BU CS 332 – Theory of Computation

Lecture 22:

- NP-Completeness Example
- Space Complexity
- Savitch's Theorem

Reading:

Sipser Ch 8.1-8.2

Mark Bun

April 22, 2020

NP-completeness

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")

Theorem: If $C \in \mathrm{NP}$ and $B \leq_{\mathrm{p}} C$ for some NP-complete language B, then C is also NP-complete

$3SAT$ (3-CNF Satisfiability)

Definition(s):

- A literal either a variable of its negation x_5 , $\overline{x_7}$
- A clause is a disjunction (OR) of literals Ex. $x_5 \vee \overline{x_7} \vee x_2$
- A 3-CNF is a conjunction (AND) of clauses where each clause contains exactly 3 literals

Ex. $C_1 \wedge C_2 \wedge \ldots \wedge C_m =$

$$(x_5 \vee \overline{x_7} \vee x_2) \wedge (\overline{x_3} \vee x_4 \vee x_1) \wedge \cdots \wedge (x_1 \vee x_1 \vee x_1)$$

$$3SAT = \{\langle \varphi \rangle | \varphi \text{ is a satisfiable } 3 - \text{CNF}\}$$

Cook-Levin Theorem: $3SAT$ is NP-complete

Some general reduction strategies

- Reduction by simple equivalence

  Ex. $INDEPENDENT - SET \leq_{\text{p}} VERTEX - COVER$
  and $VERTEX - COVER \leq_{\text{p}} INDEPENDENT - SET$

- Reduction from special case to general case

  Ex. $VERTEX - COVER \leq_{\text{p}} SET - COVER$

- Gadget reductions

  Ex. $3SAT \leq_{\text{p}} INDEPENDENT - SET$

Independent Set

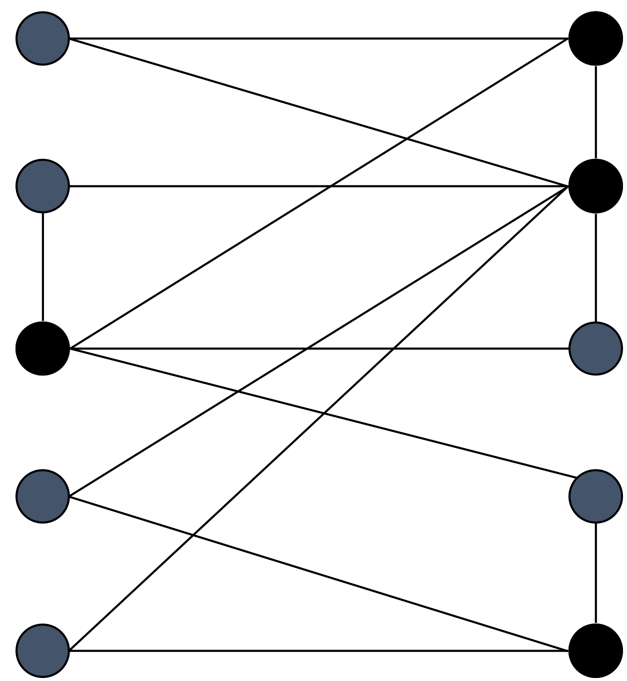
An **independent set** in an undirected graph G is a set of vertices that includes at most one endpoint of every edge.

$$INDEPENDENT-SET$$
$$= \{\langle G,k\rangle | G \text{ is an undirected graph containing an independent set with} \geq k \text{ vertices}\}$$

- Is there an independent set of size ≥ 6?
  - Yes.

independent set

- Is there an independent set of size ≥ 7?
  - No.



Independent Set is NP-complete

1) $INDEPENDENT-SET \in \mathrm{NP}$

2) Reduce $3SAT \leq_{\mathrm{p}} INDEPENDENT-SET$

Proof. "On input $\langle\varphi\rangle$, where φ is a 3CNF formula,

1. Construct graph G from φ
   - G contains 3 vertices for each clause, one for each literal.
   - Connect 3 literals in a clause in a triangle.
   - Connect literal to each of its negations.
2. Output $\langle G, k\rangle$, where k is the number of clauses in φ."

