^c})}$

Sign

Prob. Algorithm accepts $= Pr[X \geq k/2]$

By Chebyshoff, $Pr[X \geq k/2] = Pr[X-kp \geq \frac{k}{2}-kp = k(\frac{1}{2}-p)] = Pr[X-kp \geq k\frac{1}{n^c}]$

$\leq Pr[|X-E(x)| \geq k\cdot\frac{1}{n^c}] \leq \frac{\sigma_X}{k\cdot 1/n^c} =$

$Pr[X \leq k/2] \leq$ $\leq Pr[|X-E(x)| \geq k(p-\frac{1}{2})] \leq \left(\frac{\sigma_X}{k(\frac{1}{2}-p)}\right)^2 = \frac{1}{k}\frac{pq}{(\frac{1}{2}-p)^2} \leq \frac{1}{k}\cdot\frac{1}{1/n^c} = \frac{n^c}{k}$

Taking $k = 4n^c$ (say) reduces this probability to $\leq 1/4 < \frac{1}{3}$.

$0 \quad \frac{2}{3} \quad 1 \qquad \frac{1}{2} = \frac{3}{4}\cdot\frac{2}{3} = 1-$

Pf of Thm 2:

Let L, M, k, p, q be as above, ~~now p≥~~ mean $p \geq 2/3$ if desired.

By Chernoff Bounds, (+ since X_i's are mutually independent.)

$$\Pr\left[X \leq \tfrac{1}{2}\right] \leq \Pr\left[X \leq \left(1-\tfrac{1}{4}\right)\tfrac{2}{3}k\right] = \Pr\left[X \leq (1-\delta)\,E(X)\right] \qquad \text{when } \delta = \tfrac{1}{4}.$$

$$\leq \left(e^{-\delta^2/4}\right)^{E(X)} = \left(\left(e^{-1/64}\right)^{p}\right)^{k} = \left(e^{-p/64}\right)^{k}.$$

$$\leq \left(e^{-1/96}\right)^{k} = d^k \qquad \text{with } d \in [0,1).$$

Choose k so that $d^k < \frac{1}{2^{n^c}}$, i.e. $k = \underbrace{\log\left(\tfrac{1}{d}\right)}_{\text{constant}} \cdot n^c$

$$k < \log_c\left(\frac{1}{2^{n^c}}\right) = \underbrace{(\log_2 c)}_{\text{fixed constant}}\, n^c.$$

$$d^k = 2^{(\log d)k} < \frac{1}{2^{n^c}}$$

$$(\log d)\, k < -n^c$$

$$k > \left(\frac{-1}{\log d}\right) n^c \qquad (\text{Note } \log d \text{ is } < 0)$$

$$k > \left(\log \tfrac{1}{d}\right) \cdot n^c$$

Corollary ~~BPP~~ BPP $\subseteq$ P/poly.

Pf: Let $L \in BPP[1-\frac{1}{2^{n^2}}]$, accepted by machine M

For any $u \in \{0,1\}^n$, call $v \in \{0,1\}^{p(n)}$ "good for u" if $M(u,v)=1 \Leftrightarrow u \in L$.

For any u, $Pr[v \text{ is good}] \geq 1-\frac{1}{2^{n^2}}$. $Pr[v \text{ is not good}] \leq \frac{1}{2^{n^2}}$.

Lemma If $\forall u \in S_1$, $\Pr_{v\in S_2}[E(u,v)] \geq p$; then $\exists v \in S_2$ $\Pr_{u \in S_1}[E(u,v)] \geq p$.

Pf Given $\forall u$ $\sum_{v \in S_1} E(u,v) \geq p \cdot |S_1|$

Thus: $\sum_{u \in S_0} \sum_{v \in S_1} \chi(u,v) \geq p \cdot |S_1| \cdot |S_0|$

i.e. $\sum_{v \in S_1} \sum_{u \in S_0} \chi(u,v) \geq (p \cdot |S_0|)|S_1|$

So, for some v, $\sum_{u \in S_0} \chi(u,v) \geq p \cdot |S_0|$. i.e. $\Pr_{u \in S_0}[E(u,v)] \geq p$

qed

Thus, $\exists v \in \{0,1\}^{p(n)}$ $\Pr[v \text{ is good for } u : u \in \{0,1\}^n] \geq 1-\frac{1}{2^{n^2}}$

$\Pr_{u\in\{0,1\}}[v \text{ is not good for } u] \leq \frac{1}{2^{n^2}}$.

