DD2440 Advanced Algorithms

Homework A

Jim Holmström F-08 890503-7571

9 oktober 2011

1 Problem 1

$$\text{Given: } a, b \geq 0 \wedge \neg(a = b = 0) \tag{1}$$

$$\text{Definition of gcd: } \gcd(a, b) = c \iff \arg\max_c \{c|a \wedge c|b\} \tag{2}$$

1.1 Validate

In order from the top.

$$\texttt{if (b > a) return gcd(b, a)} \tag{3}$$

Sorts the arguments resulting in : $a \geq b$ and trivially holds from (2) with $a \leftrightarrow b$ and using the commutative property of "$\wedge$"

$$\texttt{else if (b == 0) return a} \tag{4}$$

Acts as base case. $i \neq 0$ from (1) $\gcd(i, 0) = i$ since $c = i$ is the largest $c : (c|i \wedge c|0)$

$$\texttt{else if (a and b are even) return 2*gcd(a/2, b/2)} \tag{5}$$

Since both a and b are even both must have at least the factor 2 in common. More generally:

$$\gcd(pm, pn) = p \cdot \gcd(m, n) \tag{6}$$

In our case $a = 2m$ and $b = 2n$

$$\texttt{else if (a is even) return gcd(a/2, b)} \tag{7}$$

Since we now that we have passed the above statement we have that b is odd thus 2 cannot be a factor, and we can thus remove the factor 2 from a without influencing the result.

$$\texttt{else if (b is even) return gcd(a, b/2)} \tag{8}$$

The same way as above but $a \leftrightarrow b$

```
else return gcd(b,a-b)
```

(3) $\Rightarrow a - b \geq 0$ and (4) $\Rightarrow b \neq 0$ and thus the parameters will at least be within the domain.

$\gcd(b, a-b) = \arg\max_c \{b|c \wedge (a-b)|c\} =$
$\{(a - \underbrace{b}_{b|c})|c \Rightarrow \{a \text{ must have a factor } c, c(a/c + b/c)|c\} \Rightarrow a|c\}$
$= \arg\max_x \{a|c \wedge b|c\} = \gcd(a, b)$ and thus the statement holds.

All the recursive calls holds, has arguments within the domain, the argumentsum is strictly smaller for all calls (except (3) but it can only be called once in a row) $\Rightarrow$ will always land in the basecase (4) and return the correct gcd. $\square$

1.2 Number of recursive calls

1.3 Bit complexity

2 Problem 2

$$N = 8905037571, a_1 = 123456789, a_2 = 987654321$$

Simple relations used:

$$c \geq 0 \Rightarrow \gcd(c+1, c) = 1 \tag{9}$$

$$c \cdot d + a \equiv a \ (mod\ d) \tag{10}$$

$$\text{Find a positive } x < N(N+1) : \begin{cases} x \equiv a_1 \ (mod\ N) \\ x \equiv a_2 \ (mod\ N+1) \end{cases} \tag{11}$$

Put$x$ as a smart linear combination of $a_{1:2}$. $x = a_1 b_1 (N+1) + a_2 b_2 N$

$$\begin{cases} x \equiv a_1b_1(N+1) + a_2b_2N & \equiv \{(10)\} \equiv a_1b_1(N+1) \ (mod\ N) \\ x \equiv a_1b_1(N+1) + a_2b_2N & \equiv \{(10)\} \equiv a_2b_2N \ (mod\ N+1) \end{cases} \tag{12}$$

(11) and (**??**) gives:

$$\begin{cases} a_1b_1(N+1) \equiv a_1 \ (mod\ N) \\ a_2b_2N \equiv a_2 \ (mod\ N+1) \end{cases} \Rightarrow \begin{cases} b_1(N+1) \equiv 1 \ (mod\ N) \\ b_2N \equiv 1 \ (mod\ N+1) \end{cases} \tag{13}$$

x satisfies (11) if $b_{1:2}$ satisfies (12)

$$b_1(N+1) \equiv b_1N + b_1 \equiv \{(10)\} \equiv b_1 \equiv 1 \ (mod\ N)$$

$$b_2N \equiv b_2(N+1) - b_2 \equiv \{(10)\} \equiv -b_2 \equiv 1 \ (mod\ N+1) \Rightarrow b_2 \equiv 1 \cdot (-1) \equiv N \ (mod\ N+1)$$

$$\begin{aligned} \therefore x &\equiv a_1(N+1) + a_2N^2 \equiv \{(\text{using pythons native big-integer support})\} \equiv \\ &\equiv \underline{71603982658724599629} \ (mod\ N(N+1)) \end{aligned}$$

The x found solves (11) and $x < N(N+1)$ $\square$

3 Problem 3

Note this has a unitcost RAM so you must check that it doesn't address more then 2^w amounth of RAM

From Fabian:

To achieve the linear time bound we use a radix sort that sorts using $\lceil \log_2(N+1) \rceil$ bits at a time (since we know that if $m = O(n^k)$,m consists of $O(k \cdot log_2 n)$ number of bits. When sorting $\lceil \log_2(n+1) \rceil$ bits at a time the largest number that occurs is $O(n)$, and we know that the number of elements is n. Then by the prevoius question we can do this sorting step in $O(n)$ time. We then proceed to sort the next $\lceil \log_2(n+1) \rceil$ bits, and repeat this $O(k)$ times.
Thus sorting the whole list takes $O(kn)$ time, though k is just some constant so we claim that the whole sorting algorigm take $O(n)$ time.

"The prevoius question"

4 Problem 4

5 Problem 5