CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 1

For each of the following questions, indicate ***every*** correct answer by marking the "Yes" box, and indicate ***every*** incorrect answer by marking the "No" box. **Assume P ≠ NP.** If there is any other ambiguity or uncertainty, mark the "No" box.

Rubric: 20 points total =
- + 1/2 for each correct answer
- − 1/4 for each incorrect answer
- + 1/8 for each IDK
- ± 0 for each unanswered

Explanations are obviously not required. Round negative total scores up to 0.

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

☐ Yes ☒ No L is finite.

Solution: 0^* is infinite. ■

☒ Yes ☐ No L^* contains the empty string ε.

Solution: By definition. ■

☐ Yes ☒ No L^* is decidable.

Solution: For any undecidable language A in 0^*, the language $(A1)^*$ is undecidable. ■

☒ Yes ☐ No If L is regular then $\Sigma^* \setminus L^*$ is regular.

Solution: If L is regular, then L^* is regular (by definition), and therefore $\Sigma^* \setminus L^*$ is regular (by flipping accept/reject states in the DFA). ■

☒ Yes ☐ No If L is the intersection of two decidable languages, then L is decidable.

Solution: If A and B are decidable, then we can decide whether a given string A is in $A \cap B$ by deciding whether $w \in A$ **and** $w \in B$. ■

☐ Yes ☒ No If L is the intersection of two undecidable languages, then L is undecidable.

Solution: Let A be any undecidable language. The complementary language $B = \Sigma^* \setminus A$ is also undecidable, but $A \cap B = \emptyset$ is trivially decidable. ■

Yes [X] No

If L^* is the complement of a regular language, then L is regular.

Solution: The language $L = \{0^{n^2} \mid n \geq 0\}$ is not regular, but $L^* = 0^*$ is the complement of a regular language (because 0^* is regular). ■

Yes [X] No

If L is undecidable, then every fooling set for L is infinite.

Solution: *Every* language has finite fooling sets. In particular, the empty set is a fooling set for *every* language. ■

Yes [X] No

L is decidable if and only if its complement $\overline{L}$ is undecidable.

Solution: In fact, L is decidable if and only if its complement $\overline{L}$ is *decidable*. ■

(b) Which of the following statements is true for ***every*** directed graph $G = (V, E)$?

Yes [X] No

$E \neq \emptyset$.

Solution: A set of vertices with no edges is still a graph. ■

Yes [X] No

Given the graph G as input, Floyd-Warshall runs in $O(E^3)$ time.

Solution: Floyd-Warshall runs in $\Theta(V^3)$ time. If G is *connected*, the $O(E^3)$ upper bound is correct, because $V = O(E)$. But in general, E could be significantly smaller than V; see the previous question. ■

Yes [X] No

If G has at least one source and at least one sink, then G is a dag.

Solution: Consider the graph with four vertices w, x, y, z and four edges $w \to x$, $x \to y$, $y \to x$, and $z \to y$. ■

Yes [X] No

We can compute a spanning tree of G using whatever-first search.

Solution: If G is disconnected, it has no spanning tree, so you can't compute one with *any* algorithm. Even if G is *weakly* connected, running WFS may not compute a spanning tree; the tree computed by WFS only contains the vertices reachable from the start vertex. ■

Yes [X] No

If the edges of G are weighted, we can compute the shortest path from any node s to any node t in $O(E \log V)$ time using Dijkstra's algorithm.

Solution: If any of the edge weights are negative, then Dijkstra's algorithm either might not find the shortest path from s to t or might run in exponential time, depending on which version of the algorithm we're talking about. ■

(c) Which of the following languages over the alphabet $\{0, 1\}$ are ***regular***?

Yes | **No** $\{0^m 1 0^n \mid m \geq n\}$

Solution: The suffix $10^{\max\{i,j\}}$ distinguishes 0^i from 0^j, so 0^* is a fooling set. ■

Yes | **No** $\{0^m 1 0^n \mid m - n \geq 374\}$

Solution: The suffix $10^{\max\{i,j\}}$ distinguishes 0^{374+i} from 0^{374+j}, so $0^{374}0^*$ is a fooling set. ■

Yes | No Binary representations of all integers divisible by 374

Solution: This language is accepted by a DFA with 374 states. ■

Yes | **No** $\{xy \mid yx \text{ is a palindrome}\}$

Solution: $\{0^n 1 0^{2n+1} 1 \mid n \geq 0\}$ is a fooling set. Specifically, the suffix 0^i distinguishes $0^i 1 0^{2i+1} 1$ from $0^j 1 0^{2j+1} 1$. (No, you weren't supposed to figure out the fooling set; you were supposed to react to the word "palindrome".) ■