Since, there are only 2^n u's, this means, v is good for all u's.

P/poly algorithm

Let $f(n)$ = some v good for all $u \in \{0,1\}^n$.

Then $u \in L \Leftrightarrow M(u, f(u)) = 1$.

qed.

Note in fact (Markov's inequality) that the fraction of v's which are good for all u's is $\geq 1 - \frac{2^n}{2^{n^2}} = 1-2^{-(n^2-n)}$.

I.e. a random $v \in \{0,1\}^{p(n)}$ is good for all $u \in \{0,1\}^n$ with high probability.

Def'n $\#P$ is the set of functions of the form

$$f(x) = \# u \in \{0,1\}^{p(|\sigma|)} \; M(u,x) \text{ accepts}.$$

Thm: $P^{PP} = P^{\#P}$.

~~Pf~~ · Def'n A $P^{\#P}$ machine can query a single $\#P$ function.

A P^{PP} machine can query ~~the~~ a single (parametrized) PP fn'l, i.e., a ~~function~~ language of the form

$$L = \{\sigma : \Pr[M(\sigma) \text{ accepts}] \geq \tfrac{1}{2}\}$$
$$= \{\sigma : (\# u \in \{0,1\}^{p(|\sigma|)} \; M(\sigma,u) \text{ accepts}) \geq \tfrac{1}{2} \cdot 2^{p(|\sigma|)}.$$

Pf: $P^{PP} \subseteq P^{\#P}$ is pretty obvious.

$\supseteq$: Suppose $L \in P^{\#P}$, accepted by a p-time procedure that makes calls to

$$f(\sigma) := \# u \in \{0,1\}^{p(|\sigma|)} (M(u,\sigma)=1)$$

Define a PP problem L' by: for $w \in \{0,1\}^{p(|w|)}$, $u \in \{0,1\}^{p(|\sigma|)+1}$,

$$N(\langle \sigma, w \rangle, u) = \begin{cases} 1 & \text{if } u = 0u' \text{ \& } M(\sigma,u')=1 \\ 1 & \text{if } u = 1u' \text{ \& } u' <_{lex} w \\ 0 & \text{o/w}. \end{cases}$$

For $w =$ binary encoding of j,

$$\langle \sigma, w \rangle \in L' \iff j + \underbrace{(\# u \in \{0,1\}^{p(|\sigma|)} \; M(u,\sigma)=1)} \geq 2^{p(|n|)}$$

i.e. $\quad (\quad) \geq 2^{p(|n|)} - j$.

Thus can binary search using calls to L', to determine the exact # of $u \in \{0,1\}^{p(|\sigma|)}$ $M(u,\sigma)=1$.

Thm Sipser-Gács [Sipser '84?].

$BPP \subseteq \Sigma_2^p \cap \Pi_2^p$

Pf: Let $L \in BPP[1-\frac{1}{2^n}]$. Suffices to show $L \in \Sigma_2^p$.

By defn of $BPP[1-\frac{1}{2^n}]$, $\exists$ PTM M, and ~~some~~ $c>0$, M is polynomial time.

$\forall u,\ u \in L \Leftrightarrow \Pr_{v \in \{0,1\}^{n^c}} M(u,v)=1 > 1-2^{-n}$, $n=|u|$.

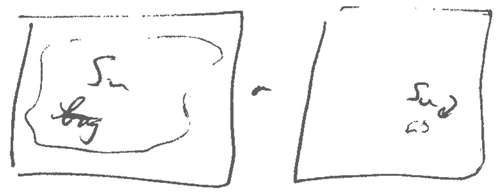
$\forall u\ \ u \notin L \Leftrightarrow \Pr_{v \in \{0,1\}^{n^c}} M(u,v)=1 < 2^{-n}$.

