EECS 376 Midterm **SOLUTIONS**

Multiple Choice (5 points each)

1. Consider the following algorithm:

> **function** FUNC($A[1, 2, \ldots, n]$)
>   **if** $n = 1$ **then**
>   **return** $A[1]$
> $x \leftarrow \text{Func}(A[1, 2, \ldots, \frac{2n}{3}])$
> $y \leftarrow \text{Func}(A[\frac{n}{3} + 1, \frac{n}{3} + 2, \ldots, n])$
> $z \leftarrow \text{Helper}(A[1, 2, \ldots, n])$
>   **return** $\min(x, y, z)$

Suppose that Helper($A[1, 2, \ldots, n]$) takes $O(n^2)$ time. Which of the following is the tightest bound on the runtime complexity of Func($A[1, 2, \ldots, n]$)?

- ○ $O(n)$
- ○ $O(n^{\log_{2/3} 2})$
- ● $O(n^2)$
- ○ $O(n^2 \log n)$

> **Solution:** $O(n^2)$.
> The recurrence for the runtime of this algorithm is $T(n) = 2T(\frac{2}{3}n) + O(n^2) = 2T(\frac{n}{3/2}) + O(n^2)$. We have $k = 2, b = 3/2, d = 2$, so $k/b^d = 2/(3/2)^2 = 2/(9/4) = 8/9 < 1$. From the Master theorem, we see that $T(n) = O(n^d) = O(n^2)$.

2. Consider the following algorithm:

> **function** ISSUMOFSQUARES(k (a positive integer))
>   **for** $a = 1, 2, \ldots, k$ **do**
>   **for** $b = 1, 2, \ldots, k$ **do**
>   **if** $a^2 + b^2 = k$ **then**
>   **return** true
>   **return** false

This algorithm runs in polynomial time with respect to the size of the input k.

- ○ True
- ● **False**

> **Solution: False**.
> The size of the input is $n = |k| = O(\log k)$, while the algorithm runs in time $k^2 = O(2^{2n})$, which is exponential in $n = |k|$.

3. Suppose Alg is a bottom-up dynamic-programming algorithm that works on a one-dimensional table of size n when given an input of size n. Then Alg (○ always / ● **sometimes** / ○ never) has a runtime complexity of $O(n)$.

> **Solution: Sometimes.**
>
> Consider the following problem: Suppose a message containing letters from A-Z is being encoded to numbers using the following mapping: 'A' $\rightarrow$ 1, 'B' $\rightarrow$ 2,..., 'Z' $\rightarrow$ 26. Given a non-empty string containing only digits, determine the total number of ways to decode it. Given a string of size n, there is a dynamic programming algorithm using $O(n)$ time.
>
> An example of such an algorithm that takes more than linear time is the bottom-up implementation for longest increasing subsequence (LIS), which takes $O(n^2)$ time.

4. ~~Suppose a country is considering a set of coin denominations with values \$1, \$5, and \$k. Then for (○ all / ● some / ○ no) values of $k \leq 10$, the greedy strategy for making change for $n \geq 1$ dollars always results in the minimal number of coins.~~

5. Which one of the following sets is uncountable?
   - ○ The set of all recognizable languages
   - ○ The set of all finite languages
   - ● **The set of all irrational numbers**
   - ○ The set Σ^* where Σ is the set of ASCII characters
   - ○ None of the sets are uncountable

> **Solution:** The set of all irrational numbers.
>
> Since the set of rationals $\mathbb{Q}$ is countable, but the set of reals $\mathbb{R}$ is uncountable, their difference $\mathbb{R} \setminus \mathbb{Q}$ must be uncountable. The union of two countable sets is always countable, so $\mathbb{R}$, the union of the rationals (which are countable) and the irrationals, can only be uncountable if the irrationals are uncountable.
>
> Since each program recognizes exactly one language, there cannot be more recognizable languages than programs. Since the set of programs is countable, so is the set of recognizable languages.
>
> The finite languages are a subset of the decidable (and therefore the recognizable) languages, so they must also be countable.
>
> We demonstrated that Σ^* is countable for any alphabet Σ – list the elements of Σ^* in lexicographic order.

