ECE 374 B: Algorithms and Models of Computation, Fall 2024

Midterm 3 – December 05, 2024

- **You will have 75 minutes (1.25 hours) to solve all the problems. Most have multiple parts.** Don't spend too much time on questions you don't understand and focus on answering as much as you can!
- ***BUDGET YOUR TIME WISELY***. I highly recommend working on the questions you know first and the questions you need to think about second.
- *No* resources are allowed for use during the exam except a multi-page cheatsheet and scratch paper on the back of the exam. ***Do not tear out the cheatsheet or the scratch paper!*** It messes with the auto-scanner.
- You should write your answers *completely* in the space given for the question. We will not grade parts of any answer written outside of the designated space.
- Please *use a dark-colored pen* unless you are *absolutely* sure your pencil writing is forceful enough to be legible when scanned. We reserve the right to take off points if we have difficulty reading the uploaded document.
- Unless otherwise stated, assume $\mathsf{P} \neq \mathsf{NP}$.
- Assume that whenever the word "reduction" is used, we mean a (not necessarily polynomial-time) *mapping/many-one* reduction.
- You can only refer to the cheat sheet content as a black box.
- ***Don't cheat.*** If we catch you, you will get an F in the course.
- ***Good luck!***

Name: _____

NetID: _____

Date: _____

1 Short Answer I (10 questions) - 20 points

For each of the problems circle *true* if the statement is *always* true, circle *false* otherwise. There is no partial credit for these questions.

(a) If a problem is in NP, then it must also be NP-hard.

True False

(b) There exists a polynomial time reduction from every NP-hard problem to every NP-complete problem.

True False

(c) If $A \leq_P B$ and B is NP-Complete, then A is NP-Complete.

True False

(d) There exists a polynomial time reduction from every problem in NP to every problem in P.

True False

(e) All recursively enumerable problems are NP-hard.

True False

(f) If a problem is NP-hard and co-NP-hard, then it must be in NP.

True False

(g) If a language is not recursively enumerable then its complement must be recursively enumerable.

True False

(h) All languages that are decidable are solvable in polynomial space.

True False

(i) Determining if the language represented by a context-free grammar accepts all strings is decidable.

True False

(j) Ignoring $\emptyset$ and Σ^* [1], even if P=NP, there can still be some problems that are in P/NP, but not are NP-hard.

True False

[1] I had to put this qualifier because technically you can't do reductions with trivial languages because reductions map yes to yes and no to no and those trivial sets either have only yes's or only no's.

2 Short Answer II (5 questions) - 10 points

Consider the "**NoTwoOnes**" problem defined by language L.

1. Let L be the language of strings that do not contain the sub-string **11**.
2. Let D be a decider for L (it accepts strings that do not contain the sub-string **11**).

We reduce from *3SAT* to the *NoTwoOnes* problem. A decider $S(x)$ for the 3SAT problem is constructed as follows, where ϕ is a 3-CNF formula.

```
S(φ):

   # Encode a Turing machine M_x as follows.
      M_x(w):
         for all possible variable assignments (x):
            if φ(x) == True and w does not contain 11
               return True
         return False

   return D(⟨M_x⟩)
```

(a) Circle all complexity classes that L belongs.

P NP Decidable Undecidable

(b) Is the reduction from **3SAT** to the **NoTwoOnes** correct? If so, prove the forward and backwards directions. If not, explain where the error is.

(c) Does this reduction show that **NoTwoOnes** problem is NP-hard? Why or why not?

(d) Is the language $L(M_x)$ Decidable or Undecidable?

(e) Is the reduction from **3SAT** to the **NoTwoOnes** problem polynomial in time?

3 Classification I (P/NP) - 14 points

Is the following problem in P, NP, or some combinations of complexity classes? For each of the following problems, circle all the complexity classes that problem belongs to. Whatever class it is in, prove it!

A **LostMyPhone cycle** asks given a *directed* graph G, does G contain a path that first visits all n vertices exactly once and then visits those vertices in the reverse order. So if the graph has 3 vertices and is fully connected, a LostMyPhone cycle might look like $n_1 \to n_2 \to n_3 \to n_3 \to n_2 \to n_1$.

