Interlude

• In an ideal world, given language L
  – state an algorithm deciding L
  – prove that no algorithm does better

• we are pretty good at part 1

• we are currently completely helpless
  when it comes to part 2, for most problems that we care about
Interlude

• in place of part 2 we can
  – relate the difficulty of problems to each other via \textit{reductions}
  – prove that a problem is a “hardest” problem in a complexity class via \textit{completeness}

• powerful, successful surrogate for lower bounds
Reductions

• **reductions** are the main tool for relating problems to each other

• given two languages $L_1$ and $L_2$ we say “$L_1$ reduces to $L_2$” and we write “$L_1 \leq L_2$” to mean:
  
  – there exists an efficient (for now, poly-time) algorithm that computes a function $f$ s.t.
    
    • $x \in L_1$ implies $f(x) \in L_2$
    • $x \notin L_1$ implies $f(x) \notin L_2$
Reductions

• positive use: given new problem $L_1$ reduce it to $L_2$ that we know to be in $\mathbf{P}$. Conclude $L_1$ in $\mathbf{P}$ (how?)
  – e.g. bipartite matching $\leq$ max flow
  – formalizes “$L_1$ as easy as $L_2$”
Reductions

• **negative use**: given new problem \( L_2 \) reduce \( L_1 \) (that we believe not to be in \( P \)) to it. Conclude \( L_2 \ not \ in \ P \) if \( L_1 \ not \ in \ P \) (how?)
  
  – e.g. satisfiability \( \leq \) graph 3-coloring
  – formalizes “\( L_2 \) as hard as \( L_1 \)”
Reductions

• Example reduction:

  – \text{3SAT} = \{ \phi : \phi \text{ is a 3-CNF Boolean formula that has a satisfying assignment} \} \quad (3\text{-CNF} = \text{AND of OR of } \leq 3 \text{ literals})

  – \text{IS} = \{ (G, k) | G \text{ is a graph with an independent set } V' \subseteq V \text{ of size } \geq k \} \quad (\text{ind. set = set of vertices no two of which are connected by an edge})
Ind. Set is NP-complete

The reduction f: given

\[ \varphi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (\ldots) \]

we produce graph \( G_\varphi \):

- one triangle for each of \( m \) clauses
- edge between every pair of contradictory literals
- set \( k = m \)
Reductions

\[ \varphi = (x \lor y \lor \neg z) \land (\neg x \lor w \lor z) \land \ldots \land (\ldots) \]

- Claim: \( \varphi \) has a satisfying assignment if and only if \( G \) has an independent set of size at least \( k \)
  - proof?
- Conclude that 3SAT \( \leq \) IS.
Completeness

• complexity class $\mathcal{C}$
• language $L$ is $\mathcal{C}$-complete if
  – $L$ is in $\mathcal{C}$
  – every language in $\mathcal{C}$ reduces to $L$
• very important concept
• formalizes “$L$ is hardest problem in complexity class $\mathcal{C}$”
Completeness

• Completeness allows us to reason about the entire class by thinking about a single concrete problem

• related concept: language $L$ is **C-hard** if
  – every language in $C$ reduces to $L$
Completeness

• May ask: how to show every language in $C$ reduces to $L$?
  – in practice, shown by reducing known $C$-complete problem to $L$

  – often not hard to find “1st” $C$-complete language, but it might not be “natural”
Completeness

– Example:

\( \textbf{NP} = \) the set of languages \( L \) where

\[
L = \{ x : \exists y, |y| \leq |x|^k, (x, y) \in R \}
\]

and \( R \) is a language in \( \textbf{P} \).

one \( \textbf{NP} \)-complete language “bounded halting”:

\[
\text{BH} = \{ (M, x, 1^m) : \exists y, |y| \leq |x|^k \text{ s.t. } M \text{ accepts } (x, y) \text{ in at most } m \text{ steps} \}
\]

– challenge is to find natural complete problem

– Cook 71: SAT \( \textbf{NP} \)-complete
Summary

- problems
  - function, decision
  - language = set of strings
- complexity class = set of languages
- efficient computation identified with efficient computation on Turing Machine
  - single-tape, multi-tape
  - diagonalization technique: HALT undecidable
- TIME and SPACE classes
- reductions
- $C$-completeness, $C$-hardness
Time and Space

A motivating question:
- Boolean formula with $n$ nodes
- evaluate using $O(\log n)$ space?