6. ~~Which one of the following languages is decidable?~~
   - ● $L_{n\text{-}\mathbf{HALT}} = \left\{ \begin{matrix} \langle M, n \rangle : & n \in \mathbb{N} \textbf{ and } M \textbf{ accepts some string of} \\ & \textbf{length } n \textbf{ in fewer than } n \textbf{ steps} \end{matrix} \right\}$
   - ○ ~~$L_{\text{LOOPS}} = \{\langle M \rangle : M$ loops on the string "LOOP"$\}$~~
   - ○ ~~$L_{\text{NEQ}} = \{(\langle M_1 \rangle, \langle M_2 \rangle) : L(M_1) \neq L(M_2)\}$~~
   - ○ $L_{\text{NOT-SMALL}} = \{\langle M \rangle : M$ rejects all strings of length less than 376$\}$
   - ○ None of the languages are decidable

7. Let L_1 be an undecidable language. Then L_1 (● **always** / ○ sometimes / ○ never) has some strict subset $L_2 \subsetneq L_1$ that is also undecidable.

Solution: Always.

We can see this via contradiction. Suppose every strict subset of L_1 is decidable. Pick an arbitrary such subset L_2. Then L_2 and $L_3 = L_1 \setminus L_2$ are both strict subsets of L_1 and are both decidable. On the other hand, the decidable languages are closed under union, and we have $L_1 = L_2 \cup L_3$, so L_1 is decidable. This contradicts the fact that L_1 is undecidable.

Alternatively, we can use a counting argument to demonstrate this. An undecidable language must be infinite (otherwise a program can hardcode all the elements), and the power set of a countably infinite set is uncountably infinite. Thus, there are uncountably many subsets of L_1 (which doesn't change if we exclude L_1 itself), and since there are only countably many deciders, most of those subsets are undecidable.

8. (● **All** / ○ Some / ○ No) languages that can be decided by a DFA can also be decided by a C++ program.

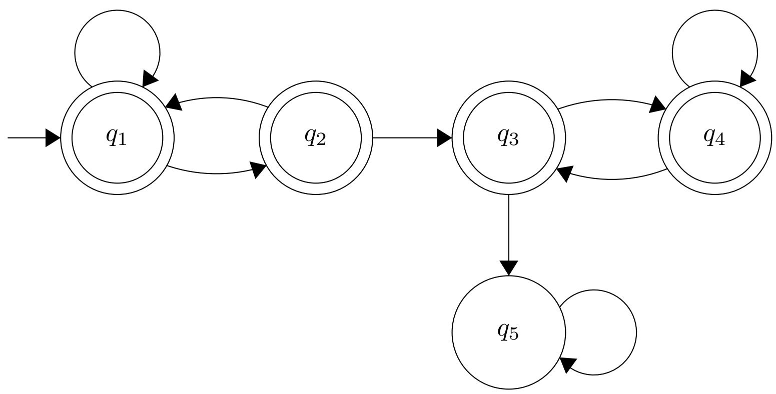
Solution: All.

C++ is Turing complete, so that any language decidable on a Turing machine is also decidable in C++. Since Turing machines are a strictly more powerful model than DFAs, all languages decidable by a DFA are decidable by a Turing machine and therefore by C++.

Written Answer (15 points each)

9. (a) Let $L_1 \subseteq \{0,1\}^*$ be the set of all binary strings that contain at most one occurrence of the substring "11". For example, 001010110 and 1100101 are both strings in L_1 but 0111001 is not, as the substring "11" occurs at both positions 1-2 and positions 2-3.

Fill in the transitions of the following DFA over the alphabet $\{0,1\}$ so that the DFA decides the language L_1.

