CSE 2011A Fall 2019 – Exam fun
Instructor: Jeff Edmonds

Questions:

1. First Order Logic
2. ADT
3. Software Eng
4. Pointers
5. Loop Invariants
6. Running Time
7. BigOh Notation and Sums
8. Recursive
9. Balanced Binary Search Trees
10. Heaps
11. Hash Tables
12. Probabilistic Alg
13. Graphs
14. Linear Programming (Network Flows)
15. Greedy Alg
16. Dynamic Programming
17. NP

Time 3 hrs.
Keep your answers short and clear.
Do not repeat the question.
USE A PEN OR DARK PENCIL SO THE SCAN IS CLEAR
DON'T WRITE OUTSIDE SPECIFIED AREAS.
DON'T WRITE OUTSIDE SPECIFIED AREAS.
No question needs more than 4 or 5 sentences.

1. First Order Logic: Use Jeff's game to prove or disprove:
   $\exists y,\ \forall x,\ x \times y = 0$, ie "There exists a y such that for all x, $x \times y = 0$"
   What does this sentence say about mathematics? ie Meaning.
2. What are the Abstract Data Types we learned, what are their operations, how are they used, how are they stored, what are their loop invariant, what are their algorithms, what are their running times. ...
3. Software Eng:
   (a) Software Eng: $(7 = 1 + 1 + 1 + 4$ marks)

i. Stacks, Priority Queues, Dictionaries, and so on are all examples of what? Give its three word name.

ii. A *Stack* is an ordered list of objects in which the operations acting on it are restricted to *Pushing* a new object on its *top* end, *Popping* and returning the most recently pushed object from the top, and determining if the stack is empty.
What is this description an example of?

iii. What does this such a description specifically not specify?

iv. Give four software engineering principles that this facilitates. Explain.

(b) (3 marks) From the book we got the concept of a *position*.

i. What was position an example of?

ii. Abstractly, what did it represent?

iii. Then in the assignment we *instantiated* it. What data structures did it become?

4. Pointers: Write code to play with pointers to play with linked lists, trees, ...

5. Loop Invariant:

(a) What are three most important steps in proving a program works using loop invariant. (Not counting defining the LI, the exit cond, or anything to do with progress.) Write as "This and this gives you this and this".'

(b) What is a loop invariant and what does it mean to maintain it?

(c) What is the loop invariant for binary search? How do you maintain this loop invariant while making progress?

(d) Give the loop invariant and follow each of the three key steps in defining an iterative algorithm that on input M outputs $S = \sum_{i=1}^{M} i^2$. (Each of these steps is of the form "This and this gives you this and this" is not the exit cond, or anything to do with progress.)

6. Running Time: I give you code and you tell me the $\mathcal{O}(running\ time)$.

(a) Multiple loops

(b) logs

(c) Recurrence relations

(d) Complexity measured wrt size and size is the number of bits to write down the input.

(e) Searching for solution vs verifing solution.

7. BigOh Notation and Sums

(a) Give the exists forall statement defining BigOh

(b) Does $\Theta(1)$ include $sin(n^2) + 5$? Explain this class of functions.

(c) **Summation:** Approximate the sum $\Theta(\sum_{i=1}^{n} f(i))$. Which result did you use and how did you use it? What type of sum is it?

i. $f(n) = 5 \cdot n^{5.5} \log^{100}(n)$

ii. $f(n) = 5 \cdot 3^{4n} n^{100}$

iii. $f(n) = 5 + sin(n^2)$

8. Recursion:

(a) Give a recursive algorithm for summing up all the values in a binary tree.

(b) Describe a Merge Sort like algorithm whose running time is given by $T(n) = 3T(n/3) + \Theta(n)$.

(c) Solve this recurrence relation $T(n) = 3T(n/3) + \Theta(n)$. How does it compare with that of Merge Sort?

(d) When making a recursive call in any recursive program, what are the requirements on the instance that you give it?
When describing a recursive algorithm or proving that it works, would Jeff want you to trace out the tree of stack frames? Why or why not.
How does he say you should relate to your (recursive) friends.

9. Balanced Binary Search Trees: What are their operations, how are they used, how are they stored, what are their loop invariant, what are their algorithms, what are their running times, trace out an example.

10. Heaps: What are their operations, how are they used, how are they stored, what are their loop invarients, what are their algorithms, what are their running times, trace out an example.