Yes | **No** $\{\langle M \rangle \mid M \text{ accepts a finite number of non-palindromes}\}$

Solution: This language is undecidable (by Rice's Theorem), but all regular languages are decidable. ■

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

Yes | No Binary representations of all integers divisible by 374

Solution: Does $\lfloor n/374 \rfloor \cdot 374 = n$? ■

Yes | No $\{xy \in \{0,1\}^* \mid yx \text{ is a palindrome}\}$

Solution: Given an input string w, for each possible split $w = yx$, check whether xy is a palindrome by brute force. ■

Yes | **No** $\{\langle M \rangle \mid M \text{ accepts the binary representation of every integer divisible by 374}\}$

Solution: By Rice's Theorem. ■

Yes ☒ No

$\{\langle M\rangle \mid M \text{ accepts a finite number of non-palindromes}\}$

Solution: By Rice's Theorem. ■

☒ Yes No

The set of all regular expressions that represent the language $\{0,1\}^*$. (This is a language over the alphabet $\{\emptyset, \varepsilon, 0, 1, \star, +, (,)\}$.)

Solution: Regular expressions have a well-defined syntax, which is easy to parse. (More formally: This language is context-free.) ■

(e) Which of the following languages can be proved undecidable ***using Rice's Theorem***?

☒ Yes No

$\{\langle M\rangle \mid M \text{ accepts an infinite number of strings}\}$

Solution: Let $\mathscr{L}$ be the set of all infinite languages. The machine "return FALSE" accepts $\emptyset \notin \mathscr{L}$, and the machine "return TRUE" accepts $\Sigma^* \in \mathscr{L}$. ■

Yes ☒ No

$\{\langle M\rangle \mid M \text{ accepts either } \langle M\rangle \text{ or } \langle M\rangle^R\}$

Solution: The description of the accepting language refers to the machine encoding. ■

Yes ☒ No

$\{\langle M\rangle \mid M \text{ accepts } 001100 \text{ but rejects } 110011\}$

Solution: Rice's theorem considers *only* the accepting language (or, after some simple modification, only the rejecting language). There is almost certainly a *generalization* of Rice's theorem that states that the language

$$\{\langle M\rangle \mid \text{ACCEPT}(M) \in \mathscr{L}^+ \wedge \text{REJECT}(M) \in \mathscr{L}^-\}$$

is undecidable for any non-trivial disjoint language families $\mathscr{L}^+$ and $\mathscr{L}^-$, but I don't think it's reasonable to call that generalization "Rice's Theorem". ■

Yes ☒ No

$\{\langle M\rangle \mid M \text{ accepts some string } w \text{ after at most } |w|^2 \text{ steps}\}$

Solution: This language is decidable! ■

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

$$\text{CHALMERS} := \{\langle M\rangle \mid M \text{ accepts both } \texttt{STEAMED} \text{ and } \texttt{HAMS}\}$$

Professor Skinner suggests a reduction from the standard halting language

$$\text{HALT} := \{\langle M\rangle \# w \mid M \text{ halts on inputs } w\}.$$

Specifically, suppose there is a Turing machine Ch that decides CHALMERS. Professor Skinner claims that the following algorithm decides HALT.

```
DECIDEHALT(⟨M⟩#w):
  Encode the following Turing machine:
      AURORABOREALIS(x):
        if x = STEAMED or x = HAMS or x = ALBANY
            run M on input w
            return TRUE
        else
            return FALSE
  return Ch(⟨AURORABOREALIS⟩)
```

Which of the following statements is true for all inputs $\langle M \rangle \# w$?

Yes [X] No — If M accepts w, then AURORABOREALIS accepts CLAMS.

Solution: In fact, AURORABOREALIS *always* rejects CLAMS. ■

[X] Yes No — If M rejects w, then AURORABOREALIS rejects UTICA.

Solution: In fact, AURORABOREALIS *always* rejects UTICA. ■

Yes [X] No — If M hangs on w, then AURORABOREALIS accepts every input string.

Solution: In fact, if M hangs on w, then AURORABOREALIS never accepts. ■

[X] Yes No — If M accepts w, then Ch accepts $\langle$AURORABOREALIS$\rangle$.

Solution: ALBANY is a red herring. ■

[X] Yes No — DECIDEHALT decides the language HALT. (That is, Professor Skinner's reduction is actually correct.)

