Lab Worksheet 1

Question 1. Given that there is an unsorted array with length longer than 3, you want to sort an array such that it is neither non-increasing nor non-decreasing for any consecutive three elements in an array. For example, if you are given with 1,2,3,4, then you want to sort it as 1,3,2,4. Can you provide a $O(n)$ time algorithm for this problem?

Question 2. Computing the time complexity of recursion is a common problem in CS2040 midterm. However, some questions are challenging to compute the time complexity even though one knows the recursive equation. For instance, consider the recursion with $\mathrm{T}(n) = 3 \cdot \mathrm{T}(\frac{2}{3}n) + 1$.

(n-(n*2/3 ^ x))*3 ->3n , 9n + xn/2 + x/2

(a) Solve the above recursive formula and compute the time complexity.

In those cases, one powerful tool to compute the time complexity is to use Master's Theorem, which is given as follows:

(Master Theorem)

Let the recursive formula be $\mathrm{T}(n) = a \cdot \mathrm{T}(\frac{n}{b}) + f(n)$ where $f(n)$ is well-defined function. Then following holds:

(Case 1) If $f(n) \in O(n^{log_b a - \epsilon})$ for small $\epsilon \in \mathbb{R}_{>0}$, then $T(n) = \Theta(n^{log_b a})$.

(Case 2) If $f(n) \in \Theta(n^{log_b a} \times (log_2 n)^k)$ for some constant $k \geq 0$, then $T(n) = \Theta(n^{log_b a} \times (log_2 n)^{k+1})$.

(Case 3) If $f(n) \in \Omega(n^{log_b a + \epsilon})$ for small $\epsilon \in \mathbb{R}_{>0}$, and if $af(\frac{n}{b}) \leq cf(n)$ for some constant $c \in \mathbb{R}_{>0}$ where $0 < c < 1$, then $T(n) = \Theta(f(n))$.

Proof) Omitted. Out of the scope of this module (and in fact this theorem also).

(b) In which case of Master theorem does the above recursion belong to? Solve the recursion using Master's theorem. case 1, O(n^(log(1.5)(3))

(c) Solve following Recursions:

(1) $T(n) = 5T(n/5) + n^{1/3}$. O(n)

(2) $T(n) = 5T(n/3) + n^3$ O(n^3)

(3) $T(n) = 3T(n/3) + 7n\log_2 n$ O(n)

Remark) Master's theorem does not always work. Consider the recursion $T(n) = 25 \cdot T(\frac{n}{5}) + \frac{n^2(lg\ 5)^2}{(lg\ n)^2}$. Note that lg is a log with base 2.

(d) Does Master Theorem work for above example? If no, can you explain why it doesn't work for above recursion.

(e) Solve the above recurrence.

(f) In CS2040 midterm, there are some useful identities to remember.

(1) $\sum_{i=1}^{N} \frac{1}{i} \in O(logN)$

(2) $\sum_{i=1}^{N} i \in O(N^2)$

(3) $\sum_{i=1}^{N} i^2 \in O(N^3)$

(4) $\sum_{i=1}^{N} a \cdot r^i \in O(r^n)$

Question 3. Rank the following functions in increasing order of growth; that is, the function $f(n)$ is before function $g(n)$ in your list, if and only if $f(n) = O(g(n))$.

(a) $n^{n/2}$, (b) $(\frac{n}{2})^n$, (c) $(log\, n)^{log\, n}$, (d) $8^{log\, n}$, (e) $n^{3/4}$, (f) $n^3 - n^2$, (g) $2^{(log\, n)^2}$, (h) $n \cdot log\, n$.

e, h, c, d, g, f, a, b

Question 4 (CS2040S AY22/23 Sem 2 Midterm).

Given the following function:

```
void Func1(int arr[], int i, int j, boolean flag) {
  if (j <= 0)
    return;
  else {
    if (flag)
      // arr.length gives the size of the array
      for (int k = arr.length-i; k < i; k++)
        System.out.println(arr[k]);
    else
      for (int k = arr.length; k > i-1; k--)
        System.out.println(arr[k-1]);
    Func1(arr,i-1,j/2,true);
    Func1(arr,i-1,j/2,false);
  }
}
```

(a) Establish the recurrence relationship. T(j) = 2*T(j/2)+ 2T(j/2)*T(i)

(b) Solve the recurrence and compute time complexity. (j-j*1/2^n)*2 -> 2j, O(n*2^n)

Question 5 (Past Midterm).

Given an arraylist A containing N ($N \geq 1$) unsorted arraylists each containing up to M ($M \geq 1$) possibly repeated integer values, give an algorithm which takes in A and print out all integers that are common to all the N arraylists in ascending order. The integers can take values from 1 to 10,000,000, and each list may contain repeated integer values. You must ensure the time complexity of your algorithm is $O(M * N)$ in the worst case. For example, if A contains 3 unsorted lists with the following content:

initialise a m by n array and initialise with false: O(N)
each value in array n change the index of array to true: O(MxN)
check if column all true: O(n)

10, 13, 30, 100, 2, 5, 11, 2
2, 11, 13, 5, 13
13, 3, 2, 1, 7, 3, 4, 10, 11, 100, 30

the output from your algorithm should be 2,11,13.

Question 6. (Amortized Analysis, Queue & Stack)
Consider a standard (FIFO) queue that supports the following operations:
(1) PUSH(x): Add item x at the end of the queue.
(2) PULL(): Remove and return the first item present in the queue.
(3) RemoveAll(): Remove every elements in the queue.
(4) SIZE(): Return the number of elements present in the queue.

We can easily implement such a queue using singly-linked list so that PUSH, PULL, and SIZE operations take at most c1, c2, c3 worst-case time for some constants c1, c2, c3 > 0.
Now suppose that you are asked to implement this (FIFO) queue using 2 Stacks.

(a) Show that on average RemoveAll() Method takes O(1) time
(it means that if n number of operations are executed in total, then the time complexity of RemoveAll divided by n has O(1) time complexity).

(b) Provide a pseudo-code on how to implement queue with 2 stacks which allows the queue to maintain average O(1) time for every operation provided.