

15-110 Recitation Week 14

Reminders

- HW6 due TOMORROW at noon! NO REVISIONS!
 - Always double check that your code is passing/not breaking the autograder, and stop by OH tonight get debugging help if it is
- Final exam:
 - Monday, April 27, 8:30 - 11:30 am: GHC 4401, GHC 4307
 - Review session this weekend!
 - Small groups this weekend!
- [Recitation feedback form](#)

Overview

Review topics:

- Runtime/Big-O and Tractability
- Aliasing code reading
- Recursion
- Simulation - MVC and Monte Carlo
- Data Analysis
- Search Algorithms

Problems

REVIEW TOPICS:

Big-O and Tractability

What are the Big-O runtimes of the following mystery functions? List the Big-O of each line in the function to help you find the overall runtime

```
def mystery1(matrix): # matrix is an n by 10 2D list of integers
    total = 0
    for row in matrix:
        row.sort() #sort runs in nlogn time
        for elem in row[:2]:
            total += elem
    return total
```

Overall runtime:

```
def mystery2(n): # n is an integer
    newN = n**2
    result = []
    for i in range(0, newN, n):
        print(i * n)
        for j in range(2, newN):
            result.append(i * j)
    return result
```

Overall runtime:

Tractability/P vs. NP questions

What are the Useful NP Problems?

If you try to solve the subset sum problem by finding the sum of all subsets of the input, is this tractable or intractable? Explain.

If someone gives you a potential solution to the subset sum problem, is verifying whether the solution is correct tractable or intractable?

What complexity class is the subset sum problem in and why?

Aliasing Code Reading

What values do the following variables have after the code has finished running?

```
A = [1, 2, 3]
```

```
B = A[1:]
```

```
C = A
```

```
D = C
```

```
E = [1, 2, 3]
```

```
B.append(A[0]*2)
```

```
C.extend([4, 5, 6])
```

```
C.append(7)
```

```
C = C + [4]
```

```
A.pop(0)
```

```
D.pop()
```

A = _____

B = _____

C = _____

D = _____

E = _____

Which values are still aliased at the end of the code?

--

Recursive Functions

Function 1:

Write a recursive function that returns all capital letters in a string in reverse order of appearance

`#reverseCaps("EnjoY a rEaLlY Long wintEr brEak")` returns **"EELYLEYE"**

Function 2:

Write a recursive function `evaluate(numbers, operations)` that takes in a list of numbers of size `n` and a list of operations of size `n - 1` and recursively evaluates the expression `numbers[0] operations[0] numbers[1] ... operations[n-1] numbers[n]`.

For example, if `numbers = [5, 6, 4]` and `operations = ["+", "-"]` then `evaluate(numbers, operations)` will return `5 + 6 - 4 = 7`. Note the only operations you will get are "+" and "-" so you don't have to worry about order of operations. You can assume `len(numbers)` will always be at least 1 and that `len(operations)` will always be `len(numbers) - 1`.

Recursive Number Tree

Write a recursive function `getLargestNum(tree)` that takes in a tree with non-negative integers as nodes and returns the **largest number** in the tree.

For example, the following tree should return 10:

```
t = {"contents" : 6,
     "left" : {"contents" : 0,
               "left" : {"contents" : 10,
                         "left" : None,
                         "right" : None},
               "right" : {"contents" : 7,
                         "left" : None,
                         "right" : None}},
     "right" : {"contents" : 2,
               "left" : None,
               "right" : {"contents" : 9,
                         "left" : None,
                         "right" : None}}}
```



Simulation - Model, View, Controller

1. Imagine we have the following parts of a simulation:

Set the starting position and radius of a circle and draw it in the center of the canvas.

What functions would we use for this?

The circle moves horizontally left and right across the canvas based on time (when it reaches one end it switches directions).

What functions do we need?

2. Now go to your starter file where we have provided this simulation! Let's add the following elements inside keyPressed function:
 - If we hit the return/enter key, we make the circle stop or start moving
 - If we hit the up arrow key, we move the circle up the canvas by 10 pixels
 - If we hit the down arrow key, we move the circle down the canvas by 10 pixels

Recall that in `keyPressed(data, event)` we can use `event.keysym` to get the "name" for characters we can't show in strings. For example:

- Enter/Return: `event.keysym == "Return"`
- Up arrow key: `event.keysym == "Up"`
- Down arrow key: `event.keysym == "Down"`

Simulation - Monte Carlo

Write a Monte Carlo Simulation to compute how long you have to listen to songs on your Spotify playlist until you hit "Headlines" by Drake. The `runTrial` function you write should take in a 2D list of songs, where each inner list has two elements. The 0th index of the inner list is the song name, and the 1st index is the length of the song in seconds. This 2D list will be given to you in the `getExpectedValue` function. Remember that on each trial, the order of the songs in the playlist will be randomly shuffled. You can use the `random.shuffle` function to help you!

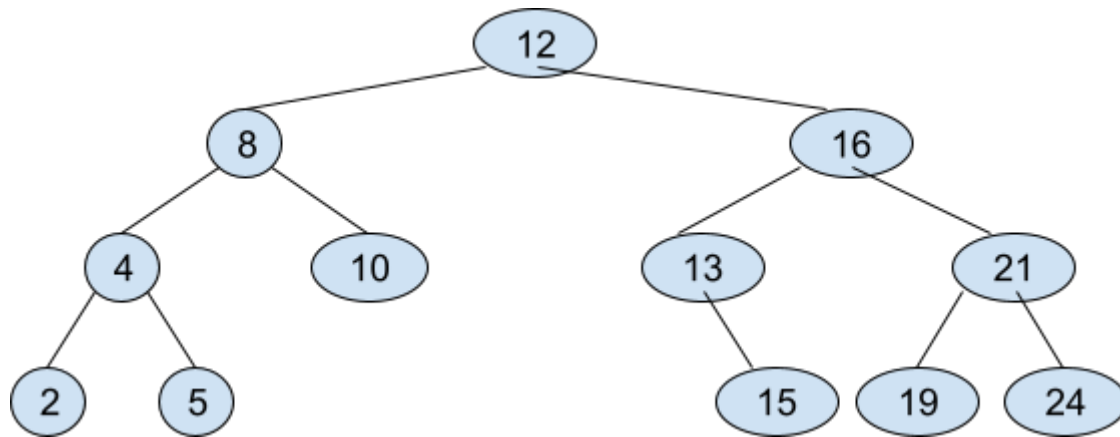
Data Analysis

Given the following types of data, name the best visualization method to display that data:

- A. Numerical $\times$ Ordinal
- B. Numerical
- C. Ordinal $\times$ Ordinal
- D. Categorical
- E. Numerical $\times$ Numerical $\times$ Numerical

Write a function `geneExpression(filename, sample)` that takes in a text filename and a sample number and creates a bar plot visualization of the gene expression levels for the corresponding sample. First, read the file corresponding to the filename passed in. This file is in CSV format with several lines. For each string percentage in a line, convert the string percentage into a float and add it to a temporary list. Then for each line, add the temporary list to an overall list, creating a 2D list of gene expression percentages. Each row will correspond to a specific gene and each column will correspond to a specific sample. Then, plot the expression level for each gene for the input sample using the matplotlib bar plot function. On the x-axis, set the values to the string “Gene x ” for every index x and on the y-axis, let the values be the expression percentage for each gene for the given sample.

Search Algorithms



Perform binary search looking for the following elements in the BST. Write down all elements that are searched until the element in question is found or not.

15:

4:

18:

9: