FACULTY OF
SCIENCE & TECHNOLOGY

Faculty Name: Science and Technology
Department Title: Computer Science

Module: Algorithms and Complexity

Module Code: ECSC504

Date of Exam: 30/04/2015

Time Exam Starts: 10:00am

Time allowed for exam: 2 hours

Instructions for Candidates:

You need to answer ALL the questions.

Q1 [10 Marks]: Give the order of growth (as a function of *N*) of the running times of each of the following code fragments:

1.

```
int sum = 0;
for (int n = N; n > 0; n /= 2)
   for (int i = 0; i < n; i++)
      sum++;
```

2.

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for(int j = 0; j < i; j++)
      sum++;
```

3.

```
int sum = 0;
for (int i = 1; i < N; i *= 2)
   for (int j = 0; j < N; j++)
      sum++;
```

Answer:

1. linear (N + N/2 + N/4 + ...); [3 Marks]
2. linear (1 + 2 + 4 + 8 + ...); [3 Marks]
3. linearithmic (the outer loop loops lg N times). [4 Marks]

Q2 [8 Marks]: Develop, by completing the code of the program below, a brute-force based algorithmic solution for the 4-sum problem, i.e., given N integers, all quadruples (combinations of four integers) shall be identified the sum of which equals to zero. You may also use pseudo-code. Explain why this is a Brute-Force strategy.

```
public class FourSum {

    // print distinct 4-tuples (i, j, k, l) such that a[i] + a[j] + a[k] + a[l] = 0
    public static int printAll(long[] a) {
        int N = a.length;
        int cnt = 0;

        // Missing code

        }
        return cnt;
    }
```

Answer:

```
public class FourSum {

    // print distinct 4-tuples (i, j, k, l) such that a[i] + a[j] + a[k] + a[l] = 0
    public static int printAll(long[] a) {
        int N = a.length;
        int cnt = 0;
        for (int i = 0; i < N; i++) {                              [1 Mark]
            for (int j = i+1; j < N; j++) {                        [1 Mark]
                for (int k = j+1; k < N; k++) {                    [1 Mark]
                    for (int l = k+1; l < N; l++) {                [1 Mark]
                        if (a[i] + a[j] + a[k] + a[l] == 0) {      [1 Mark]
                            StdOut.println(a[i] + " " + a[j] + " " + a[k] + " " +
a[l]);
                        }
                    }
                }
            }
        }
        return cnt;
    }
```

Explanation: It takes into consideration all 4-combinations of integers resulting into a combinatorial explosion regardless the distribution of positive and/or negative integers and the fact that it makes no sense to scan through combinations of integers, if they are all, for instance, greater than 1. [3 Marks]

Q3 [9 Marks]: Suppose that an intermixed sequence of push and pop operations are performed on a *Stack*. The pushes push the integers 0 through 9 in ascending order of reading; the pops delete/remove the number and print out the return value. Which of the following printouts of sequence(s) of numbers could not occur? Justify why in terms of the stack contents.

```
(a)  4 3 2 1 0 9 8 7 6 5

(b)  4 6 8 7 5 3 2 9 0 1

(c)  2 5 6 7 4 8 9 3 1 0

(d)  4 3 2 1 0 5 6 7 8 9

(e)  1 2 3 4 5 6 9 8 7 0

(f)  0 4 6 5 3 8 1 7 2 9

(g)  1 4 7 9 8 6 5 3 0 2

(h)  2 1 4 3 6 5 8 7 9 0
```

Answer:

(b), (f), and (g). [3 Marks]

Justification: (b) – after 9 has been displayed/removed, the remaining numbers on the stack are {0:bottom, 1:top} and can only popped out as {… 9 1 0}, [2 Marks]

(f) - after 8 has been displayed/removed, the next number to be popped out cannot be 1 as there is number 2 on top of the stack, [2 Marks]

(g) – same as (b) where after having inserted all numbers (inclusion of 9), the remaining numbers on the stack are {2:top, 0:bottom}, therefore, they can only be displayed as {… 3 2 0}. [2 Marks]

Q4 [6 Marks]: Suppose that a client performs an intermixed sequence of (queue) *enqueue* and *dequeue* operations. The *enqueue* operations put the integers 0 through 9 in ascending reading order onto the queue; the dequeue operations delete/remove and print out the return value. Which of the following sequence(s) could not occur? Give an justification for your choices.

