Algorithms Qualifying Exam Information  
Computer Science & Software Engineering Department  
Auburn University  
May 2017

Note to students: This document contains several sample questions (but this is not a sample exam) that provide an indication of the skills that we expect you to demonstrate in this topic area. See the end of the document for a list of these skills, topics covered by this qualifying examination and suggested reading. In general, you should expect questions that address the listed skills in any of the listed topic areas. These topics are those covered by typical junior/senior/first-year-graduate courses on algorithms. What this means is that a specific topic may or may not be covered by the undergraduate/graduate courses you took or by COMP 3270 or 7270. You are still responsible for knowing this material. To become familiar with these topics you may read relevant chapters from the suggested book or any other books on algorithm design and analysis.

Sample Questions

True or False? it is possible to have an array-based implementation of a queue in which both enqueue and dequeue have constant running time.

Consider the problem of finding the kth smallest number in an array of n numbers, 1<=k<=n. True or False? A modification of the Quicksort algorithm can solve this problem in less than quadratic time.

Suppose you are writing an application to support an online music store. The user functions that you must make extremely efficient are (i) the ability to search for songs by a given artist and (ii) the ability to display all the songs available for purchase in alphabetic order of artist. Songs will be removed from the store very rarely but there is a high volume of new songs added each day. Which of the following data structure is best suited for this problem?
(A) Array-based lists (B) linked lists (C) balanced binary search trees (D) hash tables (E) None of these are suitable

Which of the following best reflects the technical meaning of the statement “an algorithm is correct?”
(A) The algorithm produces the correct answer for every input or inputs.
(B) The algorithm produces the correct answer for every legal input or inputs.
(C) The algorithm always halts and produces an answer for every legal input or inputs.
(D) The algorithm halts and produces the correct answer for every legal input or inputs.
(E) None of the above reflects the technical meaning of the statement “an algorithm is correct.”

What is its order of complexity of the algorithm below?

Algorithm F(n: integer)
if n ≤ 1 then return 1
else
    f = 1; i = 1
while i < n
    f = f * i
    i = i + 1
return f

(A) O(logn)    (B) O(n)    (C) O(nlogn)    (D) O(n^2)    (E) None of these

What does the algorithm F compute?
(A) Σ_{i=1}^{n} i
(B) n!
(C) 2^n
(D) n^2
(E) None of these

The recursive algorithm below computes an arithmetic operation on n. What is it?
Algorithm A(n: non-negative integer)
1      if n ≤ 1 then return 0
2      else return 1+A(n-2)
(A) ceiling(n/2)
(B) (n/2)
(C) floor (n/2)
(D) (n-2)
(E) None of these

What is the first recurrence relation, corresponding to the base cases, of this algorithm? Assume that a comparison, returning a value, addition and subtraction each requires exactly 1 unit of time.
(A) T(n)=4
(B) T(n)=4, n=1
(C) T(n)=4, n≤1
(D) T(n) ≤4, n≤1
(E) None of the above

What is the second recurrence relation, corresponding to the base cases, of this algorithm?
Assume that a comparison, returning a value, addition and subtraction each requires exactly 1 unit of time.
(A) T(n)=T(n-2)+7, n>1
(B) T(n)=T(n)+7, n>1
(C) T(n)=(n+2)+7, n>1
(D) T(n)=T(n/2)+7, n>1
(E) None of the above

What is the algorithm’s order of complexity?
(A) O(logn)    (B) O(n)    (C) O(nlogn)    (D) O(n^2)    (E) None of these

Suggested Book: Cormen, Leiserson, Rivest & Stein, Introduction to Algorithms, the textbook for COMP 3270 & 7270 – has chapters on almost all topics listed below.
Skills

- know fundamental data structures and operations on them
- understand algorithms
- be able to mentally simulate the operation of algorithms on input data
- debug algorithms
- modify an algorithm to solve a different problem
- analyze correctness and complexity of algorithms
- compare algorithms in terms of efficiency
- design algorithms

Topics

(May change slightly from year to year; the list below is valid for Academic Year 2017-18)

Data structures and associated operations

- arrays
- linked lists of various kinds
- stacks
- queues
- binary trees, balanced trees, binary search trees
- hash tables: linear probing, quadratic probing, double hashing
- priority queues: binary heaps
- disjoint sets: find, union, path compression
- graphs: adjacency matrix and list representations, graph algorithms such as shortest path

Recursive algorithms

- recursion trees
- divide and conquer
- developing recurrence relations
- solving recurrence relations: recursion tree method, substitution method, master method, backward/forward substitution method

Analyzing algorithms

- analyzing simple statements, conditionals, loops, nested loops
- expressing complexity as $T(n)$
- determining growth functions and orders of complexity: Big-Oh, Omega etc.

Correctness of algorithms: proofs by counterexample, contradiction, loop invariants and induction

Fundamental algorithms

- sorting: insertion sort, selection sort, bubble sort, merge sort, quick sort, heap sort, counting sort, radix sort, bucket sort
- searching: linear search, binary search
- graph problems: breadth-first search, depth-first search, minimum spanning tree, single source shortest paths, topological sort, strongly connected components

Designing algorithms

- designing recursive algorithms: divide and conquer, dynamic programming (DP)
- designing iterative algorithms: simple iteration, DP, greedy algorithms

Basic knowledge about NP-completeness: classes P, NP, NP-Complete and NP-Hard, common NP complete problems such as Traveling Salesman, Hamiltonian Circuit, Vertex Cover, etc.