• depth-first traversal requires storing intermediate values
• idea: short-circuit ANDs and ORs when possible
Time and Space

• Can we evaluate an $n$ node Boolean circuit using $O(\log n)$ space?
Time and Space

• Recall:
  – $\text{TIME}(f(n))$, $\text{SPACE}(f(n))$

• Questions:
  – how are these classes related to each other?
  – how do we define robust time and space classes?
  – what problems are contained in these classes? complete for these classes?
Outline

• Why big-oh? Linear Speedup Theorem
• Hierarchy Theorems
• Robust Time and Space Classes
• Relationships between classes
• Some complete problems
Linear Speedup

**Theorem**: Suppose TM $M$ decides language $L$ in time $f(n)$. Then for any $\varepsilon > 0$, there exists TM $M'$ that decides $L$ in time

$$\varepsilon f(n) + n + 2.$$

**Proof:**
- simple idea: increase “word length”
- $M'$ will have
  - one more tape than $M$
  - $m$-tuples of symbols of $M$
  $$\Sigma_{\text{new}} = \Sigma_{\text{old}} \cup \Sigma_{\text{old}}^m$$
- many more states
Linear Speedup

• part 1: compress input onto fresh tape

```
  a b a b b a a a
aba bba aa_
```

...
Linear Speedup

• part 2: simulate $M$, $m$ steps at a time

... $b\ b\ a\ a\ b\ a\ b\ a\ a\ a\ a\ b\ b\ \ldots$

\[
\begin{array}{cccccc}
\text{abb} & \text{aab} & \text{aba} & \text{aab} & \text{aba}
\end{array}
\]

\[\rightarrow\ m\ \rightarrow\ m\ \rightarrow\ \]

– 4 (L,R,R,L) steps to read relevant symbols, “remember” in state

– 2 (L,R or R,L) to make $M$’s changes
Linear Speedup

- accounting:
  - part 1 (copying): $n + 2$ steps
  - part 2 (simulation): $6 \frac{f(n)}{m}$
  - set $m = \frac{6}{\varepsilon}$
  - total: $\varepsilon f(n) + n + 2$

**Theorem**: Suppose TM $M$ decides language $L$ in space $f(n)$. Then for any $\varepsilon > 0$, there exists TM $M'$ that decides $L$ in space $\varepsilon f(n) + 2$.

- Proof: same.
Time and Space

• Moral: big-oh notation necessary given our model of computation
  – Recall: \( f(n) = O(g(n)) \) if there exists \( c \) such that \( f(n) \leq c g(n) \) for all sufficiently large \( n \).
  – TM model incapable of making distinctions between time and space usage that differs by a constant.

• In general: interested in course distinctions not affected by model
  – e.g. simulation of \( k \)-string TM running in time \( f(n) \) by single-string TM running in time \( O(f(n)^2) \)
Hierarchy Theorems

• Does genuinely more time permit us to decide new languages?

• how can we construct a language $L$ that is not in $\text{TIME}(f(n))$…
  • idea: same as “HALT undecidable” diagonalization and simulation
Recall proof for Halting Problem

The existence of $H$ which tells us yes/no for each box allows us to construct a TM $H'$ that cannot be in the table.
Time Hierarchy Theorem

- Turing Machines
- \( (M, x): \text{does } M \text{ accept } x \text{ in time } f(n) \)?
- \( \text{TM SIM} \) tells us yes/no for each box in time \( g(n) \)
- rows include all of \( \text{TIME}(f(n)) \)
- construct TM D running in time \( g(2n) \) that is not in table
Time Hierarchy Theorem

**Theorem** (Time Hierarchy Theorem): For every proper complexity function $f(n) \geq n$:

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n^3)).$$

- more on “proper complexity functions” later…
Proof of Time Hierarchy Theorem

• Proof:
  – SIM is TM deciding language
    \{ <M, x> : M accepts x in \leq f(|x|) \text{ steps} \}
  – Claim: SIM runs in time \( g(n) = f(n)^3 \).
  – define new TM D: on input <M>
    • if SIM accepts <M, M>, reject
    • if SIM rejects <M, M>, accept
  – D runs in time \( g(2n) \)
Proof of Time Hierarchy Theorem

