BU CS 332 – Theory of Computation

https://forms.gle/muSXckPgw2Rw97rz6

Lecture 23:

- NP-completeness example
- Space complexity

Reading:

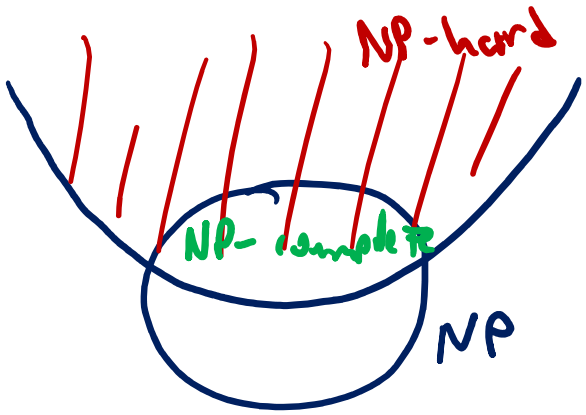
Sipser Ch 7.4-7.5,

8.1-8.2

Alexander Poremba & Mark Bun

April 28, 2026

NP-completeness review



Definition: A language B is NP-complete if

1) $B \in \mathrm{NP}$, and
2) **Every** language $A \in \mathrm{NP}$ is poly-time reducible to B, i.e., $A \leq_{\mathrm{p}} B$ ("B is NP-hard")

The usual way to prove NP-completeness:

If

1) $C \in \mathrm{NP}$, and
2) There is an NP-complete language B (e.g., 3-SAT, VERTEX-COVER, IND-SET, …) such that $B \leq_{\mathrm{p}} C$

then C is NP-complete.

∃ a poly-time comp. f s.t. ∀ w, w∈B ⇔ f(w)∈C

Subset Sum

SUBSET-SUM = $\{\langle w_1, \ldots, w_m, t\rangle \mid$
there exists a subset of natural numbers $w_1, \ldots, w_m$ that sum to $t\}$

Ex: $\langle 1, 3, 5, 7, 11\rangle \in$ SUBSET-SUM because $1+3+7=11$

but $\langle 1, 3, 5, 7, 256\rangle \notin$ SUBSET-SUM

Theorem: SUBSET-SUM is NP-complete

Claim 1: SUBSET-SUM is in NP

Verifier: Certificate C describes a subset $S \subseteq [m]$ of indices

On input $\langle w_1, \ldots, t\rangle$, C):
Accept iff $\sum_{w_i \in S} = t$

Claim 2: SUBSET-SUM is NP-hard

Prove that 3SAT $\leq_p$ SUBSET-SUM

$3SAT \leq_p$ SUBSET−SUM

poly-time alg. performs this

Goal: Given a 3CNF formula φ on v variables and k clauses, construct a SUBSET-SUM instance $w_1, \ldots, w_m, t$ such that

φ is satisfiable iff there exists a subset of $w_1, \ldots, w_m$ that sum to t

$x_1, \bar{x}_1, x_2, \bar{x}_2, \ldots$

First attempt: Encode each literal ℓ of φ as a k-digit *decimal* number $w_\ell = c_1 \ldots c_k$ where

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ 0 \text{ otherwise} \end{cases}$$

Example first attempt reduction

$$\varphi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

clause 1 clause 2

Encode literal ℓ as $w_\ell = c_1 \ldots c_k$ where

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ 0 \text{ otherwise} \end{cases}$$

$\ell \mapsto w_\ell =$	c_1	c_2
x_1	0	1
$\overline{x_1}$	1	0
x_2	1	0
$\overline{x_2}$	0	1
x_3	1	1
$\overline{x_3}$	0	0

Satisfying assignment:

$x_1 = 0 \quad x_2 = 0 \quad x_3 = 1$

$$\begin{array}{r} 1\,0 \\ 0\,1 \\ +\ 1\,1 \\ \hline 2\,2 \end{array}$$

w_ℓ's for literals corresponding to a sat. assignment add up to ≥ 11 "digitwise"

$3SAT \leq_p$ SUBSET−SUM

First attempt: Encode each literal ℓ of φ as a k-digit *decimal* number $w_\ell = c_1 \ldots c_k$ where

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ 0 \text{ otherwise} \end{cases}$$

Claim: If φ is satisfiable, then there exists a subset of the w_ℓ's that "digit-wise" add up to "at least" $111 \ldots 11$

Two issues:

1) Need to enforce that exactly one of $\ell, \bar{\ell}$ is set to 1
2) Need the subset to add up to **exactly** some target

$3SAT \leq_p$ SUBSET−SUM

vars

clauses

Actual reduction: Encode each literal ℓ of φ as a $(v+k)$-digit *decimal* number $w_\ell = b_1 \ldots b_v | \, c_1 \ldots c_k$ where

$$b_i = \begin{cases} 1 \text{ if } \ell \in \{x_i, \overline{x_i}\} \\ 0 \text{ otherwise} \end{cases} \qquad c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ 0 \text{ otherwise} \end{cases}$$

Enforce choosing exactly one of each var or its negation

Also, include two copies each of 000…0|100…0, 000…0|010…0, … 000…0|0…01

"padding" numbers enable adding up to exactly on target

Claim: φ is satisfiable if and only if there exists a subset of the numbers that add up to $t = 111 \ldots 11 | 333 \ldots 33$

Example of the reduction

$$\varphi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

Encode literal ℓ as $w_\ell = b_1 \dots b_v | c_1 \dots c_k$ where

$$b_i = \begin{cases} 1 \text{ if } \ell \in \{x_i, \overline{x_i}\} \\ 0 \text{ otherwise} \end{cases}$$

$$c_i = \begin{cases} 1 \text{ if } \ell \text{ appears in clause } i \\ 0 \text{ otherwise} \end{cases}$$

Include two copies each of 000…0|100…0, 000…0|010…0, … 000…0|0…01

$\ell \mapsto w_\ell =$	b_1	b_2	b_3	c_1	c_2
x_1	1	0	0	0	1
$\overline{x_1}$	1	0	0	1	0
x_2	0	1	0	1	0
$\overline{x_2}$	0	1	0	0	1
x_3	0	0	1	1	1
$\overline{x_3}$	0	0	1	0	0

Satisfying assignment:

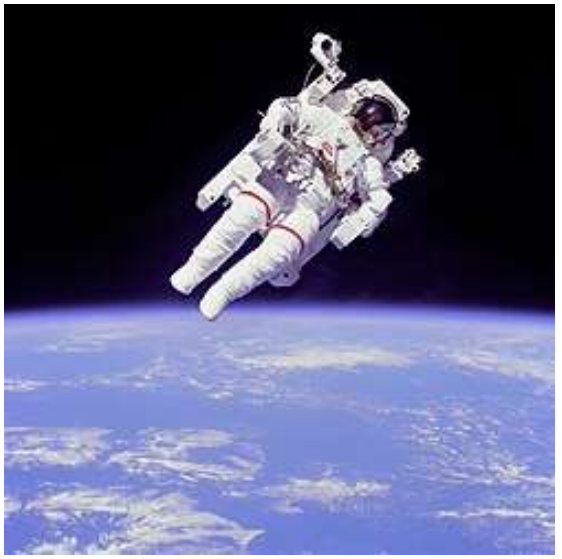
$x_1 = 0 \quad x_2 = 0 \quad x_3 = 1$

$p_{11} =$	0	0	0	1	0
$p_{12} =$	0	0	0	1	0
$p_{21} =$	0	0	0	0	1
$p_{22} =$	0	0	0	0	1
Target t =	1	1	1	3	3

1 0 0 1 0
0 1 0 0 1
0 0 1 1 1
0 0 0 1 0
0 0 0 0 1
1 1 1 3 3

Ensure at least one literal in each clause is "on"

A Brief Tour of Space (Complexity)



Space analysis

Space complexity of a TM (algorithm) = maximum number of tape cells it uses on a worst-case input

Formally: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. A TM M runs in space $f(n)$ if on every input $w \in \Sigma^*$, M halts on w using at most $f(|w|)$ cells

For nondeterministic machines: Let $f : \mathbb{N} \rightarrow \mathbb{N}$. An NTM N runs in space $f(n)$ if on every input $w \in \Sigma^*$, N halts on w using at most $f(|w|)$ cells on every computational branch

Space complexity classes

Let $f : \mathbb{N} \rightarrow \mathbb{N}$

A language $A \in \mathrm{SPACE}(f(n))$ if there exists a basic single-tape (deterministic) TM M that

1) Decides A, and
2) Runs in space $O(f(n))$

A language $A \in \mathrm{NSPACE}(f(n))$ if there exists a single-tape nondeterministic TM N that

1) Decides A, and
2) Runs in space $O(f(n))$

Example: Space complexity of SAT

Theorem: $SAT \in \mathrm{SPACE}(n)$

Proof: The following deterministic TM decides SAT using linear space

On input $\langle \varphi \rangle$ where φ is a Boolean formula:

1. For each truth assignment to the variables $x_1, \ldots, x_m$ of φ:
2. Evaluate φ on $x_1, \ldots, x_m$
3. If any evaluation $= 1$, accept. Else, reject.

