COP 3503 Spring 2024 Section 1 Final Exam Solutions

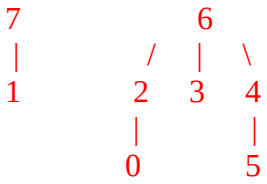
Date: 4/25/2024

1) (10 pts) Write a method that takes in a TreeSet of Strings, which are all lowercase names and returns a TreeMap mapping each name to its 0-based sorted ranking. So for example, if the input set contained the names "Quan", "Alice", "Fred" and "Jamal", then the TreeMap should have the following mappings: Alice → 0, Fred → 1, Jamal → 2 and Quan → 3.

```
TreeMap<String,Integer> getMap(TreeSet<String> names) {

  // Grading: 2 pts for initialization.
  TreeMap<String,Integer> map = new TreeMap<String,Integer>();
  int id = 0;

  // Grading: 2 pts for loop condition.
  while (names.size() > 0) {

    // Grading: 2 pts to retrieve next name.
    String name = names.pollFirst();

    // Grading: 2 pts for put, 1 pt for id update
    map.put(name, id++);
  }

  // Grading: 1 pt
  return map;
}
```

2) (5 pts) Draw the corresponding picture for the disjoint set stored by this array:

Index	0	1	2	3	4	5	6	7
Value	2	7	6	6	6	4	6	7



Grading: 1 pt for first tree on left, 4 pts for second tree, grader decides partial

3) (20 pts) Let students, numbered 0, 1, 2, … in a class of n students, where n = rc, originally sit in a grid of seats with r rows and c columns as follows:

0	1	…	c-1
c	c+1	…	2c-1
…	…	…	…
n – c	n – c + 1	…	n - 1

Imagine that each student must get up and sit in a different seat that belongs to both a different row and different column than they originally were seated. Complete the program below so that it finds such an arrangement. (Specifically, it'll find the first lexicographical arrangement that is valid, if one exists.)

In the method, both the permutation array and used array are of size n, k represents the current slot of the permutation array to be filled in, and numC represents the number of columns in the classroom grid. Assume that n is divisible by numC and that the permutation array is storing the classroom with the first row in indexes 0 to c – 1, the second row in indexes c to 2c – 1, etc.

For example, calling the method with an empty permutation array for n = 12, numC = 4 returns true and fills the permutation array as follows:

5	4	7	6	9	8	11	10	1	0	3	2

This corresponds to a classroom arrangement of:

5	4	7	6
9	8	11	10
1	0	3	2

Fill in the necessary code in the go method to complete the solution:

```
import java.util.*;

public class q3 {
  public static void main(String[] args) {
    int n = 12, c = 4;
    int[] p = new int[n];
    boolean[] u = new boolean[n];
    boolean res = go(p, u, 0, c);
    if (res) {
      for (int i=0; i<n; i++) {
        System.out.print(p[i]+"\t");
        if (i%c == c-1) System.out.println();
      }
    }
  }
```

```
  public static boolean go(int[] perm, boolean[] used,
                           int k, int numC) {
    int n = perm.length;
    if (k == n) return true;
    for (int i=0; i<n; i++) {

      // Grading: 2 pts
      if (used[i]) continue;

      // Grading: 3 pts for each condition, row, col
      if (i/numC == k/numC) continue;
      if (i%numC == k%numC) continue;

      // Grading: 2 pts for each of these.
      used[i] = true;
      perm[k] = i;

      // Grading: 4 pts
      boolean tmp = go(perm, used, k+1, numC);

      // Grading: 2 pts for each of these.
      if (tmp) return true;
      used[i] = false;
    }

    return false;
  }
}
```

4) (6 pts) In the Divide and Conquer multiplication algorithm (Karatsuba's) when integers A and B are multiplied, three recursive multiplications are performed. When A = 178 and B = 117, assuming we split A and B into 8 bits, what are the three recursive multiplications that occur? Please express your answers in decimal. (Don't calculate the product, just give the two numbers that get multiplied. Credit is only given for the answers, 1 pt per slot.

