ECE 374 B: Algorithms and Models of Computation, Fall 2022

Midterm 3 – December 01, 2022

- **You will have 75 minutes (1.25 hours) to solve XX 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!
- *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 bring (sharpened) ***pencils and an eraser*** to take your exam with, unless you are *absolutely sure* you will not need to erase. We will *not* provide any additional scratch paper if you write in pen and make mistakes, nor will we provide pencils and erasers.
- 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.
- ***Don't cheat.*** If we catch you, you will get an F in the course.
- ***Good luck!***

Name: ______________________________

NetID: ______________________________

Date: ______________________________

1 Short Answer I (3 questions) - 20 points

For each of the problems provide a brief and concise solution. These are short answer questions and partial credit will be limited.

(a) (8 POINTS) Consider the problem NEGATIVEHAMCYCLE, which asks if a graph contains a Hamiltonian cycle with negative total weight. Is NEGATIVEHAMCYCLE in NP?

Solution: This problem is in NP. The most direct way to determine this is to provide a certifier and certificate for YES instances. An example would be that the certificate is a cycle C in graph G. The certifier would check if ever vertex is visited once and if the sum of the edge weights is negative. We can traverse the cycle to verify if these conditions hold which can be done in polynominal time. ■

Solution: Another approach to verify this would be showing how this NHC problem reduces to a known NP problem, for example to Hamiltonian Path, i.e. to show NEGATIVEHAMCYCLE $\leq_p$ HAMCYCLE.

However, note that a reduction in the other direction, HAMCYCLE $\leq_p$ NEGATIVEHAMCYCLE does not answer this question. This show that NEGATIVEHAMCYCLE is NP-hard, not that it is in NP. ■

(b) (6 POINTS) Suppose we knew that NEGATIVEHAMCYCLE could *NOT* be decided in polynomial time. Would this imply $P \neq NP$?

Solution: Since NEGATIVEHAMCYCLE is in NP, this would imply $P \neq NP$, as if were $P = NP$, then there would be a polynomial solution to decide this problem. In other words, because this can not be decided in polynomial time then this implies $P \neq NP$. ■

(c) (6 POINTS) Consider the problem TRIVIALHAMCYCLE, which asks if all of the edges in a graph form a Hamiltonian cycle. In other words, a YES instance of this problem is a graph (V, E) with $|V| = |E|$ such that E is a Hamiltonian cycle. This problem is in P since it can be decided in polynomial time. Does this imply $P = NP$? Why or why not?

Solution: We know TRIVIALHAMCYCLE is in P as the problem states (and by more reasoning below). This implies that TRIVIALHAMCYCLE $\leq_p$ HAMCYCLE, but a reduction in this direction only proves that $textscTrivialHamCycle$ is in NP, not that it is NP-hard. Only a reduction in the direction from HAMCYCLE to TRIVIALHAMCYCLE would imply that $P = NP$, but we're unlikely to find one.

To see why TRIVIALHAMCYCLE is in P, notice that the following approach solves the problem. This problem states that every edge in the graph must be used to form a Hamiltonian cycle. A graph that satisfies this condition must therefore be connected, have vertices all with degree two, and the number of vertices must equal the number of edges. In otherwords, this graph would be a ring. Verifying if a graph has a ring topology only takes $O(V + E)$ time. ■

2 Short Answer II (3 questions) - 20 points

For each of the problems provide a brief and concise solution. These are short answer questions and partial credit will be limited.

(a) (8 POINTS) Is the Halting Problem in NP? Why or why not?

Solution: NP is a subset of decidable problems, the halting problem is undecidable so it cannot be in NP. ∎

(b) (6 POINTS) Consider the following $2SAT$ expression:

$$(a \vee b) \wedge (\neg d \vee e)$$

Give an equivalent $3SAT$ expression.

Solution: We add the literals x_1, x_2 to pad the clauses.
$(a \vee b) \wedge (\neg d \vee e)$ is equivalent to $(a \vee b \vee x_1) \wedge (a \vee b \vee \neg x_1) \wedge (\neg d \vee e \vee x_2) \wedge (\neg d \vee e \vee \neg x_2)$.

A 2SAT clause $(m \vee n)$ is equivalent to the 3SAT clauses $(m \vee n \vee p) \wedge (m \vee n \vee \neg p)$ because if the 3SAT clauses are true then m or n must be true as p and $\neg p$ cannot both be true. If the 2SAT clause is true then both of the 3SAT clauses are clearly true. ∎

(c) (6 POINTS) Consider the following $4SAT$ expression:

$$(a \vee \neg a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c \vee d)$$

Give an equivalent $3SAT$ expression.

