BU CS 332 – Theory of Computation

https://forms.gle/z4RnBWnou8DU9Dsu8

Lecture 24:

- NP-completeness example
- Space complexity

Mark Bun

December 7, 2021

Reading:

Sipser Ch 7.4-7.5,

8.1-8.2

HW 10 : Th. 12/9

In class final:
Wed 12/15 3-5

Take-home final:
Th 12/16 (11:59 PM)

NP-completeness review

Definition: A language B is NP-complete if

      1) $B \in \text{NP}$, and

      2) Every language $A \in \text{NP}$ is poly-time reducible to

      B, i.e., $A \leq_{\text{p}} B$ ("B is NP-hard")

The usual way to prove NP-completeness:

If

      1) $C \in \text{NP}$, and

      2) There is an NP-complete language B (e.g., SAT,

          VERTEX-COVER, IND-SET, ...) such that $B \leq_{\text{p}} C$

then C is NP-complete.

Subset Sum

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

Ex: $\langle 3, 7, 8, 4, 4, 14 \rangle \in$ SUBSET - SUM
$\quad\quad w_1 \; w_2 \; \dots w_5 \quad t$

because $\quad w_1 + w_2 + w_5 = 3 + 7 + 4 = 14$

Theorem: SUBSET-SUM is NP-complete

Claim 1: SUBSET-SUM is in NP

NTM: Nondeterministically guess set of indices $S \subseteq [m]$
$\quad\quad$ Check $\sum_{i \in S} w_i = t$

Claim 2: SUBSET-SUM is NP-hard $\quad$ 3-SAT $\leq_P$ SUBSET-SUM

3SAT $\leq_p$ SUBSET–SUM

$\varphi = (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x}_2 \vee x_3)$

$v = 3 \qquad k = 2$

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

$x_1, \overline{x}_1, \ldots, x_3, \overline{x}_3$

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 of the first attempt reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3)$$