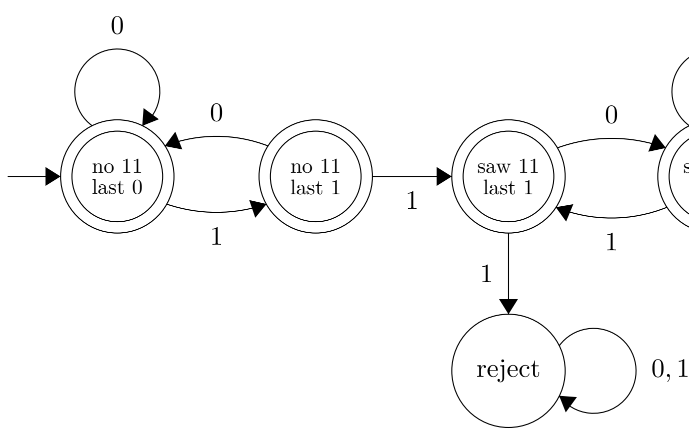


Solution: There are two pieces of information we need to keep track of in our states:

- Have we seen the substring "11" already?
- What was the last symbol we saw? If it was a "1", then seeing another "1" means we've encountered the substring "11". If it was a "0", then seeing a "1" does not mean we've encountered "11" yet.

Thus, we need four states to keep track of every combination. In addition, we need a "disposal" state that results in a rejection when we see a second "11".

The labeled states and transitions are as follows:



(b) Let $L_2 \subseteq \{a,b,c\}^*$ be the set of all strings over the alphabet $\{a,b,c\}$ **except** those that contain both at least one b and at least one c. For example, aa, aba, cca are all in L_2, but abc is not as it contains both a b and a c.

Write a DFA over the alphabet $\{a,b,c\}$ that decides the language L_2.

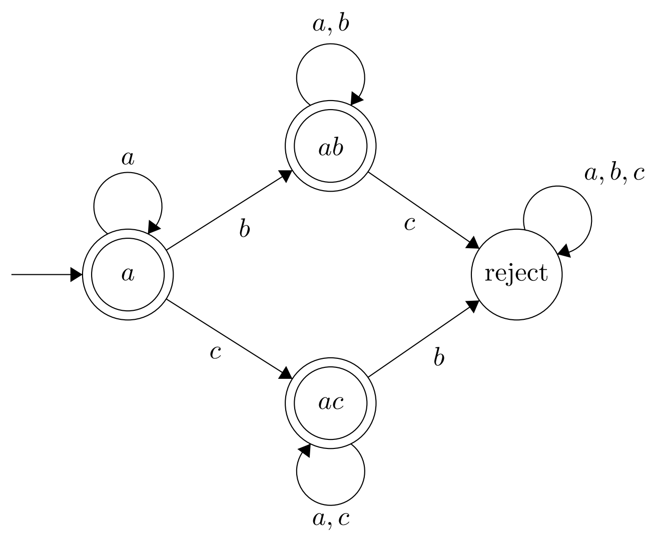
Hint: The DFA needs at most four states to decide L_2.

Solution: We need the following states:

- No b's or c's have been encountered yet. Encountering a b or c moves to the following two states, respectively.
- A b has been encountered. We stay in this state as long as we only see a's and b's. As soon as we see a c, we move to a disposal state that causes a rejection.
- A c has been encountered. We stay in this state as long as we only see a's and c's. As soon as we see a b, we move to a disposal state that causes a rejection.
- The disposal state.

The first three states are all accepting states – as long as we're in one of those states, we have not seen both a b and c in the input.

The full machine is as follows:



10. Suppose input is a function that returns a user-specified positive integer. For each of the following programs, determine if the program halts for all possible valid inputs x.

    Either provide a proof of termination for all possible valid inputs x, or provide a specific input that causes the program to loop along with a brief explanation for why it loops on that input.

    Hint: Consider how the value of x changes after two iterations of the loop.

(a)

```
x ← input()
while x > 10 do
    if x is odd then
        x ← x + 3
    else
        x ← x/2
```

Solution: The algorithm terminates on all inputs. It terminates when the value of $x \leq 10$. For $x > 10$, the value of x strictly decreases every two iterations. Let x_i be the value of x after i iterations.

- **Case 1:** x_i is odd. The algorithm increments the value by 3, which makes $x_{i+1} = x_i + 3$ even. In the subsequent iteration, the algorithm divides the value by 2, so $x_{i+2} = \frac{x_i+3}{2}$. The difference between x_i and x_{i+2} is

$$\begin{aligned} x_i - x_{i+2} &= x_i - \frac{x_i + 3}{2} \\ &= \frac{1}{2}(2x_i - x_i - 3) \\ &= \frac{1}{2}(x_i - 3) \end{aligned}$$

  This difference is at least 1 when:

$$\begin{aligned} \frac{1}{2}(x_i - 3) &\geq 1 \\ x_i - 3 &\geq 2 \\ x_i &\geq 5 \end{aligned}$$

- **Case 2:** x_i is a multiple of 4. Then x_i is even, so $x_{i+1} = x_i/2$. This is also even, so $x_{i+2} = x_i/4$, which is strictly less than x_i.

