Dokuz Eylül University, Computer Engineering Department

CME-2001 Data Structures and Algorithms, 2016-2017 Fall Semester  
Assignment #1  
  
Design and code a recursive sorting algorithm which sorts positive integers (32 bit unsigned integers) in increasing order according to the following approach using Java:

Here is an example of 8-bit integers:  
Think about the integers 132, 14, 2 and 9.  
The binary representations of these integers are:

132 = 10000100 14 = 00001100 2 = 00000010 9 = 00001001

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7. bit | 6.bit | 5.bit | 4.bit | 3.bit | 2.bit | 1.bit | 0.bit |
| 132 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

In each call, the algorithm replaces the integers in the region specified by parameters “p” and “q” and depending on bit values indicated by the parameter “digit”. At that point, the algorithm places the integers whose current bit values are 0 to the left part of the region specified by “p” and “q”; places the integers whose current bit values are 1 to the right part. For that purpose, in a loop, the algorithm compares bit values of two integers in the pth and qth positions of the array. If necessary, these two numbers are shifted mutually and “p” and/or “q” are updated accordingly. If not necessary, updating “p” and/or “q” is sufficient. In the comparison of two bits, there are four possibilities: 0-0, 0-1, 1-0, 1-1. Updating “p” and/or “q” is performed depending on these four cases. An example is shown in the following table. Here, starting from the most significant bit (7. bit), the algorithm changes the positions of the integers as shown in the following table. The bold lines in the table indicates the regions processed by recursive calls.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 132 | 14 | 2 | 9 |
| Start: | 10000100 (132) | 00001110 (14) | 00000010 (2) | 00001001 (9) |
| After 7.bit placement | **0**0001001 (9) | **0**0001110 (14) | **0**0000010 (2) | 10000100 (132) |
| After 6.bit placement | 0**0**001001 (9) | 0**0**001110 (14) | 0**0**000010 (2) | 10000100 (132) |
| After 5.bit placement | 00**0**01001 (9) | 00**0**01110 (14) | 00**0**00010 (2) | 10000100 (132) |
| After 4.bit placement | 000**0**1001 (9) | 000**0**1110 (14) | 000**0**0010 (2) | 10000100 (132) |
| After 3.bit placement | 0000**0**010 (2) | 0000**1**110 (14) | 0000**1**001 (9) | 10000100 (132) |
| After 2.bit placement | 00000**0**10 (2) | 00001**0**01 (9) | 00001**1**10 (14) | 10000100 (132) |
| After 1.bit placement | 00000010 (2) | 00001001 (9) | 00001110 (14) | 10000100 (132) |
| After 0.bit placement | 00000010 (2) | 00001001 (9) | 00001110 (14) | 10000100 (132) |
| Resulting array: | 2 | 9 | 14 | 132 |

public void Our\_sort(UInt32[] array, int p, int q, int digit)

Here;

array = the array which will be sorted by Our\_sort function.

p = starting index  
q = last index

Initially, you should run the algorithm as Our\_sort (array, 0, array.Length-1, 32).

* Your implementation namespace must be named “DS”. The name of your class should be “Sort”. You only need to submit your class. Do NOT submit your entire Java project and Do NOT compress your class.
* Besides the “public void Our\_sort(UInt32[] array, int p, int q, int digit)” function, your “Sort” class must include the following attributes which store **your** name, surname and number:
  + public String strYour\_name = “Onur Çakırgöz”;
  + public long Your\_number = 201690006;

* Due date of this assignment is 04.11.2016 - 09:30 am.

Be sure that your definitions are in the correct format. Otherwise, your class cannot be tested by our testing program and you cannot get any points.

* Do not define your class and attribute as static.
* ***Cheating will not be tolerated!***