Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Engineering, Mathematics & Science

School of Computer Science and Statistics

Integrated Computer Science Programme **Trinity Term 2018**
BA (Mod) Computer Science and Language
BA (Mod) Computer Science and Business
M.S.I.S.S
Year 2 Annual Examinations

CS2010: Algorithms and Data Structures

Thursday 3 May 2018 **RDS Main Hall** **09.30-12.30**

Dr. Ivana Dusparic, Dr. Vasileios Koutavas

Instructions to Candidates:

- This exam paper has TWO PARTS.
- Use SEPARATE answer books for each part.
- Answer TWO of the three questions in PART A.
- Answer TWO of the three questions in PART B.
- Each question is worth 25 marks.

Materials permitted for this examination:

- None.

PART A

Question 1 [25 marks]

(a) The following class implements a generic **Minimum Priority Queue** ADT.

```
public class MinPQ<Key extends Comparable<Key>>
{
  ...
  int size(){...} //returns the number of elements in the PQ
}
```

(i) Write the signatures of the methods implementing the remaining API of the Minimum Priority Queue and briefly describe what they do. *[4 marks]*

(ii) Using asymptotic notation, give the **worst-case running time** of each of these operations in the standard implementation of Priority Queue with a *Binary Heap*. For example, the worst-case running time of `size` in such an implementation is $\Theta(1)$. *[4 marks]*

(b) The following method inputs an array 'a' of N Integers and outputs the k-th largest integer.

```
Integer kthLargest(Integer[] a, int k)
{
  ...
}
```

(i) Write Java code that **efficiently implements** this method. Your code should use a minimum priority queue with the API from question (a) above. You can assume that $0 < k \leq N$. *[9 marks]*

(ii) Using the asymptotic notation, calculate a tight bound of the worst-case running time of this method. This bound should contain both 'N' and 'k'. You should explain your calculation. *[8 marks]*

Question 2 [25 marks]

(a) When studying the performance of algorithms we can use three types of asymptotic notation: O, Θ, Ω.

(i) Explain the difference between the three notations. Use examples in your explanation. *[6 marks]*

(ii) Describe situations where you would use each of the three notations. *[3 marks]*

(b) A **binary tree** has nodes which are objects of the following class:

```
class TreeNode
{
  String key;
  TreeNode left, right;
}
```

(i) Explain in English what is the **postorder** of the keys of a binary tree. *[8 marks]*

(ii) Implement the method

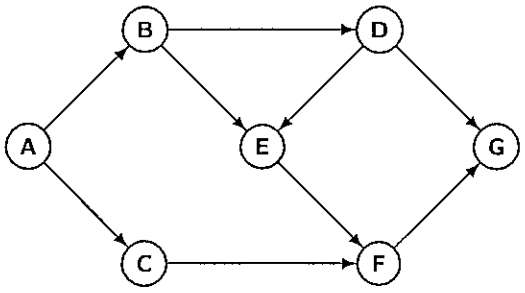
```
String printPostorder(TreeNode rt)
```

which traverses the binary tree with root `rt`, and returns the concatenation of all keys in the tree, in postorder. *[8 marks]*

Question 3 [25 marks]

(a) Explain in English the difference between the Depth-First Search (DFS) and the Breadth-First Search (BFS) traversals of a directed graph. *[7 marks]*

(b) Write the vertices of the following graph in the order that are marked as visited by DFS, starting from vertex **A**.



[4 marks]

(c) Write the vertices of the same graph in the order that are marked as visited by BFS, starting from vertex **A**. *[4 marks]*

(d) Explain in English the algorithm that calculates the **topological sort order** of the vertices of a directed graph. *[5 marks]*

(e) Write the vertices of the above graph in topological sort order. *[5 marks]*

PART B

Question 4 [25 marks]

(a) Explain the following approaches to algorithm design, giving an example of a well known algorithm for each. Explain why the algorithm you chose fits into that category, using pseudo code or diagrams to illustrate your explanations.

