Data structures, exercise 1, 25-29 January

Note that the deadline of the first TRAKLA2 question set is 31.1.

1. We know that a certain algorithm takes 0.5 ms to run for input size $n = 200$. How long does it take for input size 5000, when the running time of the algorithm is

   (a) $O(\log n)$
   (b) $O(n)$
   (c) $O(n \log n)$
   (d) $O(n^2)$
   (e) $O(2^n)$?

   In each case assume that the lower order terms can be neglected. For example, in part (d) assume that the running time is cn^2 for some constant c.

2. An algorithm takes 0.5 ms for input size 200, like in the previous problem, and we consider the same possible asymptotic running times (a)–(e). For each of these possibilities, what is the largest input size for which the algorithm runs in under one minute?

3. Which of the following claims are true, which not? Give a proof/motivation for your answer.

   (a) $2n^2 + 7n + 3 = O(n^3)$
   (b) $2n^3 + 7n + 3 = O(n^2)$
   (c) $\log n = O(10^6)$
   (d) $10^6 = O(1)$
   (e) $\log(n^2) = O(\log n)$

4. Sum $1 + 2 + 3 + \ldots + n$ can be calculated with the following algorithm:

```
sum = 0
for i = 1 to n
    sum = sum + i
```

   Define the running time complexity (the running time) and space complexity (the amount of extra memory) of the algorithm.

   Define an invariant for the algorithm and use that to show that algorithm works correctly.

5. The following algorithm evaluates the polynomial function $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_kx^k$, when the constant coefficients $a_0, \ldots, a_k$ are stored in an array $A[0\,..\,k]$:

Evaluate-Polynomial(A,x)

```
1  s = 0
2  for i = 0 to k
3      z = 1
4      for j = 1 to i
5          z = z * x
6      s = s+A[i] * z
7  return s
```

Simlate algorithm with function $p(x) = 2x^3 - 3x^2 - 7$ when $x = 2$

What is the running time of the algorithm? (Assume arithmetic operations take a constant time.) What is the loop invariant for the outer for loop?

6. From the following it is enough to do either (a) or (b).

(a) An array of size $n - 1$ contains all the integers from $1 \ldots n$ except one. A value can not be twice in the array.

Eg. the next array has all values from $1 \ldots 8$ except 6.

2	3	7	1	4	8	5

Define an algorithm that finds out what is the missing value. Your algorithm should have time complexity $\mathcal{O}(n)$? and space complexity $\mathcal{O}(1)$?.

(b) Define a datastructure that contains n values. Inside the data structures, values have an order:

1st	2nd	3rd	4th	5th	6th	7th
2	5	-1	3	0	2	8

Besides the ones that allow to enter the values inside to the data structure, there is only one operation: $subsum(a, b)$ that shows what is the sum of values stored in the data structure in places a, $a + 1$, $\ldots$, $b - 1$, b.

For example, in the above situation $subsum(3, 6)$ would be 4, since $-1 + 3 + 0 + 2 = 4$.

An easy solution would be to calculate the sumbsum always when the operation is called. The time complexity of the operation would then be $O(n)$.

Define your data structure so that the time complexity of operation $subsum(a, b)$ is only $O(1)$.