BU CS 332 – Theory of Computation

https://forms.gle/z4RnBWnou8DU9Dsu8

Lecture 24:

- NP-completeness example
- Space complexity

Reading:

Sipser Ch 7.4-7.5,
8.1-8.2

Mark Bun

December 7, 2021

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

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 $t\}$

Theorem: SUBSET-SUM is NP-complete

Claim 1: SUBSET-SUM is in NP

Claim 2: SUBSET-SUM is NP-hard

$3\text{SAT} \leq_p$ SUBSET–SUM

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

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

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

$\text{3SAT} \leq_p$ SUBSET−SUM

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

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

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

$3\text{SAT} \leq_p$ SUBSET−SUM

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

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

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

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

Example of the reduction

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

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

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

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

Example of the reduction

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

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 $A \in \mathrm{SPACE}(f(n))$ 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.

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_{accept}

Rejecting configuration: q = q_{reject}

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) $kn2^n$

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

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

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

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

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

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

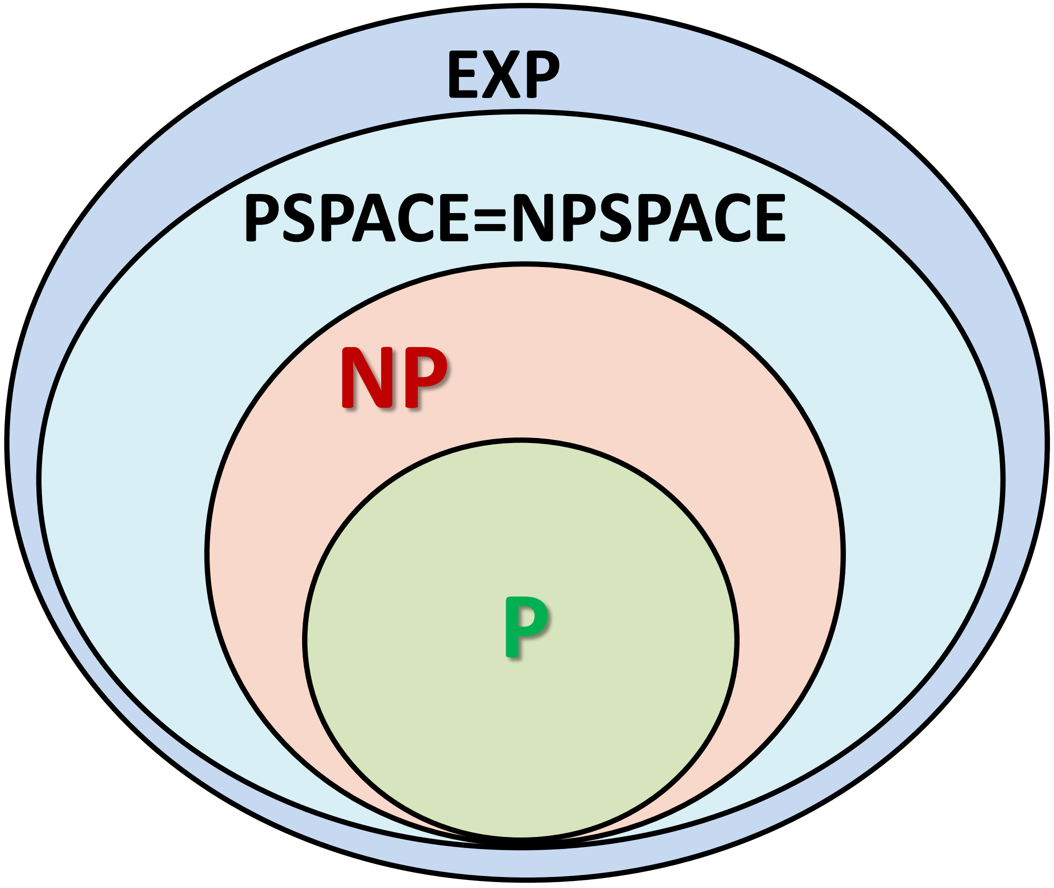
Definition: NPSPACE is the class of languages decidable in polynomial space on a single-tape (nondeterministic) TM

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

Relationships between complexity classes

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

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



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

   (via time hierarchy)

Which containments in (1) are proper?

Unknown!

Course Evaluations

bu.campuslabs.com/courseeval