DSA Homework 1

B03902089 林良翰

1.1

(1)

The art of computer programming, written by Professor Donald Knuth, is a series of books that introduces computer programming with various mathematical concepts and logics, like basic arithmetic, mathematical induction, permutations and combinations, powers and logarithms, polynomials…and so on. Besides, it also includes introduction to data structures which is important to making efficient use with data, and many kinds of algorithms which are very essential to enhancing the qualities of programming. In conclusion, these books provide great information for mastering computer programming.

(2)

1 STRLEN(character array str)

2 i ← 0

3 **while** str[i] is not ‘end of string’ **do**

4 i ← i + 1

5 **end while**

6 **return** i

1. If the length of string is 1 :

Line 2 i = 0

Line 3 str[0] isn’t the end of string, go into while loop

Line 4 i = 1 (0 + 1)

Line 3 str[1] is the end of string, break from while loop

Line 6 i = 1 the length of string is 1

…correct

**2.** If the length of string is 2, continue from the case when i = 1 :

Line 3 str[1] isn’t the end of string, go into while loop again

Line 4 i = 2 (1 + 1)

Line 3 str[2] is the end of string, break from while loop

Line 6 i = 2 the length of string is 2

…correct

**3.** By “Mathematical Induction”, we can know that the cases when the length of string is 3, 4, 5, 6…are all correct because we have already proved the cases of length 1 and 2. Finally, we proved the algorithm is correct.

1.2

GCD-By-Reverse-Search(positive integer a, positive integer b)

1 for i ← min(a, b) to 1 do

2 if a mod i = 0 and b mod i = 0 then

3 return i

4 end if

5 end for

(1)

**1.** First we assume if a > b, b = 1 :

Line 1 min(a, b) = 1, the variable ’i’ runs from 1 to 1

Line 2 the condition statement in ‘if’ is true (a%1 = 0 & 1%1 = 0) go into if statement.

Line 3 i = 1, the gcd of a and b (a and 1) is 1

…correct

**2.** Again we assume if a > b, but b = 2 :

Line 1 min(a, b) = 2, the variable ‘i’ runs from 2 to 1, now i = 2

Line 2 case1: ‘if’ condition statement is correct (‘a’ is even), return the value in line3 (i = 2, the gcd of a and b is 2)

case2: ‘if’ condition statement isn’t correct (‘a’ is odd), ‘i’ changes to 1, again go into line2, and the ‘if’ condition statement is correct. Finally returns the value in line3 (i = 1, the gcd of a and b is 1)

…correct

**3.** By “Mathematical Induction”, we can know that the cases when

“a > b, b = 3, 4, 5, 6…”are all correct because we have already proved the cases of b = 1 and 2. Besides, we can assert that the answer is the greatest of all the common divisor because the variable ‘i’ runs from b to 0 (b must > 0). Moreover, we can even assume similar case ”b > a, a = 1, 2, 3…” and find it correct, too. Finally, we proved the algorithm is correct.

(2)

The minimum number of iteration is 1. This will happen when ‘b’ is the divisor of ‘a’.

1.3

(1)

Since ‘a’ and ‘b’ have a common divisor ‘l’, we can assert that the answer of gcd(a, b) will have a divisor ‘l’ and the rest of the divisor gcd(a/l, b/l). From this point, we can prove that “gcd(a, b) = l \* gcd(a/l, b/l)”.

(2)

□ should be 1. Because we can find that inside the function “GCD-By-Filter” call itself again, we are able to know that it is a recursive function. The answer will be returned and modified again and again by the recursive function until it is returned to the first function we called, and finally we get the right answer.

(3)

△ should be 2. Inside the recursive function, we can observe that the variable ‘i’ changed from small to big. Since we want to find all the common divisors, we have to check the numbers from 2.

1.4

1 GCD-By-Binary(positive integer a, positive integer b)

2 n ← min(a, b), m ← max(a, b), ans ← 1

3 **whil**e n ≠ 0 and m ≠ 0 **do**

4 **if** n is even and m is even then

5 ans ← ans × 2

6 n ← n/2

7 m ← m/2

8 **else if** n is even and m is odd then

9 n ← n/2

10 **else if** n is odd and m is even then

11 m ← m/2

12 **end if**

13 **if** n > m then

14 swap(n, m)

15 **end if**

16 m ← (m − n)

17 **end while**

18 **return** n × ans

(1)

a = 56, b = 14

init n = 14, m = 56, ans = 1

1st n = 7, m = 21, ans = 2 (line 4, line 15)

2nd n = 7, m = 14, ans = 2 (line 15)

3rd n = 7, m = 0, ans = 2 (line 10, line 15, line 18)

(2)

The while runs for finite times because ‘a’ and ‘b’ are finite numbers. Through while loop iterations, ‘a’ and ‘b’ will become smaller and closer to 1. At last, they will be turned to 1 and break the while loop, and that we can know that while loop will run for finite times.

(3)

Let a = mt, b = nt, gcd(m, n) = 1, m, n, t is integer

We can assert gcd(a, b) = t \* gcd(m, n) = t

Then, gcd((a - b), b) = gcd(t(m - n), tn) = t \* gcd((m - n), n) = t

So we prove that “gcd(a, b) = gcd((a - b), b)”

1.5

1 GCD-By-Euclid(positive integer a, positive integer b)

2 n ← min(a, b), m ← max(a, b)

**3 while** m mod n ≠ 0 **do**

4 tmp ← n

5 n ← m mod n

6 m ← tmp

**7 end while**

**8 return** n

(1)

a = 56, b = 14

init n = 14, m = 56

1st n = 14, m = 56, tmp = ?

(2)

T times.

1.6

(2)

GCD-By-Reverse-Search 11254 times

GCD-By-Filter 6491 times

GCD-By-Filter-Faster 6491 times

GCD-By-Binary 18 times

GCD-By-Euclid 8 times