Product #1: **11 x 7**

A = 11 x 16 + 2 (A_H = 11, A_L = 2)
B = 7 x 16 + 5 (B_H = 7, B_L = 5)

Product #2: **2 x 5**

Product #3: (11 + 2) x (7 + 5) = **13 x 12**
Note: Middle Term = 13 x 12 – 11 x 7 – 2 x 5 = 69, so we get:
178 x 117 = 11 x 7 x 16^2 + 69 x 16 + 2 x 5 = 19712 + 1104 + 10 = 20826 (correct)

Grading: Products can be in any order, try your best to match them. 1 pt for each correct value in a product.

5) (15 pts) Determine the fewest number of multiplications to calculate the product ABCDE, for matrices A, B, C, D, E with the following dimensions:

Matrix	Dimensions
A	2x7
B	7x3
C	3x4
D	4x1
E	1x5

In order to get full credit you must fill out the chart below appropriately, as shown in class. Please include your calculations below the chart.

	A	B	C	D	E
A	0	42	66	47	57
B	X	0	84	33	68
C	X	X	0	12	27
D	X	X	X	0	20
E	X	X	X	X	0

AB = 2 x 7 x 3 = 42
BC = 7 x 3 x 4 = 84
CD = 3 x 4 x 1 = 12
DE = 4 x 1 x 5 = 20

(AB)C = 42 + 2 x 3 x 4 = 42 + 24 = **66**
A(BC) = 84 + 2 x 7 x 4 = 84 + 56 = 140

(BC)D = 84 + 7 x 4 x 1 = 84 + 28 = 112
B(CD) = 12 + 7 x 3 x 1 = 12 + 21 = **33**

(CD)E = 12 + 3 x 1 x 5 = 12 + 15 = **27**
C(DE) = 20 + 3 x 4 x 5 = 20 + 60 = 80

(ABC)D = 66 + 2 x 4 x 1 = 66 + 8 = 74
(AB)(CD) = 42 + 12 + 2 x 3 x 1 = 54 + 6 = 60
A(BCD) = 33 + 2 x 7 x 1 = 33 + 14 = **47**

(BCD)E = 33 + 7 x 1 x 5 = 33 + 35 = **68**
(BC)(DE) = 84 + 20 + 7 x 4 x 5 = 244
B(CDE) = 27 + 7 x 3 x 5 = 132

(ABCD)E = 47 + 2 x 1 x 5 = 47 + 10 = **57**
(ABC)(DE) = 66 + 20 + 2 x 4 x 5 = 126
(AB)(CDE) = 42 + 27 + 2 x 3 x 5 = 99
A(BCDE) = 68 + 2 x 7 x 5 = 138

Grading: First 2 diagonals, 1 pt each (7 total)
Third diagonal: 2 pts each (4 total)
Final answer: 4 pts

6) (9 pts) Consider counting the number of ways to make change for 15 cents using 1 cent, 2 cent, 5 cent and 9 cent coins. Fill in the table below showing the work that the dynamic programming algorithm to solve the problem would do to solve the problem. The first row has been filled out for you for convenience.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8
5	1	2	2	3	4	5	6	7	8	10	11	13	14	16	18
9	1	2	2	3	4	5	6	7	9	11	13	15	17	20	23

Grading: 3 pts per row, give 3 pts if fully correct, 2 pts if 2/3 correct, 1 pt if 1/3 – 2/3 correct.

7) (10 pts) Consider needing to schedule a single room for events. Each event gives a request with a start and end time. For this problem, we aim to maximize the amount of time the room is scheduled (not the number of events). Assume that all requests are ordered pairs of integers, (s, e), where s > 0 and e < 10000, and that there are no more than 1000 requests. Describe a solution to this problem that utilizes Dijkstra's algorithm to correctly solve it. (Hint: think of each event as a vertex in a graph.) Note: there is a dynamic programming solution to this problem. If you very, clearly describe that, you'll get full credit as well.