Let $m = n^c$, $c \geq 1$

Let $S_u = \{v \in \{0,1\}^m : M(u,v)=1\}$

So $|S_u| \geq 2^m - 2^{m-n}$ (big).

or $|S_u| < 2^{m-n}$ (small)



S_u is either "big" or "small"

<u>Claim 1</u> If S is small, $|S| < 2^{m-n}$, and $u_1 \ldots u_k \in \{0,1\}^m$, $k = \lceil \frac{m}{n} \rceil + 1$

then $\bigcup_{i=1}^{k} (S+u_i) \neq \{0,1\}^m$.

Here $S+u_i = \{w+u_i : w \in S\}$ where "+" means component-wise addition Mod 2 (XOR).

<u>Pf</u> $|S+u_i| = |S|$. So $\left|\bigcup_{i=1}^{k} (S+u_i)\right| \leq k|S| \leq (\frac{m}{n}+2)\, 2^{m-n} < 2^m$

<u>Claim 2</u>: If S is large, then exist $u_1 \ldots u_k$ s.t. $\bigcup_{i=1}^{k} (S+u_i) = \{0,1\}^m$.

Pf: We'll show this works for a randomly chosen $u_1 \ldots u_k \in \{0,1\}^m$.

Fix $w \in \{0,1\}^m$. $\Pr_{u_0 \in \{0,1\}^m} [(S+u_0) \text{ contains } w]$ ~~is~~

$w \in (S+u_0)$

$= \left[\Pr_{u_0 \in \{0,1\}^m} (u_0 + w) \in S\right] = \frac{|S|}{2^m} > 1 - 2^{-n}$.

Since u_i's are independent, $\Pr_{u_1 \ldots u_k}\left[\bigvee_{i=1}^{k} (S+u_i) \text{ contains } w\right] > 1 - (2^{-n})^k \geq 1 -$

$w \in (S+u_i)$

$\geq 1 - (2^{-n})^{\frac{m}{n}+1} > 1 - 2^{-m-n}$

I.e., $\Pr_{u_1 \ldots u_k}\left[\bigwedge_{i=1}^{k} w \notin (S+u_i)\right] < 2^{-m} \cdot 2^{-n}$

Hence $\Pr_{u_1 \ldots u_k}\left[\exists w \in \{0,1\}^m \bigwedge_{i=1}^{k} w \notin (S+u_i)\right] < 2^{-m} 2^{-n} \cdot 2^m = 2^{-n} < 1$.

Therefore

$\forall x \in \{0,1\}^n$,

$$x \in L \Rightarrow \Pr_{u_1 \cdots u_k}\left[\exists w \in \{0,1\}^m \bigwedge_{i=1}^{k} w \notin (S_x + u_i)\right] < 2^{-n}$$

$$x \notin L \Rightarrow \quad '' \quad = 1$$

Thus. $$x \in L \Leftrightarrow \Pr_{\substack{u_1 \cdots u_k \\ \in \{0,1\}^m}}\left[\forall w \in \{0,1\}^m \bigvee_{i=1}^{k} (w \in (S_x + u_i))\right] > 1 - 2^{-n}$$

$$x \notin L \Leftrightarrow \quad '' \quad = 0$$

So $$x \in L \Leftrightarrow \exists u_1, \ldots, u_k \in \{0,1\}^m \ \forall w \in \{0,1\}^m \ w \in \bigcup_{i=1}^{k} (S_x + u_i)$$

$u_i + w \in S_x$ for some i

$$\Leftrightarrow \underbrace{\exists u_1 \ldots u_k \in \{0,1\}^n}_{\substack{(\frac{m}{n}+1)m \text{ bits} \\ = (\frac{n^c}{n}+1)n^c \text{ bits}}} \ \forall w \in \{0,1\}^n, \underbrace{(\text{for some } i \le \tfrac{n^c}{n}+1,\ M(x, u_i+w)=1)}_{\text{poly time}}$$

So this is a Σ_2^P-computation.

