PACE UNIVERSITY SEIDENBERG SCHOOL OF CSIS

# CS608-SPRING2023: ALGORITHMS & COMPUTING THEORY

## Assignment#4 - TOTAL POINTS: 100

## DUE DATE: 04/20/2022 (April 20th)

## Team Assignment

| S.No. | Questions | Points | Self-Assessment |
|---|---|---|---|
| 1 | Given an array of integers “nums”, sort the array in ascending order and return it. You must solve the problem **without using any built-in functions** and using **merge sort** in O(nlog(n)) time complexity.<br>Example 1:<br>Input: nums = [5,2,3,1]<br>Output: [1,2,3,5]<br>**Explanation**: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).<br>Example 2:<br>Input: nums = [5,1,1,2,0,0]<br>Output: [0,0,1,1,2,5]<br>**Explanation**: Note that the values of nums are not necessarily unique. | **25** | The code implements the merge sort algorithm to sort a list of integers entered by the user. It recursively divides the list into two halves, sorts each half separately, and then merges them back together. The merge function combines the two sorted sub-arrays into a single sorted array. The time complexity of merge sort is O(nlogn).<br>points 25 |
| 2 | Assume you are an awesome friend and want to give your friends some cookies. But, you should give each friend at most one cookie.<br>Each friend ‘i’ has a greed factor g[i], which is the minimum size of a cookie that the friend will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the friend i, and the friend i will be content. Your goal is to maximize the number of your content friends and output the maximum number.<br>Example 1:<br>Input: g = [1,2,3], s = [1,1]<br>Output: 1<br>**Explanation**: You have 3 friends and 2 cookies. The greed factors of 3 friends are 1, 2, 3. And even though you have 2 cookies, since their size is both 1, you could only make the friend whose greed factor is 1 content. You need to output 1.<br>Example 2:<br>Input: g = [1,2], s = [1,2,3]<br>Output: 2 | **25** | The "Assign Cookies" problem, in which we must distribute cookies to kids whose greed factors and cookie sizes are supplied, is solved by the code. It then iterates through the two arrays, assigning cookies to each child whose greed factor can be satisfied by a cookie. The two arrays are sorted in ascending order. The sorting step causes the algorithm's time complexity to be O(nlogn).<br>points 25 |

| | | | |
|---|---|---|---|
| | **Explanation**: You have 2 friends and 3 cookies. The greed factors of 2 friends are 1, 2. You have 3 cookies, and their sizes are big enough to gratify all the friends, You need to output 2. | | |
| **3** | You have some apples and a basket that can carry up to 5000 units of weight. Given an integer array weight where weight[i] is the weight of the ‘i’$^{th}$ apple, return the maximum number of apples you can put in the basket.<br>Example 1:<br>Input: weight = [100,200,150,1000]<br>Output: 4<br>Explanation: All 4 apples can be carried by the basket since their sum of weights is 1450.<br>Example 2:<br>Input: weight = [900,950,800,1000,700,800]<br>Output: 5<br>**Explanation**: The sum of weights of the 6 apples exceeds 5000 so we choose any 5 of them. | **25** | The "Maximum Number of Apples" problem, which asks us to determine how many apples can fit in a basket with a 5000 unit weight limit, is solved using the code. It then iteratively adds apples to the basket until the maximum allowed weight is reachec by sorting the provided array of apple weights in ascending order. The sorting step causes the algorithm's time complexity to be O(nlogn).<br>points 25 |
| **4** | 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 $5.<br>Note that you do not have any change in hand at first.<br>Given an integer array bills where bills[i] is the bill the i$^{th}$ customer pays, return true if you can provide every customer with the correct change, or false otherwise.<br>Example 1:<br>Input: bills = [5,5,5,10,20]<br>Output: true<br>**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.<br>Example 2:<br>Input: bills = [5,5,10,10,20]<br>Output: false<br>**Explanation:** From the first two customers in order, we collect two $5 bills. For the next two customers in order, we collect a $10 bill and give back a $5 bill. For the last customer, we cannot give the change of $15 back because we only have two $10 bills. Since not every customer received the correct change, the answer is false. | **25** | A list of bills paid by consumers is input into the lemonade_change function, which then determines whether the correct change can be given to each customer who used a $5, $10, or $20 bill. It returns and records the $5 and $10 bills that are available as change. True if accurate change can be given to every consumer; otherwise, False.<br>Points 25 |

## Submission

- Submit a python **notebook**(of file type **.ipynb**) with comments above each code block/line explaining its purpose. Also submit **screenshot** of the result/output you get.
- You may not be graded full points if your program doesn't execute or produce intended results.
- Late submission up to one week after **due date** will incur **10% loss** of total points earned. 5% every week thereafter until end date.
- **Be careful not to share your code. You may lose points by sharing your work. Similarity scores will be checked.**
- Attach this file with self-assessment. This is for your reference if you answered question completely.