Runs in $2^{O(n)}$ time but only uses linear space

← Can reuse same space

← Uses linear space

Space vs. Time

$$\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$$

Simulating Time With Square-Root Space*

Ryan Williams†
MIT

Feb 2025

Abstract

We show that for all functions $t(n) \geq n$, every multitape Turing machine running in time t can be simulated in space only $O(\sqrt{t \log t})$. This is a substantial improvement over Hopcroft, Paul, and Valiant's simulation of time t in $O(t/\log t)$ space from 50 years ago [FOCS 1975, JACM 1977].

$\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(\sqrt{f(n) \log f(n)})$ on multi-tape TMs

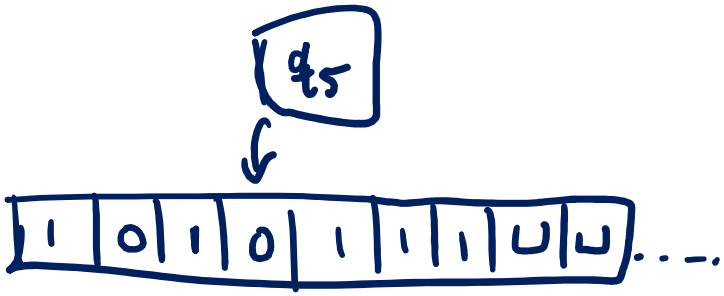
How about the opposite direction? Can low-space algorithms be simulated by low-time algorithms?

Reminder: Configurations

A configuration is a string uqv where $q \in Q$ and $u, v \in \Gamma^*$

- Tape contents = uv (followed by blanks ⊔)
- Current state = q
- Tape head on first symbol of v

Example: $101q_50111$



Start configuration: q_0w

Accepting configuration: $q = q_{\text{accept}}$

Rejecting configuration: $q = q_{\text{reject}}$

Reminder: Configurations

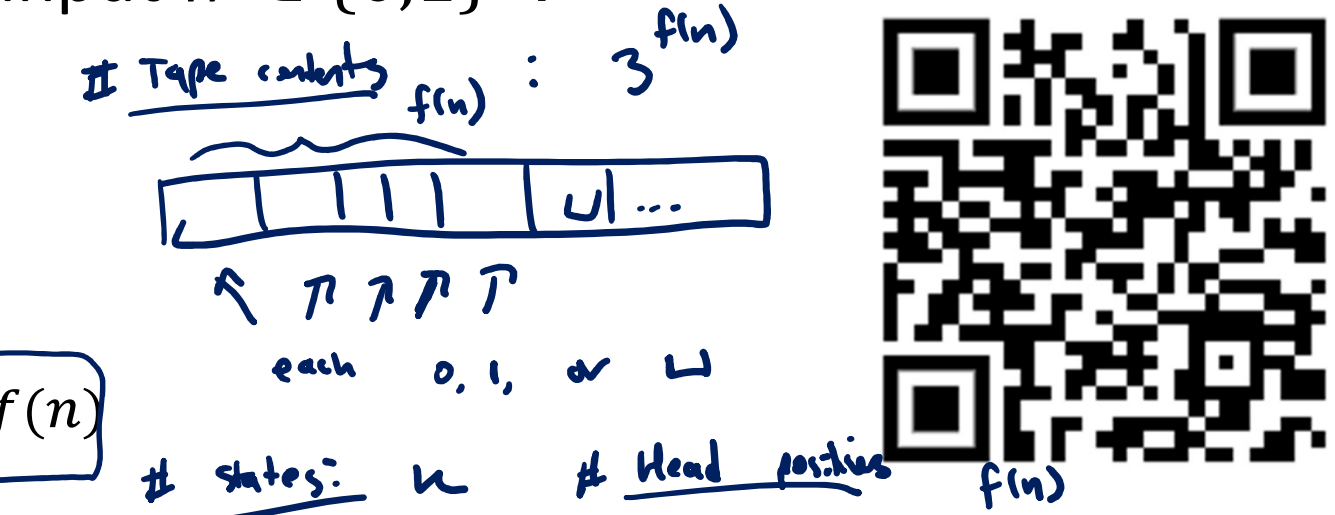
Consider a TM with k states, tape alphabet $\{0, 1, \sqcup\}$, and space bound $f(n)$. How many configurations are possible when this TM is run on an input $w \in \{0,1\}^n$?

