Data Structures and Analysis of Algorithms
EECE 330, Sections 6, & 7
Final Exam — Crib sheet allowed — Dec 16, 2017

The exam has 10 problems for **120 total possible points** with additional **10 bonus points** in problem 3. Points should provide you with good timing hints. You are expected to work against **95 points**. The first five problems cover basic material for **(45 pts.)** The sixth and seventh problem cover queues and stacks for **(20 pts.)** Graph application problem eight is worth **(25 pts.)** Graph application problems nine and ten together are worth **(30 pts.)** *More efficient solutions are worth more points across all problems except problem 10.*

Sprint for grades. (45 pts.)

Problem 1. Function behavior. (6 pts.)

Order the following functions in terms of their asymptotic behavior.

n	$\log_2 n$	$\log_8 n$	$2n^2$
$n \log n$	n^n	$n!$	2^n

Problem 2. Array behavior. (7 pts.)

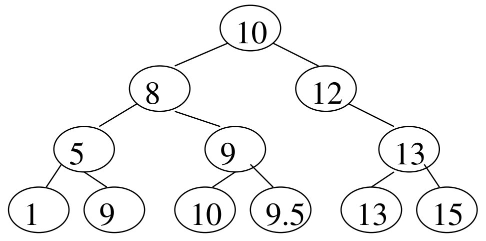
Provide the asymptotic behavior of the following 10 classical algorithms given an array of size n. Order them accordingly.

- Find the median
- Create a binary search tree
- Create a red black balanced binary search tree
- Create a priority queue organized as a binary heap
- Perform binary search on a sorted version of the array
- Find the minimum
- Sort with insertion sort
- Sort with heap sort
- Sort with quick sort
- Sort with merge sort

Problem 3. Binary "search" tree misbehaving. (10 pts.)

The binary tree on the right has exactly two nodes that violate the BST properties. Find both violations, explain why they are violations, and suggest fixes that turn the binary tree into a BST.

Bonus:+(10 pts.) Provide an algorithm that traverses a binary tree T and transforms it into a BST T' by finding and fixing nodes that violate the BST properties. The number of nodes in T' must be equal to those in T and the non-violating nodes must still exist in T'. Reason about the run time of your algorithm.



Problem 4. Runtime analysis. (12 pts.)

Formula $pow(x, n)$ defines a recursive method to compute x^n where $x \in \mathbb{R}$ and n is a non-negative integer.

$$pow(x,n) = \left\{ \begin{array}{ll} 1, & n = 0 \\ x * pow(x, (n-1)/2)^2, & n > 0 \textit{ is odd} \\ pow(x, n/2)^2, & n > 0 \textit{ is even} \end{array} \right\}$$

- Write a recursive algorithm that implements *pow* **(4 pts.)**
- Determine the recurrence relation that characterizes the runtime $T(n)$ of your algorithm **(4 pts.)**
- Provide a closed form solution for $T(n)$ **(4 pts.)**

Problem 5. Graph representation. (10 pts.)

- Would you use an adjacency *matrix* or an adjacency *list* to represent a dense graph? Why? **(3 pts.)**
- Write an algorithm that takes an $n \times n$ Boolean adjacency matrix A that represents a directed graph G of n nodes and returns an adjacency list representation of G. $A[i][j] = true$ denotes an edge exists from node i to node j, and $A[i][j] = false$ denotes no edge exists from i to j. **(7 pts.)**

Take a breath with stacks and queues. (20 pts.)

Problem 6: Array implementation of a queue. (10 pts.)

Consider the array-based implementation of a first-in first-out queue below.

```
int a[N];// N is the capacity
int h, t; // indices of head and tail
int n; // size of the queue

// initially array is empty
AB-Queue() { h=t=n=0;}
//head, tail, size and empty implementations
head() { return a[h];}
tail()  { return a[t];}
size() {return n;}
empty(){return n==0;}
```

```
// enqueue an element at tail
enqueue(e) {
 if (n==N){throw 0;}//throw an exception and exit
 a[t] = e;
 t = t+1 mod N;
 n = n+1; }
//dequeue an element from head
dequeue(e) {
 if (n==0){throw 0;}//throw an exception and exit
 h = h + 1 mod N;
 n = n -1; }
```

- Start from an empty queue q with capacity $N = 4$ and show the state of the array a and the variables h, t and n after the following sequence of calls.
  - `AB-Queue q(); q.enqueue(5); q.enqueue(3); q.dequeue(); q.enqueue(7); q.enqueue(8); q.dequeue(); q.enqueue(6)`
- Find a relation between n, h and t. In other words, can you define n in terms of h and t?
- List one advantage of using array-based queues over using linked-list based queues.
- List one disadvantage of using array-based queues over linked-list based queues.