- INPUT: A directed graph G.
- OUTPUT: TRUE if there exists a path that first visits all n vertices exactly once, and then visits the vertices in the reverse order. FALSE otherwise.

Which of the following complexity classes does this problem belong to? Circle ***all*** that apply:

P NP NP-hard NP-complete

4 Classification II (P/NP) - 14 points

Is the following problem in P, NP, or some combinations of complexity classes? For each of the following problems, circle all the complexity classes that problem belongs to. Whatever class it is in, prove it!

The MostofSAT (**MostofSAT**) problem asks whether a conjunctive normal form formula ϕ has a truth assignment that satisfies the majority ($>m/2$) of the clauses.

- INPUT: A SAT formula ϕ.
- OUTPUT: TRUE if there exists a truth assignment that satisfies the majority of the SAT clause in ϕ. FALSE otherwise.

Which of the following complexity classes does this problem belong to? Circle ***all*** that apply:

P NP NP-hard NP-complete

5 Classification I (Decidability) - 14 points

Are the following languages decidable? For each of the following languages,

- Circle one of "decidable" or "undecidable" to indicate your choice.
- If you choose "decidable", show your choice correct by describing an algorithm that decides that language. If you choose "undecidable", show your choice correct by giving a reduction proving its correctness.
- Regardless of your choice, explain *briefly* (i.e., in few sentences, diagrams, *clear* pseudo-code) why your choice is valid.

$$\textsc{Accept4Strings}_{TM} = \{\langle M \rangle \mid M \text{ is a } \boldsymbol{TM} \text{ and accepts exactly 4 strings. } (|L(M)| = 4)\}$$

decidable undecidable

6 Classification II (Decidability) - 14 points

Are the following languages decidable? For each of the following languages,

- Circle one of "decidable" or "undecidable" to indicate your choice.
- If you choose "decidable", show your choice correct by describing an algorithm that decides that language. If you choose "undecidable", show your choice correct by giving a reduction proving its correctness.
- Regardless of your choice, explain *briefly* (i.e., in few sentences, diagrams, *clear* pseudo-code) why your choice is valid.

$$\textsc{AcceptExpSpace}_{TM} = \{\langle M, w\rangle \mid M \text{ is a } \boldsymbol{TM} \text{ and accepts } w \text{ in } 2^{|w|} \text{ space. }\}$$

decidable undecidable

7 Classification (Mixed) - 14 points

Same type of problem as before, except now you have to choose between a mix of computational and algorithmic complexity classes.

Given the language:

$$\text{HOLIDAY}_{TM} = \{\langle M \rangle \mid M \text{ rejects only on the string "Told you so"}\}$$

Which of the following classes does this language belong to? Whatever you choose, *succinctly* prove it!

P NP NP-hard NP-complete decidable undecidable

EXTRA CREDIT (1 pt)

What does NP stand for?

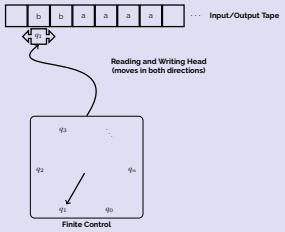
This page is for additional scratch work!

ECE 374 B Reductions, P/NP, and Decidability: Cheatsheet

Turing Machines

Turing machine is the simplest model of computation.

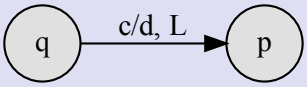
- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).
- Ever step: Read character under head, write character out, move the head right or left (or stay).
- Every TM **M** can be encoded as a string $\langle M \rangle$



Transition Function: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow, \square\}$

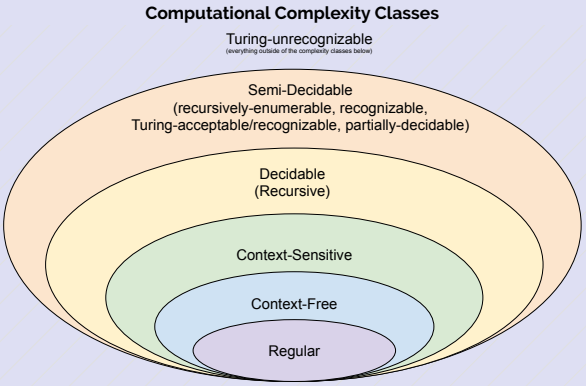
$\delta(q, c) = (p, d, \leftarrow)$

- q: current state.
- c: character under tape head.
- p: new state.
- d: character to write under tape head
- $\leftarrow$: Move tape head left.

