DATA STRUCTURES AND ALGORITHMS

SEMINAR 2

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș - Bolyai University
Computer Science and Mathematics Faculty

2018 - 2019

```
subalgorithm s1(n) is:
   for i ← 1, n execute
      j ← n
      while j ≠ 0 execute
         j ← [j/2]
      end-while
   end-for
end-subalgorithm
```

```
subalgorithm s2(n) is:
   for i ← 1, n execute
      j ← i
      while j ≠ 0 execute
         j ← [j/2]
      end-while
   end-for
end-subalgorithm
```

```
subalgorithm s3(x, n, a) is:
    found ← false
    for i ← 1, n execute
        if x_i = a then
            found ← true
        end-if
    end-for
end-subalgorithm
```

```
subalgorithm s4(x, n, a) is:
   found ← false
   i ← 1
   while found = false and i < n execute
      if x_i = a then
         found ← true
      end-if
      i ← i + 1
   end-while
end-subalgorithm
```

- x is an array, with elements $x_i \leq n$

```
subalgorithm s5(x, n) is:
   k← 0
   for i ← 1, n execute
      for j ← 1, x_i execute
         k ← k + x_j
      end-for
   end-for
end-subalgorithm
```

- Consider the following problems and find an algorithm (having the required time complexity) to solve them:
  - Given an arbitrary array with numbers $x_1 ... x_n$, determine whether there are 2 equal elements in the array. Show that this can be done with $\Theta(n \cdot log_2 n)$ time complexity.

- Consider the following problems and find an algorithm (having the required time complexity) to solve them:
    - Given an arbitrary array with numbers $x_1 ... x_n$, determine whether there are 2 equal elements in the array. Show that this can be done with $\Theta(n \cdot log_2 n)$ time complexity.
    - Given an arbitrary array with numbers $x_1 ... x_n$, determine whether there are two numbers whose sum is k (for some given k). Show that this can be done with $\Theta(n \cdot log_2 n)$ time complexity.

- Consider the following problems and find an algorithm (having the required time complexity) to solve them:
  - Given an arbitrary array with numbers $x_1 ... x_n$, determine whether there are 2 equal elements in the array. Show that this can be done with $\Theta(n \cdot log_2 n)$ time complexity.
  - Given an arbitrary array with numbers $x_1 ... x_n$, determine whether there are two numbers whose sum is k (for some given k). Show that this can be done with $\Theta(n \cdot log_2 n)$ time complexity.
  - Given an ordered array $x_1 ... x_n$, in which the elements are distinct integers, determine whether there is a position such that $A[i] = i$. Show that this can be done with $O(log_2 n)$ complexity.

```
subalgorithm s6(n) is:
   for i ← 1,n execute
      @elementary operation
   end-for
   i ← 1
   k ← true
   while i ≤ n - 1 and k execute
      j ← i
      k1 ← true
      while j ≤ n and k1 execute
         @ elementary operation (k1 can be modified)
         j ← j + 1
      end-while
      i ← i + 1
      @elementary operation (k can be modified)
   end-while
end-subalgorithm
```

```
subalgorithm p(x, l, r) is:
   if l < r then
      m ← [(l+r)/2]
      for i ← l, r-1, execute
         @elementary operation
      end-for
      for i ← 1,2 execute
         p(x, l, m)
      end-for
   end-if
end-subalgorithm
```

- Initial call: p(x, 1, n)

subalgorithm s7(n) **is:**
  s ← 0
  for i ← 1, n^2 **execute**
    j ← i
    while j ≠ 0 **execute**
      s ← s + j
      j ← j - 1
    end-while
  end-for
end-subalgorithm

subalgorithm s8(n) **is:**
    s $\leftarrow$ 0
    for i $\leftarrow$ 1, n^2 **execute**
        j $\leftarrow$ i
        while j $\neq$ 0 **execute**
            s $\leftarrow$ s + j - 10 * [j/10]
            j $\leftarrow$ [j/10]
        end-while
    end-for
end-subalgorithm