CS6033 Design and Analysis of Algorithms I
Homework Assignment 7 Part 2
December 11, 2022

Group Members:
rk4305 (Sai Rajeev Koppuravuri)
sg7372 (Sriharsha Gaddipati)
vt2182 (Venu Vardhan Reddy Tekula)
vt2184 (Veeravenkata Raghavendra Naveenkumar Tata)

---

1. (10 points) You are given a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. You wish to schedule all the activities using as few lecture halls as possible. Give an efficient greed algorithm to determine which activity should use which lecture hall.

   Ans:
   Let's say, we have S set of activities (each activity(interval) has a start time and end time). First, we sort the start time and end time of activities separately. We maintain the maximum number of rooms required at any point in time. If any meeting ends, we decrease the current rooms and if any meeting starts, we increase the current rooms.

```
def minMeetingRooms(intervals):
    starts = sorted([interval[0] for interval in intervals])
    ends = sorted([interval[1] for interval in intervals])
    meet_count, cur_count = 0, 0
    s_ptr, e_ptr = 0, 0
    while s_ptr < len(intervals):
        if starts[s] < ends[e]:
            s_ptr += 1
            cur_count += 1
        else:
            e_ptr += 1
            cur_count -= 1
        meet_count = max(meet_count, cur_count)
    return meet_count
```

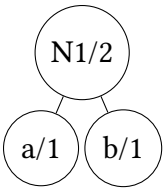
   Time complexity : O(NlogN)

   Space complexity : O(N)

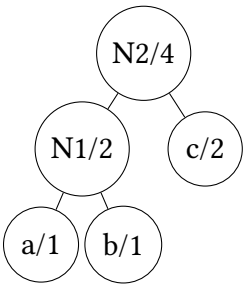
2. (10 points) What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

   a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

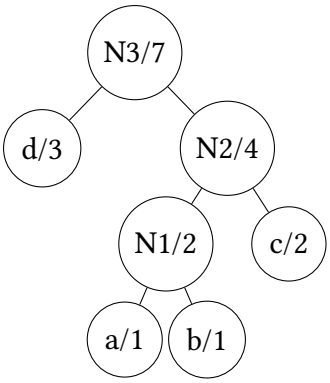
   Ans:



