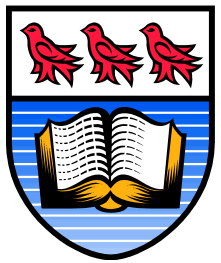
Algorithms & Data Structures I
CSC 225

Ali Mashreghi

Fall 2018



Department of Computer Science, University of Victoria

Final review

- Final is on **Wednesday December 12** at **2pm** (check the final exam table for location)

- **You can bring 2 pieces of papers (front and back)** of cheat sheets. You cheat sheets can be printed as well.

- Calculator is not allowed and not needed.
- Don’t bring scrap papers; they’ll be provided.
- It’s worth 30%

Final review

- The exam is **about 2 hours** long.

- There are **40-50 multiple choice** questions.
- There are **3-5** written questions.

Final review

- It covers the following:

1. All lectures
2. All lab materials
3. 4 written assignments
4. 8 programming assignments

✓ Read the textbook to understand the material on the lectures.
✓ There will be no questions that *only* appear in the textbook, unless something similar has been discussed in lectures, labs or assignments

Mathematical background

- You must know the summations and log properties.

- Learn the proof techniques that we have discussed:

1. Proof by contradiction
2. Proof by contrapositive
3. Proof by induction

✓ **There will be** at least one question from **proof by induction**. So, learn everything that relates to induction such as substitution method, induction on graphs, using induction to prove a closed form formula, dynamic programming.

Time complexity and asymptotic notations

- The definitions of asymptotic notations and proofs about asymptotic notations.

- There will be several questions like this:

- What is the time complexity of this for loop in terms of n?

$x = 1$

for $i = 0$ **to** $n/5$

$x = x+1$

1. $\Theta(1)$ 2. $\boldsymbol{\Theta(n)}$ 3. $\Theta(n^2)$ 4. $\Theta(n \log n)$

Time complexity and asymptotic notations

Algorithm Loop4(n):

$s \leftarrow 0$
for $i \leftarrow 1$ **to** $2n$ **do**
  for $j \leftarrow 1$ **to** i **do**
    $s \leftarrow s + i$

Algorithm Loop5(n):

$s \leftarrow 0$
for $i \leftarrow 1$ **to** n^2 **do**
  for $j \leftarrow 1$ **to** i **do**
    $s \leftarrow s + i$

Algorithm Loop1(n):

$s \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
  $s \leftarrow s + i$

Algorithm Loop2(n):

$p \leftarrow 1$
for $i \leftarrow 1$ **to** $2n$ **do**
  $p \leftarrow p \cdot i$

Algorithm Loop3(n):

$p \leftarrow 1$
for $i \leftarrow 1$ **to** n^2 **do**
  $p \leftarrow p \cdot i$

Time complexity and asymptotic notations

- Given two functions like $f(n) = n^2$ and $g(\mathrm{n}) = 5n^2 + 10n$ you should be able to prove that $f(n) = O(g(n))$

- Also, knowing about little-oh and little-omega is necessary.

Solving recurrences

- Know the 3 methods for solving recurrences.

- There is a PDF file on Lab 3 on Connex with more than 20 solved examples on Master theorem.

Algorithms and pseudocodes

- Learn every single algorithm in detail along with how you can analyze it.

- For sorting algorithms learn the properties of all sorting algorithms and know when one is preferred over another.

- Know the difference between different kinds of analyses: worst-case, best-case, average-case, etc.

Algorithms and pseudocodes

Binary-Search(*A, x*)
low = 1 // beginning of search range
high = *n*+1 // end of search range
while *low* < *high*
    mid = $\left\lfloor \frac{(low+high)}{2} \right\rfloor$
    if *x* == *A*[*mid*]
        (1)
    elseif *x* > *A*[*mid*]
        (2)
    else
        (3)
return -1

One type of question that you might be asked is something like this.

Given a pseudocode from the slides, complete the blanks (1)-(3) so the algorithm works correctly.

Algorithms and pseudocodes

```
BINARY-SEARCH(A, x)
1    low = 1 // beginning of search range
2    high = n+1 // end of search range
3    while low < high
4        mid = ⌊(low+high)/2⌋
5        if x == A[mid]
6            return mid
7        elseif x > A[mid]
8            low = mid + 1
9        else
10           high = mid
11   return -1
```

Or given the pseudocode for an algorithm, what is the time complexity of the algorithm.

Linear sorting

- **There will be** questions about the linear sorting algorithms.

- Learn the three linear sorting algorithms, their pseudocode and their complexity:

1. Basic counting sort and useful counting sort
2. Bucket sort
3. Radix sort

Trees

- Learn all definitions and special types of trees.

- For example, given an example tree, and some specified node; you should what are the descendants of that node or what is the level number of that node.

- Also, learn the **relation between the number nodes and height** in a complete or nearly complete binary tree.

Data structures

- We talked about a number of data structures such as **heaps**, **binary search trees**, **AVL trees**, **hash tables**, **stacks**, **queues**, **linked lists**, **arrays**.

- Know the running time of different operations such as insert, delete, search, successor, … on each of these data structures.

- There **will be no questions** about amortized time complexity.

Probability and expectation

- Just very simple questions to the extent of what you learned in the lectures.

Quicksort and order statistics

- Learn the quicksort algorithms in details.
- Also, learn about how we can partition the array (basic partition, randomized partition, and paranoid partition)
- The discussion about **quicksort**, **partitioning** and **the ith smallest element** in an array are similar. Learn them details.
- Learn both algorithms for finding the **the ith order statistic:** Randomized select and worst-case linear select.

Lower bounds on sorting and searching

- Just know about the lower bounds to extent that we discussed.

- I won't ask you to prove a new lower bound.

- Learn the binary search algorithm along with the examples that it can be used for.

Hashing

- Learn about different ways of making a hash function.
- Know what is a collision and given a simple hash function you should be to find a collision pair, i.e. two keys that have the same hash value.

- Learn about different strategies for resolving collisions and their analysis:

1. Using chaining
2. Using open addressing

- Given a set of keys you should be able to insert them into a hash table.

Binary search trees

- Know about the binary search tree property.
- Know about how we can do different operations such as insert, search, delete, and successor on a BST.
- Know about the walks: in-order, pre-order, post-order
- Learn the time complexity of all operations on a binary search tree.

AVL trees

- Know about the definition of balance factor.
- Know about the AVL tree property.
- Know how we can fix a violation in an AVL tree.
- **Definitely, learn rotations!**
- Learn the complexity of operations on an AVL tree.

Graphs

- **There are lot of graph questions!**

- Learn all graph terminologies and definitions such as degree, path, trail, walk, connected, disconnected, and so on.

- Know the two graph exploration algorithms, i.e. BFS and DFS, their implementation and complexity in details.

- Know for what kind of problems BFS is used and for what kind of problems DFS is used.

Graphs

- Learn the problems that we discussed on graphs:

- Shortest path
- Strongly connected components
- Eulerian tour
- Topological sort
- Graph/tree diameter
- ….

Transitive closure

- Learn the 4 algorithms for solving the transitive closure problem.

- Learn the time complexity of each and the properties of each of the solutions.

Dynamic programming

- Know the approach and know how it’s different from divide-and-conquer.
- Know the difference between the top-down and bottom-up approach.
- Know how we can compute time complexity in this approach.

Dynamic programming

- **There will be one question** about dynamic programming, for example

1. Converting a top-down solution to a bottom-up for a simple problem.

2. Memo-izig a recursive algorithm.

3. Analyzing a dynamic programming algorithm.