Another way to think about it is we have

$$BPP \subseteq \cancel{RP}\ R(coNP)$$

Question: Is $BPP \subseteq R(NP)$?

Is $BPP \subseteq ZP(NP)$?

(1) Suppose $X \in \{0,1\}$ is a random variable,

$Pr[X=1] = p \qquad Pr[X=0] = q = 1-p.$

Then $\mu := E[X] = p \qquad (= p\cdot 1 + (1-p)q)$

$\sigma^2 = Var[X] = E((X-p)^2) = p(1-p)^2 + (1-p)p^2 = p(1-p) = pq$

Std dev $(X) = \sigma = \sqrt{pq}$.

(2) Let $X_i \in [0,1]$ be random variables, $i=1..n$.

Let $X = \sum_i X_i$ be a R.V.

$E(X) = \sum_i E(X_i) = n\cdot p$

$Var(X) = \sum_i Var(X_i) = n\cdot pq$

Std Dev$(X) = \sigma = \sqrt{npq}$.

Assuming X_i's are pairwise independent!

(3) Let $Y = \frac{1}{n}X$. (X as above).

Then $E[Y] = \mu_Y = p$

$\sigma_Y^2 = Var(Y) = E((Y-p)^2) = E(\frac{1}{n^2}(X-p)^2) = \frac{1}{n^2}\cdot npq = \frac{pq}{n}$

$\sigma_Y = $ Std Dev$(Y) = \sqrt{\frac{pq}{n}}$

Application Suppose $p = 2/3$, $q = 1/3$.

$\sigma_Y =$ Std Dev$(Y) = \frac{1}{\sqrt{n}}\sqrt{\frac{2}{9}} = \frac{1}{\sqrt{n}}\cdot\frac{\sqrt{2}}{3}$.

$Pr[Y \le \frac{1}{2}] \le Pr[|Y-\frac{2}{3}| \ge \frac{1}{6}] \le \left(\frac{\frac{1}{\sqrt{n}}\cdot\frac{\sqrt{2}}{3}}{1/6}\right)^2 = \frac{(2\sqrt{2})^2}{n} = \frac{8}{n}$

by Chebyshev Inequality.

So repeating n times, reduces probability of error to $\approx 1/n$

But Chebyshev Inequality does not suppose any independence.

~~Application #2~~. ~~Suppose~~ $p = \frac{1}{2}+\varepsilon$ $q = \frac{1}{2}-\varepsilon$

ε - will be a function of n, eg $\varepsilon = 1/n,\ 1/n^2$, etc.

$\mu_Y = \frac{1}{2}+\varepsilon$

$\sigma_Y = \text{StdDev}(Y) = \sqrt{\frac{1}{n}(\frac{1}{2}+\varepsilon)(\frac{1}{2}-\varepsilon)} = \frac{1}{\sqrt{n}}\sqrt{\frac{1}{4}-\varepsilon^2} \approx \frac{1}{\sqrt{n}}\left(\frac{1}{4}-\frac{\varepsilon^2}{2}+O(\varepsilon^4)\right)$

$Pr\left[Y \le \frac{1}{2}\right] \le Pr\left[\,|Y-(\frac{1}{2}+\varepsilon)| \geq \varepsilon\right] \le \frac{(\sigma_Y)^2}{\varepsilon^2} = \frac{1}{n}\left(\frac{1}{4}-\varepsilon^2\right)/\varepsilon^2$

$= \frac{1}{n}\left(\frac{1}{4\varepsilon^2}-1\right).$

So choosing $n > \frac{1}{\varepsilon^2}$ means $Pr\left[Y \le \frac{1}{2}\right] < 1/4.$

~~Choosing $n > r\cdot\frac{1}{4\varepsilon^2}$ means $Pr\left[Y \le \frac{1}{2}\right] < \frac{1}{r}$.~~

Again we ~~have not needed any independence for Chebyshev application~~

But needed ~~pairwise independence for the calculation of the variance,~~ σ_Y of Y.