11. Hash Tables: What are their operations, how are they used, how are they stored, what are their loop invariant, what are their algorithms, what are their running times, trace out an example.

12. Probabilistic Algorithms:
We say that problem P is deterministically computable in time 100 iff
$$\exists A\ \forall I\ A(I) = P(I) \text{ and } Time(A, I) \leq 100.$$
In the first order logic game, the prover first gives the algorithm A and then the adversary, knowing A, gives the worst case input I.

    (a) Explain how this is used with Hash Tables

    (b) As done above for deterministic algorithms, give the first order logic statement to state that there is probabilistic algorithm that always gives the correct answer and has expected time at most 100 per operation.
    Use your own words to explain how the associated game is different.

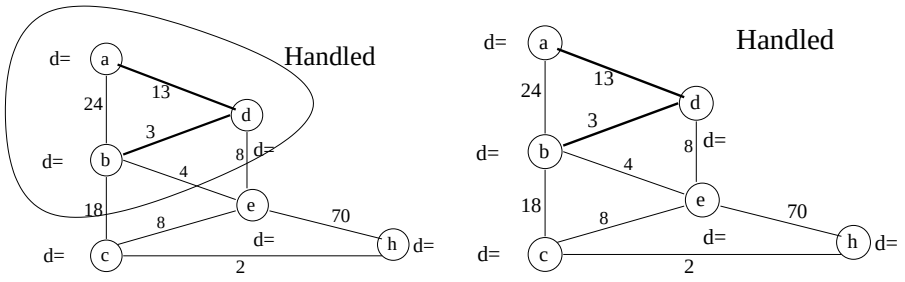
13. Graphs:

    (a) How to store it

    (b) Depth First Search (Stack) (Linear Order)

    (c) Breadth First Search (Queue)

    (d) Dijkstra's Algorithm (Priority Queue on priorities $d(v)$):
    Consider a computation of Dijkstra's algorithm on the following graph when the circled nodes have been handled. The start node is a. On the left, give the current values of d. (Be sure to think about what the loop invariant says about the values d.)



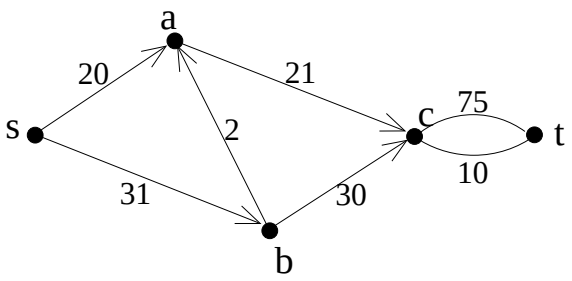
    On the right, change the figure to take one step in Dijkstra's algorithm. Include as well any πs that change.

    (e) Network Flow: Consider the following network with source s and terminal (or sink) t. The label on an edge is the capacity of the edge.

        i. Give flow along each edge in a maximum flow in this graph.

        ii. Give a minimum cut which can be used to argue that the flow cannot be more than stated.

Network



(f) Network Flow:

i. Describe the hill climbing algorithm to finding such an optimal flow.

ii. How can you prove to your boss that you can achieve a flow as good as you claim?

iii. How can you prove to your boss that it is impossible to achieve a better flow?

14. What is a Linear Program and how do you solve it.
What is Network Flows and how is it an example of Linear Programming?

15. Greedy Algorithim:

(a) What is a greedy alg?

(b) What is its loop invariant?

(c) What is its running time?

(d) Give a Greedy Algorithim for something.

16. Consider the Dynamic Programming Algorithm for finding shortest paths in a graph.

(a) If this an iterative algorithm, give it's loop invariant.
If this a recursive algorithm, what are the subinstances recursed on (given to my friends).

(b) Give one line to compute $Dist(u, v, \ell)$ giving the length of the shortest path from u to v with at most ℓ edges.

(c) What is its running time of the entire algorithm to find $Dist(u, v, n)$ for each pair $\langle u, v \rangle$?

17. NP

(a) How do you convince your boss that the computational problem he wants you to solve is likely too hard to solve in general?

(b) What does it mean for a problem P to be in NP (Non-Deterministic Polynomial Time)?

(c) Why is 3-Col in NP?

(d) What does it mean for a problem P to be NP-Complete.

18. What did Godel proof about proof systems?
Give few sentence intuition of the proof.