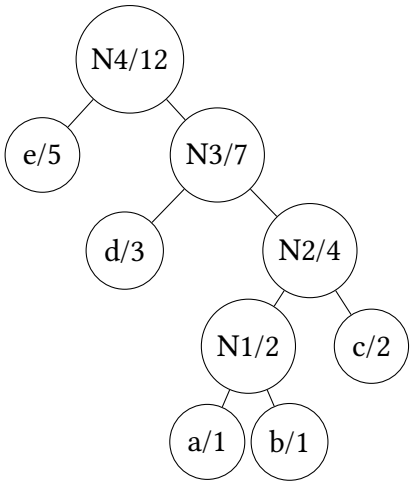
N1:2 c:2 d:3 e:5 f:8 g:13 h:21



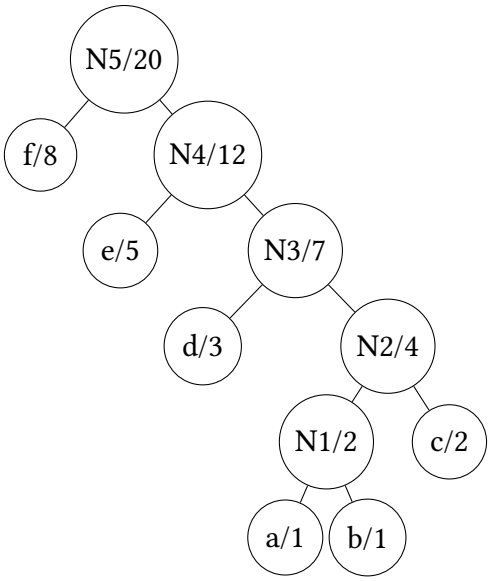
N1:2 c:2 d:3 e:5 f:8 g:13 h:21



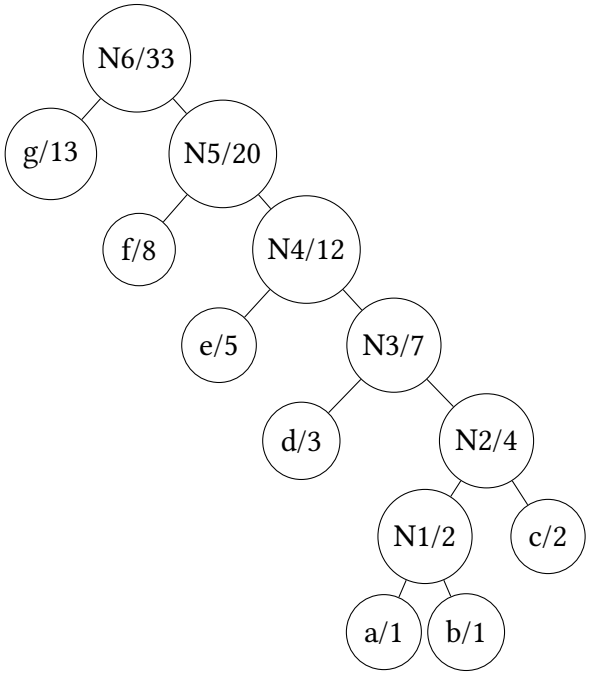
N2:4 d:3 e:5 f:8 g:13 h:21



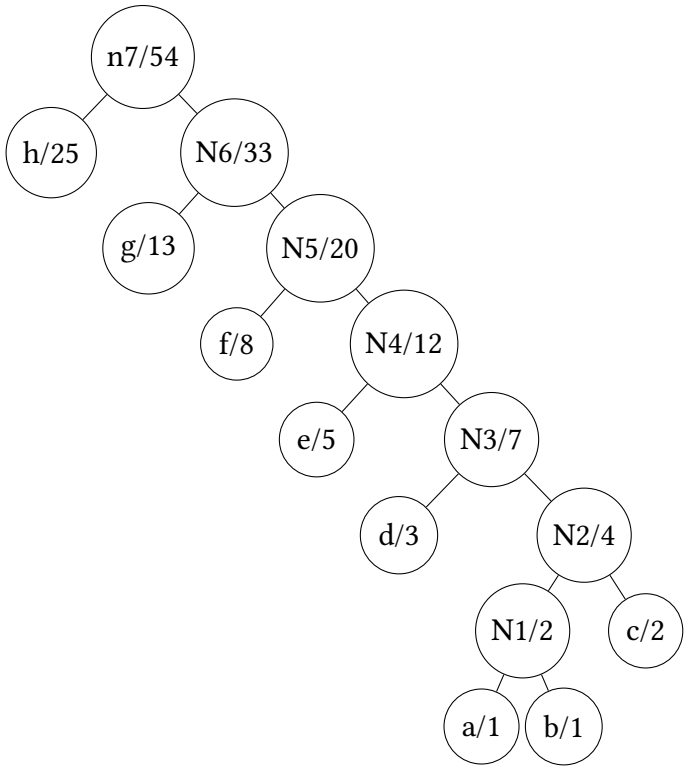
N3:7 e:5 f:8 g:13 h:21



N4:12 f:8 g:13 h:21



N5:20 g:13 h:21



N6:33 h:21

a: 1111100
b: 1111101
c: 111111
d: 11110
e: 1110
f: 110
g: 10
h: 0

3. (10 points) You are given an integer array [nums]. In one operation, you can choose an element of the array and increment it by 1. For example, if nums = [1,2,3], you can choose to increment nums[1] to make nums = [1,3,3]

   Return the minimum number of operations needed to make nums strictly increasing. Write a pseudocode and the running time for the same.

   (An array nums is strictly increasing if $nums[i] < nums[i+1]$ for all $0 \leq i < nums.len-1$. An array of length 1 is trivially strictly increasing.)

   Ans:

   (a) Initialize a variable to store the number of operations

   (b) Loop through the array starting from the second element

   (c) If the current element is not strictly greater than the previous element

      i. Increment the number of operations by the difference between the current element and the previous element plus 1. This is because we need to increment the

current element by at least 1 in order to make it strictly greater than the previous element.

ii. Increment the current element by the difference between the previous element and the current element plus 1 to make it strictly greater than the previous element

enumerate

(d) Return the total number of operations

```
def minOperations(nums):
    numOperations = 0
    for i in range(1, len(nums)):
        if nums[i] <= nums[i-1]:
            numOperations += nums[i-1] - nums[i] + 1
            nums[i] = nums[i-1] + 1
    return numOperations
```

The running time of this algorithm is O(n), where n is the length of the array nums. This is because the algorithm only loops through the array once, and all other operations run in constant time.

