CSC236 Week 12

Larry Zhang

Test 2 Recap

- Class average: 17.7/28 (63%). Median: 18/28. Highest mark 28/28
- Q1: 76.5%, Q2: 64.1%, Q3: 35%.
- Solutions posted on course web page
- Remarking request form posted on course web page
    - attach it to your test, and submit by December 5.
- **Make sure MarkUs has your mark correctly recorded.**
- Understand the mistakes that you made, practice so that you don't make similar mistakes again.

This week

- Final tutorial exercise
- PS9 due on Friday

Today

- review the course content
- exam prep tips

Review
what we’ve learned in CSC236

From Week 1 Slides...

These are the **fundamentals** of any interesting stuff you might want to do with computers.

- **Induction (simple, complete, structural)**
  - The technique for “**reasoning**”
- **Asymptotic Notations (Big Oh/Omega/Theta)**
  - Describe program runtimes.
- **Recursion**
  - Recursion in CSC148 was magic; now you can explain the magic.
  - We will learn how to analyse and design recursive programs.
- **Program correctness, invariant, variant**
  - Formal way to prove a (iterative/recursive) program is correct
- **Regular expressions and finite automata**
  - Models for a computer’s understanding of **languages**.

The difference between

an average programmer who can code stuff

and

a university-trained **computer scientist** who can understand and design elegant and efficient programs

You CANNOT call yourself a "computer science person" before taking CSC236.

Induction

- Simple induction, complete induction, structural induction.
- What are the differences between them?
- How to choose which induction to use for a proof?
  - It is possible that a statement can be proven using different types of induction.
- Structure of induction proof:
  - predicate
  - base case
  - induction step

Asymptotic Notations

- They describe the rate of growth of functions.
- What are the definitions of big-Oh, big-Omega, big-Theta
- How to prove if a function of n is in big-Oh, big-Omega, big-Theta of some other function of n.
  - e.g., n^2 - 100n + 5 in Ω(n logn)
- Key: find n_0 and c.

Recursion

- Develop a recurrence
  - e.g., from lecture: 2-element subsets
- Develop the recurrence for a recursive algorithm’s worst-case runtime
  - worst-case: if there are multiple possible paths, pick the worst path (with the longest runtime)
- Use repeated substitution to find the closed form of a recursively defined function.

Recursive algorithms

- Develop divide-and-conquer algorithms
- Use the master theorem to find an asymptotic bound on the algorithm’s runtime.
- When does or doesn’t the master theorem apply?

Correctness of Recursive Programs

- Stating preconditions and postconditions
- Finding program paths
- Prove the correctness of the program
  - For paths with recursive calls, check
    - if precondition of the recursive call is satisfied
    - if recursive call is on a smaller input.
    - check postcondition of the original function call, assuming the recursive call satisfies the postcondition

Correctness of Iterative Programs

- State preconditions and postconditions
- State the loop invariant
- Prove loop correctness using the invariant
  - Base case
  - Induction step
- Prove loop termination using a variant
  - always decreasing
  - >= 0