However, we can ~~do~~ better with full ~~mutual independence~~.

Thm $BPP\left[\frac{1}{2}+\frac{1}{n^c}\right] \le BPP$ ~~by iterating~~ $4\frac{1}{(n^c)^2}$ $= n^c$ times.

~~Proof of~~ Chernoff Bounds.

Thm Let X_i be $\{0,1\}$ valued $\mu_i = E(X_i)$ $(=p_i)$. $i=1,\dots,n$.
Suppose X_i's are mutually independent.
Let $X = \sum_i X_i$ $\mu = E(X) = \sum \mu_i$
Let $\delta > 0$.

Then $$\left.\begin{aligned} \Pr[X \ge (1+\delta)\mu] &\le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\ \Pr[X \le (1-\delta)\mu] &\le \left(\frac{e^\delta}{(1-\delta)^{1-\delta}}\right)^\mu. \end{aligned}\right\} \le e^{-\min\left(\frac{\delta^2}{4}, \frac{\delta}{2}\right)\cdot\mu}$$

Corollary Let $Y = \frac{1}{n}$. Suppose each $\mu_i = p$ (same values), $p = \frac{1}{2} + \varepsilon$

$$\Pr[Y \le \tfrac{1}{2}] = \Pr\left[Y \le \left(1 - \frac{2\varepsilon}{1+2\varepsilon}\right)\left(\tfrac{1}{2}+\varepsilon\right)\right]$$ (use $\delta = \frac{2\varepsilon}{1+2\varepsilon}$)

$$\le \left(\frac{e^\delta}{(1-\delta)^{1-\delta}}\right)^{\mu_X} \quad \text{since } Y = \tfrac{1}{n}X.$$

Later: if $\varepsilon = \frac{1}{n^c}$, $\delta \approx 2\frac{1}{n^c}$)
$1-\delta = \frac{1}{1+2\varepsilon}$

$$= \left(\frac{e^\delta}{(1-\delta)^{1-\delta}}\right)^{n\left(\frac{1}{2}+\varepsilon\right)}$$

$$= \left(\left(\frac{e^\delta}{(1-\delta)^{1-\delta}}\right)^{\frac{1}{2}+\varepsilon}\right)^n$$

Can just and $\delta^2/4$

$$\le e^{-\min\left(\frac{\delta^2}{4}, \frac{\delta}{2}\right)\left(\frac{1}{2}+\varepsilon\right)n}$$

For $\varepsilon \approx 0$, $\delta \approx 0$, $e^{-\min\left(\frac{\delta^2}{4}, \frac{\delta}{2}\right)\left(\frac{1}{2}+\varepsilon\right)} \approx e^{-\delta^2/8} \approx 1 - \frac{\delta^2}{8}$

So $\Pr[Y \le \frac{1}{2}] \lesssim \left(1 - \frac{\delta^2}{8}\right)^n = \left(1 - \frac{\delta^2}{8}\right)^{\frac{8}{\delta^2}\left(\frac{\delta^2}{8}n\right)} \approx \left(\frac{1}{e}\right)^{\frac{\delta^2}{8}n}$

If such for $\varepsilon \approx 0$, $\delta \approx 2\varepsilon$, so $\Pr[Y \le \frac{1}{2}] \lesssim \left(\frac{1}{e}\right)^{\frac{\varepsilon^2}{2}n} \le \left(\frac{1}{2}\right)^{\frac{\varepsilon^2}{2}n}$

i.e. $n = r \cdot \frac{2}{\varepsilon^2}$ puts probability of error at $< \frac{1}{2^r}$.

Thus $BPP\left[\frac{1}{2n^c}\right] \subseteq BPP$ by using $2 \cdot \frac{2}{(1/n^c)^2} = 4n^{2c}$ iterations.

