CS151
Complexity Theory

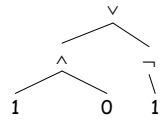
Lecture 2

April 1, 2004

Time and Space

A motivating question:

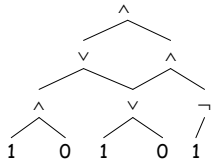
- Boolean formula with n nodes
- evaluate using O(log n) space?



- 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:
  - **TIME(f(n))**, **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 $\epsilon > 0$, there exists TM M' that decides L in time

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

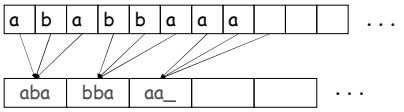
- Proof:
  - simple idea: increase "word length"
  - M' will have
    - one more tape than M
    - m-tuples of symbols of M

$$\Sigma_{new} = \Sigma_{old} \cup \Sigma_{old}^{m}$$

    - many more states

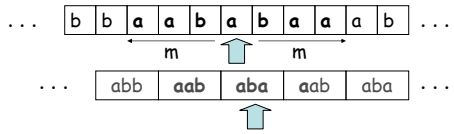
Linear Speedup

- part 1: compress input onto fresh tape



Linear Speedup

- part 2: simulate M, m steps at a time



  - 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 (f(n)/m)
  - set m = 6/ϵ
  - total: ϵf(n) + n + 2

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

- Proof: same.

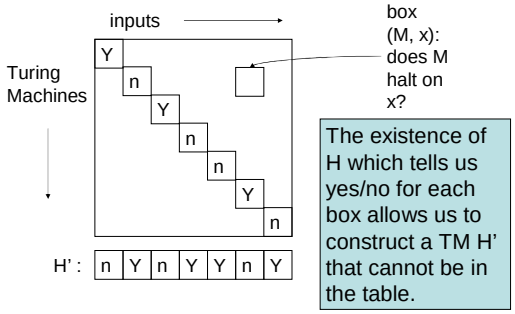
Time and Space

- Moral: big-oh notation **necessary** given our model of computation
  - Recall: f(n) = O(g(n)) if there exists c such that f(n) ≤ 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

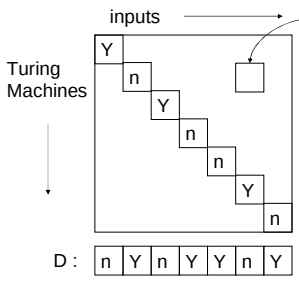
- Does **genuinely** more time permit us to decide new languages?
- how can we construct a language L that is *not* in **TIME(f(n))…**
  - idea: same as "HALT undecidable" diagonalization and simulation

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



• TM SIM tells us yes/no for each box **in time g(n)**
• rows include all of **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) ≥ n:
TIME(f(n)) ⊊ 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 ≤ f(|x|) 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 **TIME(f(n))** decides L(D)
    - M(<M>) = SIM(<M, M>) ≠ D(<M>)
    - but M(<M>) = D(<M>)
  - contradiction.

Proof of Time Hierarchy Theorem

- Claim: there is a TM SIM that decides
  {<M, x> : M accepts x in ≤ f(|x|) 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) + n)

Proper Complexity Functions

- Definition: f is a **proper complexity function** if
  - f(n) ≥ 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: ∃ non-proper f such that **TIME(f(n))** = **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) ≥ log n:

SPACE(f(n)) ⊊ **SPACE(f(n)logf(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 = **SPACE(log n)**

PSPACE = $\cup_k$ **SPACE(n^k)**

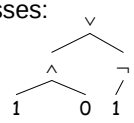
P = $\cup_k$ **TIME(n^k)**

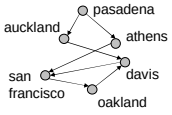
EXP = $\cup_k$ **TIME(2^{n^k})**

Time and Space Classes

- Problems in these classes:

L : FVAL, integer multiplication, most reductions…

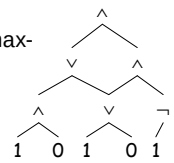




PSPACE : generalized geography, 2-person games…

Time and Space Classes

P : CVAL, linear programming, max-flow…



EXP : SAT, all of NP and much more…

Relationships between classes

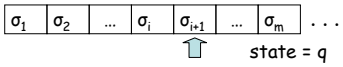
- How are these four classes related to each other?
- Time Hierarchy Theorem implies
  $$\mathbf{P} \subsetneq \mathbf{EXP}$$
  – $P \subset TIME(2^n) \subsetneq TIME(2^{(2n)3}) \subset EXP$
- Space Hierarchy Theorem implies
  $$\mathbf{L} \subsetneq \mathbf{PSPACE}$$
  – $L=SPACE(\log n) \subsetneq SPACE(\log^2 n) \subset PSPACE$

Relationships between classes

- Easy: $\mathbf{P} \subset \mathbf{PSPACE}$
- L vs. P, PSPACE vs. EXP ?

Relationships between classes

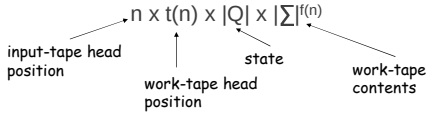
- Useful convention: Turing Machine configurations. Any point in computation



  represented by string:
  $$C = \sigma_1 \sigma_2 \ldots \sigma_i \mathbf{q} \sigma_{i+1} \sigma_{i+2} \ldots \sigma_m$$
- start configuration for single-tape TM on input x: $\mathbf{q_{start}} x_1 x_2 \ldots x_n$

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)



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 $\mathbf{q_{accept}}$ or $\mathbf{q_{reject}}$ from start configuration by exploring config. graph of size n^k (e.g. by DFS)
- Conclude: $\mathbf{L} \subset \mathbf{P}$

Relationships between classes

- if $t(n) = n^c$, at most
  $$n \times n^c \times c_0 \times c_1^{n^c} \leq 2^{n^k}$$
  configurations.
- can determine if reach $\mathbf{q_{accept}}$ or $\mathbf{q_{reject}}$ from start configuration by exploring config. graph of size 2^{n^k} (e.g. by DFS)
- Conclude: **PSPACE** $\subset$ **EXP**

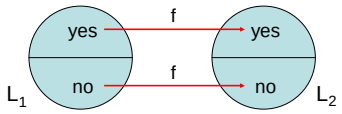
Relationships between classes

- So far:
  $$\mathbf{L} \subset \mathbf{P} \subset \mathbf{PSPACE} \subset \mathbf{EXP}$$
- believe all containments strict
- know $\mathbf{L} \subsetneq \mathbf{PSPACE}$, $\mathbf{P} \subsetneq \mathbf{EXP}$
- even before any mention of NP, two **major** unsolved problems:

$$\mathbf{L} \stackrel{?}{=} \mathbf{P} \qquad \mathbf{P} \stackrel{?}{=} \mathbf{PSPACE}$$

A **P**-complete problem

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



A **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 **P**-complete, then L_2 in **L** implies **L** = **P** (homework problem)

A **P**-complete problem

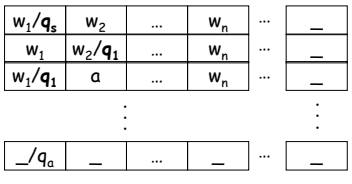
- Circuit Value (CVAL): given a variable-free Boolean circuit (gates $\wedge$, $\vee$, $\neg$, 0, 1), does it output 1?

Theorem: CVAL is **P**-complete.

- Proof:
  - already argued in **P**
  - L arbitrary language in **P,** TM M decides L in n^k steps

A **P**-complete problem

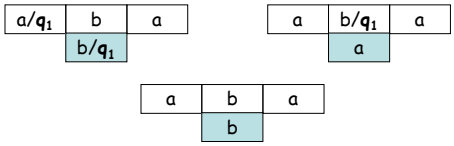
- Tableau (configurations written in an array) for machine M on input w:



- height = time taken = $|w|^c$
- width = space used $\leq |w|^c$

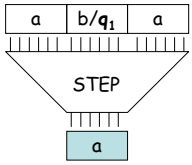
A **P**-complete problem

- Important observation: contents of cell in tableau determined by 3 others above it:



A **P**-complete problem

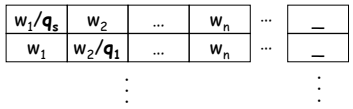
- Can build Boolean circuit STEP
  - input (binary encoding of) 3 cells
  - output (binary encoding of) 1 cell



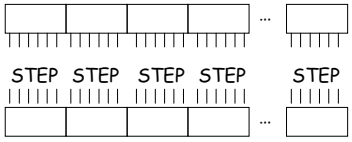
- each output bit is some function of inputs
- can build circuit for each
- size is independent of size of tableau

A **P**-complete problem

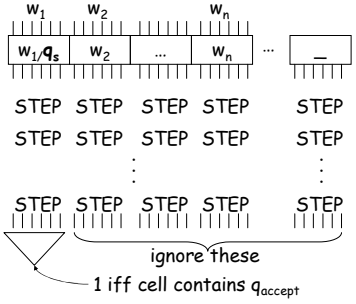
Tableau for M on input w



- $|w|^c$ copies of STEP compute row i from i-1



A **P**-complete problem



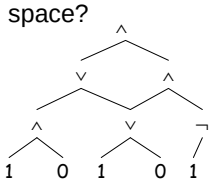
This circuit $C_{M, w}$ has inputs $w_1w_2…w_n$ and C(w) = 1 iff M accepts input w.

logspace reduction

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

Answer to question

- Can we evaluate an n node Boolean **circuit** using O(log n) space?
- NO! (probably)
- CVAL in **P** if and only if **L** = **P**



Padding and succinctness

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

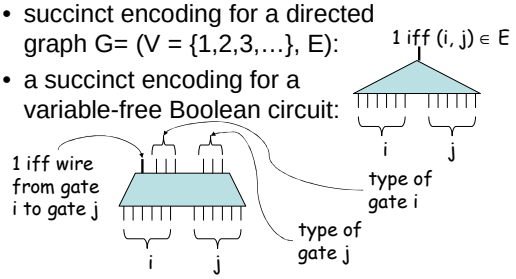
- "padding"
  - suppose L ∈ **EXP**, define
    $PAD_L = \{ x\#^N : x \in L, N = 2^{|x|^k} \}$
  - same TM decides L (ignore #s)
  - running time now polynomial !

Padding and succinctness

- converse: "succinctness"
  - suppose L is **P-complete**
  - intuitively, some inputs are "hard" -- require full power of **P**
  - $SUCCINCT_L$ = input encoded in exponentially shorter form than L
  - if "hard" inputs encodable this way, then candidate to be **EXP**-complete

An **EXP**-complete problem

- succinct encoding for a directed graph G= (V = {1,2,3,…}, E):
- a succinct encoding for a variable-free Boolean circuit:



An **EXP**-complete problem

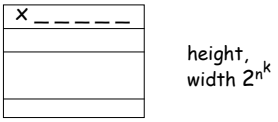
- Succinct Circuit Value: given a **succinctly encoded** variable-free Boolean circuit (gates $\wedge$, $\vee$, $\neg$, 0, 1), does it output 1?

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 **EXP**-complete problem

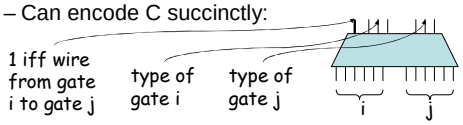
- **tableau** for input $x = x_1x_2x_3…x_n$:



- Circuit C from CVAL reduction has size $O(2^{2n^k})$.
- TM M accepts input x iff circuit outputs 1

An **EXP**-complete problem

- Can encode C succinctly:



  - 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, PSPACE, EXP
- First separations (via simulation and diagonalization):

  $P \neq EXP$, $L \neq PSPACE$
- First major open questions:

  $L \stackrel{?}{=} P$ $P \stackrel{?}{=} PSPACE$
- First complete problems:
  - CVAL is **P**-complete
  - Succinct CVAL is **EXP**-complete

Summary