a) $3k \cdot f(n)$

b) $k + f(n)$

c) $k \cdot 3^{f(n)}$

d) $k \cdot f(n) \cdot 3^{f(n)}$



$= 2^{O(f(n))}$

Observation: If a TM enters the same configuration twice when run on input w, it loops forever

Corollary: A TM running in space $f(n)$ also runs in time $2^{O(f(n))}$

$\Rightarrow$ SPACE($f(n)$) $\subseteq$ TIME($2^{O(f(n))}$)

Space vs. Time

$$\mathrm{TIME}\big(f(n)\big) \subseteq \mathrm{NTIME}\big(f(n)\big)$$
$$\subseteq \mathrm{SPACE}\big(f(n)\big) \subseteq \mathrm{NSPACE}\big(f(n)\big)$$
$$\subseteq \mathrm{TIME}\big(2^{O(f(n))}\big)$$

Now how about the relationship between SPACE and NSPACE?

Savitch's Theorem: Deterministic vs. Nondeterministic Space

1970

Theorem: Let f be a function with $f(n) \geq n$. Then $NSPACE(f(n)) \subseteq SPACE\left(\left(f(n)\right)^2\right)$.

Proof idea:

- Let N be an NTM deciding A in space $f(n)$
- We construct a TM M deciding A in space $O\left(\left(f(n)\right)^2\right)$
- Actually solve a more general problem:
  - Given configurations c_1, c_2 of N and natural number t, decide whether N can go from configuration c_1 to c_2 in $\leq t$ steps on some nondeterministic path
  - Procedure called CANYIELD(c_1, c_2, t)

Savitch's Theorem

Theorem: Let f be a function with $f(n) \geq n$. Then $NSPACE(f(n)) \subseteq SPACE\left((f(n))^2\right)$.

Proof idea:

- Let N be an NTM deciding A in space $f(n)$

M = "On input w:

1. Output the result of CANYIELD$(c_1, c_2, 2^{Cf(n)})$"

start config of N

accept config of N

upper bound on # of N's steps

where CANYIELD(c_1, c_2, t) decides whether N can go from configuration c_1 to c_2 in $\leq t$ steps on some nondeterministic path

WTS

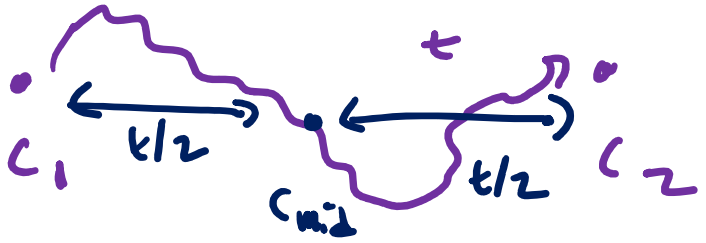
can be implemented in $O(f(n)^2)$ space

Savitch's Theorem

CANYIELD(c_1, c_2, t) decides whether N can go from configuration c_1 to c_2 in $\leq t$ steps on some nondeterministic path:

CANYIELD(c_1, c_2, t) =

1. If $t = 1$, accept if $c_1 = c_2$ or c_1 yields c_2 in one transition.
   Else, reject.
2. If $t > 1$, then for each config c_{mid} of N with $\leq f(n)$ cells:
3. Run CANYIELD$(c_1,\ c_{mid},\ t/2)$.
4. Run CANYIELD$(c_{mid},\ c_2,\ t/2)$.
5. If both runs accept, accept.
6. Reject.



$S(t) = S(t/2) + O(f(n))$ ← stores c_{mid}

$= \log t \cdot O(f(n))$

Complexity class PSPACE

Definition: PSPACE is the class of languages decidable in polynomial space on a basic single-tape (deterministic) TM

$$\mathrm{PSPACE} = \bigcup_{k=1}^{\infty} \mathrm{SPACE}(n^k)$$

Definition: $\mathrm{NPSPACE}$ is the class of languages decidable in polynomial space on a single-tape (nondeterministic) TM