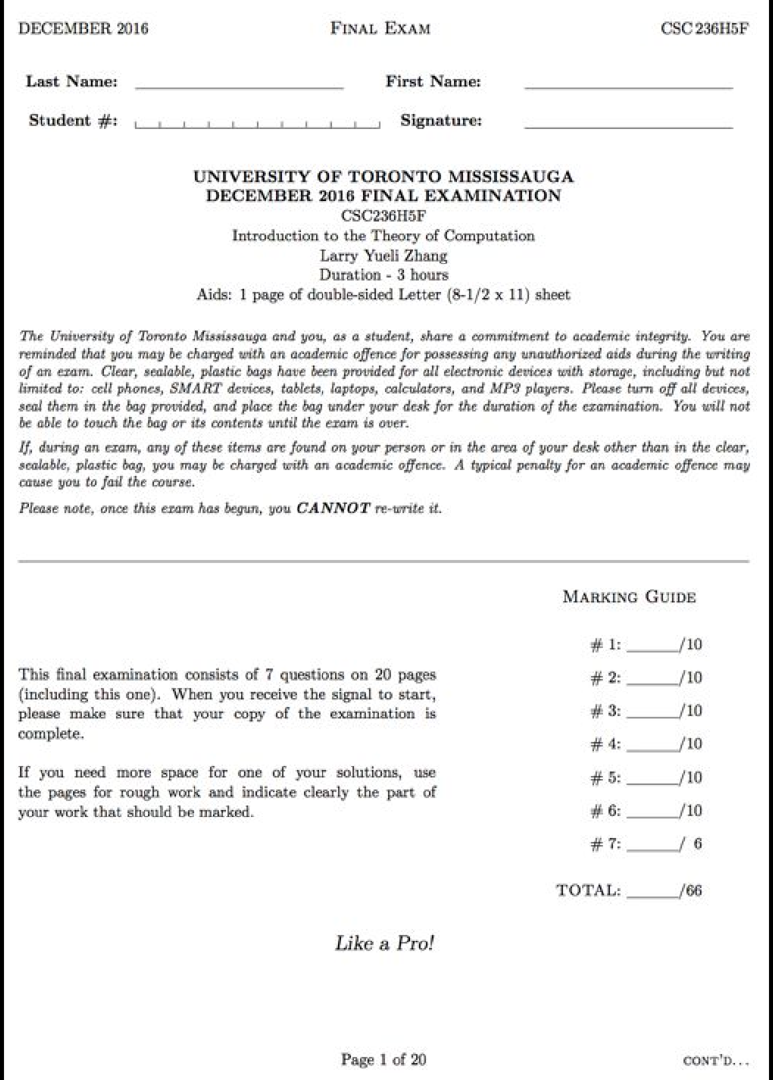
Regular Expressions

- Terminologies and their definitions
  - alphabet, language, regular language, operations on languages
    - union, concatenation, Kleene star
- Given a regular language, write a regular expression that matches it
- Given a regular expression, describe the language being matched.
- Tell why a regular expression is incorrect.
  - A string is in the language but not matched by the regex
  - A string is matched by the regex but is not in the language

DFA and NFA

- Design a DFA that recognizes a given language (with minimum number of states)
- Given a DFA, tell what language is matched
- Write state invariants for a DFA and prove DFA correctness
- Prove the minimum number of DFA states for a language
- Prove that a language is not regular
- Design an NFA to recognize the given language
- Given an NFA, tell what language is matched
- Convert an NFA into a DFA using subset construction.

Final Exam Questions



Last Name: ______________ **First Name:** ______________

Student #: ______________ **Signature:** ______________

UNIVERSITY OF TORONTO MISSISSAUGA
DECEMBER 2016 FINAL EXAMINATION
CSC236H5F
Introduction to the Theory of Computation
Larry Yueli Zhang
Duration - 3 hours
Aids: 1 page of double-sided Letter (8-1/2 x 11) sheet

This final examination consists of 7 questions on 20 pages (including this one). When you receive the signal to start, please make sure that your copy of the examination is complete.

If you need more space for one of your solutions, use the pages for rough work and indicate clearly the part of your work that should be marked.

Marking Guide

# 1:	______/10
# 2:	______/10
# 3:	______/10
# 4:	______/10
# 5:	______/10
# 6:	______/10
# 7:	______/ 6
TOTAL:	______/66

Like a Pro!

Questions

- 7 questions
  - 2 questions on Test 1 topics (induction, recurrence)
  - 2 questions on Test 2 topics (divide-conquer, master theorem, program correctness)
  - 3 questions on other topics (regex, DFA, NFA)
- Aid allowed: one double-sided 8.5”x11” sheet
- You need to get at least 40% on the final exam to pass the course.

Question types

- For questions on Test 1 and Test 2 topics, they are very similar to the question types of Test 1 and Test 2
- For regex, DFA, NFA
    - Given language, write regex
    - Given regex, describe language
    - Argue about regex being incorrect, finding counterexamples
    - Given language, design a DFA
    - Prove correctness of a DFA using state invariants
    - Prove about minimum number of states in DFA
    - Given a language, design an NFA
    - Given an NFA, convert to DFA using subset construction

