Today

- Fortnow’s time/space lower bound on SAT.
- Randomized Computation.

Power of Alternation

- Basic notion.
- Captures Time/Space differently.
- Next application shows how powerful it is.

Fortnow’s theorem

For today, will use LIN to mean the class of computations in NEARLY-LINEAR TIME:

\[ \text{LIN} = \bigcup_{c} \text{TIME}(n(\log n)^{n}) \]

- Belief: \( \text{SAT} \not\in L \).
- Belief: \( \text{SAT} \not\in \text{LIN} \).
- Can’t prove any of the above.
- Fortnow’s theorem: Both can not be false!

Formal theorem + Proof

Theorem: [Fortnow ’97] If SAT \( \in L \), then \( \exists \epsilon > 0 \) s.t. SAT \( \not\in \text{Time}(n^{1+\epsilon}) \).

Proof: Assume SAT \( \in L \), and SAT \( \in \bigcap_{\epsilon>0} \text{Time}(n^{1+\epsilon}) \).

Then .... will get contradiction (after few slides).
Proof Idea

1. SAT in $\text{Time}(n^{1+\epsilon})$, implies non-determinism is not very powerful, & so alternation is not very powerful.

2. SAT is complete for $\text{NTIME}(n)$ implies SAT is very powerful.

3. SAT in $L$ implies small space computation is very powerful.

4. Savitch’s theorem implies alternation is powerful in small space computation, and hence very powerful for all computation.

5. Contradiction to (1)!

Fortnow: Step 1

Fact 1: If SAT $\in L$, then $\text{TIME}(T(n)) \subseteq \text{SPACE}c \cdot \log T(n)$

Proof: Padding + completeness of SAT under Logspace reductions.

Fortnow: Step 2

Fact 2: $\text{SPACE}(s) \subseteq \text{ATIME}[i, i2^s/i]$.  

Proof:

- Draw depth $i$ tree of width $w$ having $2^w$ leaves.

- At top level, Guess $w$ intermediate configurations $c_1, \ldots, c_w$ and for all successive pairs $c_j, c_{j+1}$ verify reach from $c_j$ to $c_{j+1}$ in $w^{i-1}$ steps.

Corollary: (with $\text{TIME}(T) \subseteq \text{ATIME}[i, (T)^c/i]$).
Fortnow: Step 3

Fact 3: If, say, SAT $\in$ \textsc{Time}(\(n^{1+c}\)), then \textsc{ATime}[a,t] \subseteq \textsc{Time}(\(1+c)^{2^i}\).

Proof:

- Induction on \# alternations.
- Use strong form of Cook’s theorem at every step.
- Take care to make sure numbers work out.

Contradiction?

Have

\[
\text{Time}(T(n) = 2^{\sqrt{\log n}}) \\
\subseteq (\log T) \\
\subseteq \text{ATime}[i, T^{c/i}] \\
\subseteq \text{Time}(T^{(c/i)}(1+c)^{2^i}).
\]

Contradicts if \((c/i)(1 + c)^{2^i} < 1\). Can be arranged by picking \(i = 10c\) and \(\epsilon = 1/(2i)\).

Randomized computation

- Physicists’ Belief: Natural phenomena have randomness built into them.
- How does this affect our belief that “polynomial time” is all that is feasible?
- Should study formally.

Randomized algorithms/Turing machines

- Model 1: Machine can enter a random state whenever it wishes. Takes one of two outgoing transitions randomly.
- (Equivalent) Model 2: Machine has two inputs: (1) The actual input and (2) the outcome of many independent random coin tosses.
Randomized machines and languages

Machine $M$ for Language $L$ has:

**Completeness** $c$ if $c = \inf_{x \in L} \Pr_y[M(x, y)\text{accepts}]$
(Assume uniform distribution on $\ell(|x|)$ bit strings.

**Soundness** $s$ if $s = \sup_{x \not\in L} \Pr_y[M(x, y)\text{accepts}].$

$M$ seems to decide membership in $L$ if $c > s$.
But even better if $c = 1$ (and/or $s = 0$).

Time-bounded randomization

- **BPP**: (Bounded Probability Polynomial-time): Both kinds of errors allowed (two-sided error): $L \in BPP$ if there exists a two-input deterministic machine $M$ running in time poly in first input such that:

  $$ x \in L \iff \Pr_y[M(x, y)\text{accepts}] \geq 2/3. $$

  (Completeness $= 2/3$; Soundness $= 1/3$).

- **RP**: (Randomized Polynomial-time): Only false negatives (one-sided error):

  $$ x \in L \Rightarrow \Pr_y[M(x, y)\text{accepts}] \geq 2/3. $$

  (Completeness $= 2/3$; Soundness $= 1/3$ (perfect)).

Time-bounded randomization (contd.)

- **co-RP**: complements of RP languages.

- **ZPP**: Error happens with probability zero.
So what does randomness do? Running time is not guaranteed to be polynomial. Only expected to be polytime.
Space-bounded randomization

Similar collection of four classes:

- BPL, RL, co-RL, ZPL.
- Catch 1: In two-input model, have one way access to second input.
- Catch 2: Machines bounded to run in polynomial time.

Looking ahead

- $2/3, 1/3$ arbitrarily chosen. For definition of BPP suffices to have $c > s$. Similarly for RP, suffices to have $c > 0$ etc.
- Randomness more powerful than deterministic?
  - Belief: No.
  - Current evidence: Yes. There exist problems in RP that we can show to be in P. (Example: Primality testing.) There exist problems in RL that we can’t show to be in L. (Example: USTCON - connectivity in undirected graphs.)

Looking further ahead

- How do RP, BPP etc. relate to familiar complexity classes.
- Obviously: ZPP in RP & co-RP; and all are in BPP.
- RP in NP (by definition).
- BPP? Don’t quite know:
  - BPP in $P^{/poly}$.
  - BPP in PH.