$$\mathrm{NPSPACE} = \bigcup_{k=1}^{\infty} \mathrm{NSPACE}(n^k)$$

Relationships between complexity classes

1. $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$

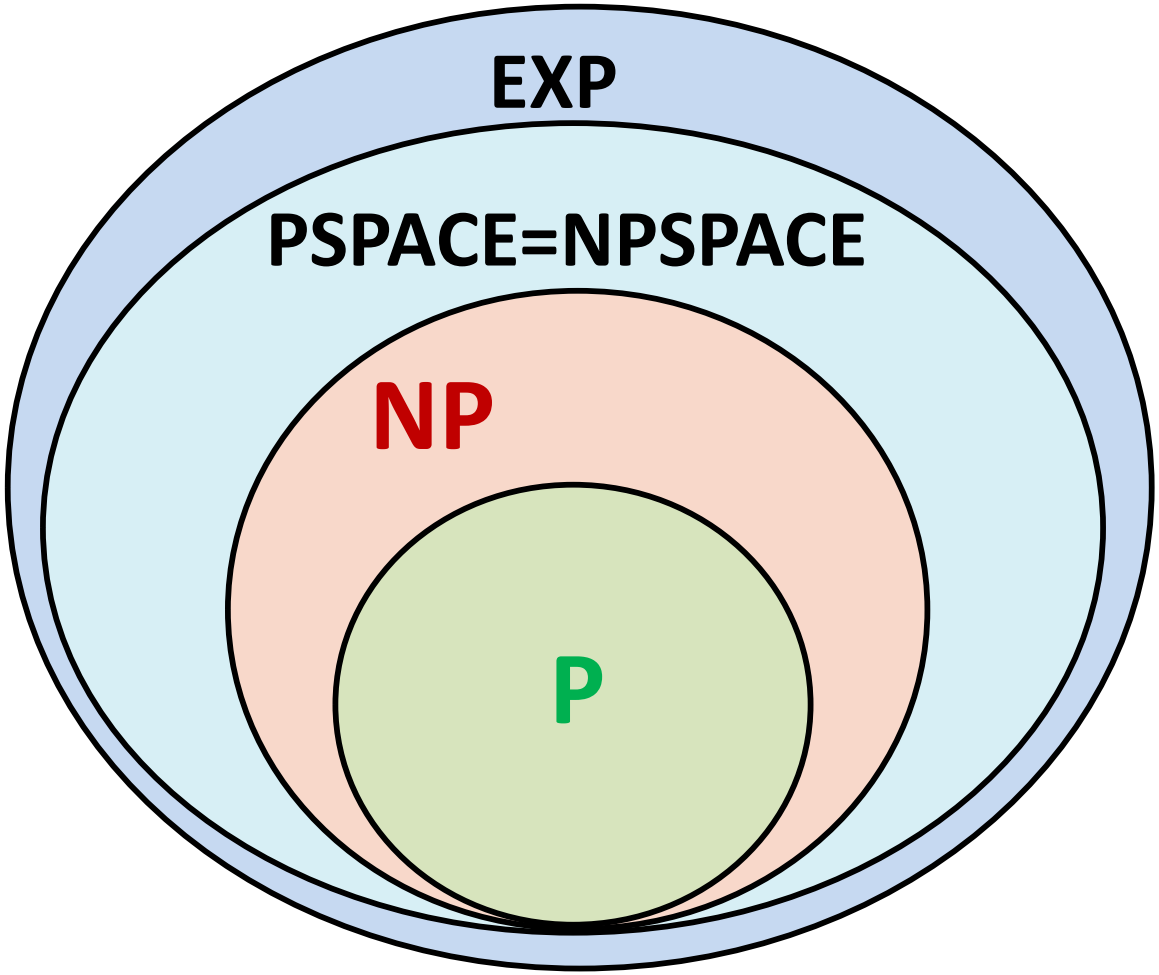
since $\mathrm{SPACE}\big(f(n)\big) \subseteq \mathrm{TIME}(2^{O(f(n))})$

2. $\mathrm{P} \neq \mathrm{EXP}$

   (via time hierarchy)

Which containments in (1) are proper?

Unknown!



Things we didn't get to talk about

- Details of polynomial and logarithmic space
- Alternating TMs, polynomial hierarchy
- Relativization and the limits of diagonalization
- Boolean circuits
- Randomized algorithms / complexity classes
- Interactive and probabilistic proof systems
- Complexity of counting

For all of this and more, Mark is teaching CS 535 in Fall '26

https://cs-people.bu.edu/mbun/courses/535_F23

Course Evaluations

bu.edu/courseeval/