Example of the reduction

$$\varphi = (\overline{x_1} \lor x_2 \lor x_3) \land (x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_4)$$

Proof of correctness for reduction

Let k = # clauses and l = # literals in φ

Claim: φ is satisfiable iff G has an ind. set of size k

$\Longrightarrow$ Given a satisfying assignment, select one literal from each triangle. This is an ind. set of size k

$\Longleftarrow$ Let S be an ind. set of size k

- S must contain exactly one vertex in each triangle
- Set these literals to true, and set all other variables in an arbitrary way
- Truth assignment is consistent and all clauses satisfied

Runtime: $O(k + l^2)$ which is polynomial in input size

Space Complexity

Complexity measures we've studied so far

- Deterministic time TIME
- Nondeterministic time NTIME
- Classes P, NP

Many other resources of interest:

Space (memory), randomness, parallel runtime / #processors, quantum entanglement, interaction, communication, …

Space analysis

Space complexity of a TM (algorithm) = maximum number of tape cell 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 $w \in \Sigma^*$, 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 halts on w using at most $f(n)$ cells on every computational branch

Space complexity classes

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

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.

Example: NFA analysis

Theorem: Let $ALL_{NFA} = \{A \mid A \text{ is an NFA with } L(A) = \Sigma^*\}$

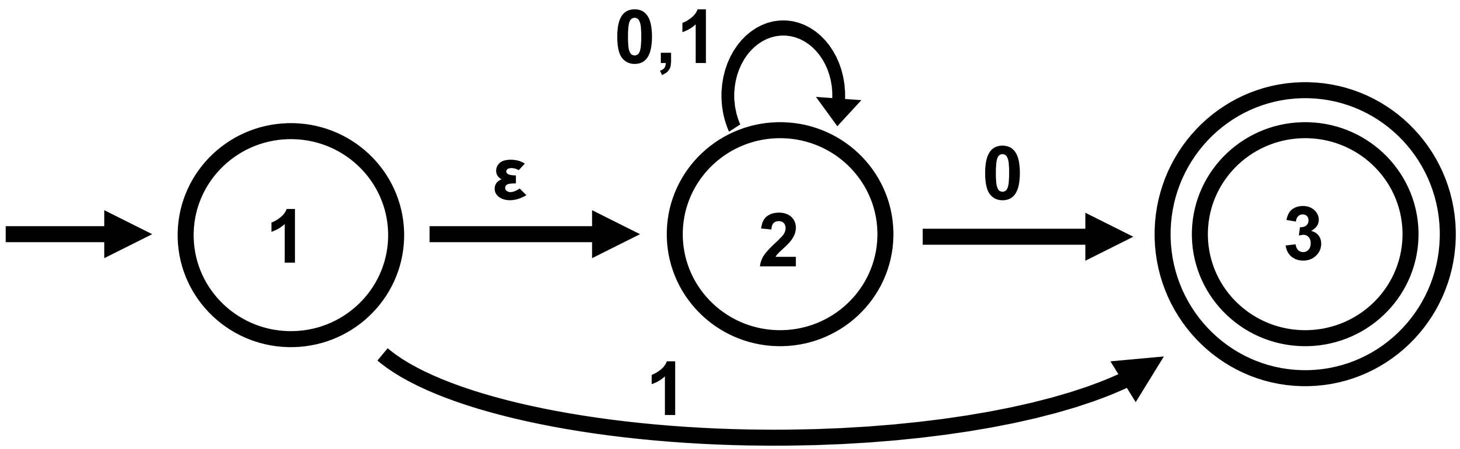
Then $\overline{ALL_{NFA}} \in \text{NSPACE}(n)$.

Proof: The following NTM decides $\overline{ALL_{NFA}}$ in linear space

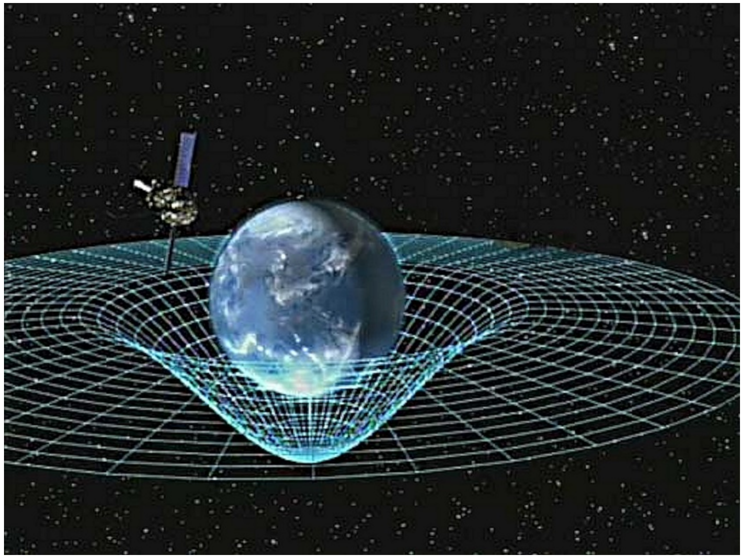
On input $\langle A \rangle$ where A is an NTM:

1. Place a marker on the start state of A.
2. Repeat 2^q times where q is the # of states of A:
3. Nondeterministically select $a \in \Sigma$.
4. Adjust the markers to simulate all ways for A to read a.
5. Accept if at any point *none* of the markers are on an accept state. Else, reject.