Let there be n events, then we will create an input graph with n+2 vertices. Create a separate start vertex labeled with time t = 0 and a separate end vertex labeled with time t = 10000. We can treat these as dummy events with duration zero, so for these two vertices s = e. For each event we create a vertex as well. Add a directed edge from u to v if the end time for vertex u is less than or equal to the start time for vertex v. The weight of the edge will be the start time of vertex v minus the end time of vertex u. In essence, an edge in the graph represents scheduling event u followed by event v, which means that the edge weight between the two represents the time the room is idle with no event. Add edges from the start vertex to all other vertices with the weight of the start time of the events. Add edges from each vertex to the end vertex with the weight 10000 – vertex end time. (Essentially, follow the same exact rules for edge formation when considering our two extra vertices.)

Once this graph is formed, the shortest distance between our added start and end vertices represents the minimum amount of time any schedule "wastes" which corresponds to the maximum amount of time the room could be scheduled for. (Every valid path in the graph corresponds to a valid schedule for the room and vice versa.) So the final answer to the question (maximum time the room can be scheduled) is just 10000 – shortest path in the graph.

Grading: 2 pts for adding extra start and end vertices
5 pts for how to add edges
3 pts for explaining how each path in the graph corresponds to a schedule and that the shortest path represents the "least time wasted" which corresponds to the schedule that maximizes the amount of time the room is used.

Note: A DP solution similar to longest increasing subsequence works as well.

8) (10 pts) In the game of 301, a player throws darts at a dart board. Each throw is worth in between 1 and 20 points and gets subtracted from the player's score. The goal of the game is to get to zero. We will make several simplifications to the real game to make the mathematics required for this question easier. Define t(n) to be the expected number of throws it will take a player to complete the game when they start with n points, to a score of 0, assuming that the probability the player scores x points on a single throw is $\frac{1}{20}$, for each integer x in between 1 and 20, inclusive. If a player overshoots their score, so for example, if their current score is 7 but they get 12 points on the dart they throw, then their score reverts back to 7. (In short, a throw that's too big gets ignored.)

(a) Prove that t(1) = 20.

With probability 1/20, if we start with a score of 1, we win. With probability 19/20, we revert back to the same state. This gives us the following recurrence relation:

$$t(1) = 1 + \frac{19}{20}t(1)$$

Solve for t(1): $t(1) - \frac{19}{20}t(1) = 1 \rightarrow$ t(1): $\frac{1}{20}t(1) = 1 \rightarrow$): $t(1) = 20$, as desired. (Note: The first 1 represents the first turn itself.)

Grading: 1 pt for writing down recurrence relation, 1 pt for explanation, 1 pt to solve for t(1).

(b) Under the assumption that t(1) = t(2) = t(3) … = t(x) = 20, where x is an arbitrarily chosen integer in between 1 and 19, prove that t(x + 1) = 20 as well.

If our current score ix x+1, then with probability 1/20, in one shot we'll have a score of x, x – 1, x – 2, etc., 1. Also, with probability 1/20, we'll win. Finally, with probability $\frac{20-x-1}{20} = \frac{19-x}{20}$, our score will revert back to x+1. Thus, we can set up the following recurrence relation:

$$t(x+1) = 1 + \left[\frac{1}{20}\sum_{i=1}^{x} t(x)\right] + \frac{19-x}{20}t(x+1)$$

Using our assumption that t(x) = 20, we have:

$$t(x+1) - [\frac{19-x}{20}t(x+1)] = 1 + \left[\frac{1}{20}\sum_{i=1}^{x} 20\right]$$

