Last time

1. Diagonalization

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>$\neq$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td></td>
<td>$\neq$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td></td>
<td></td>
<td>$\neq$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td></td>
<td></td>
<td></td>
<td>$\neq$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\neq$</td>
<td></td>
</tr>
</tbody>
</table>

Cell $(i,j) = \text{output of } M_i \text{ on } j$.

Prove that certain language $L$ is not decidable within resource bound by showing that in each row one cell contains wrong answer.

Namely on the diagonal — **DIAGONALIZATION**

2. Time hierarchy theorem

More computation time $\Rightarrow$ more problems solved

3. Ladner's theorem

If $P \neq NP$, then there are (infinitely many) complexity classes in between

(Only sketched proof in class — details on problem set)
What exactly is "proof by diagonalization"?

How far can diagonalizing techniques take us? Maybe prove P\neq NP if work hard enough?!

Probably NO, unfortunately...

Our diagonalization proofs relied on:

1. Representation of TMs by strings (every integer is a TM)

2. Efficient simulation of TM by other (universal) TM without much overhead (in time or space)

Such an approach works even if we give TM access to certain subroutines for free and don't charge for time spent in such calls during running time analysis.

Such TMs are known as "oracle Turing machines."

Might sound very strange - for us it will just be a program with a subroutine that can be called free of charge.
**DEF Oracle TM (informal—see textbook)**

An oracle Turing machine is a usual TM except
- has special read-write oracle tape
- has special oracle state

To execute M, specify oracle language 0 ≤ \( O \)

At any time, M can
- write string on oracle tape (takes several steps)
- jump to oracle state (one time step)
- get answer whether \( y \in 0 \) or not
  written as bit 1 if \( y \in 0 \), 0 if \( y \notin 0 \)
  on oracle tape (one time step)

Output of \( M \) on \( x \) when run with oracle 0 denoted \( M^0(x) \)

Nondeterministic oracle machines defined analogously

\[ \mathcal{P}^0 = \{ \text{all languages decidable by poly-time deterministic TM with oracle } 0 \} \]

\[ \mathcal{NP}^0 = \{ \text{all languages decidable by poly-time nondeterministic TM with oracle } 0 \} \]

Also say that TM \( M \) has "oracle access" to language \( 0 \).
Examples

1. \text{UNSAT} \in \text{PSAT}
   
   Write down formula on oracle tape
   
   Make query to SAT
   
   Give the opposite answer as output

2. If \(0 \in \text{P} \) then \( \text{P}^0 = \text{P} \)
   
   Oracle calls are not needed.
   
   Can compute answer by simulating machine deciding \(0\) in poly time.

3. \(\text{EXP} = \{<M, x, 2^n> : M \text{ outputs } 1 \text{ on } x \text{ within } 2^n \text{ steps}\}\)

   \[\text{Then: } \text{P}^{\text{EXP}} = \text{NP} \subseteq \text{EXP} = \text{EXP}\]

   Recall \(\text{EXP} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{cn})\)

Proof

Suppose \(L \in \text{DTIME}(2^{cn})\) decided by \(M\)

Write down \(M, x\), and then \(2^{cn}\) (i.e. \(n^c \leq 1/5\)) on oracle tape

Query \(\text{EXP} \) and answer the same.

\(\Rightarrow \text{EXP} \subseteq \text{P}^{\text{EXP}}\)

\(P^0 \subseteq \text{NP}^0\) for any oracle language \(O\) (why?)

Suppose \(L' \in \text{NP}^{\text{EXP}}\) decided by \(M'\)

running in time \(O(n^c)\)

At most \(2 \cdot O(n^c)\) nondeterministic choices

which is exponential

At most that many oracle calls - can also be computed in exponential time.

Exponential \(\times\) exponential = exponential, so \(L' \in \text{EXP}\).
Regardless if the oracle is
(Ⅰ) and (Ⅱ) holds for oracle
TMs (if simulating machine is also given the
oracle 0).

Hence, any theorem about TMs
that uses only (Ⅰ) + (Ⅱ) holds for
oracle TMs (the theorem
relativizes).

The answer to $P \neq NP$ can't be
a relativizing theorem, since there
are oracles to flip the answer both ways.