Problem 7: Array implementation of a stack. (10 pts.)

Provide an array based implementation of a stack. The stack should support `size()`, `empty()`, `top()`, `push(e)` and `pop()`. Provide the runtime of each function. Make sure you go over problem 6 first.

Finish line with graph applications. (55 pts.)

Problem 8: The internet. (25 pts.)

The internet is a huge network. It can be visualized as a graph where clients, servers and routers are nodes and weighted edges connect them. The nodes and the edges of the internet are not saved on one single machine. Rather, frequent routes that are typically accessed through routers are cashed on the routers in routing tables. Thus, the problem of merging and updating partial routing tables arises.

A route $\rho =< (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k) >$ is indexed by its start and end nodes (v_1, v_k). A routing table T has a list of routes $\{\rho_1, \rho_2, \ldots, \rho_{|T|}\}$, a latency estimate w_i for each route $\rho_i \in T$, and a timestamp t_i denoting the last time the latency of ρ_i was assessed.

- Define a quick mechanism for a router to store the routing table such that it quickly returns the route between two nodes (u, v) if the table has it.
- Write an algorithm that takes a routing table T and constructs a graph $G = (V, E)$ that expresses the latest information in T.
  - Decide first what V and E are.
  - Assume each edge e on a route ρ contributes equally to the latency of the route: $\mathit{weight}(e) = \mathit{latency}(\rho)/|\rho|$.
  - Label the edges with the corresponding *latest* estimated latency.
  - Reason about the runtime of your algorithm.
- Write an algorithm that uses G to update the latency of each route in T. Reason about the runtime.
- Give an algorithm that takes routing tables T_1 and T_2 and merges them into an up-to-date routing table T. Table T should have updated routes for all the existing indexed routes of T_1 and T_2. The routes in T should be the routes with the least possible latency given the information from T_1 and T_2.

Problem 9: Graphs for traffic, pollution and noise reduction. (15 pts.)

It is believed that turning two-way streets into one-way streets reduces traffic and thus reduces pollution, noise and fuel consumption. Consider a map of Beirut streets where nodes are intersections and edges are two-way streets. The map has all its intersections connected through the existing two-way streets.

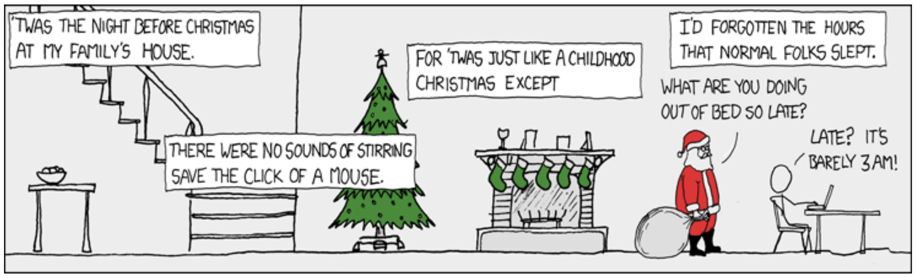
- Devise an algorithm that selects the maximum number of streets that can be turned into one way streets without compromising connectivity. Reason about the runtime of your algorithm.

Problem 10: Graphs in communications. (15 pts.)

Consider a communication channel where the transmitter sends a signal from the set $S = \{a, b, c, d, e\}$ and the receiver receives a signal from the set $\{\alpha, \beta, \gamma, \sigma, \kappa\}$. Unfortunately, due to noise in the channel, signal a may be received as α or β, signal b may be received as β or γ, signal c may be received as γ or σ, signal d may be received as σ or κ, and signal e may be received as κ or α. Consequently, the receiver can not decode the signal accurately. However, if the transmitter never sends signals b and e, then the receiver can always decode signal a correctly.

- Find out the largest subset of S that the transmitter can send without causing ambiguity at the receiver.
- Consider a set of transmission symbols S, a set of reception symbols Φ and a relation $R = \{(s, \alpha) : s \in S, \alpha \in \Phi\}$. R characterises the noise on the channel such that $(s, \alpha) \in R$ denotes that s may be received as α.
  - Devise an algorithm that finds the largest subset of S that results in non-ambiguous reception. The algorithm does not have to be efficient.

Finally, some fun with xkcd. Studying for EECE 330:


'TWAS THE NIGHT BEFORE CHRISTMAS AT MY FAMILY'S HOUSE.
THERE WERE NO SOUNDS OF STIRRING SAVE THE CLICK OF A MOUSE.
FOR 'TWAS JUST LIKE A CHILDHOOD CHRISTMAS EXCEPT
I'D FORGOTTEN THE HOURS THAT NORMAL FOLKS SLEPT.
WHAT ARE YOU DOING OUT OF BED SO LATE?
LATE? IT'S BARELY 3 AM!

Good luck!