• Proof (continued):
  – suppose $M$ in $\text{TIME}(f(n))$ decides $L(D)$
    • $M(<M>) = \text{SIM}(<M, M>) \neq D(<M>)$
    • but $M(<M>) = D(<M>)$
  – contradiction.
Proof of Time Hierarchy Theorem

• Claim: there is a TM SIM that decides
\[ \{<M, x> : M \text{ accepts } x \text{ in } \leq f(|x|) \text{ steps}\} \]
and runs in time \( g(n) = f(n)^3 \).

• Proof sketch: SIM has 4 work tapes
  • contents and “virtual head” positions for M’s tapes
  • M’s transition function and state
  • \( f(|x|) \) “+”s used as a clock
  • scratch space
Proof of Time Hierarchy Theorem

- contents and “virtual head” positions for M’s tapes
- M’s transition function and state
- \( f(|x|) \) “+”s used as a clock
- scratch space
  - initialize tapes
  - simulate step of M, advance head on tape 3; repeat.
  - can check running time is as claimed.
- Important detail: need to initialize tape 3 in time \( O(f(n)) \)
Proper Complexity Functions

• Definition: $f$ is a proper complexity function if
  
  – $f(n) \geq f(n-1)$ for all $n$
  
  – there exists a TM $M$ that outputs exactly $f(n)$ symbols on input $1^n$, and runs in time $O(f(n) + n)$ and space $O(f(n))$. 
Proper Complexity Functions

• includes all reasonable functions we will work with
  – $\log n$, $\sqrt{n}$, $n^2$, $2^n$, $n!$, …
  – if $f$ and $g$ are proper then $f + g$, $fg$, $f(g)$, $f^g$, $2^g$
    are all proper

• can mostly ignore, but be aware it is a genuine concern:

• Theorem: $\exists$ non-proper $f$ such that $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$. 
Hierarchy Theorems

• Does genuinely more space permit us to decide new languages?

**Theorem** (Space Hierarchy Theorem): For every proper complexity function $f(n) \geq \log n$:

$$\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n) \log f(n)).$$

• Proof: same ideas.
Robust Time and Space Classes

• What is meant by “robust” class?
  – no formal definition
  – reasonable changes to model of computation shouldn’t change class
  – should allow “modular composition” – calling subroutine in class (for classes closed under complement…)
Robust Time and Space Classes

- Robust time and space classes:

\[ L = \text{SPACE}(\log n) \]
\[ \text{PSPACE} = \bigcup_k \text{SPACE}(n^k) \]
\[ \text{P} = \bigcup_k \text{TIME}(n^k) \]
\[ \text{EXP} = \bigcup_k \text{TIME}(2^{n^k}) \]
Time and Space Classes

• Problems in these classes:

\[ L \]: FVAL, integer multiplication, most reductions…

\[ \text{PSPACE} \]: generalized geography, 2-person games…

\[ \begin{array}{c}
\lor \\
\land \\
\land \\
1 \\
0 \\
1
\end{array} \]
Time and Space Classes

**P**: CVAL, linear programming, max-flow...

```
1 0 1 0 1
\^ \v \wedge
```

**EXP**: SAT, all of NP and much more...
Relationships between classes

• How are these four classes related to each other?

• Time Hierarchy Theorem implies

\[ P \subsetneq \text{EXP} \]
\[ P \subseteq \text{TIME}(2^n) \subsetneq \text{TIME}(2^{(2n)^3}) \subseteq \text{EXP} \]

• Space Hierarchy Theorem implies

\[ L \subsetneq \text{PSPACE} \]
\[ L = \text{SPACE}(\log n) \subsetneq \text{SPACE}(\log^2 n) \subseteq \text{PSPACE} \]
Relationships between classes

• Easy: $P \subseteq \text{PSPACE}$
• $L$ vs. $P$, $\text{PSPACE}$ vs. $\text{EXP}$?
Relationships between classes

