BU CS 332 – Theory of Computation

Lecture 24:

- Final review

Reading:

Sipser Ch 7.1-8.3, 9.1

Final Exam:
5 PM 5/5 – 5PM 5/7

Extra office hours 5/4, 5/5 TBD

Mark Bun

April 29, 2020

Final Topics

Everything from Midterms 1 and 2

- Midterm 1 topics: DFAs, NFAs, regular expressions, pumping lemma, context-free grammars, pushdown automata, pumping lemma for CFLs

  (more detail in lecture 9 notes)

- Midterm 2 topics: Turing machines, TM variants, Church-Turing thesis, decidable languages, countable and uncountable sets, undecidability, reductions, unrecognizability, mapping reductions

  (more detail in lecture 17 notes)

Time Complexity (7.1)

- Asymptotic notation: Big-Oh, little-oh, Big-Omega, little-omega, Theta
- Know the definition of running time for a TM and of time complexity classes (TIME / NTIME)
- Understand how to simulate multi-tape TMs and NTMs using single-tape TMs and know how to analyze the running time overhead

P and NP (7.2, 7.3)

- Know the definitions of P and NP as time complexity classes TIME / NTIME
- Know how to analyze the running time of algorithms to show that languages are in P / NP
- Understand the verifier interpretation of NP and why it is equivalent to the NTM definition
- Know how to construct verifiers and analyze their runtime
- Understand the surprising implications of P = NP, esp. how to show that search problems can be solved in poly-time P = NP ⇒ can find large cliques, satisfying assignments, ...

NP-Completeness (7.4, 7.5)

- Know the definition of poly-time reducibility
- Understand the definitions of NP-hardness and NP-completeness NP-complete = in NP + NP-hard
- Understand the statement of the Cook-Levin theorem (don't need to know its proof)
- Understand several canonical NP-complete problems and the relevant reductions: SAT, 3SAT, CLIQUE, INDEPENDENT-SET, VERTEX-COVER, HAMPATH, SUBSET-SUM

Space Complexity (8.1)

- Know the definition of running space for a TM and of space complexity classes (SPACE / NSPACE)
- Understand how to analyze the space complexity of algorithms (including SAT, NFA analysis)

PSPACE and PSPACE-Completeness (8.2, 8.3)

- Know the definitions of PSPACE and NPSPACE
- Know why they're equivalent (statement of Savitch's Theorem)
- Understand how to show that languages are in PSPACE
- Know the definition of PSPACE-completeness
- You will not be asked anything about the PSPACE-complete language TQBF, or to show that any specific language is PSPACE-complete

Hierarchy Theorems (9.1)

- Know that we can prove, unconditionally, that P ≠ EXP and that PSPACE ≠ EXPSPACE
- You will not be asked about the formal statements of the time/space hierarchy theorems, but should understand how they generalize the above statements