Clause 1: 1 ∨ 0 ∨ 0

Clause 2: 0 ∨ 1 ∨ 0

Clause 1

Clause 2

0 ∨ 0 ∨ 1
1 ∨ 1 ∨ 0

1 ∨ 1 ∨ 1
0 ∨ 0 ∨ 0

Satisfying		Not satisfying
$x_1 = 0$	1	0
$x_2 = 0$	0	1
$x_3 = 0$	1	0

$x_1 \rightarrow 0\ 1$

$\overline{x_1} \rightarrow 1\ 0$ ←

$x_2 \rightarrow 1\ 0$ ←

$x_2 \rightarrow 0\ 1$

$\overline{x_3} \rightarrow 1\ 1$

$x_3 \rightarrow 0\ 0$ ←
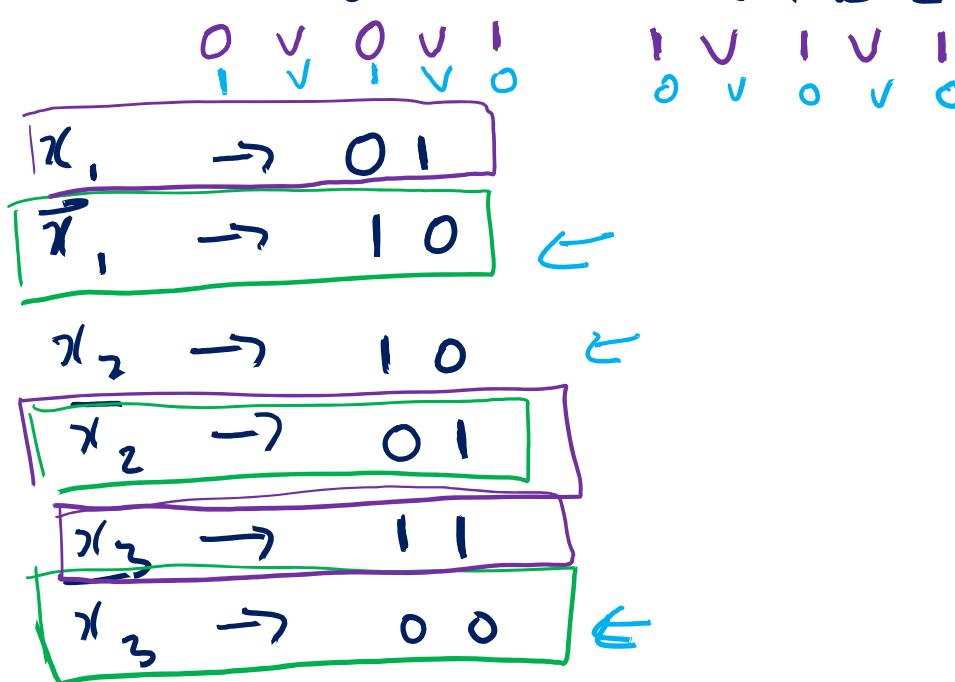
1 1

1 3

2 0

Observation: "digit wise" sum is at least 1 for a satisfying assm't

3SAT $\leq_p$ SUBSET–SUM

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

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

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

Two issues:

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

3SAT $\leq_p$ SUBSET–SUM

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

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

— Forces us to set exactly one of ℓ or $\bar{\ell}$

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

Claim: φ is satisfiable if and only if there exists a subset of the numbers that add up to $t = 111\ldots11|333\ldots33$

forces exactly one of $\ell, \bar{\ell}$ achievable iff ∃ subset of literals adding up to ≥1 digitwise

Example of the reduction

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

Handwritten above literals: 0 ✓ 0 ✓ 1   1 ✓ 1 ✓ 1

Handwritten (green): $v = 3$ $k = 2$

Handwritten (green): $x_1 = 1, \ x_2 = 0, \ x_3 = 1$

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

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

Also, include two copies each of 000...0|100...0, 000...0|010...0, ... 000...0|0...01 $\Big\}$ p's

Handwritten (green): $t = 1 1 1 1 1 \ldots 1 | 3 3 3 \ldots 3$

Example of the reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3)$$

	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	←
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
t	1	1	1		3	3

Converting a satisfying assignment to a subset summing to t:

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

Non sat assignment

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

Also need to show:

∃ subset adding up to t ⟹ ∃ satisfying assignment

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

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

Space complexity classes

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

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

1) Decides A, and

2) Runs in space $O(f(n))$

A language $A \in \text{NSPACE}(f(n))$ 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: *Takes $2^{O(m)}$ time, but only $O(m)$ space*

    2.   Evaluate φ on $x_1, \ldots, x_m$ ← *takes $O(|\varphi|)$ Space*

    3. If any evaluation $= 1$, accept. Else, reject.

Space vs. Time

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

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: $101q_5 0111$

Start configuration: $q_0 w$

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

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

Reminder: Configurations

$\{0, 1, \sqcup\}$

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

a) ~~$2kn$~~ $3k\, f(n)$

b) ~~$k + n$~~ $k + f(n)$

c) ~~$k2^n$~~ $k\, 3^{f(n)}$

d) ~~$kn2^n$~~ $k\, f(n)\, 3^{f(n)}$

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

(Configuration:
- state — k options
- head location — $f(n)$
- tape content — $3^{f(n)}$
)

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

Reminder: Configurations

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

a) $2kn$

b) $k + n$

c) $k2^n$

d) $kn2^n$ $k \cdot f(n) \cdot 3^{f(n)}$

If a TM does not loop forever, it enters every configuration at most once

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 $2^{O(f(n))}$ $=$ # of configs

Space vs. Time

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

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
$$NSPACE(f(n)) \subseteq SPACE\left((f(n))^2\right).$$

$$\left[cf. \quad NTIME(f(n)) \subseteq TIME(2^{O(f(n))}) \right]$$

Proof idea:

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

- We construct a TM M deciding A in space $O\left((f(n))^2\right)$

- Actually solve a more general problem:
  - Given configurations c_1, c_2 of N and natural number t, decide whether there exists a nondeterministic path N can follow from c_1 to c_2 in $\leq t$ steps.
  - Procedure CANYIELD(c_1, c_2, t)

Savitch's Theorem

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

Proof idea:

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

M = "On input w:

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

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

Savitch's Theorem

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

$\text{CANYIELD}(c_1, c_2, t)$ =

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 $\text{CANYIELD}(\langle c_1, c_{mid}, t/2 \rangle)$.