Example question

Language: L = {w in {0,1}* | the number of 0’s in w is a multiple of 3}

Regex: (1*01*01*01*)*

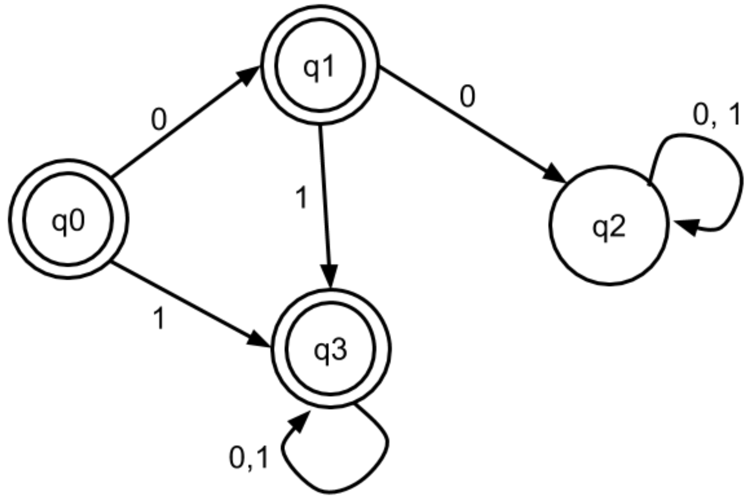
It is correct?

It’s wrong, because 11111 is in the language but is not matched by the regex.

Another example

Design a DFA that recognizes the following language

L = {w in {0, 1}* | w does NOT start with 00 }

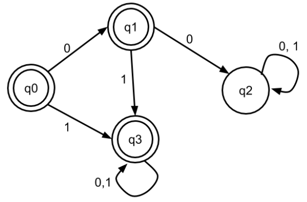


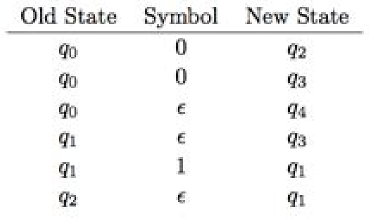
State Invariants:

- Each string w satisfies **one and only one** state's invariant
- q0: len(w) = 0
- q1: len(w) = 1 and starts with 0
- q2: w starts with 00
- q3: len(w) = 1 and is 1, or len(w) >= 2 and does not start with 00.

Some tips:

- When design DFA, q0 is the default initial state, unless specified otherwise
- Don’t forget to double-circle the accepting states.
- For subset construction, must indicate clearly the initial state and accepting states.
- You may be asked to
  - draw the state transition **diagram (circles and arrows)**, or
  - write the state transition **table (old state, symbol, new state)**
  - READ THE QUESTION CAREFULLY!





Old State	Symbol	New State
q_0	0	q_2
q_0	0	q_3
q_0	ϵ	q_4
q_1	ϵ	q_3
q_1	1	q_1
q_2	ϵ	q_1

Write program paths

Write the sequence of line numbers, but don’t go into helper functions.

For example, write 2 - 4 - 5 - 6 - 7 - 8

Do NOT write: 2 - 4 - 5 - 6 - 7 - 11 - 12 - 13 - 14 - …..

```
1   def max_seg_sum(A, low, high):
2     if low == high:
3        return max(A[low], 0)
4     mid = (low + high) // 2
5     left_sum = max_seg_sum(A, low, mid)
6     right_sum = max_seg_sum(A, mid + 1, high)
7     cross_sum = max_crossing(A, low, mid, high)
8     return max(left_sum, right_sum, cross_sum)
9
10  def max_crossing(A, low, mid, high):
11    left_sum = 0
12    s = 0
13    for i in range(mid, low - 1, -1):
14       s = s + A[i]
15       if s > left_sum:
16          left_sum = s
17    right_sum = 0
18    s = 0
19    for i in range(mid + 1, high + 1):
20      s = s + a[i]
21      if s > right_sum:
22         right_sum = s
23   return left_sum + right_sum
```

What to put on the aid sheet

- Recipes
  - induction proofs, repeated substitution, correctness proof for recursive program, proof of invariant, proof for correctness of DFA, proof for minimum number of states in DFA, subset construction
- Definitions
  - all definition that you may need for a proof.
- Math facts that you are supposed to know
  - sum of arithmetic and geometric series
  - log rules, like log(ab) = log(a) + log(b), log_b a = log(a)/log(b)
  - etc…
- anything else

Practice Problems

- Lecture examples
- Problem sets
- Tutorial exercises
- Our tests
- Past tests (posted on the course website)
- Past final exams (old exam repository at UofT library)
- Exercises in Course Notes and Dan’s invariant book

Get Help

- Ask on Piazza
- Pre-exam office hours
  - December 6, 7, 8
  - 2-5pm
  - Use them!

Go to the exam, right time, right place

When: Friday, December 9th, 9:00am~12:00pm

Where: Gym A/B

Duration: 3 hours

Bring your student card.

https://www.utm.utoronto.ca/registrar/current-students/examinations

See you in office hours!