Things we didn't get to talk about

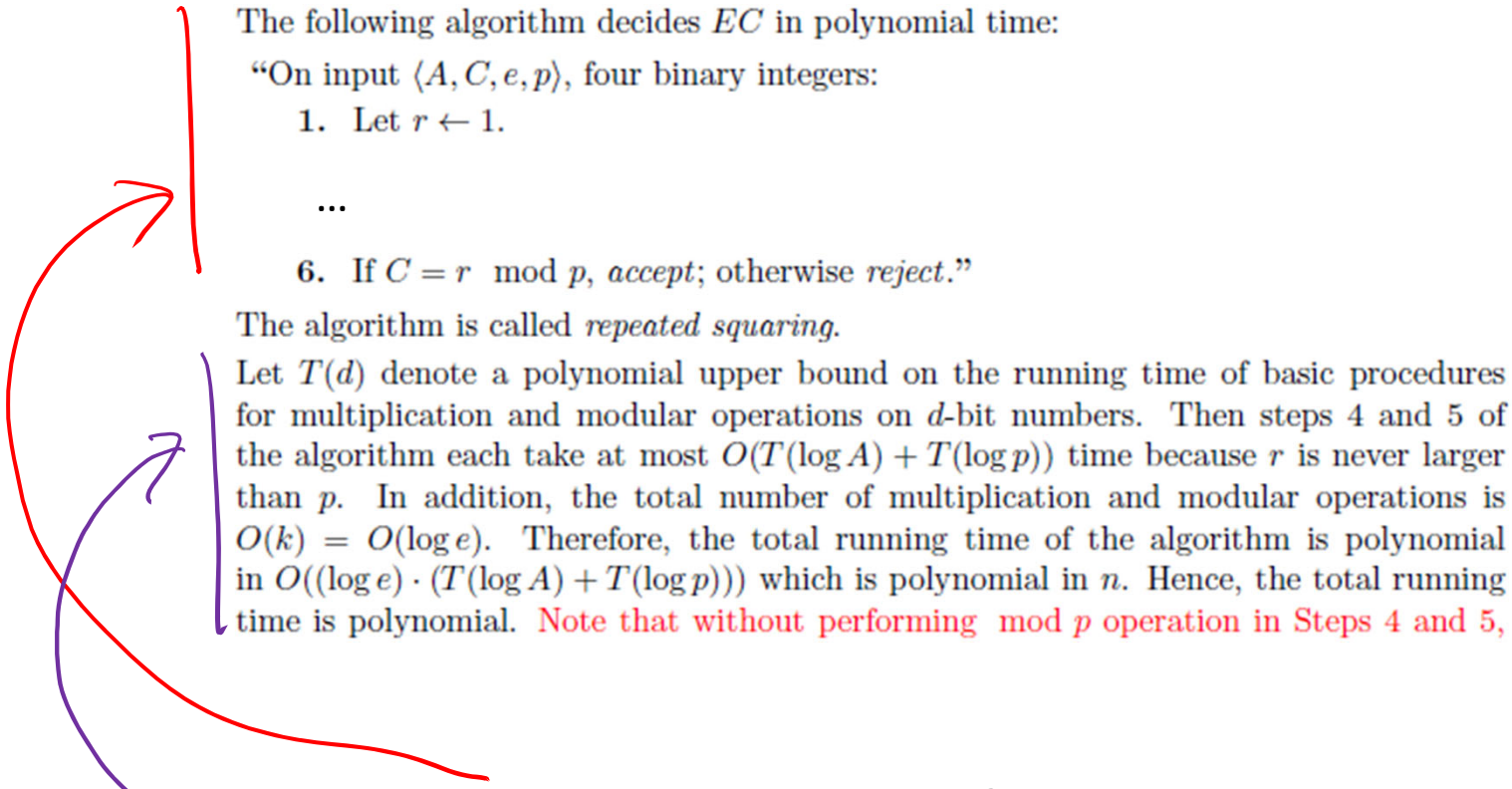
(in complexity theory)

- Additional classes between NP and PSPACE (polynomial hierarchy)
- Logarithmic space (streaming algorithms)
- Relativization and the limits of diagonalization
- Boolean circuits
- Randomized algorithms / complexity classes (derandomization)
- Interactive proof systems (generalization of NP) IP = PSPACE
- Complexity of counting

https://cs-people.bu.edu/mbun/courses/535_F20/

Tips for Preparing Exam Solutions

Designing (nondeterministic) time/space-bounded deciders



The following algorithm decides EC in polynomial time:

"On input $\langle A, C, e, p \rangle$, four binary integers:

1. Let $r \leftarrow 1$.

...

6. If $C = r \mod p$, *accept*; otherwise *reject*."

The algorithm is called *repeated squaring*.

Let $T(d)$ denote a polynomial upper bound on the running time of basic procedures for multiplication and modular operations on d-bit numbers. Then steps 4 and 5 of the algorithm each take at most $O(T(\log A) + T(\log p))$ time because r is never larger than p. In addition, the total number of multiplication and modular operations is $O(k) = O(\log e)$. Therefore, the total running time of the algorithm is polynomial in $O((\log e) \cdot (T(\log A) + T(\log p)))$ which is polynomial in n. Hence, the total running time is polynomial. Note that without performing mod p operation in Steps 4 and 5,

- Key components: High-level description of algorithm, analysis of running time and/or space usage
- A good idea: Explain correctness of your algorithm

Designing NP verifiers

Make sure alg. runs in poly-time as function of input length

We give a poly-time verifier for $TEAM$. A certificate c for our verifier is a subset of M of size k.

"On input $\langle n, X, Y, Z, M, k; c\rangle$ where $\langle n, X, Y, Z, M, k\rangle$ is a $TEAM$ instance and c is a certificate:

1. If $k > \min(n, |M|)$, *reject*.
2. Check whether $|c| = k$ and $c \subseteq M$.
3. Check whether all elements of triples in c are different.
4. If any of these checks fails, *reject*; otherwise, *accept*."

Step 1 is performed to ensure that the running time is polynomial in n even for large k. Step 2 can be run in $O(k \cdot |M|) = O(|M|^2)$ time, by iterating through M and marking elements. Step 3 can be implemented to run in $O(|c| \log |c|)$ time by first sorting the elements of c. This verifier runs in polynomial time; hence, $TEAM \in$ NP.

- Key components: Description of certificate, high-level description of algorithm, analysis of running time
- A good idea: Explain correctness of your algorithm

NP-completeness proofs

To show a language L is NP-complete:

1) Show L is in NP (follow guidelines from previous two slides)

2) Show L is NP-hard (usually) by giving a poly-time reduction $A \leq_p L$ for some NP-complete language A

- High-level description of algorithm computing reduction
- Explanation of correctness: Why is $w \in A$ iff $f(w) \in L$ for your reduction f?
- Analysis of running time

Practice Problems

$NP = \bigcup_{k=1} NTIME(n^k)$ [class of decision problems (languages) decidable in poly-time on a NTM]

$= \{L \mid L$ has a polynomial-time verifier$\}$

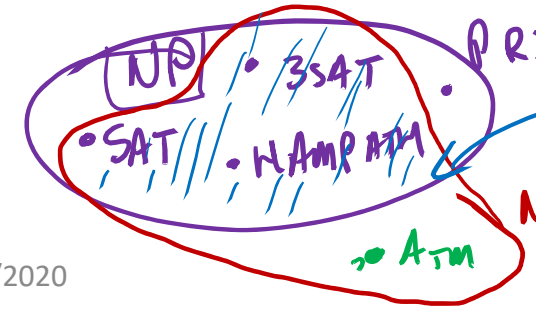
An algorithm $V(\langle w, c\rangle)$ s.t. $w \in L \iff \exists\, c$ s.t. $V(\langle w, c\rangle)$ accepts

V runs in time poly. in $|\langle w\rangle|$

$NP\text{-hard} = \{L \mid \forall\, A \in NP\ \exists$ poly-time reduction from A to $L\}$

$= \{L \mid \forall\, A \in NP\ \ A \leq_p L\}$

$NP\text{-complete} = NP \cap NP\text{-hard} = \{L \mid L \in NP$ and L is NP-hard$\}$



NP-complete

PRIMES $\in P$

PRIMES is NP-hard $\iff P = NP$

Diagram = state of the world if $P \neq NP$

P

Give examples of the following languages: 1) A language in P. 2) A decidable language that is not in P. 3) A language for which it is unknown whether it is in P.

1) 2SAT, $\{1\}$, $\emptyset$, Σ^*, PRIMES

2) ~~A_{TM}~~ recognizable but not decidable; ~~A_{DFA}~~; $\overline{SA}_{TM,EXP}$ $A_{NFA} \in PSPACE$ lecture 23

3) TSP
SAT
HAMPATH
3SAT
TQBF

Give an example of a problem that is solvable in polynomial-time, but which is not in PSPACE

deterministic

in the worst case

no certificate

L a language corresponds to computational problem "Given w, is $w \in L$?"

FP

$\approx \{\langle S \rangle \mid S \text{ is a sorted list}\}$

1) Given a list of n numbers, sort the list

2) Compute the nth Fibonacci number (on input 1^n)

$\{\langle 1^n, f \rangle \mid f \text{ is the nth Fib. number}\}$

3) Given $\langle x, y \rangle$, compute $\langle xty \rangle$

$\{\langle x, y, z \rangle \mid z = xty\}$

Let $L =$
$\{\langle w_1, w_2\rangle | \exists$ strings x, y, z such that $w_1 = xyz$ and $w_2 = xy^R z\}$.

Show that $L \in$ P.

Ex: w_1 = abracadabra (x = abra, y = cad, z = abra)

w_2 = abdacarabra (x = ab, y = dacar, z = abra)

copy = abracadabra

w_1 = rat

w_2 = rat

w_1 = tac

w_2 = atc

Non-examples

w_1 = rat w_2 = chicken

w_1 = rat w_2 = dog

Running time: $n^2 \cdot O(n) = O(n^3)$

On input $\langle w_1, w_2\rangle$:

1. Check if $|w_1| = |w_2|$, reject o.w. Let the common length of the strings be n
2. If $w_1 = w_2 = \epsilon$, accept
3. For $i = 1, \ldots, n$:
4. For $j = i, \ldots, n$:
5. copy w_2 into new variable "copy" with substring between indices i & j reversed
6. compare w_1 and copy. If equal, accept
7. Reject

$O(n)$

Which of the following operations is P closed under? Union, concatenation, star, intersection, complement.