But also $BPP \subseteq BPP\left[1 - \frac{1}{n^c}\right]$ Pf ~~Each~~ With $\varepsilon = \frac{1}{3}$, iterate $3n^c$ times. (or at least constant)

Proof of Chernoff Bounds

We do the $1-\delta$ case, See Arora-Barak for the $1+\delta$ case.

New dummy variable t (will equal $\ln(1-\delta)$). Let $Z = \exp(tX)$

Replace "$-t$" with "t" everywhere!

$$E[\exp(tX)] = E[\exp(\sum_i -tX_i)] = E[\prod_i \exp(-tX_i)]$$

$$= \prod_i E[\exp(-tX_i)] \qquad \text{by mutual independence}$$

$$E[\exp(-tX_i)] = e^0(1-p_i) + e^{-t}(p_i) = 1 + p_i(e^{-t}-1) \leq \exp(p_i(e^{-t}-1))$$

So
$$E[\exp(-tX)] \leq \prod_i \exp(p_i(e^{-t}-1))$$

$$= \exp(\sum_i p_i(e^{-t}-1)) = \exp(\mu(e^t-1))$$

~~$\beta \exp(p_i(e^{-t}-1))$ where $\beta = 2\delta e$, $\beta = 2e$~~

Now,

$$\Pr[X \leq (1-\delta)\mu] = \Pr[X-\mu \leq -\delta\mu] = \Pr[\mu - X \geq \delta\mu]$$

~~$= \Pr[t(\mu-X) \geq t\delta\mu]$~~

~~$= \Pr[\exp(-t(\mu-X)) \geq \exp(-t\delta\mu)]$~~

$$= \Pr[t(X-\mu) \geq -t\delta\mu] \quad \text{if } t<0 \quad !$$

$$= \Pr[\exp(t(X-\mu)) \geq \exp(-t\delta\mu)]$$

$$\leq \frac{E[\exp(t(X-\mu))]}{e^{-t\delta\mu}} \qquad \text{by Markov's inequality}$$

$$= \frac{e^{-t\mu} E[\exp(tX)]}{e^{-t\delta\mu}} = \frac{E[\exp(tX)]}{e^{t(1-\delta)\mu}}$$

$$\leq \frac{\exp(\mu(e^t-1))}{\exp(t(1-\delta)\mu)} = \left(\frac{(e^{-\delta})^\mu}{(1-\delta)^{t(1-\delta)\mu}}\right) = \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^\mu$$

$t = \ln(1-\delta)$

$e^{-t} = \frac{1}{1-\delta}$

$e^t = 1-\delta$

$e^t - 1 = -\delta$

q.e.d.

~~ln(1+x) = x~~ P. 5 (redo)

<u>Lemma</u> $$\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \le e^{-\min\left(\frac{\delta^2}{4},\, \frac{\delta}{2}\right)}$$

<u>Pf</u> Suffices to show:

$$-\delta - (1-\delta)\ln(1-\delta) \le -\min\left(\frac{\delta^2}{4}, \frac{\delta}{2}\right)$$

Power series expansion of $\ln(1-\delta) = -\left(\delta + \frac{1}{2}\delta^2 + \frac{1}{3}\delta^3 + \dots\right)$

~~Try $\delta = \frac{1}{2}$~~

$$\cancel{\frac{\sqrt{e}}{\left(\frac{1}{2}\right)^{1/2}} \le e^{-\min\left(\frac{1}{8}, \frac{1}{4}\right)}}$$

$$\cancel{\frac{\sqrt{e^1}}{\left(\frac{1}{2}\right)}} \le e^{-1/8}$$

$$-\frac{1}{2}\delta^2 - \frac{1}{6}\delta^3 - \dots \le -\min\left\{\frac{\delta^2}{4}, \frac{\delta}{2}\right\}$$

$\frac{\delta^2}{2}$ works for $\delta \ge 0$

For the $1+\delta$ case, need the $\min\left\{\frac{\delta^2}{4}, \frac{\delta}{2}\right\}$.