Solution: If M halts on w, then AURORABOREALIS accepts both STEAMED and HAMS, so Ch accepts $\langle$AURORABOREALIS$\rangle$, so DECIDEHALT accepts $\langle M \rangle \# w$. On the other hand, if M hangs on w, then ■

Yes [X] No — DECIDEHALT actually runs (or simulates) M.

Solution: In fact, DECIDEHALT merely modifies the encoding M to produce the encoding $\langle$AURORABOREALIS$\rangle$. ■

[X] Yes No — We could have proved CHALMERS is undecidable using Rice's theorem instead of this reduction.

Solution: Let $\mathscr{L}$ be the set of all languages that contain both STEAMED and CLAMS. ■

(g) Consider the following pair of languages:

- 3COLOR := $\{G \mid G \text{ is a 3-colorable undirected graph}\}$
- TREE := $\{G \mid G \text{ is a connected acyclic undirected graph}\}$

(For concreteness, assume that in both of these languages, graphs are represented by adjacency matrices.) Which of the following ***must*** be true, assuming P≠NP?

[X] Yes [ ] No TREE ∪ 3COLOR is NP-hard.

Solution: Every tree is 3-colorable, so TREE ∪ 3COLOR = 3COLOR. ■

[ ] Yes [X] No TREE ∩ 3COLOR is NP-hard.

Solution: Every tree is 3-colorable, so TREE ∩ 3COLOR = TREE. ■

[ ] Yes [X] No 3COLOR is undecidable.

Solution: Try each of the 3^V possible 3-colorings by brute force. ■

[ ] Yes [X] No There is a polynomial-time reduction from 3COLOR to TREE.

Solution: Such a reduction would imply a polynomial-time algorithm for 3COLOR. ■

[X] Yes [ ] No There is a polynomial-time reduction from TREE to 3COLOR.

Solution: We can decide whether a given graph is a tree in polynomial time even without calling the 3COLOR oracle. ■

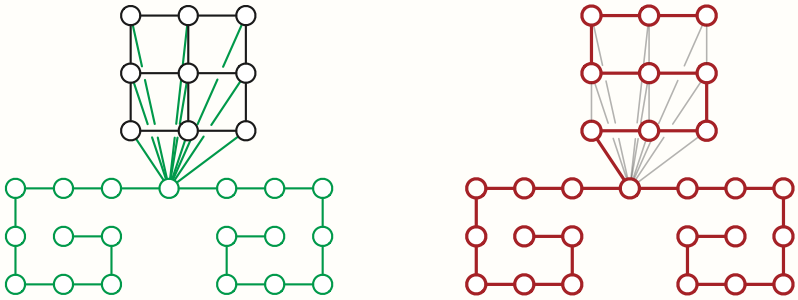
CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 2

A ***wye*** is an undirected graph that looks like the capital letter Y. More formally, a wye consists of three paths of equal length with one common endpoint, called the *hub*.

Prove that the following problem is NP-hard: Given an undirected graph G, what is the largest wye that is a subgraph of G? The three paths of the wye must not share any vertices except the hub, and they must have exactly the same length.

Solution: We can prove this problem is NP-hard by reduction from the Hamiltonian path problem in undirected graphs.

Let G be an arbitrary undirected graph, and suppose G has n vertices. We construct a new graph H from G by adding a path of $2n+1$ new vertices $x_n, x_{n-1}, \ldots, x_1, y, z_1, z_2, \ldots, z_n$, along with edges from the middle vertex y to every original vertex of G. The resulting graph H has $3n+1$ vertices. I claim that H contains a wye whose arms have length n if and only if G contains a Hamiltonian path.



$\Longrightarrow$ Suppose G contains a Hamiltonian path $v_1, v_2, \ldots, v_n$. Adding the edge yv_1 and the new path $x_n, x_{n-1}, \ldots, x_1, y, z_1, z_2, \ldots, z_n$ to this Hamiltonian path gives us a wye in H with arm-length n.

