1 Short Answer (2 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. Also assume $P \neq NP$.

(a) (12 POINTS) Assuming the reductions below can be proven,circle all the classes that the problem X *may* belong to:

- $3SAT \leq_P X$

Solution: P NP NP-hard NP-complete

decidable undecidable

■

- $X \leq_P CLIQUE$

Solution: P NP NP-hard NP-complete

decidable undecidable

■

- $X \implies A_{TM}$

Solution: P NP NP-hard NP-complete

decidable undecidable

■

(b) (8 POINTS) *Briefly* describe a reduction that shows:

$$SAT \implies A_{TM}$$

Solution: Let $A_{TM}(\langle M, w\rangle)$ be an oracle. Then SAT can be reduced to A_{TM} as the following.

```
SAT(φ):
  Encode the following Turing machine M':
      M'(x):
        Try all possible assignments on φ
        If there is a satisfying assignment for φ:
            accept
  return A_TM(⟨M', x⟩)
```

Where x can be any string. ∎

2 Classification I (P/NP) - 20 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!

General shortest-**simple**-path problem. Given a graph G, assuming every edge can be taken only once (recall that, that's what simplicity means, every edge can only be used once), **does there exist a path from s to t that is less than k length.** The graph **may have negative cycles**, but that doesn't mean there isn't a shortest **simple** path because every edge can only be taken once.

- INPUT: A graph G and vertices s, t, and integer k.
- OUTPUT: TRUE if there exists a simple path $\leq k$ length.

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

Solution: P NP NP-hard NP-complete

To show NP-hard we do a reduction from the LongestPath(G, k). Construct G' by multiplying all edges weights by -1. Then we run ShortestSimplePath$(G', s, t, -k)$ for all distinct vertex pairs s, t. If one pair returns True then return True, else return False.

$\Rightarrow$ Suppose LongestPath(G, k) returns True. So G has a path length $\geq k$, say length r. Then let s, t be the start and end vertices of this path. This path in G' will have length $-r \leq -k$. Therefore ShortestSimplePath$(G', s, t, -k)$ would return True.

$\Leftarrow$ Suppose ShortestSimplePath$(G', s, t, -k)$ returns True for some distinct vertex pair s, t. So there is a path from s to t in G' with length $\leq -k$, say length $-r$. This path in G will have length $r \geq k$. Therefore LongestPath(G, k) would return True.

G' is constructed in polynomial time and ShortestSimplePath is run a polynomial number of times. So ShortestSimplePath reduces to LongestPath and is therefore NP-Hard.

To prove NP we use a Certifier. The Certificate is P, a simple path in G. The Certifier checks by summing up the edge weights on this path and comparing it to k. The sum takes $O(|V|)$ time. So the Certifier is efficient. Therefore ShortestSimplePath is NP.

ShortestSimple Path is NP and NP-Hard so it is NP-Complete ■

3 Classification II(P/NP) - 20 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!

Specialized (DAG) shortest-**simple**-path problem. Same problem as before, but this time G is a directed-acyclic graph (DAG). Given a graph G, assuming every edge can be taken only once, **does there exist a path from s to t that is less than k length.**

- INPUT: A DAG G ***and*** an integer k.
- OUTPUT: TRUE if there exists a simple path $\leq k$ length.

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

Solution: P NP-hard NP-hard NP-complete

Since there is no Cycles in the graph, the possibility for a negative cycle is eliminated. We can find the shortest path from s to t. Since there might be negative edges, we can not use Dijkstra's algorithm, instead we can use Topological Sorting and find the shortest distance from s to t using the method in HW 7 P2 a. Then we can check whether the shortest path is less than k.

The runtime for this is O(V+E) and hence the problem can be solved in Polynomial time. ■

4 Classification I (Decidability) - 20 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, diagrams, *clear* pseudo-code) why the proof of the choice you gave is valid.

$$\text{EndWith0}_{DFA} = \{\langle D \rangle \mid D \text{ is a } \textbf{\textit{DFA}} \text{ and all w in } L(D) \text{ end with the character } 0\}$$

Solution: decidable undecidable

To prove that the language $EndWith0_{DFA}$ is decidable, we need to show that there exists a Turing machine that decides $EndWith0_{DFA}$.

The following TM T decides $EndWith0_{DFA}$.

T = "On input $\langle D \rangle$, where D is a DFA:

1) Mark the accepting states of D that can be reached from the start state. Let these be set A.

2) Check if all the incoming transitions to A are using 0.