4.   Run $\text{CANYIELD}(\langle c_{mid}, c_2, t/2 \rangle)$.

5.   If both runs accept, accept.

6.   Reject.

Complexity class PSPACE

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

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

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

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

Savitch: PSPACE = NPSPACE

Relationships between complexity classes

1. $P \subseteq NP \subseteq PSPACE \subseteq EXP$

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

2. $P \neq EXP$

   (via time hierarchy)

Which containments
 in (1) are proper?
 Unknown!

EXP

PSPACE=NPSPACE

NP

P
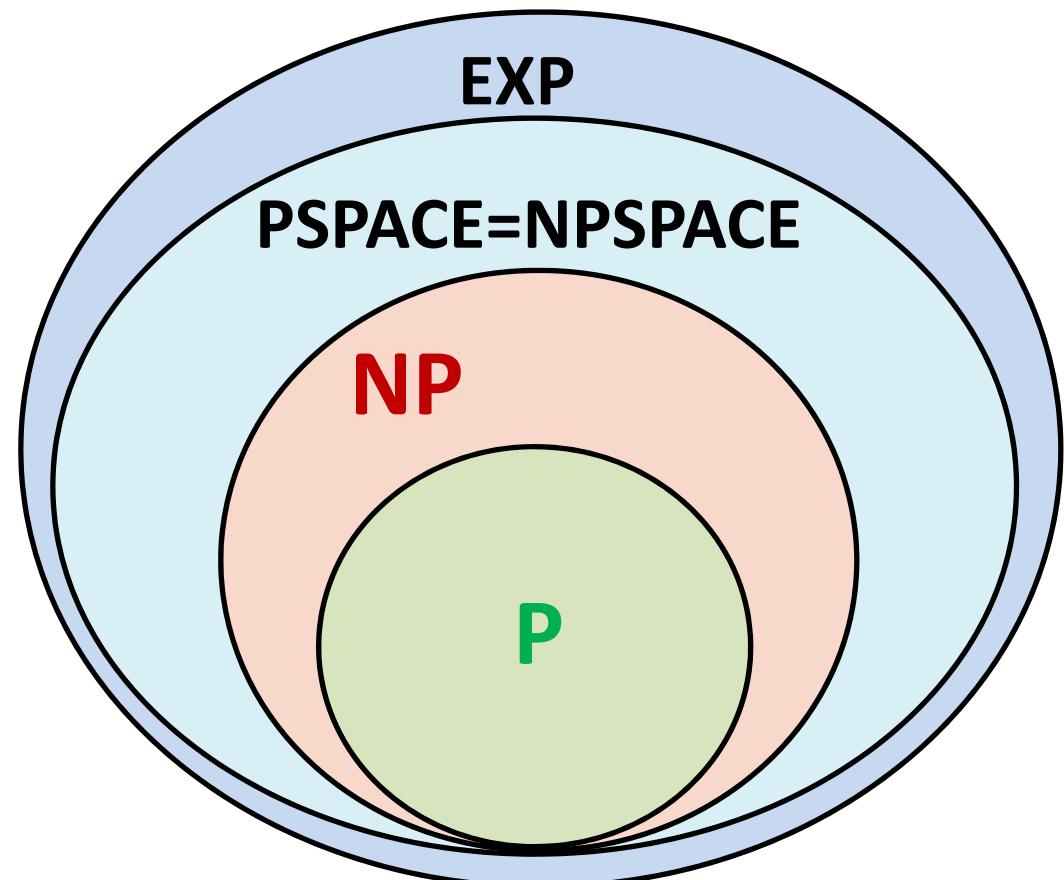
Course Evaluations

bu.campuslabs.com/courseeval