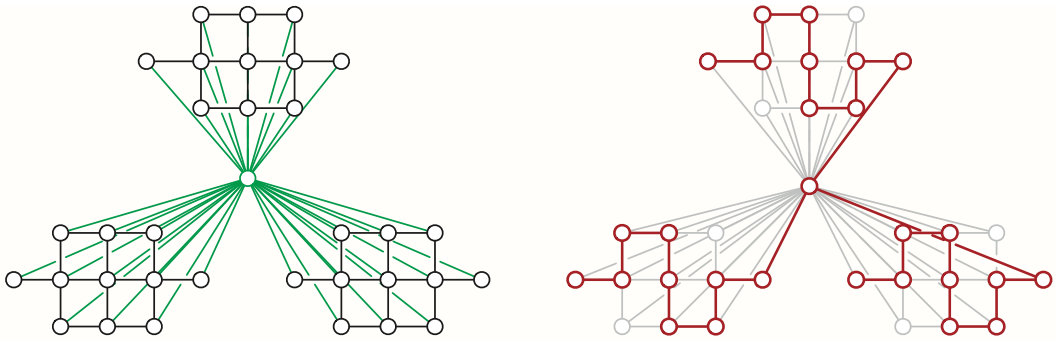
$\Longleftarrow$ Suppose H contains a wye Y whose arms have length n. Then Y must contain all $3n+1$ vertices of H, and in particular must contain the entire path $x_n, \ldots, x_1, y, z_1, \ldots, z_n$. The hub of Y must be y (the only vertex on this path with degree more than 2), and two of the arms must be $x_n, \ldots, x_1$ and $z_1, \ldots, z_n$. Let $y, v_1, v_2, \ldots, v_n$ be the third arm of Y. Every vertex v_i is a vertex of G; thus, the subpath $v_1, v_2, \ldots, v_n$ is a Hamiltonian path in G.

We can construct H from G in polynomial time by brute force. ■

Solution: We can prove this problem is NP-hard by reduction from the longest path problem in undirected graphs.

Let G be an arbitrary undirected graph, and suppose G has n vertices. We construct a new graph H from three copies G_1, G_2, G_3 of G by adding a new vertex y and new edges from y to every vertex of every copy of G. The resulting graph H has $3n+1$ vertices.

I claim that the length of the longest path in G is one less than the arm-length of the largest wye in H. As usual, we prove this claim to two steps.



$\Longrightarrow$ Suppose the longest path in G has length k. Connecting the three copies of this path to y gives us a wye in H with arm-length $k+1$.

$\Longleftarrow$ Let Y be the largest wye in H, and suppose the arms of Y have length ℓ. Clearly $\ell \geq 1$. There are two cases to consider:

- Suppose the hub of Y is the central vertex y. Then each arm of Y contains a path of length $\ell - 1$ in one of the copies of G.
- On the other hand, suppose the hub of Y is *not* the central vertex y. At most one path of Y can pass through the central vertex y. Thus, by removing one arm of Y, we obtain a path of length $2\ell > \ell - 1$ in one of the copies of G.[a]

In both cases, we find a path of length at least $\ell - 1$ in G.

We can construct H from G in polynomial time by brute force. ■

[a] The forward argument now implies that H contains a wye with arm-length $2\ell + 1$, contradicting our assumption that Y is the largest wye in H, but the proof is already done, so I don't care.

Rubric: 10 points, standard NP-hardness rubric. This is not the only correct solution. This solution is more detailed than necessary for full credit.

Many people forgot to prove that the hub of the largest wye in H must be y.

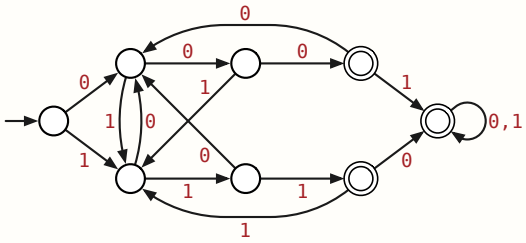
CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 3

Fix the alphabet $\Sigma = \{0, 1\}$. Recall that a *run* in a string $w \in \Sigma^*$ is a maximal non-empty substring in which all symbols are equal.

(a) Let L be the set of all strings in Σ^* that contains at least one run whose length is divisible by 3. Describe both a regular expression for L and a DFA that accepts L.

Solution:

$$(\varepsilon + (0+1)^*1)\,000(000)^*(\varepsilon + 1(0+1)^*)$$
$$+$$
$$(\varepsilon + (0+1)^*0)\,111(111)^*(\varepsilon + 0(0+1)^*)$$



■

Rubric: 5 points = 2½ for regular expression + 2½ for DFA. Grade as two separate subproblems (for IDK).

(b) Let L' be the set of all strings in Σ^* that have the same number of even-length runs and odd-length runs. ***Prove*** that L' is not regular.