<u>Corollary:</u> For all δ, $\exists$ constant $c_\delta < 1$ s.t.

$$\Pr\left[\left|\sum_i X_i - \mu\right| > \delta\mu\right] \le (c_\delta)^\mu.$$

<u>In fact,</u> for $\delta < \frac{1}{2}$, $c_\delta = e^{-\delta^2/4}$ works.

Day 6
11

Outline

Def'n A Random Variable is a mapping from a probability space (discrete): i.e. a set $a_1 .. a_k$ w/ prob $p_1 .. p_k$ that sum to 1

Lm $E(X_1 + X_2) = E(X_1) + E(X_2)$
$\sum_{a_1 a_2} p_1 p_2$ $= \sum_{a_1} a_1 pr(X_1) = a_1 + \sum_{a_2} a \, Pr[X_2] = a_2$

Pf Trivial

Y Probability results

(a) Markov's inequality, for $X \geq 0$,

$$Pr[X \geq k\mu] \leq \frac{1}{k}$$

or $$Pr[X \geq \alpha] \leq \frac{\mu}{\alpha}.$$

(b) Chebyshev inequality. Let X have mean μ, std dev σ;

$$Pr[X - E(X) \geq k\sigma] \leq \frac{1}{k^2}$$

or $$Pr[X - E[X] \geq \alpha] \leq \left(\frac{\sigma}{\alpha}\right)^2$$

Pf $Var(X) = E((X - E(X))^2) = E((X-\mu)^2). = \sigma^2$
$\sigma = \sqrt{Var X}$
~~By Markov~~, $Pr[X - \mu \geq k\sigma] \leq Pr[|X-\mu| \geq k\sigma]$
$= Pr[(X-\mu)^2 \geq k^2\sigma^2] \leq \frac{1}{k^2}$ by Markov.

Def'n X_1 and X_2 are independent if

$$Pr[X_1 = a_1 \wedge X_2 = a_2] = Pr[X_1 = a_1] \cdot Pr[X_2 = a_2].$$

$\{X_1 .. X_n\}$ are mutually independent iff

$$Pr\left[\bigwedge_{i=1}^{n} X_i = a_i\right] = \prod_{i=1}^{n} Pr[X_i = a_i]$$

$\{X_1 .. X_n\}$ are pairwise independent iff $\forall i \neq j$, X_i, X_j are independent

Lemma: If X_1, X_2 independent, then $E(X_1 X_2) = E(X_1) \cdot E(X_2)$ $= \sum_{a_1, a_2} a_1 \cdot a_2 \cdot Pr[X_1 = a_1] \, Pr[X_2 = a_2]$

Pf

Thm: If $X_1 .. X_n$ are independent, then

$$Var\left(\sum_i X_i\right) = \sum_i Var(X_i)$$

Pf Let $\mu_i = E(X_i)$. $E(\sum X_i) = \sum E(X_i)$.
Let $X = \sum X_i$. So $\mu = E(X) = \sum \mu_i$

$$Var(X) = E((X-\mu)^2) = E\left(\left[\sum(X_i - \mu_i)\right]^2\right) = E\left(\sum_i (X_i - x_i)^2 + \sum_{i \neq j} E((X_i - \mu_i)(X_j - \mu_j))\right)$$

$$= E\left(\sum_i (X_i - \mu_i)^2\right) + \sum_{i \neq j} 0 \cdot 0 = \sum_i (E(X_i - \mu_i))^2 = \sum \sigma_i^2.$$

Day 6
(2)

(c) Chebyshev inequality application

Let X_i be $\{0,1\}$-valued RV's, that are pairwise independent.

Let $p_i = Pr[X_i] = 1$ $\quad 1-p_i = q_i = Pr[X_i] = 0$.

Let $X = \sum_{i=1}^{k} X_i$. Assume p_i's are all equal, and $> 1/2$

Want to lower bd the probability that $X \leq 1/2 k$ ("Majority decides")

Each X_i has mean $\mu_i = p_i$ $(= p)$

and variance $E((X_i - p_i)^2) = p_i(1-p_i)^2 + (1-p_i)p_i^2 = p_i(1-p_i)$

So X has mean kp, variance $k\, p_i(1-p_i)$.

and std dev $\sigma = \sqrt{k\, p_i(1-p_i)}$.