Example



Space vs. Time



Space vs. Time

$$TIME(f(n)) \subseteq NTIME(f(n)) \subseteq SPACE(f(n))$$

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 $\{0, 1\}$
- space $f(n)$

How many configurations are possible when this TM is run on an input $w \in \{0,1\}^n$?

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

Savitch's Theorem

Savitch's Theorem: Deterministic vs. Nondeterministic Space

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

M = "On input w:

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

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

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

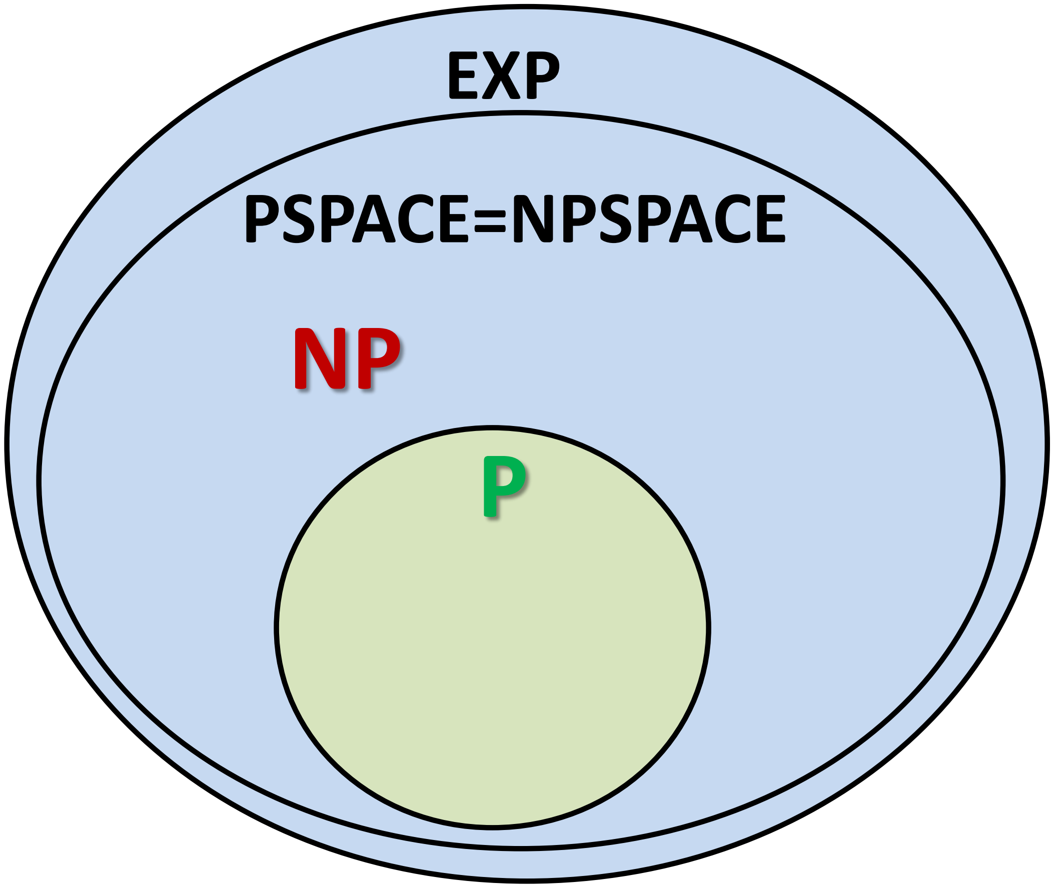
Definition: $\mathrm{NPSPACE}$ is the class of languages decidable in polynomial space on a single-tape (nondeterministic) TM

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

Relationships between complexity classes

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

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



2. $\mathrm{P} \neq \mathrm{EXP}$ (Monday)

Which containments in (1) are proper?

Unknown!