Solution: First note that the first clause contains $a \vee \neg a$ which is always true. This allows us to drop this clause. Then we add the literal y to split the remaining clause.
$(a \vee \neg a \vee b \vee \neg c) \wedge (a \vee \neg b \vee \neg c \vee d)$ is equivalent to $(a \vee \neg b \vee y) \wedge (\neg y \vee \neg c \vee d)$.

A 4SAT clause $(h \vee i \vee j \vee k)$ is eqivalent to the 3SAT clauses $(h \vee i \vee y) \wedge (\neg y \vee j \vee k)$ because if the 3SAT clauses are true then at least one of h, i, j or k must be true because y and $\neg y$ cannot both be true. Similarly if the 4SAT clause is true then at least one of h, i, j or k must be true and y can be assigned to make the other clause true. ∎

3 Classification I (P/NP) - 40 points

Are the following problems 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) Let's define a new problem: WellItsSomething-SAT (you can abbreviate to WIS_{SAT}) defined as follows:

- Input: A 3SAT formula. You may assume literals appear at most once in each clause.
- Output: True if at least 4 clauses are satisfiable.

Which of the following complexity classes does WellItsSomething-SAT belong to? Circle ***all*** that apply:

P NP co-NP NP-hard NP-complete

Solution: [✓ P] [✓ NP] [✓ co-NP] NP-hard NP-complete

The problem WellItsSomething-SAT is in P, so it is also in NP and co-NP based on algorithmic complexity classes.

The problem returns true if at least 4 clauses are simultaneously satisfiable. The number of clauses in the 3SAT formula is n. We can enumerate all n choose 4 ways to pick a subset of 4 clauses. In total, there are $O(n^4)$ possible subsets to try. For each subset of clauses, there are at most 16 variables to consider, so we can check if these are simultaneously satisfiable in $O(1)$ by trying each possible assignment. Since the overall time is polynomial, the problem WellItsSomething-SAT is in in P. ■

Solution: Alternatively, we can utilize the extra assumption that the clauses are distinct and each variable occurs at most once in a clause to see that every formula is 4-partially satisfiable in this way. Every instance of WellItsSomething-SAT is a YES instance given this assumption. ■

Solution: This problem is discussed in detail in this Piazza thread:

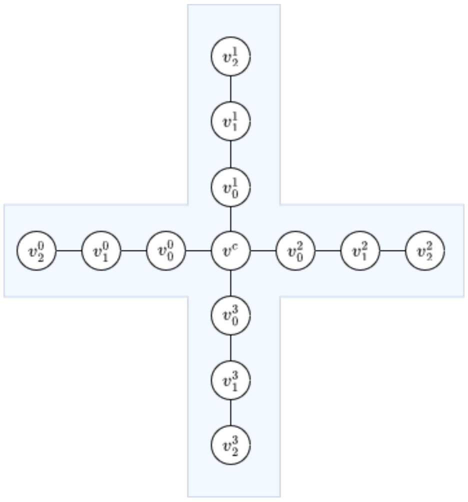
https://piazza.com/class/l6s54v47q0l55t/post/400 ■

(b) For any integer $k > 1$, a (k)-*plus* is a graph formed by 4 *path*s meeting at a particular vertex , each one on k vertices, plus another vertex linking all the graphs together (i.e., it's a graph with $2k$ vertices). See the figure below for an example of a (3)-plus. The PLUS problem is defined as follows:

- INPUT: An ***undirected*** graph G ***and*** an integer k.
- OUTPUT: TRUE if G contains a (k)-plus as a subgraph, FALSE otherwise.

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

P NP co-NP NP-hard NP-complete



Solution: P [✓ NP] co-NP [✓ NP-hard] [✓ NP-complete]

The PLUS problem is in NP-complete, so it is also in NP and NP-hard based on algorithmic complexity classes. Also, we need to prove the problem is in NP and NP-hard for proving NP-complete.

The problem is clearly in NP. Given an instance G, k, a solution is made out of four lists and one extra vertex. There are k vertices for each list. One can check in polynomial time that each list corresponds to a path of size k, and that there are four edges, and each edge is between the start/end vertex of each list and the extra vertex.

As for the reduction from an NP-Complete problem. We consider Longest Path. The goal is to produce G' such that G has a Longest Path with size k if G' contains a (k)-plus as a subgraph. This can be done by adding a vertex v^c with edges to every vertex in the original graph G. For the vertex v^c, we attach start/end vertex of three paths of size k to the vertex v^c. The new graph will be G'.

$\Rightarrow$: If there exists a Longest Path with size k in G, starting with vertex s and ending with vertex t. Both s and t are connect with the vertex v^c, and there are four paths of size k attached to vertex v^c. Then is a (k)-plus in G'.

$\Leftarrow$: In the other case, if there is a (k)-plus in G', then removing attached vertex v^c with three paths of size k will return a Longest Path with size k in G.

Clearly, the reduction is polynomial time. Therefore, the Plus problem is in NP-complete.