3) If yes then accept; otherwise, reject." ■

5 Classification II (Decidability) - 20 points

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

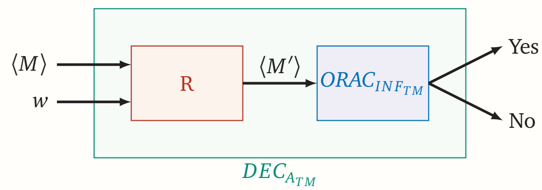
- Circle one of "decidable" or "undecidable" to indicate your choice.
- If you choose "decidable", prove your choice correct by describing an algorithm that decides that language. If you choose "undecidable", prove your choice correct by giving a reduction proving its correctness.
- Regardless of your choice, explain *briefly* (i.e., in 3 sentences maximum, diagrams, *clear* pseudo-code) why the proof of the choice you gave is valid.

$$\text{INF}_{TM} = \{\langle M\rangle \mid M \text{ is a } \textbf{\textit{TM}} \text{ and } |L(M)| = \infty\}$$

Solution: decidable undecidable

This is a lab question and we do a reduction from the accept language:

$$A_{TM} \Rightarrow INF_{TM}$$



The reduction is as follows. On input $\langle M, w\rangle$ we encode the following machine:

$\underline{M'(x)}$:
run M on input w and return TRUE if M accepts w
otherwise return false

$\underline{DEC_{ATM}(w)}$:
Construct M' using M and w
if $ORAC_{INF_{TM}}(< M' >)$
return TRUE
else
return FALSE

In this case, if $ORAC_{INF_{TM}}$ output true, we know that the language M' represents is infinite which is only possible if M accepts w. If M does not accept w, then the language represented by M' is not infinite and hence the oracle $ORAC_{INF_{TM}}$ correctly returns a false. ■

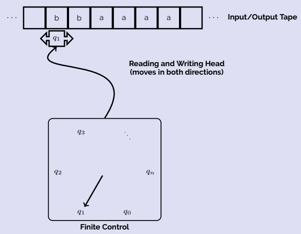
This page is for additional scratch work!

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

Turing Machines

Turing machine is the simplest model of computation.

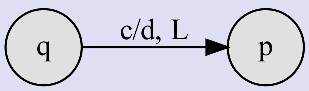
- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).
- Ever step: Read character under head, write character out, move the head right or left (or stay).
- Every TM **M** can be encoded as a string $\langle M \rangle$



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

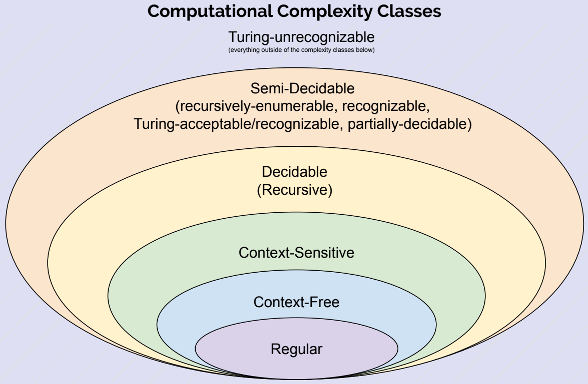
$\delta(q, c) = (p, d, \leftarrow)$

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

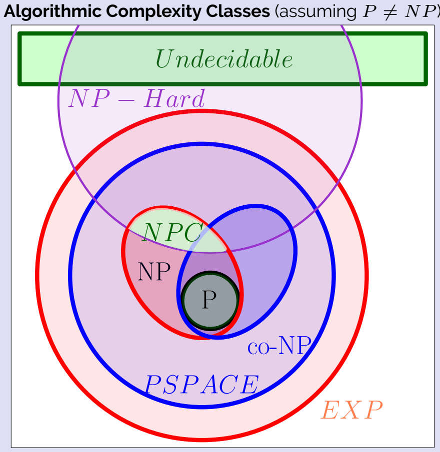


Complexity Classes

Computational Complexity Classes



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



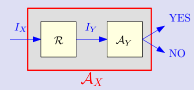
Reductions

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: $A_{TM} = \{\langle M, w \rangle \mid M$ is a TM and M accepts on $w\}$

HaltsOnInput: $Halt_{TM} = \{\langle M, w \rangle \mid M$ is a TM and halts on input $w\}$

HaltOnBlank: $HaltB_{TM} = \{\langle M \rangle \mid M$ is a TM & M halts on blank input$\}$

Emptiness: $E_{TM} = \{\langle M \rangle \mid M$ is a TM and $L(M) = \varnothing\}$

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