The University of the South Pacific

School of IT, Engineering, Mathematics & Physics

CS214: Design & Analysis of Algorithms

Test I Semester 2, 2023

Time Allowed: 50 mins
Total Marks: 12 (10% of final grade)

Name: ________________________ **Student ID: ________________________**

Campus: ________________________

NOTE:

This test covers the two learning outcomes:

CLO – 1: Evaluate the efficiency of algorithms (5%)

CLO – 2: Assess the suitability of different algorithms/data structures for solving a given problem (5%)

INSTRUCTIONS:

- All questions are compulsory.
- All answers must be written in the spaces given in this booklet.
- You are not allowed to use the course material or internet search during the exam.

1. Suppose a new search engine QUIK has been built with the help of appropriate data structures. It gives fast results for search queries based on the popularity of the websites. Results are listed in the batch of 8 weblinks per page based on their ranking. What would be the best data structure for this scenario? Justify your answer. (1+2 marks)

(10 mins)

Fast retrieval & popularity are the priority here.

I would choose combination of heap/binary tree and priority queue. The main structure is heap and every node has priority queue. Heap nodes can be key words like "movie", "song", "religion", "news" etc. The list of words should be huge. The order may depend on popularity (no of weblinks?) ~~or simply alphabets a, b, c, etc.~~

heap helps it to be updated with new pages and retrieval would be fast with the order of log_2 (n).

Priority queue would release the pages based on priority.

This can also be arranged through Hash-table but it needs to be explained.

Through indexing by keep tracking of usage of words. The higher usage would result in higher priority.

Eg. Moive: 1million links
Song: 5 million links
News 2 million link
Religion 3 million links

```
                Song (5)
         News (2)        religion (3)
Movie (1)
```

2. Some algorithms can be written in iterative as well as recursive forms. Suppose algorithm **X** has been written using both styles to analyze some big data of *tweets* from popular international news media outlets. However, it was found that the iterative approach is somewhat slower than the recursive approach by an amateur algorithm tester. She has suggested using the iterative approach. The number of operations required by both approaches for different array sizes is listed below:

Size	**Iterative**	**Recursive**
1	100	2
2	110	4
3	120	8
4	130	16
…	…	…

Do you agree with the claim of the testers? Please explain. (1+2 marks)

(10 mins)

Yes, no – 1 mark.

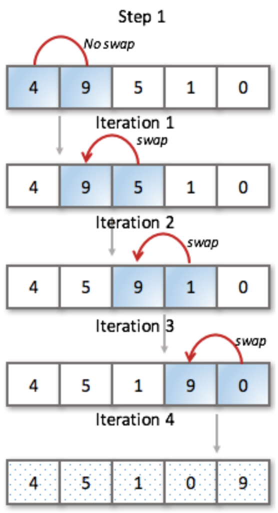
No, recursive is exponential and Iterative is linear. Since we are dealing with the big data, I would chose iterative..

3. A variation of the *bubble sort* algorithm and an example illustrating what it does are shown below. Using this algorithm answer the following questions:

```
static void bubbleSortSimple (int S[]){

    int n = S.length;
    int temp;

    for (int j = 1; j < n; j++) {
        for ( int i=0; i < n-1; i++ ) {
            if (S[i+1] < S[i]) {
                temp = S[ i ];
                S [ i ] = S [ i+1 ];
                S [ i+1 ] = temp;
            }
        }
    }
}
```



a) Evaluate the Big O order of the best and worst time complexity for this algorithm? Show all working. (2 marks)

b) Under what condition(s) you will attain the best and worst time complexities? (2 marks)

15 mins - answer both questions 3 (a and b) here. Use the last page if need more space.

This is a poorly written code. Best case for the given algorithm would be **any order** OR it will have everycase T(n): #1M

B(n) = n + n + … + n = n*n = n^2. #1M (with working otherwise 0)

Worst case is same as above: #1M

W(n) = n + n + … + n = n*n = n^2. #1M (with working otherwise 0)

4. The given bubble sort algorithm in Q.3 works for integer data only. Redesign this algorithm so that it can work for an array of any given user-defined data type. [Hint: Make use of built-in *Comparable* interface and its *int compareTo(Object x)* method] (2marks + 1 bonus)

(10 mins)

```
//Class MyArray{
public static void Sort(Object k[]) or (Comparable k[]) { //0.5 marks
    Comparable x[];
    if (!(k instanceof Comparable[])){ //1 mark bonus - only if k is Object[]
        throw new ClassCastException("Must implement Comparable");
    }
    x = (Comparable[])k; // if K is object

    int n = x.length + 1;
    boolean exchanges;
    Comparable temp;
    do {
        exchanges = false;
        for ( int i=0; i < n-1; i++ ) {
            if (x[i].compareTo(x[i+1]) > 0) { //1 marks
                temp = x[i];
                x [i] = x [i+1];
                x [i+1] = temp;
                exchanges = true;
            }
        }
    }while (exchanges); //0.5 marks if rest are as it is.
}
//}
```

Extra Sheet

END