1 Notes on 27th April

1.1 constructing g

In the proof of last class, we used the following fact:

Fact 1 *Let $k \in N$, let $f \in \#P$*
$\exists g \in \#P$, s.t.
$f(x)$ is odd $\Rightarrow g(x) \equiv 1 \ (mod \ \ 2^{n^k})$
$f(x)$ is even $\Rightarrow g(x) \equiv 0 \ (mod \ \ 2^{n^k})$

Today, we'll construct this g according to specified f.

Define: $g(n) = 3n^2 + (2^{q(n)} - 2)n^3$
$g_{i+1}(n) = g_i(g_1(n))$

We'll say later what the polynomial q is.
Claim:

$$1 \leq \log q(n) \Rightarrow \left(\begin{array}{lcl} n \equiv 0(mod\ 2) & \Rightarrow & g_i(n) \equiv 0(mod \ \ 2^{2^i}) \\ n \equiv 1(mod\ 2) & \Rightarrow & g_i(n) \equiv 0(mod \ \ 2^{2^i}) \end{array} \right) \qquad (1)$$

(This claim has an easy inductive proof.)

Proof by induction:
We need
given a $\#P$ function f,
show that for any large enough polynomial q, s.t
$g_{k \log |x|}(f(x)) \in \#P$

On input y
compute $i = k \log |x|$
Run(y, i)

Run(y, i)
if $i = 0$, then
return $f(y)$ paths
else
flip a coin
if *head* then
flip 2 coins
if all are *tails*, then
rejects
else **3 chances**
Run$(y, i-1)$
Run$(y, i-1)$
if Both return accepting paths then accepts
else **tail**

flip $q(|y|)$ coins
if the first $q(|y|)-1$ are *tails*, then
  rejects
else $^{**}2^{q(|y|)}-2$ chances**
  Run$(y, i-1)$
  Run$(y, i-1)$
  Run$(y, i-1)$
  if all return accepting then accepts

End of construction of g.

Claim:
The height of the tree of this non-deterministic algorithm is: $\leq 3^i \leq 3^{k \log n} = n^{O(1)}$

This concludes the proof of Toda's Theorem.

Originally, we had intended to give a proof of the following interesting fact, which is yet another application of the Isolation Lemma. However, there is not time to present the proof.

Fact 2

$$NL/Poly = UL/Poly \tag{2}$$

If $DSPACE$ has a set of hardness of $2^{\epsilon n}$, then $NL = UL$.

1.2 $NP = P$?

Theorem 1 $\exists A, \exists B \;\; s.t \;\; P^A = NP^A, \; P^B \neq NP^B$

Fact 3 *Most diagonalization arguments rely on simulation. Most simulation also work with oracle. Thus, no proof of $P = NP (P \neq NP)$ can use the usual diagonalization and simulation.*

Proof (of theorem 1):
A can be any $PSPACE-complete$ set.

Let's construct B
We want $\forall i, \; M_i^B \neq L(B)$
  where $M_i \in DTIME(n^{O(1)})$ and $L(B) = \{0^n : \exists x \in B, |x| = n\}$
  Trivially, $L(M) \in NP^B$

We'll construct B in stages
  $B = \bigcup_i B_i$
  $B_0 = \phi, B_i \subset B_{i+1} \subset \cdots$

To construct B_i from B_{i-1}
  Let m be the length of longest string in B_{i-1}
  Let $n > m^i$

if M_i accepts 0^n, then
$B_i = B_{i-1}$ and don't put any string of length $\leq n$ into B_i -
else $^{**}M_i(0^n)$ rejects and it queries $\leq n^i$ strings of length n^{**}
$B_i = B_{i-1} \bigcup y$ where y is some unqueried string of length n.

End (of theorem 1).

For most of the well-known open questions in complexity theory, there are oracles relative to which either answer to the open question holds. For example, the following fact is not too hard to prove (and we *would* prove it if we had time). It shows that there are oracles relative to which EXP is contained in P/poly (since the proof that BPP is in P/poly actually shows that, relative to all oracles A, BPP^A is contained in P^A/poly).

Fact 4 $\exists c, BPP^c = EXP^c$

Since a goal of complexity theory is to solve the P vs NP problem, it is important to develop "non-relativizing" proof techniques. One important example of a non-relativizing proof is the proof that PSPACE = IP (to be presented in an upcoming lecture). Actually, another important example has already been presented. We proved that if EXP is in P/poly, then EXP is equal to Σ_2^p. It turns out, however, that there is an oracle relative to which EXP is contained in P/poly, but EXP is *not* equal to Σ_2^p. (Our original proof used the fact that there are "self-reducible" sets that are complete for EXP. This does not hold relative to every oracle!)