Sogang ACM-ICPC Team

Number Theory

BEFORE STUDY

- There's two points P1 = (x1,y1) , P2 = (x2,y2) on the grid plane. How many points in the segment P1P2 except P1 and P2?
- Constraints : -10^9 <= point <= 10^9
- Answer : GCD(|x1-x2|, |y1-y2|) - 1

BEFORE STUDY

- What does GCD stand for ?
- - Greatest Common Divisor
- GCD(710, 68) ?
- - 2
- x*144 + y*28 = multiples of [ ]

HOW TO SOLVE GCD FASTER ?

- Of course, you can solve with for(int i=0; i<n; i++) if(value%i ==0) ...
- But, best method is

Euclidean Algorithm !

EUCLIDEAN ALGORITHM

- GCD(1071, 1029)

  =GCD(1029, 1071 % 1029 = 42)

  = GCD(42, 1029 % 42 = 21)

  = GCD(21, 42 % 21 = 0)

  = 21

THEN ... LET'S CODE !

- `int gcd(int k,int l){return l?gcd(l,k%l):k;}`

TIME COMPLEXITY

- GCD(a,b) -> GCD(b,a%b) -> GCD(a%b,b%(a%b))
- When b > a/2,
  a%b = a-b < a/2

  When b < a/2,
  a%b = b < a/2
- First parameter become less than half per every 2 recursion.
- O(log max(a,b))

EXTENDED EUCLIDEAN ALGORITHM

- Find integer x, y which are satisfy
  $ax + by = GCD(a,b)$
- in indeterministic equation, $ax + by = c$ can have integer solution in the case that c is multiples of GCD(a,b)
- Example : $x*144 + y*28 = 4$ (8, 12, ...)

HOW TO SOLVE

- GCD (710,68)

710 = 68*10 + 30
68 = 30*2 + 8
30 = 8*3 + 6
8 = 6*1 + 2
6 = 2*3 + 0

: GCD is 2

Let 710 to a, 68 to b
Then ,

30 = a-b*10
8 = b-30 *2
= b-(a-b*10)*2
= -2a*21b
...
6 = 7a-73b
2 = -9a+94b

: integer solution is
(-9, 94)

THEN ... LET'S CODE !

```
int extgcd(int a, int b, int &x, int &y){
  int d = a;
  if(b != 0){
    d = extgcd(b, a%b, y, x);
    y -= (a/b) * x;
  } else {
    x = 1; y = 0;
  }
  return d;
}
```

BEFORE YOU STUDY

- Determine whether N is prime or not.
- Constraints : 1<= n <=10^9

HOW TO DETERMINE FASTER

- Suppose n has divisor d, then n/d is also divisor.
- Because n is equal to d*n/d
  and min(d, n/d)<=sqrt(n) ,
  retrieval from 2 to sqrt(n) is enough.

THEN ... LET'S CODE
bool is_prime(int n)

```
#include <cstdio>
bool is_prime(int n){
    int i;
    if(n==1) return false;
    for (i=2; i*i <= n; i++){
        if (n%i == 0) {
            return false;
        }
    }
    return true;
}

int main(){
    printf("%d\n",is_prime(1));
}
```

THEN… LET'S CODE vector<int> divisor(int n)

```
#include <vector>
#include <cstdio>
using namespace std;
vector<int> divisor(int n){
	vector<int> ans;
	for(int i=1;i*i<=n;i++){
		if(n%i==0){
			ans.push_back(i);
			if(i != n/i) ans.push_back(n/i);
		}
	}
	return ans;
}
int main(){
	int n;
	scanf("%d",&n);
	vector<int> test=divisor(n);
	for(int i=0;i<test.size();i++)
		printf("%d ",test[i]);
	puts("");
	return 0;
}
```