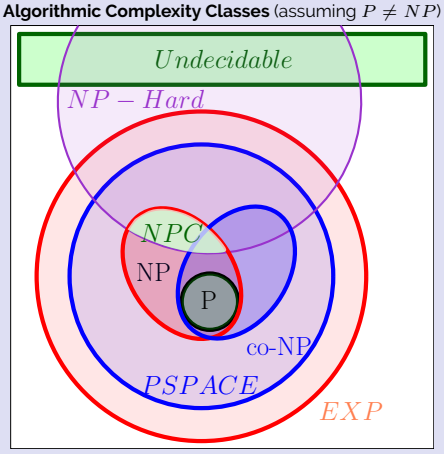


Complexity Classes

Computational Complexity Classes



Algorithmic Complexity Classes (assuming $P \neq NP$)



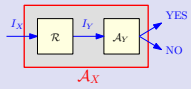
Reductions

A general methodology to prove impossibility results.

- Start with some *known* hard problem X
- *Reduce* X to your favorite problem Y

If Y can be solved then so can $X \implies Y$. But we know X is hard so Y has to be hard too. On the other hand if we know Y is easy, then X has to be easy too.

The Karp reduction, $X \leq_P Y$ suggests that there is a polynomial time reduction from X to Y.



Assuming

- $R(n)$: running time of $\mathcal{R}$
- $Q(n)$: running time of $\mathcal{A}_Y$

Running time of $\mathcal{A}_X$ is $O(Q(R(n)))$

Sample NP-complete problems

CircuitSat:	Given a boolean circuit, are there any input values that make the circuit output True?
3Sat:	Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment?
IndependentSet:	Given an undirected graph G and integer k, what is there a subset of vertices $\geq k$ in G that have no edges among them?
Clique:	Given an undirected graph G and integer k, is there a complete complete subgraph of G with more than k vertices?
kPartition:	Given a set X of kn positive integers and an integer k, can X be partitioned into n, k-element subsets, all with the same sum?
3Color:	Given an undirected graph G, can its vertices be colored with three colors, so that every edge touches vertices with two different colors?
HamiltonianPath:	Given graph G (either directed or undirected), is there a path in G that visits every vertex exactly once?
HamiltonianCycle:	Given a graph G (either directed or undirected), is there a cycle in G that visits every vertex exactly once?
LongestPath:	Given a graph G (either directed or undirected, possibly with weighted edges) and an integer k, does G have a path $\geq k$ length?

- Remember a **path** is a sequence of distinct vertices $[v_1, v_2, \ldots v_k]$ such that an edge exists between any two vertices in the sequence. A **cycle** is the same with the addition of a edge $(v_k, v_1) \in E$. A **walk** is a path except the vertices can be repeated.
- A formula is in conjunction normal form if variables are or'ed together inside a clause and then clauses are and'ed together: $((x_1 \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_4 \vee x_5))$. Disjunctive normal form is the opposite $((x_1 \wedge x_2 \wedge x_3) \vee (\overline{x_2} \wedge x_4 \wedge x_5))$.

Sample undecidable problems

AcceptOnInput:	$A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts on } w\}$
HaltsOnInput:	$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and halts on input } w\}$
HaltOnBlank:	$HaltB_{TM} = \{\langle M\rangle \mid M \text{ is a TM \& } M \text{ halts on blank input}\}$
Emptiness:	$E_{TM} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) = \varnothing\}$
Equality:	$EQ_{TM} = \left\{\langle M_A, M_B\rangle \;\middle|\; \begin{array}{l} M_A \text{ and } M_B \text{ are TM's} \\ \text{and } L(M_A) = L(M_B) \end{array}\right\}$