CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 1 Solution

For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No".

Read each statement *very* carefully; some of these are deliberately subtle!

(a) Which of the following statements are true for ***every*** language $L \subseteq \{0,1\}^*$?

- L^* is infinite if and only if L is infinite.

  Yes [ ] No [X] $L = \{1\}$ is finite but $L^* = 1^*$ is infinite.

- L has a finite fooling set.

  Yes [X] No [ ] The empty set is a fooling set for every language.

- L is context-free if and only if L is not regular.

  Yes [ ] No [X] HALT is neither regular nor context-free.

- If there is a Turing machine M that halts on every string in L, then L is decidable.

  Yes [ ] No [X] M_{ACCEPT} halts on *every* string, and therefore halts on every string in HALT.

- If both L and $\Sigma^* \setminus L$ are acceptable, then L is decidable.

  Yes [X] No [ ] Let A and B be accepting machines for L and $\Sigma^* \setminus L$. Consider a Turing machine M that runs A and B in parallel, accepts if A accepts, and rejects if B accepts. Then M decides L.

Problem 1 continues on the next page.

(b) Which of the following statements are true for ***every*** language $L \subseteq \{0,1\}^*$ and ***every*** language $A \subseteq \{0,1\}^*$ where $A \in NP$?

- If $L \in$ NP, then $(L \cup A) \in$ NP.

  [X] Yes [ ] No Accept w iff at least one of the poly-time NTMs for L and A accepts w.

- If $L \notin$ NP, then $(L \cup A) \notin$ NP.

  [ ] Yes [X] No Consider $A = \Sigma^*$.

- If $L \in$ NP, then $(L \cap A) \in$ NP.

  [X] Yes [ ] No Accept w iff both of the poly-time NTMs for both L and A accept w.

- If $L \notin$ NP, then $(L \cap A) \notin$ NP.

  [ ] Yes [X] No Consider $A = \emptyset$.

- If there is a polynomial time reduction from A to L, then L is NP-hard.

  [ ] Yes [X] No Only if A is also NP-hard.

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 2 Solution

For each statement below, there are two boxes in the answer booklet labeled "Yes" and "No". Check "Yes" if the statement is ***always*** true and "No" otherwise, and give a ***brief*** (at most one short sentence) explanation of your answer. **Assume $P \neq NP$.** If there is any other ambiguity or uncertainty about an answer, check "No".

(a) Which of the following languages are ***decidable***?

- $\{\langle M \rangle \mid M$ has at most 374 states$\}$

  [X] Yes [ ] No — The encoding of M includes an encoding of the number of states.

- $\{\langle M \rangle \mid M$ accepts a finite number of strings$\}$

  [ ] Yes [X] No — Rice's theorem.

- $\{\langle M \rangle \mid M$ diverges on every input$\}$

  [ ] Yes [X] No — Rice's divergence theorem / We saw this in class.

- $\{\langle M \rangle \mid$ There are exactly 374 palindromes that M does *not* accept$\}$

  [ ] Yes [X] No — Rice's theorem.

- $\{\langle M \rangle \mid \langle M \rangle$ is a palindrome$\}$

  [X] Yes [ ] No — Treat $\langle M \rangle$ as a raw string and compare $\langle M \rangle$ with its reversal.

Problem 2 continues on the next page.

(b) Suppose we want to prove that the following language is undecidable.

$$\textsc{Demon} := \{\langle M\rangle \mid M \text{ accepts } \texttt{SOULS} \text{ and } \texttt{SHAME} \text{ but diverges on } \texttt{SODAPOP}\}$$

Your friend Rumi suggests a reduction from the standard halting language

$$\textsc{Halt} := \{(\langle M\rangle, w) \mid M \text{ halts on input } w\}.$$

Specifically, suppose there is a machine TAKEDOWN that decides DEMON. Rumi claims that the following algorithm decides HALT.

```
DECIDEHALT(⟨M⟩, w):
  Write code for the following algorithm:
      CATCHY(x):
        if x = SOULS or x = SHAME
            run M on input w
            ⟨⟨ignore the output of M⟩⟩
            return TRUE
        else
            loop forever
  return TAKEDOWN(⟨CATCHY⟩)
```

Which of the following statements must be true for ***all*** inputs $(\langle M\rangle, w)$?

- If M halts on w, then CATCHY diverges on SODAPOP.

  [X] Yes [ ] No

  CATCHY *always* diverges on SODAPOP.

- If M diverges on w, then TAKEDOWN accepts ⟨CATCHY⟩.

  [ ] Yes [X] No

  If M diverges on w, then CATCHY diverges on SOULS, so ⟨CATCHY⟩ $\notin$ DEMON

- If M halts on w, then TAKEDOWN accepts ⟨CATCHY⟩.

  [ ] Yes [X] No

  If M halts on w, then CATCHY accepts SOULS and SHAME but diverges on SODAPOP, so ⟨CATCHY⟩ $\in$ DEMON

- DECIDEHALT decides the language HALT. (That is, Rumi's reduction is correct.)