$$t(x+1)(\frac{20}{20} - (\frac{19-x}{20})) = 1 + \frac{1}{20}(20x)$$

$$t(x+1)\left(\frac{x+1}{20}\right) = x + 1$$

$$t(x+1) = 20$$

Grading: 3 pts for initial equation, 4 pts for algebra, grader decides partial

9) (15 pts) Gerald is buying books from "Buy One Get One Free Emporium Book Store." As the title of the store suggests, if a customer buys one book, he receives a book of equal or lesser value for free (if one is in stock.) Gerald has a fixed amount of money, ***money***, and he'd like to purchase the maximum number of books possible. Complete the method below, so that it takes in an array, ***bookprices***, of the prices of all the books in the store and the fixed amount of money he's willing to spend, ***money***, and returns the maximum number of books he could buy from the store without spending more than ***money***. (Note: There are some corner cases to consider. Namely, it's possible that the total number of books purchased could be an odd number.) Your code should run in expected O(nlgn) time, where n is the size of the array ***bookprices***.

```
public static int getMaxBooks(long[] bookprices, long money) {

    Arrays.sort(bookprices);
    int res = 0;
    long zeroCost = 0, oneCost = 0;

    for (int i=0; i<bookprices.length; i+=2) {

        if (zeroCost + bookprices[i] <= money) {
            zeroCost += bookprices[i];
            res = Math.max(res, i+1);
        }

        if (i+1<bookprices.length &&
            oneCost + bookprices[i+1] <= money) {

            oneCost += bookprices[i+1];
            res = Math.max(res, i+2);
         }
    }

    return res;
}
```

Grading: **Max 12/15 if they only try buying odd indexes.**
Max 12/15 if they only try buying even indexes.
For correct approach, lay out points roughly like this:

Sort – 2 pts
Updating current "expenditures" – 4 pts
"Stopping" when we run out of money – 3 pts
Returning the amount of books based on simulation – 3 pts
Considering both options (odd vs even indexes) – 3 pts

10) (7 pts) A graph has vertices labeled 'A' through 'N'. The edges in the graph (via adjacency list) are given below. Show the order that vertices get visited in a breadth first search starting at A. Whenever choosing between multiple options to enqueue, select the one that comes first alphabetically. Note: for ease, edges are provided in both directions.

Adjacency List of Graph

A → M	H → B, N
B → F, H, M	I → E, L
C → D, E, F, J, K	J → C, F, N
D → C, E, M	K → C, M
E → C, D, I	L → I
F → B, C, J, N	M → A, B, D, K
G → N	N → F, H, J, G

BFS: A, M , B , D , K , F , H , C , E , J , N , I , G, L

Grading: ½ pt per slot, round down

11) (4 pts) In Program 4, you stored the position of several drones in a single integer. If drone #1 (least significant bits) is in row 3, column 2, and done #2 is in row 0 column 5, what single integer would store this set of drone locations, if you used the implementation requirement given?

In binary: 000 101 011 010,
convert to decimal = $2^8 + 2^6 + 2^4 + 2^3 + 2^1 = 256 + 64 + 16 + 8 + 2 = 346$

346

Grading: 2 pts binary, 2 pts to convert to decimal

12) (5 pts) In class we looked at an algorithm to determine the rank of a permutation in lexicographical order.

Using this algorithm, determine the lexicographical rank of the permutation [3, 1, 0, 5, 2, 4]. Your answer must be in between 0 and 719, inclusive.

Rank = 3 x 5! + 1 x 4! + 0 x 3! + 2 x 2! + 0 x 1! = 3 x 120 + 24 + 0 + 4 + 0 = 360 + 28 = **388**

388

Grading: 1 pt first term, 1 pt second term, 2 pts for the last set of terms, 1 pt final answer

13) (8 pts) In Program 5, you had to match drug names with corresponding codes. A drug could only match a code if the code appeared as a substring in the name of the drug. A greedy algorithm would simply go through the drugs in order and match them to the first code they match to, then repeat the process. Give an example with 4 drugs and 4 codes, where this algorithm fails to find a matching, but a matching exists. To get full credit, you must clearly order your drugs and codes, show the incomplete matching the algorithm would provide AND give a valid matching of all four drugs. (Your examples can use any alphabetic strings; they don't have to be names of real drugs.)

Drugs	Codes
1. abcd	1. ab
2. cab	2. bcd
3. jello	3. tac
4. taco	4. ello

Greedy Algorithm produces the following matching:

1. abcd	→	ab	(Note: At least one of these slots on the Right hand side must be blank.)
2. cab	→		
3. jello	→	ello	
4. taco	→	tac	

Here is a valid matching for the example:

1. abcd	1. bcd
2. cab	2. ab
3. jello	3. ello
4. taco	4. Tac

Grading: **2 pts for any set of drugs and codes**
2 pts if the given set has some valid matching
2 pts if the student gives a valid matching
2 pts if the example is such that the greedy doesn't match all 4

14) (1 pt) The dining hall '63 South is a nod to the year UCF was founded. In what year was UCF founded? **1963 (Give to All)**