• Useful convention: *Turing Machine configurations.* Any point in computation represented by string:

\[
C = \sigma_1 \sigma_2 \ldots \sigma_i \sigma_{i+1} \sigma_{i+2} \ldots \sigma_m
\]

- start configuration for single-tape TM on input \(x\): \(q_{\text{start}}x_1x_2\ldots x_n\)
Relationships between classes

- easy to tell if C yields C’ in 1 step
- configuration graph: nodes are configurations, edge (C, C’) iff C yields C’ in one step
- # configurations for a 2-tape TM (work tape + read-only input) that runs in space $t(n)$

$$n \times t(n) \times |Q| \times |\Sigma|^t(n)$$

- input-tape head position
- work-tape head position
- state
- work-tape contents
Relationships between classes

• if $t(n) = c \log n$, at most
  \[ n \times (c \log n) \times c_0 \times c_1^c \log n \leq n^k \]
  configurations.
• can determine if reach $q_{\text{accept}}$ or $q_{\text{reject}}$
  from start configuration by exploring config. graph of size $n^k$ (e.g. by DFS)

• Conclude: $L \subseteq P$
Relationships between classes

• if $t(n) = n^c$, at most

$$n \times n^c \times c_0 \times c_1^{n^c} \leq 2^{nk}$$

configurations.

• can determine if reach $q_{\text{accept}}$ or $q_{\text{reject}}$ from start configuration by exploring config. graph of size $2^{nk}$ (e.g. by DFS)

• Conclude: $\text{PSPACE} \subseteq \text{EXP}$
Relationships between classes

• So far:

\[ L \subseteq P \subseteq \text{PSPACE} \subseteq \text{EXP} \]

• believe all containments strict

• know \( L \subsetneq \text{PSPACE}, \ P \subsetneq \text{EXP} \)

• even before any mention of NP, two **major** unsolved problems:

\[ \text{L} \ ? \ = \text{P} \qquad \text{P} \ ? \ = \text{PSPACE} \]
A $\mathbf{P}$-complete problem

- We don’t know how to prove $\mathbf{L} \neq \mathbf{P}$
- But, can identify problems in $\mathbf{P}$ least likely to be in $\mathbf{L}$ using $\mathbf{P}$- completeness.
- need stronger notion of reduction (why?)

![Diagram](attachment:image.png)

$L_1$, $L_2$
A \textbf{P}-complete problem

- **logspace reduction**: \( f \) computable by TM that uses \( O(\log n) \) space
  - denoted \( L_1 \leq_L L_2 \)

- If \( L_2 \) is \textbf{P}-complete, then \( L_2 \) in \( L \) implies \( L = \textbf{P} \) (homework problem)
A \textbf{P}-complete problem

- **Circuit Value (CVAL):** given a variable-free Boolean circuit (gates $\land$, $\lor$, $\neg$, 0, 1), does it output 1?

\textbf{Theorem}: CVAL is \textbf{P}-complete.

- **Proof:**
  - already argued in \textbf{P}
  - L arbitrary language in \textbf{P}, TM M decides L in $n^c$ steps
A $\mathbf{P}$-complete problem

- **Tableau** (configurations written in an array) for machine $M$ on input $w$:

<table>
<thead>
<tr>
<th>$w_1/q_s$</th>
<th>$w_2$</th>
<th>$\ldots$</th>
<th>$w_n$</th>
<th>$\ldots$</th>
<th>$___$</th>
<th>$___$</th>
<th>$___$</th>
<th>$___$</th>
<th>$___$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_1$</td>
<td>$w_2/q_1$</td>
<td>$\ldots$</td>
<td>$w_n$</td>
<td>$\ldots$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
</tr>
<tr>
<td>$w_1/q_1$</td>
<td>$a$</td>
<td>$\ldots$</td>
<td>$w_n$</td>
<td>$\ldots$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
</tr>
<tr>
<td>$/_q_a$</td>
<td>$_$</td>
<td>$\ldots$</td>
<td>$_$</td>
<td>$\ldots$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
<td>$___$</td>
</tr>
</tbody>
</table>

- **height** = time taken = $|w|^c$
- **width** = space used $\leq |w|^c$
A \textbf{P}-complete problem

