CS 313E - Elements of Software Design

Table of Contents

About

This is my repository containing all content related from the Fall 2019 semester course, CS 313E, from the University of Texas at Austin. This course is taught by Dr. Shyamal Mitra and explores algorithms, data structures, object oriented programming paradigms and software design concepts.

Assignments

1. (Odd) Magic Sqaure
    - Create a 3x3 grid of integers such that every row, column, and the two diagonals equal the same number, which is 15 in the case of a 3x3 grid.
2. Collapsing Intervals
    - Given a list of intervals (represented by a tuple with starting and ending points), return a modified list of collapsed intervals. In order for two intervals to collapse, the ending point of one must be greater than the starting point of another. For example, the intervals (-10, 0) and (-5, 20) collapse into (-10, 20).
3. Baby Names
    - Allow for a user to query a database of the 1000 most popular names in the United States per decade.
4. Word Search
    - Given a file with a word search (grid of letters) and a list of words, write a program that returns each word in the list if it was found in the search. Remember that words can be valid whether they are represented backwards, diagonally, upside down, etc in the grid.
5. Basic Geometry
    - Familiarize yourself with Python and object oriented programming concepts. Create basic classes and methods for Point, Circle, Rectangle, and Triangle objects.
6. Office Space

- Imagine that your workplace is moving to a new location and everyone requests a specific amount and the location of their new personal office space. Find the total amounts of unallocated and contested office space and the guaranteed space allowed for each employee.

7. Convex Hull
   - A convex hull is the smallest convex polygon that encloses a certain set of points. Given a file of coordinates, find the points that comprise the convex hull solution.
8. Work 'Till You Drop
   - The scenario is that it's late at night, and you have to write n lines of code before the morning. After writing *v* lines of code, you drink a cup of coffee to regain some energy,but it actually reduces your productivity so that you only write *v // k ** 1* new lines of code. After another cup, you can only write *v // k ** 2*. Find the lowest amount of *v* lines of code so that you can write at least *n* lines of code where *1 ≤ n ≤ 10^6* and *2 ≤ k ≤ 10*.
9. Crossing the Bridge
   - There are *n* people who wish to cross a bridge at night. Only two people may cross the bridge at any time and they must have a flashlight to get across. After a pair crosses, a person must come back so that the people still needing to cross have the flashlight. Also, each person crosses the bridge a different speed so the crossing time of a pair of people is determined by the slower one's time. Find the minimum time in which everyone can cross the bridge.
10. Recursion 2 (CodingBat)
    - 8 short but challenging problems exploring recursion. They were taken from CodingBat's problem set.
11. Greatest Path Sum in a Grid
    - Given a grid of positive integers, find the greatest path sum in the grid starting from the lop left corner and ending in the bottom right corner. You are only allowed to traverse to the right or down.
12. (Even) Magic Square
    - In contrast to odd magic squares, even magic squares contain vastly more solutions for each dimension. For example, the solution set for dimension = 4 contains 880 valid grids while the solution set for dimension = 3 contains only 8 valid squares.
13. Nesting Boxes
    - Given a set a boxes (list of three dimensions represented by integers), find the largest subset of nesting boxes. A box can nest within another if it's length, width, and height are strictly less than the other box's respective dimension. Boxes are allowed to be rotated therefore the order of dimensions representing a single box does not matter.

14. Eight Queens Problem
    - Given a number *n*, denoting the number of queens and the dimension of the chessboard, print all boards containing n queens with each queen in a position so that it cannot capture or be captured by another queen.
15. Reducible Words
    - What is the longest English word that remains a valid English word as you remove one letter at a time from it? Once only one letter is left, it is only valid if it is "a", "i", or "o". Such words are known as reducible words. Consider the example, "string":
        - *string -> sting -> sing -> sin -> in -> i*
    - Given a list of valid English words (roughly 113,809 entries), write a program that finds the longest reducible word(s). To solve this problem optimally, the solution **must** implement hashing.
16. Singly Linked List
    - Write the various methods for a linked list implementation in Python (such as insertion, deletion, removal, sorting, copying, merging, searching, and so forth) and test them. Assume that the data you are handling is just integers for now (for easier comparisons with no errors in floating point precision).
17. Josephus Problem (Circularly Linked List)
    - The Josephus problem is as follows: there is a group of soldiers surrounded by an overwhelming enemy force. There is no hope for victory, so they make a pact to commit suicide. They stand in a circle and starting from a randomly selected solider, they count *n* soliders. That soldier is killed by the starting solider and the count begins again from the next solider after. One by one, each solider is killed until one is standing who is free to join the enemy forces.
    - Given an input file with total number of solders, the starting solider, and *n*, write a program that determines the last solider alive using an implementation of a linked list.
18. Polynomial Operations w/ Linked Lists
    - Assuming that the polynomials given will have non-zero integer coefficients and exponents are greater than zero, define a Linked List (and Link) class that represents a polynomial. You will be given a file, poly.txt, containing data for just two polynomials. The first line will represent the number of given terms, *n*, for the first polynomial followed by *n* lines. Each line has two integers separated by a space, with the first int representing the coefficient and the second representing the exponent. Following this will be a blank line, then the exact same format for the second polynomial.
    - Create a linked list of terms that contain the coefficient, exponent, and pointer to the next term. Then write methods to multiply and add polynomials together (do not forget a simplify / reduce method).
19. Expression Trees

    - Given a valid mathematical expression in infix notation, create an expression tree. Write methods that evaluate the expression and also print the prefix and postfix notation equivalent.

20. Cipher Encryption / Decryption with Binary Search Trees
    - Write a program that will encrypt / decrypt string inputs based on a given encryption key. First create a BST comprised of unique set of letters in the encryption key. To encrypt a string, change each character's representation to the specific traversal moves required to get the character. To decrypt, simply follow the given directions in the input.

Exams

- Exam 01
    1. Define a Triangle class (assuming a Point class has been written), with proper methods for perimeter, area, point_inside, and is_isosceles.
    2. Given a grid of 0s and 1s, find the largest rectangle of 1s (use a max area of a histogram)
    3. Given a string, find the longest possible palindrome in the string.
    4. Given a number n, find x such that x! = n
    5. Get a list of people (represented by letters), pair each one with another in a configuration such that every person's requirement (if there is one) is satisfied.
    6. (Extra credit) - Trace the Ackermann function up to m = 3 and n = 3.
        - A(m, n) = n + 1 if m = 0
        - A(m, n) = A(m - 1, 1) if m > 0 and n = 0
        - A(m, n) = A(m - 1, A(m, n - 1)) if m > 0 and n > 0

Exercises

- Data Structures
    - Stacks, queues
    - Trees (binary search trees, expression trees, and decision trees)
    - Linked lists (singly, doubly, and circularly-linked)
    - Hash tables
- Algorithms
    - Sorting (selection, bubble, insertion, merge, quick, heap)
    - Searching (sequential, binary)
    - Hashing (linear probing, quadratic probing, double hashing)

Instructions

These scripts are imported into a testing script as modules. Their tests are located within the testing.py script, which is where you can test any functionality of the particular data structure or algorithm.

```
cd <path-to/cs-313e/exercises>
python testing.py
```

Acknowledgments

I am grateful for the amazing opportunity to learn from Dr. Mitra - he was absolutely excellent and I always felt as though I was learning loads from him every class. I fully recommend this class to anyone interested or able to take it. The work and effort is worth it.

Thank you very much for reading !

back to top ↑