BU CS 332 – Theory of Computation

Lecture 24:

- Final review

Mark Bun

May 1, 2024

Final Topics

Everything from Midterms 1 and 2

- Midterm 1 topics: DFAs, NFAs, regular expressions, distinguishing set method

  (more detail in lecture 8 notes)

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

  (more detail in lecture 16 notes)

Mapping Reducibility (5.3)

- Understand the definition of a computable function
- Understand the definition of a mapping reduction
- Know how to use mapping reductions to prove decidability, undecidability, recognizability, and unrecognizability

Time Complexity (7.1)

- Asymptotic notation: Big-Oh, little-oh
- 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 runtime overhead

P and NP (7.2, 7.3)

- Know the definitions of P and NP as time complexity classes
- 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

NP-Completeness (7.4, 7.5)

- Know the definition of poly-time reducibility
- Understand the definitions of NP-hardness and NP-completeness
- 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, 8.2)

- Know the definition of running space for a TM and of space complexity classes $SPACE(f(n))$
- Understand the known relationships between space complexity classes and time complexity classes

Hierarchy Theorems (9.1)

- Formal statements of time and space hierarchy theorems and how to apply them
- How to use hierarchy theorems to prove statements like $\mathrm{P} \neq \mathrm{EXP}$

Things we didn't get to talk about

- 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

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

Theory and Algorithms Courses after 332

- Algorithms
  - CS 530/630 (Advanced algorithms)
  - CS 531 (Optimization algorithms)
  - CS 537 (Randomized algorithms)
- Complexity
  - CS 535 (Complexity theory)
- Cryptography
  - CS 538 (Foundations of crypto)
- Topics (CS 599)

  E.g., Privacy in machine learning, algorithms and society, sublinear algorithms, new developments in theory of computing, communication complexity, math tools for theoretical CS, meta-complexity, fine-grained complexity

Algorithms and Theory Research Group

- https://www.bu.edu/cs/research/theory/

- Weekly seminar: Mondays at 1:30
  https://www.bu.edu/cs/algorithms-and-theory-seminar/

  Great way to learn about research in theory of computation!

Tips for Preparing Exam Solutions

Designing (nondeterministic) time/space-bounded deciders

We give the high-level description of a non-deterministic Turing Machine N deciding CLIQUE in polynomial time. On input $\langle G, k \rangle$:

- If $k > n$, reject.
- Non-deterministically guess a subset of k vertices.
- For every pair of vertices in the subset, check that there is an edge connecting them. If any pair doesn't have an edge, reject.
- Accept.

First, we argue that this runs in non-deterministic polynomial time.

The first step always takes at most time $\log k + \log n$ (comparison can be done by subtracting the numbers in binary and comparing to 0). If $k > n$, the Turing machine N always halts in this much time.

Now, assume that $k \leq n$. If the graph has n nodes and m edges, then the size of the input is at least $n + m + \log k$ (since the adjacency list of the graph is at least size n and integer k takes $\log k$ bits to represent). Non-deterministically guessing a subset of k vertices takes time at most $O(n + \log k)$ (since this can be done by cycling through all the vertices and adding them into the subset non-deterministically, and stopping once the subset has size k). Note that checking that a pair of vertices has an edge can be done in time at most $n + m$. Hence, step 2 takes time at most $\frac{(n+m)(k(k-1))}{2}$ since there at most $\binom{k}{2} = k(k-1)/2$ pairs of vertices in a subset of vertices that has size k. Note that since $k \leq n$, this is polynomial in the input size. Hence, the Turing machine runs in polynomial time in this case as well.

Finally, we are left to argue correctness. If $\langle G, k \rangle$ is in CLIQUE, then G contains a clique of size k, and on the computational branch that guesses the corresponding k nodes, Turing machine N will accept. On the other hand, if $\langle G, k \rangle$ is not in CLIQUE, then there is no k-clique in G and hence none of the computational branches of the NTM N will accept. Thus, in this case Turing Machine N will reject. Hence, N decides CLIQUE.

- Key components: High-level description of algorithm, explanation of correctness, analysis of running time and/or space usage

Designing NP verifiers

For simplicity in analyzing our algorithm, suppose each S_i be encoded as an n bit string, where the j'th bit is set to 1 if $j \in S_i$ and is set to 0 otherwise. We will use a similar encoding for our certificate.

We give a poly-time verifier for MS as follows. The certificate is a set T encoded as an n bit string with at most k 1's. Our verifier is as follows.

"On input $\langle S_1, \ldots, S_m, n, k; T \rangle$:

1. Scan T to check that it encodes a list of at most k distinct elements of $[n]$. *Reject* if not.
2. For $i = 1, \ldots, m$:
3. Scan S_i and scan T to check that they intersect. If not, *Reject*
4. *Accept.*"

Correctness: If $\langle S_1, \ldots, S_m, n, k \rangle \in MS$, then there exists a set T of size at most k that intersects every set. The certificate which encodes this set will result in the algorithm successfully passing every check in step 3, so the algorithm will accept. On the other hand, if $\langle S_1, \ldots, S_m, n, k \rangle \notin MS$, then every set of size at most k will fail to intersect at least one S_i, so every certificate will lead to rejection.

Runtime: The encoding we are using for each set ensures that the length of the input is at least mn. Describing a certificate T takes n bits, which is hence polynomial in the input length. The loop in step 2 runs for m steps and the loop in step 3 runs for $O(n^2)$ steps, so the total runtime of the algorithm is $O(mn^2)$. This is polynomial in the input length, which again, is at least mn.

- Key components: Description of certificate, high-level description of algorithm, explanation of correctness, analysis of running time

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

Use a mapping reduction to show that $ALL_{\text{TM}} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \Sigma^*\}$ is co-unrecognizable

Use a mapping reduction to show that $ALL_{\text{TM}} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \Sigma^*\}$ is unrecognizable

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.

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

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

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

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

Prove that $LPATH$ is NP-hard

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

Which of the following statements are true?

- $TIME(2^{n^2}) \subseteq NTIME(n)$

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

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