- **Case 3:** x_i is a multiple of 2 but not a multiple of 4. Then x_i is even, so $x_{i+1} = x_i/2$, which is odd since x_i is not a multiple of 4. Thus, $x_{i+2} = x_i/2 + 3$. The difference between x_i and x_{i+2} is

$$\begin{aligned} x_i - x_{i+2} &= x_i - (x_i/2 + 3) \\ &= x_i/2 - 3 \end{aligned}$$

  This difference is at least 1 when:

$$\begin{aligned} x_i/2 - 3 &\geq 1 \\ x_i/2 &\geq 4 \\ x_i &\geq 8 \end{aligned}$$

In all cases, the value of x decreases by at least one every two iterations when $x > 10$. Since x has a lower bound of 10, this demonstrates by a potential argument that the algorithm always terminates.

Comments: The reasoning above can be shoehorned into the standard framework of a potential function by defining $s_i = x_{2i}$, i.e. the ith value of the potential is the $2i$th value of x. This is analogous to loop unrolling (`https://en.wikipedia.org/wiki/Loop_unrolling`), a standard optimization technique in compilers.

It is also a valid solution to combine cases 2 and 3 above and just show that x_i decreases after one iteration when it is even.

(b)

```
x ← input()
while x > 10 do
    if x is odd then
        x ← (x − 1)/2
    else
        x ← x + 2
```

Solution: The algorithm does not terminate for even $x > 10$. In each iteration, x is incremented by 2, which keeps it even and ensures that it will never reach the termination condition of $x \leq 10$.

11. ~~Consider the following language.~~

$$L_{\text{ALL-REJECT}} = \{\langle M \rangle : M \text{ is a Turing Machine and } M \text{ rejects all inputs}\}$$

~~Show that $L_{\text{ACC}} \leq_T L_{\text{ALL-REJECT}}$ or show that $L_{\text{HALT}} \leq_T L_{\text{ALL-REJECT}}$. (Do whichever one of the two you would prefer.)~~

12. You are organizing a trip for k students to attend the Rose Bowl, and you are looking to rent buses to take the students there. The bus company has n buses available, where bus i has $S(i)$ seats but costs $C(i)$ to rent. Your goal is to minimize the total cost to rent buses for the k students. (Each bus can only be used at most once.)

Let $MB(i, j)$ be the minimum cost to rent buses for j students, allowing only buses $1, 2, \ldots, i$ to be rented. (Define $MB(i, j) = \infty$ for the cases where buses $1, 2, \ldots, i$ cannot accommodate j students.)

(a) Provide a recurrence for $MB(i, j)$ (including base case(s)). Briefly justify your answer.

Solution: The base cases are when we have only a single bus, i.e. $i = 1$. If there are no students (or a negative number), we don't need the bus at all. If there are j students and they all fit on that bus, we can successfully take them on that single bus with cost $C(1)$. Otherwise there is no solution. Thus,

$$MB(1, j) = \begin{cases} 0 & \text{if } j \leq 0 \\ C(1) & \text{if } 1 \leq j \leq S(1) \\ \infty & \text{if } j > S(1) \end{cases}$$

Alternatively, we can define the base cases to be when there are no buses, in which case it is only viable when there are no students (or a negative number). This gives us:

$$MB(0, j) = \begin{cases} 0 & \text{if } j \leq 0 \\ \infty & \text{if } j > 0 \end{cases}$$

Now we consider general i. For a bus i, our choices are to either rent it or not. If we rent it, we have $S(i)$ fewer students that need seats out of the remaining buses, but at an additional cost of $C(i)$. Otherwise, we need to accommodate all j students on the remaining buses. Thus,

$$MB(i, j) = \min\{C(i) + MB(i - 1, j - S(i)), MB(i - 1, j)\}$$

for $i > 1$.