Mock Midterms

Section A — Analysis

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).

Question 1. The *tightest* time complexity of the following program is $O(n^2)$. **[True / False]**

```
void func(int n) {
    if (n <= 1) {
     System.out.println("CS2040");
     return;
 }
    func2(n/2);
    func2(n/2);
    func(n/2);
}

void func2(int n) {
    if (n <= 1) {
        System.out.println("CS2040");
        return;
    }
    func2(n/2);
    func2(n/2);
}

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

Question 2. The running time of the following modified quicksort algorithm is $O(n \log n)$. **[True / False]**

```
void QuickSort(int low, int high, int N) {
    if (high - low <= N / 2) {
        Insertion_Sort(low, high); // O(high - low)
    }
    else {
        int middle_idx = partition(low, high); // O(high - low)
        QuickSort(low, middle_idx);
        QuickSort(middle_idx + 1, high);
    }
}
```

Question 3. If a stack is implemented using a linked list, we can access any element within the stack in O(1) time. **[True / False]**

Question 4. In a near full hash table (number of empty spots << number of buckets) , all three probing methods (linear, quadratic and double hashing) may possibly probe forever. **[True / False]**

Section B — Applications

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.

Question 5a. Bubble Sort

Given an array of n **distinct** integers A, where $1 \leq A[i] \leq n$, design an algorithm to compute the number of passes required by the bubble sort algorithm (with early-termination) to sort the array in ascending order. *Hint: A good algorithm runs in O(n) time.*

For example, if $n = 5$ and $A = [1,2,3,4,5]$, then exactly 1 pass is needed. If $n = 5$ and $A = [5,4,3,2,1]$, then 4 passes are needed. If $n = 5$ and $A = [1,2,3,5,4]$, then 2 passes are needed.

Question 5b. More Bubble Sort

Given two integers n and k, design an algorithm that outputs an array of size n containing n distinct integers, such that when bubble sort is used to sort this array, exactly k swaps are performed. *Hint: A good algorithm runs in O(n) time.*

For example, if $n = 5$ and $k = 1$, then a valid array would be $[1,2,4,3,5]$, since exactly $k = 1$ swap is performed when bubble sort is used to sort this array.

Question 6. Most Frequent Element

You are given an array of n integers, where exactly one element appears more than $\frac{n}{2}$ times in the array. You wish to identify this element.

(i) Design an algorithm that solves this problem in $O(n)$ time, using $O(n)$ additional space.

(ii) Design an algorithm that solves this problem in $O(n)$ time, using $O(1)$ additional space.

Question 7. Meetings

You have n free time periods throughout the day, modelled as a list of time intervals with the format: `start_time end_time`. Your free time periods do not overlap with each other. All timings are given in military format (eg 0100 corresponds to 1am, 1500 = 3pm), no interval will stretch past midnight (0000), all values are in the range [0000, 2359], and we only consider hours and minutes. Your free time intervals will not overlap with each other.

You have a list of m meetings, given in the same format as your free timings. You can attend a meeting if and only if the meeting falls within a free time interval.

For example, consider the following:

Free timings = { {1200, 1400}, {1600, 1900} }

Meeting Timings = { {1200, 1400}, {1800, 2000} }

The first meeting is from 1200 to 1400. It fits entirely within the free time interval 1200, 1400. Therefore, you can attend it. The second meeting, however, cannot fit into your free time interval 1600, 1900 as it exceeds 1900 and you are not free after 1900.

Determine the maximum number of meetings that you can attend for the following scenarios.

Part 1 (Easier): The m meetings intervals do not overlap with each other.

Part 2 (Harder): The m meetings might possibly overlap with each other, and you are now free for the entire day (i.e. $n = 1$, and the free duration is [0000, 2359]).

— End of Paper —

Prepared by: Kevin Lim, Matthew Ng, Wang Zhi Jian.