THEN ... LET'S CODE

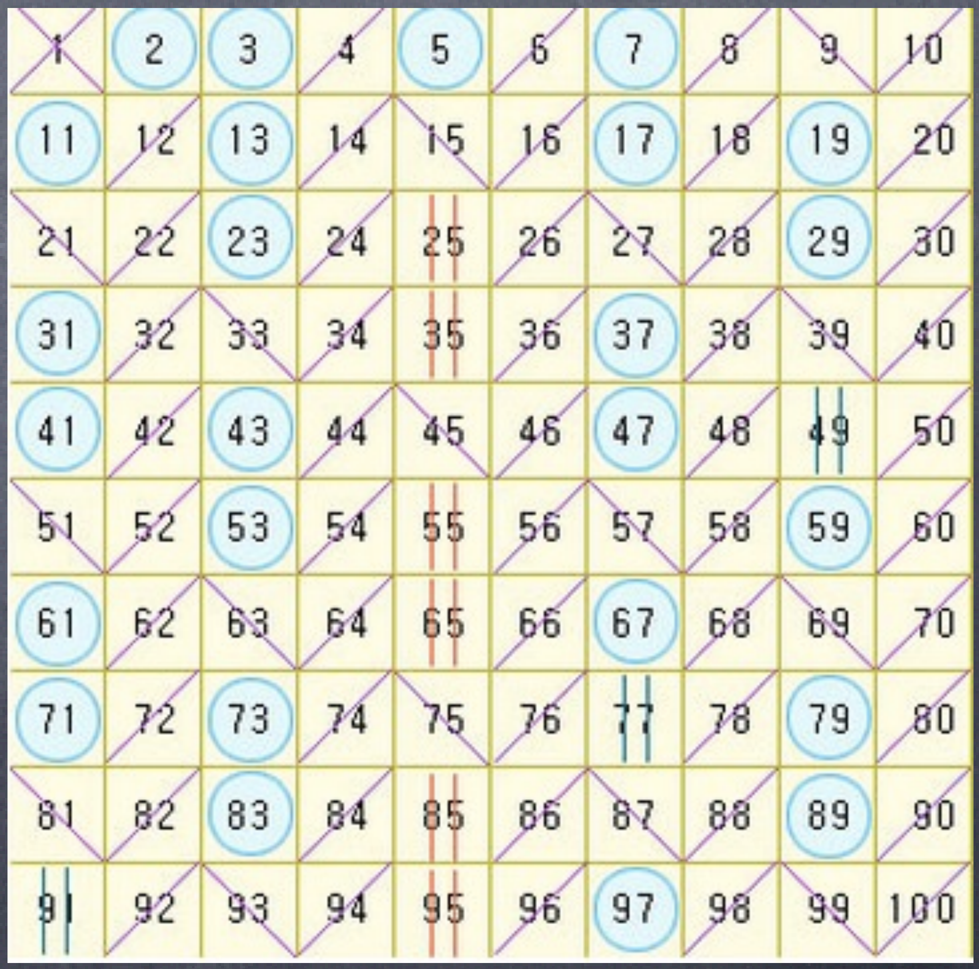
map<int,int> prime_factor(int n)

```
#include <map>
#include <cstdio>
using namespace std;
map<int,int> prime_factor(int n){
	map<int,int> ans;
	for(int i=2;i*i<=n;i++)
		while(n%i==0){
			n/=i;
			ans[i]++;
		}
	if(n != 1) ans[n]++;
	return ans;
}

int main(){
	map<int,int> test=prime_factor(48);
	for(map<int,int>::iterator it=test.begin();it!=test.end();it++){
		printf("%8d : %8d times\n",it->first,it->second);
	}
	return 0;
}
```

Eratosthenes' sieve

- If we have to determine just 1 integer or small set, O(sqrt(n)) algorithm will do.
- But what if
“ Find all primes from 1 to 10^6 “
.... OMG ....



LET'S CODE !

```
#include <cstdio>
#include <cstring>
#define n 111111
bool ck[111111];
void era() {
    memset(ck,true,n);
    ck[1] = false;
    for ( int i = 2 ; i*i <= n ; i++ ) {
        if ( ck[i] == true )
            for ( int j = i*i; j <= n ; j+=i )
                ck[j] = false;
    }
}
```

MODULO

- When result is bigger than 64bit, the problem usually say
  "if the answer is bigger than N, modulo with M"
- This can help get rid of unfairness.
- In JAVA language, Biginteger in it.

BASIC NOTATION

- a mod m = a%m (not exactly same. see appendix)
- $a \equiv b \pmod{m}$: a%m = b%m

PROPERTY

- $a+b \equiv c+d \pmod{m}$
- $a-b \equiv c-d \pmod{m}$
- $a*b \equiv c*d \pmod{m}$
- Exception in division,
  $2 \equiv 8 \pmod{6}$ ==> O
  $2/2 \equiv 8/2 \pmod{6}$ ==> X

THINK BEFORE LEARN

- < UVa NO.10006 >
- Arbitrary integer x between 1 and n.

  If n satisfies x^n≡x(mod n), we call x Carmichael number.

  Determine Carmichael number or not about Given n.
- Constraints : 2 < n < 65000

EASY SOLUTION

- Is 6 Carmichael number?
  Let's see ...
  $2^6 \equiv 2 \pmod{6}$ ... NO~
  $3^6 \equiv 3 \pmod{6}$ ... YES!
  $4^6 \equiv 4 \pmod{6}$ ... YES!
  $5^6 \equiv 5 \pmod{6}$ ... NO~
  SO, 6 is not.
- Total search algorithm is work for small number with $O(n^2)$ time complexity.

THINK FURTHER

- In case $n = 2^k$
  $x^n = (((x^2)^2) \dots)$
- we can represent any number n to addition of 2^k form, such as
  $n = 2^{k1} + 2^{k2} + 2^{k3} \dots$
- then $x^n = x^{2^{k1}} + x^{2^{k2}} + x^{2^{k3}} \dots$
- Example : $x^{22} = x^{16} * x^4 \wedge x^2$
  (22 = 0b10110)

THEN ... LET'S CODE !

```
typedef long long ll;
#define MOD 1234

ll mod_pow(ll x, ll n){
    ll res = 1;
    while(n>0){
        if(n & 1) res = res*x % MOD;
        x = x*x%MOD;
        n >>= 1;
    }
}
```

APPENDIX

- < diffrence between MOD & REM >

1) MOD(x,0) =x , whereas REM(x,0)=NaN
2) MOD(x,y) has the sign of y,
   while REM(x,y) has the sign of x
3) MOD and REM are equal
   if x and y have the same sign.

- ceiling function : minimum integer that larger or equal to contained value.

  $\lceil n/m \rceil = (n+m-1)/m$

  EX) $\lceil 5/3 \rceil = (5+3-1)/3 = 7/3 = 2$

Q & A

- Any question ?