FACTOR REFINEMENT OF A SEQUENCE OF POSITIVE INTEGERS.

JOHN CAMPBELL AND KEITH MATTHEWS

In the September 2002 issue of the College Mathematics Journal, Fred Howard [1] gave a proof of the Chinese Remainder Theorem, which depended on the following result:

Let $a_1, \ldots, a_n$ be positive integers. Then there exist positive integers $b_1, \ldots, b_n$ such that

(a) b_i divides a_i for $i = 1, \ldots, n$;
(b) $\gcd(b_i, b_j) = 1$ for $1 \leq i < j \leq n$;
(c) $b_1 \cdots b_n = \operatorname{lcm}(a_1, \ldots, a_n)$.

His construction depended on factorising the a_i into prime powers.

Instead, we give an algorithm for constructing a sequence $b_1, \ldots, b_n$ which only involves repeated taking of gcd's and divisions. First we deal with the case $n = 2$ and then apply this algorithm $\binom{n}{2}$ times for the general case. We present our algorithms as pseudocode.

The case $n = 2$. Consider the following procedure:

```
function refine(a1,a2)
input: positive integers a1,a2
output: positive integers j1,j2
        j1 divides a1, j2 divides a2
        gcd(j1,j2)=1, j1*j2=lcm(a1,a2)

d:= gcd(a1,a2)
if d = 1 {
    j1:= a1
    j2:= a2
}
else {
    j1:= a1/d
     h:= gcd(j1,a2)
    j2:= a2
    while h > 1 {
        j1:= j1*h
        j2:= j2/h
         h:= gcd(j1,j2)
    }
}
```

Explanation. If `a1` and `a2` are powers of the same prime: (`a1`,`a2`)$= (p^r, p^s), r > 0, s > 0$, at the conclusion of the `while` loop that,

$$(\mathtt{j1}, \mathtt{j2}) = \begin{cases} (p^r, 1) & \text{if } r > s, \\ (1, p^s) & \text{if } r \leq s. \end{cases}$$

This is easy to see if $r \leq s$, as $j1 = 1, h = 1$ and $j2 = p^s$ after entering the `else` section.

If $r > s$, we enter the `while` loop and eventually `j2` becomes 1. However the product `j1`*`j2`$= \mathrm{lcm}\,(p^r, p^s) = p^r$ is preserved and hence the final value of `j1` is p^r.

When we are performing the algorithm on an arbitrary pair (`a1`,`a2`), we can regard this as simultaneously performing it on all component prime power pairs.

Once a value 1 is reached in such a pair, the corresponding component (`j1`,`j2`) remains constant. The overall final (`j1`,`j2`) will then be the product of the component pairs.

We see that the final `j1` divides `a1`, `j2` divides `a2` and $\gcd(\mathtt{j1}, \mathtt{j2}) = 1$.

Finally the `while` loop preserves `j1`*`j2`$= \mathrm{lcm}\,(\mathtt{a1}, \mathtt{a2})$.

The general case $a_1, \ldots, a_n$. We can perform the above algorithm successively on all ordered pairs (a_i, a_j), $i < j$:

```
procedure refine_array(a[])
input: positive integers a[1],...,a[n], n>1
output: positive integers b[1],...,b[n]
        b[i] divides a[i] for i=1,...,n
        gcd(b[i],b[j])=1 if i < j
        b[1]*b[2]*...*b[n]=lcm(a[1],...,a[n])

for i = 1,...,n
  b[i]:= a[i]
for j = 2,...,n {
   for i = 1,...,n {
      if i < j {
         refine(b[i],b[j])
         b[i]:= j1
         b[j]:= j2
      }
   }
}
```

A straightforward induction on n shows that the final array `b[1],...,b[n]` will have the required property.

Examples.

1. $n = 2$: $(a_1, a_2) = (72, 108)$. Then the successive $(j1, j2)$ are $(2, 108), (4, 54), (8, 27)$. Also $8 \cdot 27 = 216 = \mathrm{lcm}\,(72, 108)$.
2. $n = 3$: $(a_1, a_2, a_3) = (72, 108, 150)$. Then the successive updated arrays are $(8, 27, 150), (8, 27, 75), (8, 27, 25)$. Also $8 \cdot 27 \cdot 25 = 5400 = \mathrm{lcm}\,(72, 108, 150)$.

Acknowledgement The second author is grateful for the hospitality provided by the School of Mathematical Sciences, ANU.

References

1. F.T. Howard, *A Generalized Chinese Remainder Theorem*, College Math. Journal **33** (2002) 279–282.

DEPARTMENT OF COMPUTER SCIENCE, AUSTRALIAN NATIONAL UNIVERSITY
E-mail address: John.Campbell@cs.anu.edu.au

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF QUEENSLAND, BRISBANE, AUSTRALIA, 4072
E-mail address: krm@maths.uq.edu.au