■

4 Classification II (Decidability) - 40 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", prove your choice correct by describing an algorithm that decides that language. If you choose "undecidable", prove your choice correct by giving a reduction proving its correctness.
- Regardless of your choice, explain *briefly* (i.e., in 3 sentences maximum) why the proof of the choice you gave is valid.

(a)

$$\text{AtLeastOne}_{TM} = \{\langle M\rangle \mid M \text{ is a } \textbf{\textit{TM}} \text{ and } L(M) \text{ accepts at least one string}\}$$

Solution: decidable [✓ undecidable]

This is undecidable.

We will show HALT $\leq$ AtLeastOne, since $HALT$ is a known undecidable problem.

We will assume $M_{\text{AtLeastOne}}$ exists and use it as a blackbox to solve HALT.

We define our solution $M_{\text{HALT}}(\langle M, w\rangle)$ as:

- We want to define M' as a machine that accepts at least one string if M halts on w, and M' is empty if M does not halt on w.
- Let $M'(w')$ be:

```
    run $M$ on input $w$ until it halts
    output Accept
```

- Return $M_{\text{AtLeastOne}}(\langle M'\rangle)$

Notice that when M halts on w, M' accepts at least one string (in fact, it accepts every input). And when M does not halt on w, M' does not accept any string (in fact it always diverges). That completes the reduction.

■

(b)

$$\textsc{Empty}_{NFA} = \{\langle A\rangle \mid A \text{ is a } \textbf{\textit{NFA}} \text{ and } L(A) \text{ is empty}\}$$

decidable undecidable

Solution: [✓ decidable] undecidable

This is decidable. To briefly describe an algorithm, we can consider the NFA $A = (\delta, Q, s, Acc, \Sigma)$ as a graph, where $G = (Q, E)$ where E contains (u, v) if $v = \delta(u, c)$ for some $c \in \Sigma$. We can apply a DFS graph search to ask if any accepting state $q \in Acc$ is reachable by a path from the start state s. If so, that path describes a string that the NFA accepts, hence it is not empty.

Sidenote: A good number of people attempted to use the standard halting reduction template seen in the lab and apply it to this problem. That doesn't work here because the oracle for $EMPTY_{NFA}$ doesn't take in account ■

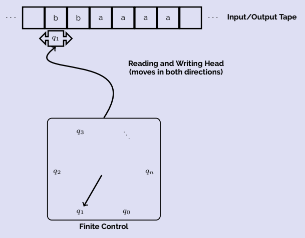
This page is for additional scratch work!

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

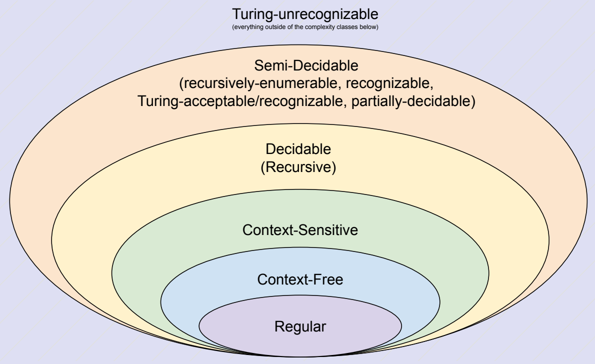
$\delta(q, c) = (p, d, \leftarrow)$

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

$$\begin{tikzcd} q \arrow[r, "c/d, L"] & p \end{tikzcd}$$

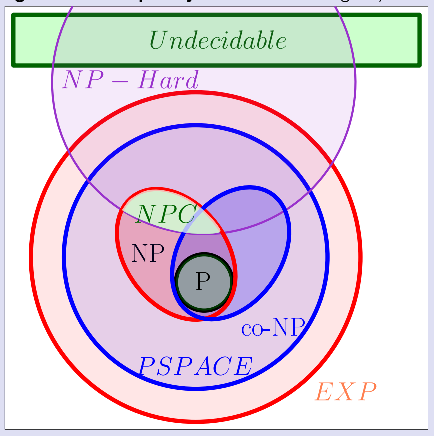
Complexity Classes

Computational Complexity Classes



Turing-unrecognizable
(everything outside of the complexity classes below)
Semi-Decidable
(recursively-enumerable, recognizable, Turing-acceptable/recognizable, partially-decidable)
Decidable
(Recursive)
Context-Sensitive
Context-Free
Regular

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



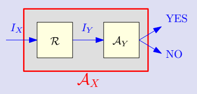
Reductions

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 consecutive 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:	$EQ_{TM} = \left\{\langle M_A, M_B\rangle \;\middle|\; \begin{array}{l} M_A \text{ and } M_B \text{ are TM's} \\ \text{and } L(M_A) = L(M_B) \end{array}\right\}$