```
(a)  0 1 2 3 4 5 6 7 8 9

(b)  4 6 8 7 5 3 2 9 0 1

(c)  2 5 6 7 4 8 9 3 1 0

(d)  4 3 2 1 0 5 6 7 8 9
```

Answer:

(b), (c), and (d) [3 Marks]

Justification: The number to be displayed (dequeued) first will always have to be zero (0). [3 Marks]

Q5 [10 Marks]: Design and describe a data type and its behaviour (API interface) in plain English that implements the following operations, *all in constant time O(1)*: push, pop, min, for N integers. The third operation *min* will always deliver, at some time point t, the minimum from all those integers being on the data structure at the same time point t. Assume that the items are `Comparable`.

Answer:

- Maintain two stacks, one which contains all of the items and another which contains the minima. [2 Marks]
- To push an item, push it on the first stack; [2 Marks]
- If it is smaller than the topmost item on the second stack, push it on the second stack as well. [2 Marks]
- To pop an item, pop it from the first stack; if it is the top item on the second stack, pop it from the second stack as well. [2 Marks]
- To find the minimum, return the top item on the second stack. [2 Marks]

Q6 [6 Marks]: What is the maximum number of exchanges involving any particular item during selection sort? What is the average number of exchanges involving an item? Justify your answer.

Answer:

- The average number of exchanges is exactly 1 because there are exactly N exchanges and N items. [3 Marks]

- The maximum number of exchanges is N (worst case scenario, when the order is inversed). [3 Marks]

Q7 [12 Marks]: Complete the matrix below by adding rows showing how *insertion sort* sorts the array. The variable (i) stands for the index position on the array, in ascending order, the variable (j) indicates the position of the array element to be considered for the *InsertionSort* algorithm. Each row depicts the image of the array elements after each sorting step, with the last row depicting the image of the sorted array.

a[ ]

i	j	0	1	2	3	4	5	6	7	8	9	10	11
		E	A	S	Y	Q	U	E	S	T	I	O	N
0	0	E	A	S	Y	Q	U	E	S	T	I	O	N

Answer:

a[ ]

i	j	0	1	2	3	4	5	6	7	8	9	10	11
		E	A	S	Y	Q	U	E	S	T	I	O	N
0	0	E	A	S	Y	Q	U	E	S	T	I	O	N
1	0	A	E	S	Y	Q	U	E	S	T	I	O	N
2	2	A	E	S	Y	Q	U	E	S	T	I	O	N
3	3	A	E	S	Y	Q	U	E	S	T	I	O	N
4	2	A	E	Q	S	Y	U	E	S	T	I	O	N
5	4	A	E	Q	S	U	Y	E	S	T	I	O	N
6	2	A	E	E	Q	S	U	Y	S	T	I	O	N
7	5	A	E	E	Q	S	S	U	Y	T	I	O	N
8	6	A	E	E	Q	S	S	T	U	Y	I	O	N
9	3	A	E	E	I	Q	S	S	T	U	Y	O	N
10	4	A	E	E	I	O	Q	S	S	T	U	Y	N
11	4	A	E	E	I	N	O	Q	S	S	T	U	Y
		A	E	E	I	N	O	Q	S	S	T	U	Y

[1 Mark per correct row]

Q8 [6 Marks]: A clerk at a shipping company is charged with the task of rearranging a number of large crates in order of the time they are to be shipped out. Thus, the cost of compares is very low (just look at the labels) relative to the cost of exchanges (move the crates). The warehouse is nearly full: there is extra space sufficient to hold any one of the crates, but not two. Which sorting method should the clerk use? Justify your answer.

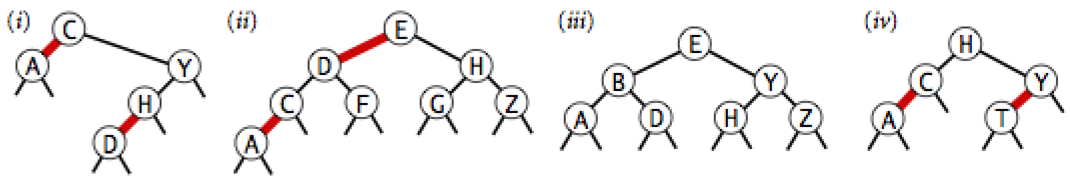
Answer:

Since there is space for holding only one crate and the cost of comparisons is low [2 marks], use selection sort [1 Mark] because it minimizes the number of exchanges. [1 Mark]

Q9 [8 Marks]: Which of the following are legally balanced red-black BSTs? Justify your answer.

Note: The red links are the bold edges A-C and D-H on (i), D-E and A-C on (ii), no red links on (iii), A-C and T-Y on

(iv)



Answer:

(iii) and (iv) only. [2 Marks]

(i) is not balanced, [3 Marks] (ii) is not in symmetric order or balanced. [3 Marks]

Q10 [6 Marks]: In contrast with the Minimum Spanning Tree with positive weights on edges, how would you find a *maximum spanning tree* of an edge-weighted graph? Suggest two different ways for your algorithmic design.

Answer:

Turn the weights of each edge to negative ones [3 Marks], or reverse the sense of comparison in the algorithmic design and implementation. [3 Marks]

Q11 [7 Marks]: Given a MST for an edge-weighted graph G, suppose that an edge in G, which does not disconnect G, is deleted. Describe how to find the MST of the new graph.

Answer:

- If the edge in not in the MST, then the old MST is an MST of the updated graph. [2 Marks]
- Otherwise, deleting the edge from the MST leaves two connected components. [2 Marks]
- Add the minimum weight edge with one vertex in each component. [3 Marks]

Q12 [12 Marks]: The percolation problem is defined as a *N*-by-*N* grid of sites, with each site being open with probability *p*, or blocked with probability 1 – *p*. Open sites in the grid may be coloured *white*, blocked ones may be coloured *black*. The system percolates if top and bottom are connected by open sites. If not, the system does not percolate. Suggest a data structure and model to represent the percolation system, as well as three possible algorithmic (problem) solution strategies for deciding whether there is a continuous flow of white sites in the grid, from top to bottom.

Answer:

- Taking a brute-force strategy would imply scanning all entries of the matrix repeatedly in order to mark the white and black boxes. Consequently, you may start exploring this N x N space by checking all neighboured boxes, horizontally and vertically, to form a continuous flow of white boxes from top to bottom. [3 Marks]

- Alternatively, you may take a smarter approach in that NOT ALL boxes need to be checked in regards with their colour. For instance, once a black box is hit to the left or right on a row other than the top or bottom row on the current path taken, there is no point of checking the contents of all boxes. [3 Marks]
- Finally, another approach would have been to randomly pick pairs of rows with high distributions of black boxes in an attempt to quickly reason that there are deadlocks found and, therefore, there is no point of exhaustively searching the whole NxN space. [3 Marks]

The data structure will a NxN matrix with cells occupied by 1s and 0s (white and black boxes, respectively), or a combination of other pair of values, e.g., true vs. false, red vs, green. [3 Marks]

END OF EXAM PAPER