4. (10 points) At a lemonade stand, each lemonade costs $5. Customers are standing in a queue to buy from you and order one at a time (in the order specified by bills). Each customer will only buy one lemonade and pay with either a $5, $10, or $20 bill. You must provide the correct change to each customer so that the net transaction is that the customer pays $5.

   Note that you do not have any change in hand at first. Given an integer array bills where bills[i] is the bill the ith customer pays, return true if you can provide every customer with the correct change, or false otherwise.

   Write a pseudocode and the running time for the same.

   Input: bills = [5,5,5,10,20]
   Output: true
   Explanation: From the first 3 customers, we collect three $5 bills in order. From the fourth customer, we collect a $10 bill and give back a $5. From the fifth customer, we give a $10 bill and a $5 bill. Since all customers got correct change, we output true.

   Ans:

   To solve this problem, we can keep a count of the number of $5, and $10 bills we have on hand. Then, as we process each customer's payment in the array, we can update our counts accordingly.

   When a customer pays with a $5 bill, we can simply increase the count of $5 bills we have on hand. When a customer pays with a $10 bill, we will need to provide them with a $5 bill as change. If we have at least one $5 bill on hand, we can decrease the count of $5 bills we have on hand and increase the count of $10 bills we have on hand. If we don't have any $5 bills on hand, we will not be able to provide the correct change, so we should return false.

When a customer pays with a $20 bill, we will need to provide them with $15 bills as change.

If we have at least one $10 and $5 dollar bills on hand, we can decrease the count of $10 bills and $5 bills by 1, and increase the count of $20 bills. Also, if you have three $5 bills on hand, we can decrease the count of $5 bills we have on hand by three and increase the count of $20 bills we have on hand. If none of the above two conditions work, we will not be able to provide the correct change, so we should return false.

Once we have processed all the payments in the array, we can return true if we were able to provide the correct change to each customer, or false otherwise.

```
def lemonadeChange(bills):
    five = ten = 0
    for bill in bills:
        if bill == 5:
            five += 1
        elif bill == 10:
            if not five:
                return False
            five -= 1
            ten += 1
        else:
            if ten and five:
                ten -= 1
                five -= 1
            elif five >= 3:
                five -= 3
            else:
                return False
    return True
```

5. (10 points) Consider the problem of making change for n cents using the smallest number of coins. As- sume that each coin's value is an integer.

   Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

   Ans:

   Set cq = [n/25] //This is the largest number of quarters that can be used to make change for n cents.

   Set nq = n − 25cq //This is the amount remaining after using cq quarters

   Set cd = [nq/10] //largest number of dimes that can be used

   Set nd = nq − 10cd //amount remaining

   Set cn = [nd/5]

   Set cp = np = nd − 5cn

   Solution uses cq quarters, cd dimes, cn nickles and cp pennies.

Proof of optimality: Assume Greedy solution G is not optimal.

Let O be an optimal solution using oq quarters, od dimes, on nickels and op pennies.

First note that if op $>=$ 5 we can replace every 5 pennies with one nickle, reducing the number of coins used, so we can assume op $<$ 5.

If od 3 replace every three dimes with 1 quarter and 1 nickle without increasing the number of coins. So, we can assume od $<=$ 2.

If on $>=$ 2 replace every 2 nickels with one dime, reducing the number of coins used, so we can assume on 1.

Now suppose that

10od + 5on + op $>=$ 25

The only way that this can happen is if

od = 2 and on = 1

In this case we can replace the two dimes and one nickle with one quarter, reducing the number of coins used, contradicting optimality of O. So this is impossible.

This means that 10od + 5on + op $<$ 25

Since n = 25oq + 10od + 5on + op

we have just shown that $cq = [n/25] = oq$.

After greedy takes off the $cq = oq$ quarters what remains is

$n' = n - 25oq = 10od + 5on + op$

From the facts that on $<=$ 1 and op $<=$ 4 we see that 5on + op $<$ 10 so Greedy chooses cd $=$ od dimes and then cnon nickles and then cp $=$ op pennies, so we are done.