CS577 HW1

09/17/2015

Keith Funkhouser

Haruki Yamaguchi

**GCD 1**: invalid algorithm.

Counter example inputs: , where .

Assuming , the first call returns since it will fall into line 5. It leads to the infinite recursive calls since the value of and won’t change.

**GCD 2**: invalid algorithm.

Counter example inputs: , where .

Assuming , the first call returns . Then the second call returns . It leads to the infinite recursive calls. The same situation occurs with .

**GCD3**: invalid algorithm.

Counter example inputs: , where (the first two condition can be expressed as as well).

There are three conditions that this algorithm returns outputs. Any whose actual GCD does **not** fall into those outputs would get invalid result; e.g., .

1. If , it calls until becomes equal to (I.e.,), the output is at th call, where .
2. If , the next recursive call is , where. Since , it leads to , where . It eventually reaches to , after this branch of recursion is called times, and as a result, it outputs .
3. If and , it keep calls while . Assuming it turns afterth call, where and , then it calls . Since and from the second condition, the final output would be .

**GCD4**: correct algorithm.

Partial correctness

According to the lecture note from the class, “partial correctness of a recursive procedure follows if for every valid input ,

* All the recursive calls that appear in the code of the program on input have valid arguments.
* Assuming all those calls return a correct output for their respective arguments, and that the program terminates on input , the program returns a correct output on input .”

For the first bullet:

* There are two possible recursive calls. Firstly, if , the arguments will be (. Since and , both arguments are valid (positive integers).

Secondly, if , the arguments will be . Again, since , both arguments are valid (positive integers).

For the second bullet:

* An output is returned at three places. Firstly, if , the output is . It is correct since , according to the proposition 6 from the lecture note.

Secondly, if , the output is . It is also correct, according to the proposition 5 from the lecture note.

Lastly, if , the output is . This gets into the next recursion call , where and. Since , it invokes the next recursive call and so on while for th call. At th call, the recursive call reaches to , where and. Since , the output becomes , which is correct according to the proposition 5 from the lecture note.

Termination

There are two possible recursion call in GCD4; or . As stated in the second bullet of partial correctness proof, the latter call always reaches to . In addition, as stated in the first bullet, the inputs are always so that the inputs of both call , which is called when , and , which is eventually called when , remain as positive values.

In short, each call (or at the end of its subsequence calls) decreases or , but they still remain as a positive value. It means that the recursion calls eventually reach to the state where at some point, so that the recursion will terminate.