By Chebyshev,

$$Pr[X \leq \tfrac{1}{2}k] = Pr[X - kp \leq (\tfrac{1}{2} - p)k] = Pr[(kp - X) \geq k(p - \tfrac{1}{2})]$$

$$\leq \frac{\sigma^2}{\alpha^2} = \frac{k(p)(1-p)}{(k(p-\frac{1}{2}))^2} \leq \frac{1}{4}\frac{1}{k}\frac{1}{(p-\frac{1}{2})^2}.$$

Corollary $BPP\left(\frac{1}{2} + \frac{1}{n^c}\right) = BPP$.

Pf: Here $p = \frac{1}{2} + \frac{1}{n^c}$. $\quad \left(p - \frac{1}{2}\right)^2 = \frac{1}{n^{2c}}$.

Want $\frac{1}{4}\frac{1}{k}\frac{1}{n^{-2c}} \leq \frac{1}{3}$, i.e. $k \geq \frac{n^{2c}}{4}$.

So running the $BPP(\frac{1}{2} + \frac{1}{n^c})$ algorithm $\frac{n^{2c}}{4}$ times and taking majority vote works!

Remark Similar use of Chebyshev gives $BPP = BPP\left(1 - \frac{1}{n^c}\right)$, $\forall c$.

But we can do better.

In particular, note the random choices are independent, but the above analysis used only pairwise independence.

(d) Chernoff bounds.

Day 6 (3)

Let $X_1 \ldots X_n$ be independent, $\{0,1\}$ valued random variables

Let $\delta > 0$. Let $X = \sum X_i$, $\mu = E(X)$, $\sigma^2 = Var(X)$.

Then $Pr[\cancel{\sum_{i=1}^{n} X_i} X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} \leq \left(e^{-\delta^2/4}\right)^{\mu}$ for $\delta < \frac{1}{2}$

$Pr[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} \leq \left(e^{-\delta^2/2}\right)^{\mu}$ for all $\delta \geq 0$.

(d.i) Application to BPP

Thm $BPP = BPP(1 - \frac{1}{2^{n^c}})$, for ~~...~~ $c \geq 1$.

Pf Do majority vote of k independent trials. ~~Further,~~

Here $p = \frac{2}{3}$. $\cancel{\delta = (1+\delta)\frac{2}{3}}$ For $x \in L$, $E(X_i) = \frac{2}{3}$,

and $\mu = \frac{2}{3}k$.

$(1-\delta)\frac{2}{3} = \frac{1}{2}$ $\quad 1-\delta = \frac{3}{4}$ $\quad \delta = \frac{1}{4}$

$Pr[X \leq \frac{1}{2}k] = Pr[X \leq (1-\frac{1}{4})\frac{2}{3}k]$ ← so $\delta = \frac{1}{4}$

$\leq \left(\frac{e^{-1/4}}{(1-\frac{1}{4})^{(1-1/4)}}\right)^{2/3 k} \leq \left(e^{-3/32}\right)^{k} = d^k$ for some $d \in [0,1]$.

$d^k = 2^{-n^c}$

~~$k \log c = d \log n$~~

$k \log_2 d = -n^c$

$k = \frac{1}{\log d} \cdot -n^c$

Taking $k = \cancel{\left(\frac{d}{\log c}\right)}\left(\frac{1}{\log(1/d)}\right) n^c$

$Pr[X \leq \frac{1}{2}k] \leq 2^{-n^c}$.

q.e.d.

(d.ii) Pf of Chernoff Bound:

See page "P.4."

Day 6
(4)

(e) Thm: $BPP \subseteq P/poly$

Lemma: on averaging

see p 35

(f) Thm Sipser-Gacs

$BPP \subseteq \Pi_2^P \cap \Sigma_2^P$.

See p 37-38

(g) $P^{PP} = P^{\#P}$.

see p. 36