NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
Midterm (14%)
AY2018/19 Semester 4

CS2040 – Data Structures and Algorithms

12 July 2019 Time allowed: 90 minutes

INSTRUCTIONS TO CANDIDATES

1. Do **NOT** open the question paper until you are told to do so.
2. This question paper contains TWO (2) sections with sub-questions. Each section has a different length and different number of sub-questions. It comprises EIGHT (8) printed pages, including this page.
3. Answer all questions in this paper itself. You can use either pen or pencil. Write **legibly**!
4. This is an **Open Book Quiz**. You can check the lecture notes, tutorial files, problem set files, CP3 book, or any other books that you think will be useful. But remember that the more time that you spend flipping through your files implies that you have less time to actually answer the questions.
5. When this Quiz starts, **please immediately write your Matriculation Number** and **Tutorial Group**.
6. The total marks for this paper is **40**.

TUTORIAL GROUP

STUDENT NUMBER:

A								

For examiners' use only		
Question	***Max***	***Marks***
Q1-4	12	
Q5	10	
Q6	10	
Q7	8	
Total	**40**	

Section A – Analysis (12 Marks)

Prove (the statement is correct) or disprove (the statement is wrong) the following statements below. If you want to prove it, provide the proof or at least a convincing argument. If you want to disprove it, provide at least one counter example. 3 marks per each statement below (1 mark for circling true or false, 2 marks for explanation):

1. The time complexity of the following program when n >> 2*m is O(n*m). **[true/false]**

```
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  int n = sc.nextInt();
  int m = sc.nextInt();
  int result = 0;

  for (int i=0; i < n; i++) {
    for (int j=m; j > 0; j--)
      result += 1;
    m -= 1;
  }
  System.out.println(result);
}
```

Ans:
False, as it is not tight enough. When n >> 2*m, once m reached 0, the inner loop will no longer be executed, and will have run for O(m^2) time, while the outer loop will continue to run. Since n is not bounded by m, the actual time complexity is O(m^2+n).

2. Given the same input, selection sort will always have time complexity equal to or worse than optimized bubblesort. **[true/false]**

Ans:
True. Selection sort will run in time O(N^2) regardless of the input, however if input is already sorted, bubblesort will run in time O(N), since it will stop after the first pass.

3. For a queue of size N, in order to access any item in the queue, the worst time complexity is O(N), assuming you are only restricted to queue operations (poll(), offer(), peek()).
[true/false]

True.
Since to access the last item in the queue, you have to poll() every other item from the queue to reach it and this takes time O(N).

4. Given strings of length d, where $d > 3$, and containing only the characters {‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’}. If characters in each position are **uniformly** distributed among the digits (i.e if there are 100 strings of length 4, then 10 strings will have ‘0’ as the 1st character, 10 strings will have ‘1’ as the 1st character and so on …), then the time to insert each string into a hashtable of size 10^3 will take at least $O(d)$ time since you need to convert each string into an integer key (e.g using the *31 method) before inserting into the hashtable in order to distribute the strings evenly into the hashtable.
[true/false]

False.
If the characters in each position of the strings are distributed uniformly, then picking any 3 character and converting them to integer to be used as index for the hashtable will distribute the strings evenly in the hashtable. Thus e.g picking the 1st 3 characters of the string and converting to integer will suffice and will only take O(1) time.

Section B – Applications (28 Marks)

Write in pseudo-code. Any algorithm/data structure/data structure operation not taught in CS2040 must be described, there must be no black boxes. Partial marks will be awarded for correct answers not meeting the time complexity required.

5. **New linked list operation [10 marks]**

Given 2 sorted non-empty linked list a and b (containing integers), give an algorithm to merge them into 1 sorted linked list c (You don't need to preserve a and b after creating c). You may assume that this is a basic linked list, that you CANNOT make use of any of the linked list operations given in the lectures/Java API, and you HAVE access to the head reference of the two input lists. (It might be easier to write as close to java code as possible for this question).

Ans:

```
Merge(a,b)
{
  Create a new linkedlist c
  If (a.head.value <= b.head.value)
    c.head = a.head
  else
    c.head = b.head
  node prev = null
  node curMove = a.head
  node curStay = b.head
  while (curMove != null) {
    if (curMove.value <= curStay.value) { // move through
      prev = curMove                       // nodes < curStay
      curMove = curMove.next
    }
    else {
      if (prev != null)       // point prev.next to curStay as
        prev.next = curStay   // everything up to prev is < curStay
      node t = curMove        // Now swap curStay and curMove and
      curMove = curStay       // continue
      curStay = t
    }
  }
  prev.next = curStay // everything up to prev is < curStay
  return c
}
```

6. **Baby name search engine [10 marks]**

After having 5 children, Jeremy and his wife realized how important it is to be able to efficiently search for good and ***unique*** names for their children. To this end, Jeremy who has a list of N ($1{,}000{,}000 \leq N \leq 500{,}000{,}000$) names for babies registered by the social service administration from 1880 to 2018 wants to use it to create a search engine for baby names.