**THEOREM (Baker, Gill, Solovay '75)**
There exist oracles $A$ and $B$ s.t.
$P^A = NP^A$ and $P^B \neq NP^B$

**Proof**
Set $A$ to be EXPGEN.

For any language $B$, let
$$U_B = \{ 1^n : \exists x \text{ s.t. } |x| = n \text{ and } x \in B \}$$

For any $B$, $U_B \in NP^B$.

On input $1^n$, guess $x$ of length $n$, write
on oracle tape, query $B$.

Want to build $B$ s.t. $U_B \notin P^B$.
If so, proof finished.
High-level intuition:

Any TM for \( U_3 \) has to run in subexponential time. Can only query vanishing small part of strings of length \( 30,15^n \) - exponentially many. Make sure any TM "queries the wrong strings."

Construction of \( B \)

\( M_i \): Turing machine encoded by (binary expansion of) integer \( i \)

Construct \( B \) in stages. Stage \( i \) will make sure \( M_i \) doesn't solve \( U_3 \) in time \( \leq 2^n / 10 \).

Initially \( B_i = \emptyset \), \( i = 1 \).

**Stage \( i \):**

\( B \) contains finite # strings so far.

Fix \( n \) s.t. no string of length \( \geq n \) has had its status w.r.t. \( B \) decided.

Run \( M_i \) on \( 2^n \) for \( 2^n / 10 \) steps.

Oracle queries \( y \)

**Case 1:** strings of \( y \) decided in previous stages — answer accordingly

**Case 2:** status of \( y \) undecided — answer \( y \notin B \)
Suppose \( M_i \) finishes on \( 1^n \) and answers 6

\[ \text{Note} - \text{Have only decided status } f \leq 2^n/10 \]
strings in \( \{0,1\}^n \)
- For all of them, answer no.
If \( b = 1 \), decide that no string in \( \{0,1\}^n \)
is in \( B \)
\[ \Rightarrow 1^n \notin U_B, \text{ and } M_i \text{ is wrong on } 1^n \]
If \( b = 0 \), pick some string \( y \in \{0,1\}^n \)
ot queried and decide that \( y \in B \)
\[ \Rightarrow 1^n \notin U_B, \text{ so } M_i \text{ is wrong on } 1^n \]

Only remaining worry. What if we didn't allow \( M_i \) to finish? What if it runs in polynomial time \( p \) s.t.
\[ p(n) > 2^n/10 \text{ for this } n \]
The TM \( M_i \) will be repeated infinitely often for larger and larger \( n \). Finally, will get some \( n' \) s.t.
\[ p(n') < 2^n/10, \text{ and for this } n', \text{ the proof will work.} \]

Recall our TM encoding has "stop marker" after which junk allowed, so each TM encoded by infinitely many integers
**SUMMARY**

- Diagonalization can be used to separate complexity classes.
  - In particular, $P \neq EXP$.
  - If $P \neq NP$, then there is an infinite hierarchy of complexity classes between $P$ and $NP$.
  - But diagonalization is not enough to settle $P \equiv NP$, since any such proof works for oracle TMs and different oracles give different answers to $P \equiv NP$.

**NEXT ON THE AGENDA**

- Memory consumption as the limiting factor
- Space-bound complexity classes
So far focused on running time as limited resource.

At the end of the day, most interesting measure [at least a lot of the time]

But also interesting to consider memory usage.

Arguably second most fundamental resource.

Get new complexity classes and complete problems for them.

Some similarities with time-bounded computation, but also some striking differences.

**DEF 1** A language \( L \subseteq \{0,1\}^* \) is in space \( (s(n)) \) if \( \exists \) constant \( c \) & Turing machine \( M \) such that

1) \( M \) correctly decides \( L \)

2) \( M \)'s heads never visit more than \( c \cdot s(n) \) distinct locations on read-write tapes excluding input tape for any input of length \( n \)

\( L \in \text{NSPACE} (s(n)) \) if \( \exists \) non-deterministic TM deciding \( L \) s.t. at most \( c \cdot s(n) \) real-write locations visited for any input of length \( n \) and any non-deterministic choices.

Note that we have NDTM here and not certificate and verifier TM.
Important points

- Input stored on read-only input tape
doesn't count towards memory

\[ \Rightarrow \text{Possible to do computations in sublinear space} \]

- Decision problem: Need not use output tape other
  than for answer yes/no (0/1).
  Focus on work tape (s)

- Look at only space-consumable \( s(n) \)
  \( \exists \text{TM that computes } s(|x|) \) in \( O(s(1 \times l)) \)
  space given input \( x \). (Technical condition
  that we will ignore.)