Solution: Let $F = (01)^*$.
Let x and y be arbitrary distinct strings in F.
Then $x = (01)^i$ and $y = (01)^j$ for some integers $i \neq j$.
Let $z = (0011)^i$.
Then $xz = (01)^i(0011)^i$ has $2i$ odd runs and $2i$ even runs, so $xz \in L$.
But $yz = (01)^j(0011)^i$ has $2j$ odd runs and $2i$ even runs, so $yz \notin L$.
We conclude that F is a fooling set for L'.
But F is infinite, so L' cannot be regular. ■

Rubric: 5 points =

- 1 for proposing an explicit infinite set F
- 2 if F is actually a fooling set
- 1 for consider arbitrary pairs of strings in F
- 1 for distinguishing suffix for every pair of strings in F

This is not the only correct solution.

CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 4

Suppose we want to split an array $A[1..n]$ of integers into k contiguous intervals that partition the sum of the values as evenly as possible. Specifically, define the *quality* of such a partition as the minimum, over all k intervals, of the sum of the values in that interval; our goal is to maximize quality. Describe and analyze an algorithm to compute the maximum quality of a partition of A into k intervals, given the array A and the integer k as input.

Solution: We define three functions:

- $PreSum(i)$ is the sum of all elements of the prefix $A[1..i]$.
- $Sum(i,j)$ is the sum of all elements of the interval $A[i..j]$.
- $MaxQ(i,\ell)$ is the maximum quality of a partition of $A[i..n]$ into ℓ intervals.

We need to compute $MaxQ(1,k)$. These functions satisfy the following recurrences:

$$PreSum(i) = \begin{cases} 0 & \text{if } i = 0 \\ A[i] + PreSum(i-1) & \text{otherwise} \end{cases}$$

$$Sum(i,j) = PreSum(j) - PreSum(i-1)$$

$$MaxQ(i,\ell) = \begin{cases} Sum(i,n) & \text{if } \ell = 1 \\ \max\left\{ \min\left\{ \begin{array}{c} Sum(i,j), \\ MaxQ(j+1,\ell-1) \end{array} \right\} \;\middle|\; i \le j \le n \right\} & \text{otherwise} \end{cases}$$

We can memoize $PreSum$ into an array $PreSum[0..n]$, which we can fill from left to right in $O(n)$ time. We don't need to memoize Sum. Finally, we can memoize $MaxQ$ into an array $MaxQ[1..n, 0..k]$, which we can fill in standard column-major order: increasing ℓ in the outer loop, and considering i in any order in the inner loop.

The overall algorithm runs in $\boldsymbol{O(n^2k)}$ ***time***. ■

This solution assumes that partitions are allowed to contain empty intervals; without loss of generality, all the empty intervals in any partition are at the end of the array. To forbid empty intervals, modify the base cases as follows:

$$MaxQ(i,\ell) = \begin{cases} -\infty & \text{if } i > n \\ Sum(i,n) & \text{if } i \le n \text{ and } \ell = 1 \\ \max\left\{ \min\left\{ \begin{array}{c} Sum(i,j), \\ MaxQ(j+1,\ell-1) \end{array} \right\} \;\middle|\; i \le j \le n \right\} & \text{otherwise} \end{cases}$$

Solution: For any integers i and ℓ, let $MaxQ(i,\ell)$ denote the maximum quality of a partition of $A[i..n]$ into ℓ intervals. We need to compute $MaxQ(1,k)$. This function satisfies the following recurrences:

$$MaxQ(i,\ell) = \begin{cases} \infty & \text{if } \ell = 0 \\ 0 & \text{if } i > n \text{ and } \ell > 0 \\ \max\left\{ \min\left\{ \sum_{h=i}^{j} A[h],\ MaxQ(j+1,\ell-1) \right\} \;\middle|\; i \le j \le n \right\} & \text{otherwise} \end{cases}$$

In particular, $MaxQ(i,0) = \infty$ because the minimum of the empty set (in this case, the set containing zero interval sums) is ∞, and $MaxQ(n+1,\ell) = 0$ when $\ell > 0$ because the only legal partition of the empty sequence consists of empty intervals, each of which sums to 0.