If M_1 decides L_1 and M_2 decides L_2 (each in poly-time)

1) Union — Run M_1 on w, run M_2 on w. If either accept, accept, else reject

2) concatenation — For each index i, run M_1 on $w_1, \ldots, w_i$ and M_2 on $w_{i+1}, \ldots w_n$. Accept if both accept.

3) Star (?) — Homework 7 ← Dynamic programming

4) Complement

5) Intersection — $A \cap B = \overline{\overline{A} \cup \overline{B}}$ or

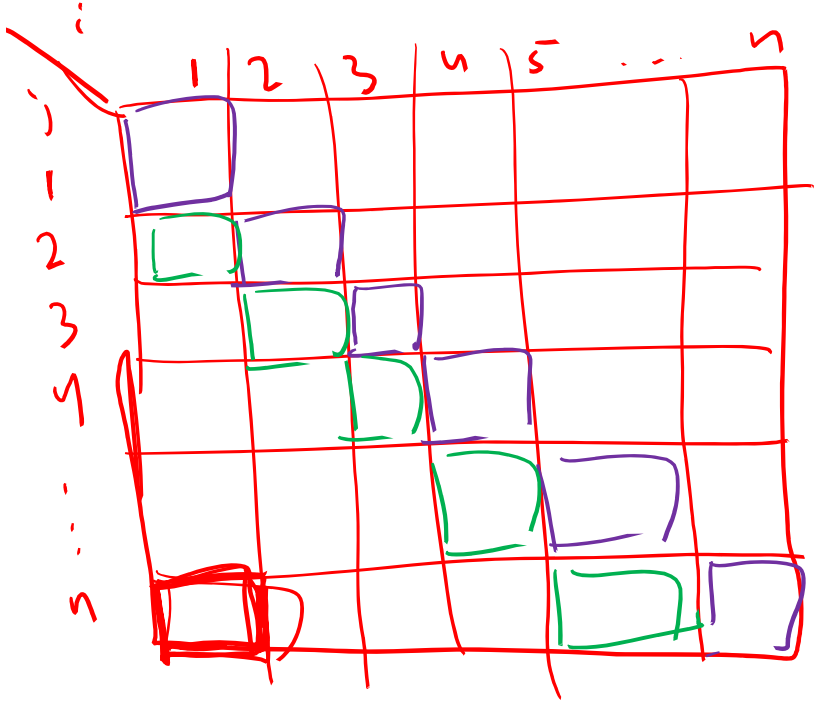
Run M_1 on w, run M_2 on w. If both accept, accept. Else reject.

Star (Dynamic programming)

Let M decide A in poly-time

Table $T[i,j] = 1 \iff w_i, \ldots, w_j \in A^*$

$w \in A \iff T[1,n] = 1 \iff w_1, \ldots, w_n \in A^*$



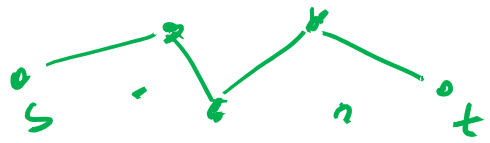
1) For $i = 1, \ldots, n$:
   Run M on w_i. If accept, set $T[i,i] = 1$
   o.w. set $T[i,i] = 0$
2) For $\ell = 2, \ldots, n$: [ℓ = substring length]
3) For $i = 1, \ldots, n - \ell + 1$ ← Test if $w_i, \ldots, w_{i+\ell-1} \in A^*$
4) For $j = 1, \ldots, \ell$:
5) If $T[i,j] = 1$ and $T[j, i+\ell-1] = 1$, set $T[i, i+\ell-1] = 1$
6) Accept iff $T[1,n] = 1$

$x \in A^* \iff \exists$ a split $x_1, \ldots, x_i \in A$
$x_{i+1}, \ldots, x_{n+1} \in A$

NP and NP-completeness

Prove that $LPATH = \{\langle G, s, t, k\rangle | G$ is an undirected graph containing a simple path from s to t of length $\geq k\}$ is in NP

No repeated vertices



length $\geq k$

Poly-time verifier:

Certificate $c = v_1, \ldots, v_m$ sequence of vertices

$V(\langle G, u, c\rangle) =$

1) Check $u \leq m \leq$ # vertices, reject o.w.
2) Check no repeated vertices in $v_1, \ldots, v_m$
3) Check that $v_1 = s$, $v_m = t$, and $(v_i, v_{i+1}) \in$ vertex set $\forall\ i.\ i+1$

1st: check $u \leq$ # vertices
2nd: read c sequentially and reject if # vertices in certificate $> k$

Running time:

$O(m)$ edges to check $\cdot$ time it takes to look up an edge

$O(|E|)$ or $O(m \log_3 m)$ for step 2

Prove that $LPATH$ is NP-hard

Undirected Hampath : $\{\langle G, s, t\rangle \mid G$ is an undirected graph and $\exists$ a Ham. path from s to t in $G\}$

Sipser 7.55 : UHAMPATH is NP-complete (reduction from HAMPATH)

Show UHAMPATH $\leq_p$ LPATH [construct poly-time reduction]

Idea: G has a Ham path $\iff$ G has a path of length $\geq$ # vertices
i.e. $\langle G, s, t\rangle \in$ UHAMPATH i.e. $\langle G, s, t, k\rangle \in$ LPATH

On input $\langle G, s, t\rangle$:

1. let k = # vertices of G
2. Output $\langle G, s, t, k\rangle$

$\Leftarrow$ pigeonhole principle

Run time: Just need to count # vertices and output it
$\Rightarrow O(n)$ reduction

Which of the following operations is NP closed under? Union, concatenation, star, intersection, complement.

Show that if P = NP, there is a polynomial-time decider for $USAT = \{\langle\phi\rangle | \phi$ is a formula with exactly one satisfying assignment$\}$

Space Complexity

TIME(f(n)) = {languages L decidable in time O(f(n))}

Which of the following statements are true?

SPACE(f(n)) = {" " space " "}

- $SPACE(2^n) = SPACE(2^{n+1})$ True

  $2^n = \Theta(2^{n+1})$ since $2^{n+1} = 2 \cdot 2^n$

- $SPACE(2^n) = SPACE(3^n)$ False

  $3^n = \omega(2^n)$ Space hierarchy theorem: If $f(n) = o(g(n))$, then $SPACE(f(n)) \subsetneq SPACE(g(n))$

  $2^n = o(3^n)$

- $NSPACE(n^2) = SPACE(n^5)$ False

  Savitch: $NSPACE(n^2) \subseteq SPACE(n^4) \subsetneq SPACE(n^5)$

  (space hierarchy theorem)

Consider the inheritance problem from HW9, except Alice and Bob now take turns drawing bags from boxes. Alice's goal is to assemble a complete collection of marbles, and Bob's is to thwart her. Prove that determining whether Alice has a winning strategy is in PSPACE.

Example run of game:

1) Alice picks bag 2 in B_3 — 1, 4
2) Bob picks bag 2 in B_4 — 1, 3, 4
3) Alice picks bag 1 in B_1 — 1, 2, 3, 4

Alice wins

Alice has a winning strategy $\iff$ $\exists (i_1, b_1) \forall (i_2, b_2) \exists (i_3, b_3) \forall (i_4, b_4) \cdots$ $\exists / \forall (i_m, b_m)$ (Alice gets a complete set of marbles)

m boxes, k marble types

i_1: Alice's first move

b_1: Bob's first move

i_3: Alice's 2nd move

Bob's i_1, $b_1 = 0$ means picks first bag, $b_1 = 1$ means picks second bag

3) If formula looks like $\forall x : \psi(x)$
   i) "
   ii) "
   iii) If both run accept, accept, else reject

Space usage: $T(m)$ = amount of space used for m quantifiers

$m = 0$ $T(0) = O(n)$

$T(m) = T(m-1) + O(1)$

$O(m+n)$ space

Looks like: TQBF

$\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_m \; \varphi(x_1, \ldots, x_m)$ ← length n

PSPACE alg. for TQBF: On input a QBF

1) If no variables, just evaluate formula [Base case]
2) If formula looks like $\exists x_i \psi(x)$: [First inductive]
   i) Run alg. on ψ with $x_i = 0$
   ii) Run alg. on ψ with $x_i = 1$
   iii) If either run accepts, accept, else reject

PSPACE Alg. for Marbles game Recursive alg. based on # boxes

1) If no boxes available ($m = 0$). {Base case}

   If $k = 0$: accept

   If $k > 0$: reject

2) If $m \geq 1$, Alice's turn:

   For each box $i = 1, \ldots, m$ and each choice of bag b_i:

   - Run alg. on game w/ 1 fewer box where Alice made that choice
   - If at least one run accepts, accept. Else reject.

3) If $m \geq 1$, Bob's turn:

   ||

   If every run accepts, accept else reject.

Space usage: $S(0) = O(n)$ $S(m) = S(m-1) + O(\log m) + O(n)$

($S(m-1)$: recursive call; $O(\log m)$: counter for current box; $O(n)$: create new instance)

$\Rightarrow S(m) \in O(m(\log m + n))$