  [X] Yes [ ] No

  See previous two statements.

- We could instead prove DEMON is undecidable using Rice's theorem.

  [ ] Yes [X] No

  The definition of DEMON is only about both acceptance and divergence

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 3 Solution

Suppose you are given a ***sorted*** array $A[1..n]$ containing n distinct positive integers, all strictly less than $2n-1$. Describe and analyze an algorithm that finds two *consecutive* integers in this array.

Solution: The following variant of binary search finds an adjacent pair in $O(\log n)$ ***time***.

```
FindAdjacent(A[1..n]):
  lo ← 1
  hi ← n
  while hi − lo > 2
      mid ← ⌊(hi + lo)/2⌋
      if A[mid] − A[lo] < A[hi] − A[mid]
          hi ← mid
      else
          lo ← mid
  return lo
```

If the subarray $A[i..j]$ does *not* contain two consecutive integers, then $A[j]-A[i] \geq 2(j-i)$, because the array A is sorted and contains distinct integers. Equivalently, if $A[j]-A[i] < 2(j-i)$, then $A[i..j]$ must contain two consecutive integers. Initially we have $A[i] \geq 1$ and $A[n] < 2n-1$, so $A[n]-A[1] < 2n-2$.

At the start of each iteration of the while loop, we have $A[hi]-A[lo] < 2(hi-lo)$, so the subarray $A[hi..lo]$ must contain a consecutive pair. ■

Solution: The following variant of binary search finds an adjacent pair in $O(\log n)$ ***time***.

```
FindAdjacent(A[1..n]):
  lo ← 1
  hi ← n
  while hi − lo > 1
      mid ← ⌊(hi + lo)/2⌋
      if A[mid + 1] = A[mid] + 1
          return mid
      else if A[mid] − A[lo] < 2(mid − lo)
          hi ← mid
      else
          lo ← mid + 1
  return lo
```

Again, at the start of each iteration of the while loop, we have $A[hi]-A[lo] < 2(hi-lo)$, so the subarray $A[hi..lo]$ must contain a consecutive pair. ■

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 4 Solution

Submit a solution to ***exactly one*** of the following problems.

(a) ***Prove*** that the following **3IN5SAT** problem is NP-hard: Given a boolean formula Φ in conjunctive normal form, with *five* literals in each clause, is there an assignment to the variables such that each clause of Φ contains *at least three* TRUE literals?

Solution: We reduce from the standard 3SAT problem.

Let Φ be any 3CNF formula with m clauses and n variables $x_1, x_2, \ldots, x_n$. We transform Φ into a 5CNF formula by adding two literals y and z to every clause, where y and z are (the same) two new variables.

$\Longrightarrow$ Suppose Φ is satisfiable. Fix a satisfying assignment to the variables $x_1, x_2, \ldots, x_n$. Every clause in Φ has at least one TRUE literal. If we add the assignments $y =$ TRUE and $z =$ TRUE, then every clause in Φ' has at least three TRUE literals, namely y, z, and one literal from Φ.

$\Longleftarrow$ Suppose there is an assignment such that every clause in Φ' has at least three TRUE literals. Then each clause of Φ' has at least one TRUE literal that is not y or z. It follows that each clause in Φ has at least one TRUE literal; in other words, Φ is satisfiable.

Constructing Φ' from Φ in polynomial time is straightforward. ■

(b) ***Prove*** that the following **67TREE** problem is NP-hard: Given an undirected graph $G = (V, E)$, does G contain a spanning tree T such that every vertex has degree at most 67 in T?

Solution: We reduce from HAMILTONIANPATH.

Let $G = (V, E)$ be an arbitrary undirected graph. We construct a new undirected graph G' by adding a "fan" of 65 edges to each vertex of G. That is, for each vertex $v \in V$, we add new vertices $v_1, v_2, \ldots v_{65}$ and edges $vv_1, vv_2, \ldots vv_{65}$. Altogether G' has $66V$ vertices (exactly $65V$ of which have degree 1) and $E + 65V$ edges.

$\Longrightarrow$ Suppose G has a Hamiltonian path P. Then P is also a spanning tree of G with maximum degree 2. For each vertex $v \in V$, add all 65 edges vv_i to P; the result is a spanning tree of G' with maximum degree 67.

$\Longleftarrow$ Suppose G' has a spanning tree T' with maximum degree at most 67. Every vertex v_i added in the construction of G' must be a leaf in T'. If we delete these $65V$ leaves, the result is a spanning tree T of G that has maximum degree at most 2. But a tree with maximum degree at most 2 is just a path. Thus, T is a Hamiltonian path in G.

Constructing G' from G in polynomial time is straightforward. ■

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 5 Solution

Suppose you are given three strings $A[1..n]$, $B[1..n]$, and $C[1..n]$. Describe and analyze an algorithm to find the length of the longest common subsequence of A, B, and C.