The information in the list of registered baby names that is used for the search engine is

I) The name of the baby where characters are only letters from the English alphabet and the maximum length of a baby names is 100 character. Note that there will be lots of repeated names in the list.

II) The gender of the baby – represented as 'M' or 'F' for male or female.

Jeremy knows you are taking CS2040 and wants your help to do this. As a start the search engine should implement the following operation:

```
List<String> search(int type, int p)
```

which will return a list of unique baby names from the list of N registered baby names, given the input parameters, which are as follows:

- The baby names must be of gender 'M' (**type** == 1), 'F' (**type** == 2) or both (**type** == 3)
- the baby names must only appear in $<= \boldsymbol{p}\%$ of the N registered baby names where $1 \leq p \leq 100$.

In order to be efficient each call to **search** must run in $O(M)$ time where M is the number of baby names returned.

a) Give the algorithm and appropriate data structure(s) for a preprocessing method **`preprocess()`** (this is a method that is called once before the search engine is used) which will process all the N baby names (which is not sorted) and store them in the mentioned data structure(s) so that later on each call to **`search`** can be done in $O(M)$ time. **`preprocess()`** must run in $O(N)$ time. [6 marks]

Ans:

1. Create a hashtable H which uses baby name as key and the pair <count, gender> as value, where count if the number of occurrence of the babyname and gender is the 1 or 2 where 1 indicates a male name and 2 indicates a female name.

2. Go through the N babynames and use each to hash into H. If a key value mapping already exists update the value.count by incrementing it by 1, otherwise insert a new mapping into the H with value.count = 1 and value.gender = 1 or 2 depending on the gender of the baby name. → O(N) time

3. Now create a 2D array of Arraylist of String of size 2x100 called A. i.e ArrayList<String> A[3][100]

4. Iterate through H. For each key value mapping <k,value> → O(N) time.
   - p = (value.count/N)*100
   - Add k to the back of the list at A[0][p] → index 0 is both M and F
   - Add k to the back of the list at A[value.gender][p] → index 1 is M, index 2 is F

b) Now give the algorithm for **`search`**. You may assume the arguments given to **`search`** is always valid. [4 marks]

Ans:

```
Search(int type, int $p$)
   List A
   if (type == 3)
     for i from 0 to p append A[0][i] to A → O(M)
   else
     for i from 0 to p append A[type-1][i] to A → O(M)
```

Each search is O(M) time.

7. **Missing family members [8 marks]**

The Addams family have just gone on a fishing trip and taken a photo. The tradition of the Addams family is to line up for their family photos, where each member of the family will wear a shirt having a number x where $1 \leq x \leq N$ (N being the number of members in the family). The oldest will wear shirt 1, the second oldest will wear shirt 2, and so on and the youngest will wear shirt N.

Now if there are 6 members in the family, they could line up as such: 1,2,4,5,6,3

After coming back from the trip, the Evve family who has been at odds with the Addams family has cast a spell on the photo and remove some of the members from the lineup in the photo. So if the original sequence is 1,2,4,5,6,3 after removing 2 and 5 from the photo it will result in the remaining subsequence 1,4,6,3.

In order to fix the photo, Michael Addams the patriarch of the family needs to know the exact sequence of the lineup in the photo. However, no one can quite remember the exact lineup except that if one were to order all permutations of 1 to N in ascending order of the permutation sequences (e.g 1,2,3 will have the permutations in ascending order as {1,2,3}, {1,3,2}, {2,1,3}, {2,3,1}, {3,1,2}, {3,2,1}), the original sequence in the photo will be the first such permutation that contains the remaining subsequence (sequence of the members remaining in the photo).

You are now tasked with finding the original lineup sequence given N and the remaining subsequence S (you can assume it is given in the form of a integer array). Give an algorithm that can do so in time $O(N)$.

Ans:

Since the original sequence is the 1st permutation in ascending order to include the remaining subsequence S, it also means that the missing subsequence S' in that permutation must be in ascending order (otherwise it cannot be the 1st permutation in ascending order to include S)! To piece back the original simply merge S and S' using merge method of merge sort!

read S into an array a of integers → O(N) time.

To create an array b containing S',

1. create a boolean array c of size N initialize to false. Go through a and for each integer i in a, set $c[i]$ = true.

2. go through c again. For each $c[i]$ that is false insert $c[i]$ into back of b

After step 2, b will contain the missing integers in ascending order which makes it S'. This takes O(N) time.

Now perform merging on a and b and this will give the original sequence. → O(N) time.

== END OF PAPER ==