We know:

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

Today we prove these separations:

The method we will prove is based on diagonalization where we designed a new machine that did the opposite of the ith machine M_i on input $\langle M_i \rangle$

list of all TMs		$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\cdots$
	M_1				
	M_2				
	M_3				
	$\vdots$				

We will do something along the same lines for listing all space-bound (or time-bounded TMs)

The construction is similar in the two cases but easier for space.

The general idea is that a bit more space will let TMs to do more, but that only works for "nice" space bounds.

Space Hierarchy Theorem

Def^n A function $f: \mathbb{N} \to \mathbb{N}$ is space constructible iff
$f(n) \geq \log_2 n$ and
the map $1^n \mapsto$ binary representation of $f(n)$, $\langle f(n) \rangle$
is computable by an $O(f(n))$-space TM.

The general idea of a listing of all space-bound TMs is that for a space constructible $f(n)$
and any $\langle M \rangle$ we can simulate M on input x
using space $O(f(n))$ s.t.
The simulation does what M does if
- M doesn't use more than $f(|x|)$ storage
- M doesn't run for more than $2^{f(|x|)}$ steps
(which implies that M doesn't run forever)

"On input $\langle M \rangle$ and x:
- Use space constructability of f to compute the binary string $\langle f(|x|) \rangle$ on the work tape
(pretend each symbol of x is a 1)
- Mark off $f(|x|)$ cells on a separate section of the work tape
- Create a counter $2^{f(|x|)}$ using another $f(|x|)$ cells.
- Simulate M on input x keeping track of the # of steps
  - subtract 1 from counter each step
  - stop simulation if it moves off the marked cells & reject
  - stop when counter reaches 0 & reject"

Using this idea we prove

Theorem If $f(n)$ is space constructible Then there is language A decidable using space $O(f(n))$ but not $o(f(n))$.

Proof Define A as the language decided by the following TM for a "diagonal language"

Almost final alg.

On input x:
1. Use space constructibility of f to compute $\langle f(|x|)\rangle$ on the work tape
2. Mark off $f(|x|)$ cells on the work tape
3. If x is not of the form $\langle M\rangle 01^n$ then reject
4. Simulate M on input x counting steps
   If more than $2^{f(|x|)}$ steps then stop & accept
   If more than $f(|x|)$ cells used stop & accept
5. If M accepts then reject.
   If M rejects then accept

Here is our fix

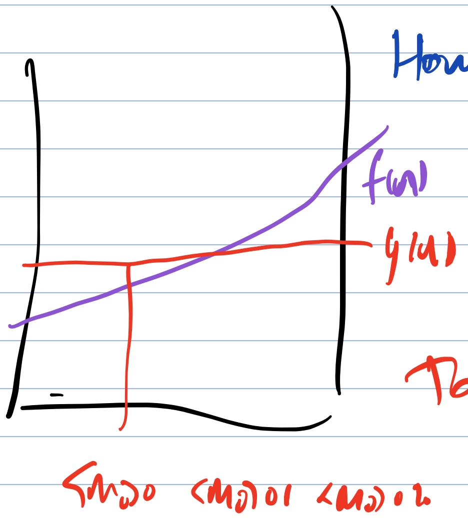
Claim A is different from every language decided using space $o(f(n))$

Suppose not. Then $A = L(M_i)$ for some M_i that uses space $g(n) = o(f(n))$

Consider whether A includes $\langle M_i\rangle$
If M_i runs on input $\langle M_i\rangle$ using $\leq f(|\langle M_i\rangle 01^n|)$ cells and time at most $2^{f(|\langle M_i\rangle 01^n|)}$

what our fixes do

Then we get a contradiction since we flipped the answer in defining A.



However even though $g(n)$ is $o(f(n))$ $n = \langle M_i \rangle$ might be small enough that $f(n) < g(n)$, in which case there wouldn't be a contradiction.

To get around this we flip an infinite # of values for each M_i, and not just the diagonal.

We use

$x = \langle M_i \rangle 0 1^k$ for all integers k

which will allow us to tell which machine is associated.

Now for any input $x\langle M_i \rangle 0 1^k$ such that k makes $f(\langle M_i \rangle 0 1^k) \geq g(\langle M_i \rangle 0 1^k)$ is good enough

and we get a contradiction $\boxtimes$

Cor If $S_1(n)$ is $o(S_2(n))$ then

$$SPACE(S_1(n)) \subsetneq SPACE(S_2(n))$$

Note: most natural functions are space constructible

n^k, $\log n$, $n \log n$ etc.

eg $\log n$: On input 1^n count # of bits onto work tape: gets n in binary which takes $\log n$ bits.

Now count # of bits in that: $\log n$ in binary

Cor $NL \subseteq SPACE(\log^2 n) \subsetneq PSPACE$

Time Hierarchy

Def^n $f(n) \geq n \log n$ is **time constructible**
iff $1^n \to$ binary of $f(n)$ is computable
in time $O(f(n))$

Thm If $t(n)$ is time constructible there
is a language decidable in time
$O(t(n))$ but not $o(t(n)/\log t(n))$

possible gap.

Proof idea: Essentially the same as the
one for bounded space except
that on input x
we use time constructability
to compute $t(|x|)$ in binary
and use it as a timer for
the computation.
(count down to 0 subtracting
1 per step)
reject if it exceeds the time

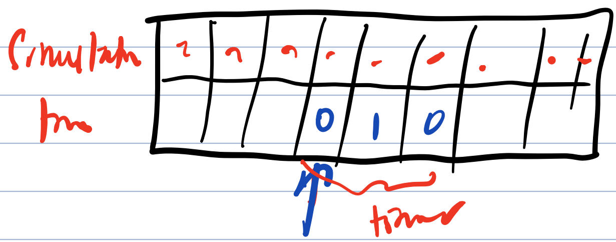
Unlike with space complexity we have
to count # of steps to update
the timer.
The timer takes $\log t(|x|)$ bits
to represent & update

In the course we used multitape TMs for
time. The book used 1-tape TM
The proof is different in the two cases.

Multitape TM version: We need a fixed # of tapes for the machine defining A but the other TMs M_i might use more tapes:

We use simulation of k-tape TM by 2-tape TM
$t(n)$ steps become $O(t(n)\log t(n))$ steps & it keeps track of a counter

1-tape version: Maintain the counter like a pocket watch that is carried by the TM near the read head:



(think of it as on a separate track of the tape)
shift the timer left or right at each time step.
$O(\log t(n))$ steps per original step.

If $t'(n)$ is $o(t(n)/\log t(n))$ then both of these can be done in $\leq t(n)$ steps

∎