Алгоритм решения

После недолгого раздумья алгоритм кажется очевидным:

- найти НОД двух чисел, и разделить исходные числа на это НОД. Так, если, например, исходные числа 9 и 147, (НОД=3),то получается 3 и 49;
- затем применив третье правило, первое число умножим на семь, а второе разделим на 7. Получим 21 и 7.
- НОД равно 7, разделим числа на 7,
- поменяем числа игроков
- ответ 1, 3.

Несколько запутано.

Но можно заметить. Что решением должны быть два числа, первое из которых всегда равно 1, т.к. всегда первый игрок может свое число поделить само на себя, а второй умножить на это же число (второе правило игры).

Отметим еще одну особенность: если второе число содержит делитель точного квадрата некоторого числа d, то сначала, разделив второе число на d, а первое, умножив на d , а затем, поделив оба числа на d, то получаем новое правило, если одно из чисел содержит множитель равный квадрату целого числа, то это число можно просто поделить на этот квадрат.

Действительно.

1, b*a*a; (первое число умножить на a. второе разделить на a)

a, b*a; (оба числа разделить на a)

1, b;

Вот теперь очевидно, т.к. первое число решения должно быть всегда 1, то второе - произведение исходных чисел, деленное на все квадраты целых чисел, входящие в это произведение.

Таким образом, можно предложить следующий алгоритм решения задачи:

1. начало;
2. ввод чисел a и b.
3. Если a=b
4. тогда решение 1,1,
5. иначе
6. c:=a*b
7. перебрать все делители d от 2 до тех пор пока d^3 <=c
   пока c делится нацело на d^2 , нц

8. c:= c div d^2 кц все;
9. Вывод “1”,c
10. конец .

Так как возможны случаи, когда некоторый делитель будет входить в произведение более трех раз, то пришлось внутри цикла «Для» организовать еще один цикл «Пока». Но, если квадраты чисел перебирать с максимального значения (a*b), до минимального (2), то алгоритм упрощается. А время работы программы увеличивается

```
Var x,a,b.i :longint;
Begin
Readln(a,b);
X:=a*b; a:=1;
For i:=a*b downto  2 do
  if x mod (i*i)=0 then x:=x div (i*i);
Write(1,’ ‘,x);
End;
```

Однако работу программы можно ускорить, немного усложнив алгоритм.

```
uses CRT;
  var
      a,b,x,i,d        :longint;
begin
clrscr;
 readln(a,b);
 x:=a*b;
a:=1;
i:= trunc(sqrt(x))+1;
 repeat
    dec(i);
    if x mod (i*i)=0
      then   begin
                  x:= x div(i*i);
                  if i*i>x then i:=trunc(sqrt(x));
                end;
 until i<2;
 writeln(1, x:10);
end.
```

Какой же можно сделать вывод?

Как правило, при разработке алгоритма надо помнить, что самая хорошая программа – это компромисс между сложностью алгоритма и ее быстродействием. Чем сложнее алгоритм, чем больше он требует памяти, тем быстродействие программы выше и наоборот.