4CCS1DST – Data Structures

Lecture 3 – Exercises

Exercise 1: asymptotic notation

$10\,n$ is: $O(n)$ $O(n^2)$ $O(\log n)$ $\Theta(n^2)$ $\Theta(n)$

$n/2 + 5 \log n$ is: $O(n)$ $O(n^2)$ $\Theta(n \log n)$ $\Theta(n^2)$ $\Theta(n)$

$7n^3 - 9n^2$ is: $O(n)$ $O(n^2)$ $\Omega(n^2)$ $O(\log n)$ $\Theta(n^2)$ $\Theta(n^3)$

Circle all correct answers.

Exercise 2

The following Java method determines whether the elements in a given range of array arr are all unique.

```
public static boolean isUniqueLoop(int[] arr, int start, int end) {
    for ( int i = start; i < end; i++ )
        for ( int j = i+1; j <= end; j++ )
            if ( arr[i] == arr[j] )
                return false;     // the same  element at locations i and j
    // all elements are unique
    return true;
}
```

What is the worst-case running time of this method, in terms of the number *n* of elements under consideration (*n* = end – start + 1) ?

Is there a better (faster) way to find out if all elements are unique?

Exercise 3

The following Java method determines whether three sets of integers, given in arrays a, b and c, have a common element.

```
public static boolean haveSameElement(int[] a, int[] b, int[] c) {
    for ( int i=0; i < a.length; i++ )
        for ( int j=0; j < b.length; j++ )
            for ( int k=0; k < c.length; k++ )
                if ( (a[i] == b[j]) && (b[j] == c[k]) )
                    return true;     // a common element found
    // no common element
    return false;
}
```

What is the worst-case running time of this method, if each array is of size *n* ?

Is there a better (faster) way to find out if the arrays have a common element?

Exercise 4

Design the following algorithm and implement it as a Java method

```
Algorithm countOnes($A$, $n$)
    Input two-dimensional $n$ x $n$ binary array $A$ (each entry is either 0 or 1)
    Output two-dimensional $n$ x $n$ array $S$, where $S[i][j]$ is the number of 1's
    in the "subarray" with the top-left corner at (0,0) and the bottom-right
    corner at ($i$, $j$).
```

Example. Input:

	0			***j***	
0	1	0	1	1	0
	0	1	1	1	0
i	1	0	1	0	1
	1	1	0	0	1
	0	0	1	1	0

Output:

				j	
	1	1	2	3	3
	1	2	4	6	6
i	2	3	6	8	9
	3	5	8	10	12
	3	5	9	12	14

What is the running time of your algorithm in terms of n?
Try to obtain the running time as low as you can.
The target running time is $\Theta(n^2)$.