- Space bounds of interest \( \geq \log n \) - want TM
  to be able to remember positions on input tape.

Clearly, \( \text{DTIME}(s(n)) \leq \text{SPACE}(s(n)) \)

Can visit at most one tape position per time step
But space can be reused
Use space \( s(n) \) to count from \( 0 \) to \( 2^{s(n)} - 1 \).
This is (almost) all that we know.

\[ \text{TMM 2} \]

\[ \text{DTIME}(s(n)) \leq \text{SPACE}(s(n)) \leq \text{NSPACE}(s(n)) \leq \text{DTIME}(2^{O(s(n))}) \]

(Will be proven shortly.)
In fact, can do slightly better

**THEM 3** (Hopcroft, Paul, Valiant '77)

$$\text{DTIME} \left( s(n) \right) \leq \text{SPACE} \left( s(n) / \log s(n) \right)$$

So space is strictly more powerful than time as resource. (Probably won't do anything close to proving Thm 3.)

One more point:

- What about termination in Def 1?
  - Can require it. Not necessary, really.

After $2^{O(s(n))}$ steps, multispace has looked exactly the same, all heads have been exactly the same place, (NO) TM state has been exactly the same, etc. at two time steps $t_1, t_2$. So can ignore computation during interval $[t_1 + 1, t_2]$.

If $\exists$ accepting computation, only $2^{O(s(n))}$ steps needed. Can equip any TM with "clock" that terminates after $2^{O(s(n))}$ steps. Only $O(s(n))$ space needed to count the time.

Back to proof of Thm 2...
Configuration graph of TM M

- program counter/ state
- head positions
- consists of all tape positions that can possibly be visited (for some input length)

Configuration graph of M on input \( x \in \{0,1\}^* \)

Directed graph \( G_{M,x} \) [space 5(h) TM]

Vertices: all possible configs \( C \) with \( \text{input} = x \) and \( c \cdot s(1x) \) worktape cells

Edges: \( (C, C') \) if \( C' \) can be reached from \( C \) in one step acc to M's transition function.

Deterministic TM: out-degree 1
Nondeterministic TM: out-degree 2

Assume M has "clean-up phase" erasing all worktapes before halting \( \Rightarrow \) one unique accepting config Caccept.

\( M \text{ accepts } x \iff \exists \text{ path in } G_{M,x} \text{ from Start to Caccept} \)

CLAIMS

1. Every vertex in \( G_{M,x} \) can be described using \( K = s(n) \) bits (\( K = O(n) \) depending on alphabet, \# tapes, \# states)

2. \( G_{M,x} \) has at most \( 20(n) \) vertices

3. \( \exists 0(\log n) \)-size CNF formula \( \Phi_{M,x} \) s.t.

\[ \Phi_{M,x}(C, C') = 1 \iff (C, C') \text{ edge in } G_{M,x} \]
Proof

1. Sort of by description in Def 1
2. Follows from 1.
3. Use Cook-Levin-style reasoning
   Formula contains lots of local consistency checks
   - type contents are correct on one step
   - jump to correct state given read bits and
     previous state
   - et cetera
   \( O(s(n)) \) checks
   Each check involves constant \# bits = \# variables, \( n^c \)

Proof of Thm 2

Only need to show \( \text{NSPACE}(s(n)) \leq \text{DTIME}(2^{O(s(n))}) \).
Construct \( G_{\text{Mix}} \) in \( 2^{O(s(n))} \) time.
Do BFS to check if \( C \) is reachable from \( C_{\text{start}} \)

DEF6: Some complexity classes of particular interest

\[
\begin{align*}
\text{PSPACE} &= \bigcup_{c \in \text{nat}} \text{SPACE}(n^c) \\
\text{NSPACE} &= \bigcup_{c \in \text{nat}} \text{NSPACE}(n^c) \\
L &= \text{SPACE}(\log n) \\
NL &= \text{NSPACE}(\log n)
\end{align*}
\]
Next time

Talk about \textit{PSPACE, NPSPACE, L, NL}

Relate to other complexity classes
such as \textit{NP}

leads to discussion of
complete problems

And more ...