PRG – ETSInf – THEORY – Academic year 2017-18
Retake First Mid Term Exam – 19 June 2018 – Duration: 2 hours.

Note: Maximum mark for this exam is 10 points, but its weight in the final grade of PRG is **3 punts**.

1. 4 points Let `a` be an array of `double` whose elements are the values of the y-coordinate of a sequence of points belonging to a line with positive slope. In other words, the array is sorted in ascending order.

   You have to design and implement a **recursive** method for finding the x-intercept of the line, i.e. $y = 0$. The method must return the the index in the array where it is located the value of y equal to zero. Otherwise must return -1.

   For instance, if the contents of the array is {`-7.4, -1.3, 0.0, 1.8, 2.3, 3.6`} the method must return 2. But if the contents is {`-7.4, -1.3, 1.8, 2.6, 3.6`}, then the method must return -1.

   What To do:

   a) (0.75 points) Profile of the method with the formal parameters needed to solve the problem recursively. The preconditions affecting one or more parameters must be expressed also.

   b) (1.25 points) Description of the trivial and general cases.

   c) (1.50 points) Java code of the method.

   d) (0.50 points) Initial call for the method for applying the solution to a whole array. You can use a wrapper method.

Solution:

1. A possible solution is the following one, where the binary search strategy has been used:

```
/** Searches the value 0 within a[left..right].
 *  Precondition: 0 <= left, right < a.length,
 *  a[left..right] is sorted in ascending order.
 */
public static int intercepts(double[] a, int left, int right) {
    if (left > right) { return -1; }
    else {
        int middle = (left + right) / 2;
        if (a[middle] == 0) { return middle; }
        else if (a[middle] > 0) {
            return intercepts(a, left, middle - 1);
        } else {
            return intercepts(a, middle + 1, right); }
    }
}
```

   The initial call for solving the problem for a given array `a` must be `intercepts(a, 0, a.length - 1)`.

2. Another possible solution is the following method which haves a different profile and the strategy is the linear search:

```
/** Searches the value 0 within a[0..i]. Precondition: 0 <= i < a.length,
 *  a[0..i] is sorted in ascending order.
 */
public static int intercepts(double[] a, int i) {
    if (i <  0) { return -1; }
    else if (a[i] == 0) { return i; }
    else if (a[i] < 0) {return -1;}
    else {
        return intercepts(a, i - 1);
    }
}
```

The initial call for solving the problem for a given array `a` must be `intercepts(a, a.length - 1)`.

For the first method $T(n) \in \Omega(1) \cap O(\log n)$, and for the second one $T(n) \in \Omega(1) \cap O(n)$, where $n =$ `a.length` in both cases.

2. 3 points Let `m` be a matrix (2-dimensional array) of `char` and let `c` a `char`. The following method writes on standard output the words or sequences of characters stored in each row of the matrix, but omitting each appearance of `c`.

```
/** Precondition: m is an squared matrix (2-dimensional array). */
public static void ommitChar(char[][] m, char c) {
    int dim = m.length;
    for (int i = 0; i < dim; i++) {
        for (int j = 0; j < dim; j++) {
            if (m[i][j]) != c) {
                System.out.print(m[i][j]);
            }
        }
        System.out.println();
    }
}
```

For instance, if `m` = {{'e','e','l','e'}, {'m','e','m','e'}, {'n','u','l','l'}, {'c','a','s','e'}}, and `c` = 'e', then the output writes:

```
l
mm
null
cas
```

What To Do:

a) (0.25 points) Describe the input size of the problem and give an expression for it.

b) (0.50 points) Choose a critical instruction for using it as reference for counting program steps.

c) (0.75 points) Is the method sensible to different instances of the problem for the same input size? In other words, is the critical instruction repeated more or less times depending on the input data for the same input size?
If the answer is yes describe best and worst cases.

d) (1.00 points) Obtain an expression of the temporal cost function for each case if the answer to the previous question was yes and a unique expression if the answer was no.

e) (0.50 points) Use the asymptotic notation for expressing the behaviour of the temporal cost function for large enough values of the input size.

Solution:

a) $n =$ `m.length`

b) `j < dim` – It could be also used the comparison in the `if`: `m[i][j] != c`.

c) No, the temporal cost of the method doesn't depend on which values are stored in the matrix `m`.

d) $T(n) = 1 + \sum_{i=0}^{n-1}\left(1 + \sum_{j=0}^{n-1} 1\right) = 1 + \sum_{i=0}^{n-1}(1+n) = 1 + n + n^2$

e) $T(n) \in \Theta(n^2)$

3. 3 points The following recursive method checks whether $x > 1$ have no divisors in the range $[d, x[$, where d must be a value lower than or equal to x, i.e. $d \leq x$.

```
/** Precondition: (x > 1 && 1 < d <= x) */
public static boolean noDivisors(int x, int d) {
    if (x == d) { return true; }
    else {
        if (x % d == 0) { return false; }
        else { return noDivisors(x, d + 1); }
    }
}
```

What to do:

a) (0.25 points) Describe the input size of the problem and give an expression for it.

b) (0.25 points) Choose a critical instruction for using it as reference for counting program steps.

c) (0.50 points) Is the method sensible to different instances of the problem for the same input size? In other words, is the critical instruction repeated more or less times depending on the input data for the same input size?

If the answer is yes describe best and worst cases.

d) (1.50 points) Obtain an expression of the temporal cost function for each case if the answer to the previous question was yes and a unique expression if the answer was no. As in this case the method is a recursive one, you have to apply the substitution method.

e) (0.25 points) Use the asymptotic notation for expressing the behaviour of the temporal cost function for large enough values of the input size.

f) (0.25 points) If we run the method with this call `noDivisors(x,2)`, the method returns whether `x` is a prime number. What will be the asymptotic temporal cost of depending only on `x`?

Solution:

a) $n = x - d$

b) $x == d$

c) The strategy is a search for finding the first divisor of `x` greater than or equal to `d`. Best case, when d is a divisor of x. Worst case, when x is a prime number or has no divisors in $[d, x[$.

d) In the best case $T^b(n) = 1$.

In the worst case we have to take into account the trivial case and the general case:

$$T^w(n) = \begin{cases} 1 + T^w(n-1) & \texttt{if } n > 0\texttt{, i.e. } x > d \\ 1 & \texttt{if } n = 0\texttt{, i.e. } x = d \end{cases}$$

Applying the substitution method:

$T^w(n) = 1 + T^w(n-1) = 2 + T^w(n-2) = 3 + T^w(n-3) = \ldots = k + T^w(n-k) = \ldots = n + T^w(0) = n + 1$

e) $T^b(n) \in \Theta(1)$ and $T^w(n) \in \Theta(n) \Rightarrow T(n) \in \Omega(1) \cap O(n)$

f) In this particular case the input size is $n = x - 2$ and we can rewrite the asymptotic notation as $T(x) \in \Omega(1) \cap O(x)$