(i) Divide and Conquer

(ii) Dynamic Programming

[6 marks]

(b) You are given an array of integers and you need to output the array containing the same integers sorted in ascending order (from smallest to largest). Name two algorithms you can use to accomplish this. For each algorithm state the following:

(i) best-case running time

(ii) worst-case running time

(iii) additional space requirements

(iv) an example of when would using this algorithm be suitable

[8 marks]

(c) Consider the code for Insertion sort given below.

```
public void iterInsertSort(Comparable[] list) {
    int N = list.length;
    for(int i=1; i<N; i++) {
          for(int j=i; j>0; j--) {
                if(list[j].compareTo(list[j-1]) < 0) {
                  Comparable temp = list[j];
                  list[j] = list[j-1];
                  list[j-1] = temp;
                }
          }
    }
}
```

Assume the method is given as input the array $\{5, 4, 2, 5, 1\}$. Trace the execution of the Insertion sort, showing the state of i, j, and the array content after every execution of the inner **for** loop. *[6 marks]*

(d) Consider the two methods below, used as a part of recursive implementation of the Merge Sort algorithm.

```
public static void sort(Comparable[] a) {
  Comparable[] aux = new Comparable[a.length];
  sort(a, aux, 0, a.length-1);
}

private static void merge(Comparable[] a, Comparable[] aux,
  int lo, int mid, int hi) {

    for (int k = lo; k <= hi; k++) {
          aux[k] = a[k];
    }

    int i = lo, j = mid+1;
        for (int k = lo; k <= hi; k++) {
           if      (i > mid)                a[k] = aux[j++];
           else if (j > hi)                 a[k] = aux[i++];
           else if (less(aux[j], aux[i]))   a[k] = aux[j++];
           else                             a[k] = aux[i++];
        }
 }
```

Write the implementation of the missing method

```
public static void sort(Comparable[] a, Comparable[] aux,
   int lo, int hi)
```

[5 marks]

Question 5 [25 marks]

(a) List three substring search algorithms and for each one discuss the following:

(i) the number of character compares to search for a pattern of length M in a text of length N

(ii) the extra space required by the algorithm

(iii) an example of when using that algorithm would be preferable to using the other two

[6 marks]

(b) You are given a task to check whether a DNA sub-sequence ATATG is present in a string containing a large DNA sequence (consisting only of letters A, C, G, and T representing four nucleotides found in DNA). Construct the DFA (deterministic finite state automaton) for the search string, according to the Knuth-Morris-Pratt algorithm. For this DFA you should show its:

(i) internal array representation, by filling in the table below

counter j	0	1	2	3	4
char at position j	A	T	A	T	G
dfa[A][j]					
dfa[C][j]					
dfa[G][j]					
dfa[T][j]					

(ii) graphical representation, by adding state transitions to the following states

[10 marks]

(c) Provide the trace of sorting the array of strings given in the table below using LSD sort, providing an equivalent table for each of the 3 passes of the algorithm.

A	R	M
C	A	T
C	A	B
A	R	T
B	U	G
B	U	S

[4 marks]

(d) Consider the following method that implements the LSD (least significant digit) radix sort algorithm.

```
public static void sort(String[] a, int w) {
        int n = a.length;
        int R = 256;   // extend ASCII alphabet size
        String[] aux = new String[n];

        for (int d = w-1; d >= 0; d--) {
            //TODO
        }
}
```

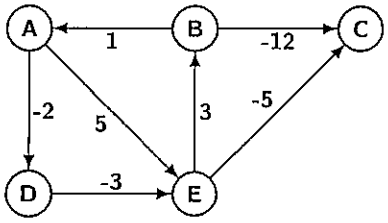
Write the missing code in the **for** loop marked with TODO.

[5 marks]

Question 6 [25 marks]

(a) Explain in English the Bellman-Ford algorithm to calculate the single-source shortest path in a graph. Specify the characteristics of the graphs in which the algorithm can be used. *[5 marks]*

(b) Run the Bellman-Ford shortest path algorithm for the graph given below, starting from vertex **A**. Fill in the table to contain the current values of the shortest path to each vertex in each iteration of the algorithm.



path to	A	B	C	D	E
iteration number 0	0	∞	∞	∞	∞
iteration number 1					
iteration number 2					
iteration number 3					
iteration number 4					

[10 marks]

(c) Can Dijkstra's algorithm be used to find the shortest path in the above graph? If yes, please detail the performance implications of using Dijkstra instead of Bellman-Ford. If not, explain why. *[5 marks]*

(d) Can the Floyd-Warshall algorithm be used to find the shortest path in the above graph? If yes, please detail the performance implications of using Floyd-Warshall instead of Bellman-Ford. If not, explain why. *[5 marks]*