CSCI-141 Final Exam Review Packet: SI Edition

Section 01: Professor Deever’s Class

1. Hashing
   1. Your client, Mr. Client, has hired you to construct a password database for his application. You decide that a **hash table** sounds like a good data structure to use. Assuming that each user of Mr. Client’s application is going to have a *username* and a *password*, how might you store these in the hash table?
   2. Your deadline is fast approaching, so you decide that a good **hashing function** is a simple one: the **hash code** will be the ordinal value of the first letter of the key, with ord(a) = 0 and ord(z) = 25. (assume all usernames/passwords will begin with a lowercase letter)

Show the hash table of size 10 thus constructed, given the following collection of user information (use open-addressing to handle collisions):

Username: Password:

bob1234 kewlkat

code\_monkey\_93 csIsFun

l33t\_sk33lz soIsSI

fred5678 smiles

enter\_username enter\_password

pr0gr4mm3r may\_04\_94

freeSpirit07 creativiT

* 1. Write pseudo-code for finding a given user’s information in the hash table. Be sure to include collision handling. (Hint: a given user might not exist in the database yet. How would you check for this?)
  2. What is the best case runtime (think Big-O) of your pseudo-code from part c? What is the worst case runtime? (Hint: what causes a decrease in runtime efficiency?)

1. Stacks & Queues
   1. Draw a diagram representing the state of a stack after calling each of the following operations on that stack:

push( ‘fun!’ )

push( ‘is’ )

push( ‘Studying’ )

pop()

push( ‘computer science’ )

pop()

pop()

pop()

* 1. Assuming that pop() returns what it pops, show the output of the above operations, if you were to print what pop() returns each time.
  2. Draw a diagram representing the state of a queue after calling each of the following operations on that queue:

enqueue( ‘Studying’ )

enqueue( ‘computer science’ )

enqueue( ‘is’ )

dequeue()

enqueue( ‘fun!’ )

dequeue()

dequeue()

dequeue()

* 1. Assuming that dequeue() returns what it dequeues, show the output of the above operations, if you were to print what dequeue() returns each time.

1. Linked Lists & Arrays
   1. Which operations does a Linked List perform quicker than an Array? Which operations does and Array perform faster than a Linked List? Based on that, when should we use Arrays, and when should we use Linked Lists?
   2. Mr. Client from problem 0 is back, and has declared that his application is going to include a feature to allow users to search for friends. Mr. Client wants users to be able to search the database for a given username. Assuming that you have User objects which contain (among other user information) the username, would you rather use a Linked List or an Array to store them? Why?
   3. What would be the Big-O runtime of searching for a given user in the data structure you chose in part b?
2. Trees
   1. After working on Mr. Client’s friend-search feature for a while, you decide that maybe a tree-based data structure would have been a better fit. Show the state of the Binary Search Tree created by calling each of the following operations in order (usernames are to be sorted lexicographically):

add( ‘fred5678’ )

add( ‘code\_monkey\_93’ )

add( ‘bob1234’ )

add( ‘l33t\_sk33lz’ )

add( ‘enter\_username’ )

add( ‘pr0gr4mm3r’ )

add( ‘freeSpirit07’ )

* 1. What is the Big-O runtime of searching for a given user now?

1. Heaps

Given the following array-based min-heap:

[ 1, 4, 2, 5, 7, 3, 6 ]

* 1. What is the parent-value of value 3? Of value 7? (Hint: if you cannot remember the formulas for finding parents/children, draw the heap as a tree and number the values in order. The numbers you just assigned are the indices of the values in the heap-list)
  2. What is the left-child of value 4? What is the right-child of value 1?
  3. Draw the resulting heap-tree after a single remove operation has been performed on the given list.

1. Sorting II: Optimal Algorithms

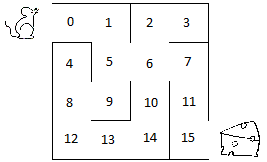
Given the following list:

[ 2, 5, 1, 4, 7, 8, 3, 6 ]

* 1. Perform merge-sort. At each step, split the data based on even/odd indices. Draw a diagram of the sort at each step.
  2. Perform quick-sort. At each step, select the middle-index (round down) as the pivot. Show the results of the sort at each step.

1. Backtracking

Being evil scientists, part of our job is running rats through mazes. It is common knowledge that rats solve mazes in a backtracking manner. Given the following maze:



* 1. What information do we know about the maze? List all of the information that is pertinent to solving the maze. What do we call group of information?
  2. What possible moves could the rat make from any given arbitrary location? What conditions would make any of these moves invalid? Can any of the moves which are valid be pruned? (Hint: We prune when we know that a given branch won’t lead to a solution. Which valid branches do we know this about?)
  3. What data structure(s) could be used to represent the maze in memory? (For those who have done this problem in SI already, try picking a different data structure)
  4. The recursive backtracking algorithm as seen in the lecture notes is as follows:

**def Solve( config ):**

**if isGoal( config ):**

**return config**

**else:**

**for child in Successors( config ):**

**if isValid( child ):**

**solution = Solve( child )**

**if( solution != None ):**

**return solution**

**return None**

Given that code, write Python code for Successors( config) and isValid( config ).

Just for funsies, over the break you could code up a solution to this problem, add some graphical output for the maze, and impress all of your friends and family with your ace programming skills :n)

Some thoughts:

-Remember, the goal is to come up with a solution to (a.k.a. a path through) the maze.

-How will you detect that you have reached the end of the maze? You could hardcode in that square 15 is the end, or you could create a FinishNode class, or you could come up with some solution of your own.

-How will you get the path through, once you find the end of the maze? How will you display it? (because of course we want to brag that our rat found its way through the maze)

Have a great break!

--Chester Taylor

--Your Friendly Neighborhood SI