• Important observation: contents of cell in tableau determined by 3 others above it:
A \textbf{P}-complete problem

• Can build Boolean circuit STEP
  – input (binary encoding of) 3 cells
  – output (binary encoding of) 1 cell

\begin{array}{ccc}
\text{a} & \text{b/q}_1 & \text{a} \\
\hline
\text{STEP} & \hline
\text{a} & \hline
\end{array}

• each output bit is some function of inputs
• can build circuit for each
• size is independent of size of tableau
A $P$-complete problem

Tableau for $M$ on input $w$

\[
\begin{array}{cccc}
\frac{w_1}{q_s} & w_2 & \ldots & w_n \\
 w_1 & \frac{w_2}{q_1} & \ldots & w_n \\
 \vdots & \vdots & \ddots & \vdots \\
\end{array}
\]

- $|w|^c$ copies of STEP compute row $i$ from $i-1$
A P-complete problem

This circuit $C_{M, w}$ has inputs $w_1w_2\ldots w_n$ and $C(w) = 1$ iff $M$ accepts input $w$.

Logspace reduction

Size = $O(|w|^{2c})$

1 iff cell contains $q_{accept}$
Answer to question

• Can we evaluate an n node Boolean circuit using $O(\log n)$ space?

• **NO!** (probably)

• **CVAL in L if and only if** $L = P$
Padding and succinctness

Two consequences of measuring running time as function of input length:

• “padding”
  – suppose $L \in \text{EXP}$, and define
    $$\text{PAD}_L = \{ x#^N : x \in L, N = 2^{|x|^k} \}$$
  – TM that decides $\text{PAD}_L$: ensure suffix of $N$ #s, ignore #s, then simulate TM that decides $L$
  – running time now polynomial!
Padding and succinctness

• converse (intuition only): “succinctness”
  – suppose $L$ is $P$-complete
  – intuitively, some inputs are “hard” -- require full power of $P$
  – $SUCCINCT_L$ has inputs encoded in different form than $L$, some exponentially shorter
  – if “hard” inputs are exponentially shorter, then candidate to be $EXP$-complete
Succinct encodings

• succinct encoding for a directed graph $G = (V = \{1, 2, 3, \ldots\}, E)$:
  
  1 iff $(i, j) \in E$
  
• a succinct encoding for a variable-free Boolean circuit:

  1 iff wire from gate $i$ to gate $j$
  
  type of gate $i$
  
  type of gate $j$
An EXP-complete problem

• **Succinct Circuit Value**: given a *succinctly encoded* variable-free Boolean circuit (gates $\land$, $\lor$, $\neg$, 0, 1), does it output 1?

**Theorem**: Succinct Circuit Value is EXP-complete.

• **Proof**:
  – in EXP (why?)
  – $L$ arbitrary language in EXP, TM $M$ decides $L$ in $2^{n^k}$ steps
An $\text{EXP}$-complete problem

- **tableau** for input $x = x_1x_2x_3\ldots x_n$:

  $\begin{array}{c}
x \\
\hline
_ _ _ _ _ \\
_ _ _ _ _
\end{array}$

  height, width $2^{nk}$

- Circuit C from CVAL reduction has size $O(2^{nk})$.

- TM M accepts input $x$ iff circuit outputs 1
An **EXP**-complete problem

– Can encode C succinctly:

1 iff wire from gate i to gate j

<table>
<thead>
<tr>
<th>type of gate i</th>
<th>type of gate j</th>
</tr>
</thead>
</table>

• if i, j within single STEP circuit, easy to compute output

• if i, j between two STEP circuits, easy to compute output

• if one of i, j refers to input gates, consult x to compute output
Summary

• Remaining TM details: big-oh necessary.
• First complexity classes:
  \[ L, P, \text{PSPACE}, \text{EXP} \]
• First separations (via simulation and diagonalization):
  \[ P \neq \text{EXP}, L \neq \text{PSPACE} \]
• First major open questions:
  \[ L \overset{?}{=} P \quad P \overset{?}{=} \text{PSPACE} \]
• First complete problems:
  – \text{CVAL} is \text{P}-complete
  – \text{Succinct CVAL} is \text{EXP}-complete
Summary

EXP
PSPACE
P
L