Solution: Let $LCS3(i, j, k)$ denote the length of the longest common subsequence of the suffixes $A[i..n]$, $B[j..n]$, and $C[k..n]$. We need to compute $LCS3(1, 1, 1)$. This function obeys the following recurrence:

$$LCS3(i,j,k) = \begin{cases} 0 & \text{if } i > n \text{ or } j > n \text{ or } k > n \\ \max\left\{\begin{array}{c} 1 + LCS3(i+1, j+1, k+1) \\ LCS3(i+1, j, k) \\ LCS3(i, j+1, k) \\ LCS3(i, j, k+1) \end{array}\right\} & \text{if } A[i] = B[j] = C[k] \\ \max\left\{\begin{array}{l} LCS3(i+1, j, k) \\ LCS3(i, j+1, k) \\ LCS3(i, j, k+1) \end{array}\right\} & \text{otherwise} \end{cases}$$

We can memoize this function into a three-dimensional array $LCS3[1..n, 1..n, 1..n]$. We can fill this array using three nested loops, decreasing i, decreasing j, and decreasing k, in $\boldsymbol{O(n^3)}$ ***time***. (The nesting order of the three loops doesn't matter.) ■

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 6 Solution

Exactly one of the following languages is regular. Which one?

1. $\{0^a 1^b 0^c \mid a \le b \le c\}$
2. $\{0^a 1^b \mid a \bmod 3 \le b \bmod 3\}$

Indicate which one of these two languages is regular. Describe a DFA or NFA that accepts the regular language, and ***prove*** that the other language is not regular.

Solution: ***The language in part (b) is regular.***

(a) Let $F = 1^*$.

Fix any two strings x and y from F. Then $x = 1^i$ and $y = 1^j$ for some integers $i \neq j$.

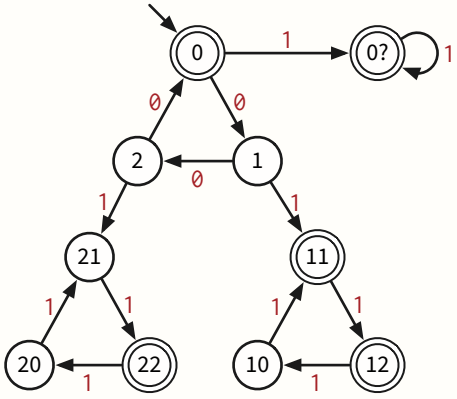
Without loss of generality, we can assume $i < j$ and therefore $0 \le i \le j-1$. (Otherwise, swap the variable names $x \leftrightarrow y$ and $i \leftrightarrow j$.)

Let $z = 0^{j-1}$.

- Then $xz = 0^0 1^i 0^{j-1} \in L$ because $0 \le i \le j-1$.
- But $yz = 0^0 1^j 0^{j-1} \in L$ because $j > j-1$.

We conclude that F is a fooling set for L. Because F is infinite, L cannot be regular.

(b) All missing transitions in the following DFA lead to a hidden dump state.



The states are named as follows:

- Each single-digit state i means that we have read $0^a 1^b$ where $a \bmod 3 = i$ and $b = 0$.
- The state 0? means that we have read $0^a 1^b$ where $a \bmod 3 = 0$ and $b > 0$.
- Each double-digit state ij means we have read $0^a 1^b$ where $a \bmod 3 = i$ and $b > 0$ and $b \bmod 3 = j$. ■

CS/ECE 374 A ✧ Fall 2025
Conflict Final Exam Problem 7 Solution

Suppose you are given a directed graph $G = (V, E)$, where each edge has a positive weight, two vertices s and t, and an integer k. Describe an algorithm that computes the length of the shortest walk from s to t that traverses ***at most k edges in the wrong direction***. Analyze your algorithm in terms of V, E, and k.

Solution: We construct a new layered graph $G' = (V', E')$ as follows:

- $V' = V \times \{0, 1, \ldots, k\}$. Each vertex (v, i) means we have reached vertex v after traversing exactly i backward edges.
- E' is the union of two subsets $E^+ \cup E^-$ defined as follows:
  - Forward edges $E^+ = \big\{(u,i)\to(v,i) \;\big|\; u\to v \in E \text{ and } 0 \le i \le k\big\}$.
    Each forward edge $(u,i)\to(v,i)$ has weight $w(u\to v)$.
  - Backward edges $E^- = \big\{(u,i)\to(v,i+1) \;\big|\; v\to u \in E \text{ and } 0 \le i \le k-1\big\}$.
    Each backward edge $(u,i)\to(v,i+1)$ has weight $w(v\to u)$.

We need to compute the length of the shortest path from $(s, 0)$ to any vertex (t, i). We can compute all k of these shortest-path lengths using a single call to Dijkstra's algorithm in $O(E' \log V') = \boldsymbol{O(kE \log kV)}$ ***time***. ■