We can memoize $MaxQ$ into an array $MaxQ[1..n+1, 0..k]$, which we can fill row-by-row bottom up in the outer loop, filling each row in arbitrary order in the inner loop. We can compute each entry $MaxQ[i,\ell]$ in $O(n)$ time by implicitly memoizing the summation $\sum_{h=i}^{j} A[h]$ as follows:

```
MaxQuality(A[1..n, k]):
  MaxQ[n+1, 0] ← ∞
  for ℓ ← 1 to k
      MaxQ[n+1, ℓ] ← 0
  for i ← n down to 1
      MaxQ[i, 0] ← ∞
      for ℓ ← 0 to k
          sum ← 0
          MaxQ[i, ℓ] ← −∞
          for j ← i to n
              sum ← sum + A[j]
              tryj ← min{sum, MaxQ[j+1, ℓ−1]}
              MaxQ[i, ℓ] ← max{MaxQ[i, ℓ], tryj}
  return MaxQ[1, k]
```

The overall algorithm runs in $\boldsymbol{O(n^2k)}$ ***time***. ■

Rubric: 10 points =
- \+ 8 for $O(n^3k)$-time dynamic programming algorithm (standard DP rubric, scaled)
- \+ 2 for $O(n^2k)$-time algorithm, either by memoizing *Sum* (either directly in $O(n^2)$ time or via prefix sums in $O(n)$ time as above), or by computing $Sum(i,j)$ on the fly in the inner loop for *MaxQ*.

We don't care whether you allow or forbid empty intervals, as long as you're consistent.

CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 5

(a) Fix the alphabet $\Sigma = \{0, 1\}$. Describe and analyze an efficient algorithm for the following problem: Given a DFA M over Σ, does M reject *any* string? Equivalently, is $L(M) \neq \Sigma^*$?

Solution: We have $L(M) \neq \Sigma^*$ if and only if some non-accepting state is reachable from the start state s of M. We can find all states reachable from s in $O(V+E) = O(|Q|)$ time using whatever-first search in the graph of the DFA. ■

Rubric: 7 points; standard graph-reduction rubric. The graph is the DFA.

(b) Recall from Homework 10 that the corresponding problem for NFAs is NP-hard. But any NFA can be transformed into equivalent DFA using the incremental subset construction. So why doesn't your algorithm from part (a) imply that P=NP?

Solution: Because we haven't described how to transform an arbitrary NFA into an equivalent DFA in polynomial time.

In fact, it is ***impossible*** to transform an arbitrary NFA into an equivalent DFA in polynomial time! For any integer n, the language $(0+1)^*0(0+1)^{n-1}$—binary strings whose nth-to-last bit is 0—is accepted by the NFA with states $\{0, 1, 2, \ldots, n\}$, start state 0, the unique accepting state n, and the following transition function:

$$\delta(q, 0) = \begin{cases} \{0,1\} & \text{if } q = 0 \\ \{i+1\} & \text{if } 0 < q < n \\ \emptyset & \text{if } q = n \end{cases} \qquad \delta(q, 1) = \begin{cases} \{0\} & \text{if } q = 0 \\ \{i+1\} & \text{if } 0 < q < n \\ \emptyset & \text{if } q = n \end{cases}$$

On the other hand, let $F = (0+1)^n$ be the set of all n-bit strings. Consider two arbitrary distinct strings $x, y \in F$; there must be some index i such that $x_i \neq y_i$. Without loss of generality, assume $x_i = 0$ and $y_i = 1$. Then the suffix $z = 1^{i-1}$ distinguishes x and y, because $xz \in L$ but $yz \notin L$. Thus, F is a fooling set for L, which implies that any DFA for L has at least $|F| = 2^n$ states. ■

Rubric: 3 points. Yes, the black text is enough for full credit.

CS/ECE 374 A ✧ Spring 2018
Final Exam ♫ Problem 6

A ***number maze*** is an $n \times n$ grid of positive integers. A token starts in the upper left corner; your goal is to move the token to the lower-right corner. On each turn, you are allowed to move the token up, down, left, or right; the distance you may move the token is determined by the number on its current square. For example, if the token is on a square labeled 3, then you may move the token three steps up, three steps down, three steps left, or three steps right. However, you are never allowed to move the token off the edge of the board.

Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution.

Solution: Let $M[1..n, 1..n]$ be the input maze. We construct a directed graph $G = (V, E)$ whose vertices correspond to grid cells and whose edges correspond to legal moves. More formally, we define

$$V := \{(i, j) \mid 1 \le i, j \le n\}$$

and

$$E := \{(i, j) \to (i, k) \mid |j - k| = M[i, j]\} \cup \{(i, j) \to (k, j) \mid |i - k| = M[i, j]\}.$$

Any sequence of legal moves in M corresponds to a directed path in G. Thus, we need to find the shortest path from $(1, 1)$ to (n, n) in G. We can compute this shortest path using breadth-first search in $O(V + E) = \boldsymbol{O(n^2)}$ ***time***. ■

Rubric: 10 points, standard graph reduction rubric