Lecture : Modular Arithmetic

Agenda

- Revision
- Problems
  - A % m = B % m
  - Count pairs
  - Rearrange array
- Inverse modulo and fermat little theorem

Revision

$\% \Rightarrow$ remainder $[\ 17 \% 5 = 2\ ]$

$a \% m \longrightarrow$ min value $= 0$ Range $= [0, m-1]$

max value $= m-1.$

Modular Arithmetic

$[\ +, -, *, /\ ]$

1. $(a+b)\% m = [a\% m + b\% m]\% m.$

2. $(a*b)\% m = [a\% m * b\% m]\% m.$

3. $(a-b)\% m = [a\% m - b\% m]\% m.$

Eg: $a = 17$ $b = 8$ $m = 5.$

LHS: $(a-b)\% m = (17-8)\% 5$

$= 9\% 5 = 4.$

RHS: $[a\% m - b\% m]\% m = [17\% 5 - 8\% 5]\% 5$

$= [2 - 3]\% 5$

$= -1\% 5$

$\downarrow$

$[-1 + 5]\% 5 = \text{LHS}$

$\uparrow$

add.

$$(a-b)\% m = [a\% m - b\% m + m]\% m$$

4. $a \% m = (((a \% m) \% m) \% m) \% m$ - ....

5. $a^b \% m = [(a \% m)^b] \% m.$

Qu1 Given 2 numbers A and B [A > B]

find count of possible M such that

$A \% m = B \% m$ [m > 1]

Eg: a = 13 b = 7

m = 2 , 3 , 6

13 % 2 = 1 7 % 2 = 1

13 % 3 = 1 7 % 3 = 1

13 % 6 = 1 7 % 6 = 1

ans = 3

Brute force:

```
int countMs( int a , int b) {
                     ↑       ↑
                    max     min
  int cnt = 0;
  for( i=2;  i <= a;  i++) {
          if( a % i == b % i) {
                cnt++;
          }
  }
  return cnt;
}
```

TC: O(A)

SC: O(1)

Approach2:

$a \% m = b \% m.$

$a \% m - b \% m = 0$

Add m on both sides

$a \% m - b \% m + m = m.$

Take % m on both sides

$(a \% m - b \% m + m) \% m = m \% m$

$(a - b) \% m = 0$

m is a multiple of a - b

cnt = [count factors of a - b]

return cnt - 1;

because m > 1.

```
int countMs( int a, int b) {

    int factors = countFactors(a-b);

    return factors -1;

}
```

TC: $O(\sqrt{n})$

SC: $O(1)$

Qu2 Given arr[n], find count of pairs such that arr[i] + arr[j] is divisible by m.

constraints: i != j

m <= n. [size of array]

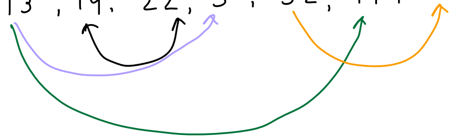
Eg:

	0	1	2	3	4	5
arr[] =	4	7	6	5	5	3

m = 5   pairs ⇒ (0, 2) (1, 5) (3, 4)   ans = 3

arr[] = [13, 14, 22, 3, 32, 19, 16]

m = 4



ans = 4

Brute force:

```
for( i=0; i<n; i++) {
    for(j = i+1; j<n; j++) {
        if( arr[i] + arr[j] % m ==0) {
            cnt ++;
        }
    }
}
return cnt;
```

TC: $O(n^2)$

SC: $O(1)$

Approach2:

$[17 + \underline{x}] \% 4 = 0$

$\% 4$ → 1

x → 3, 7, 11, 15, 19, 23, ⋮ → $\% 4 = 3$

4 ← m.

$sum = 17 \% 4 + x \% 4 = 4$

$(arr[i] + arr[j]) \% m = 0$

$[arr[i] \% M + arr[j] \% m] \% m = 0$

0 ⟋ ⟍ m.

$m = 8$

$arr[i] \% 8 = 0$
&
$arr[j] \% 8 = 0$

$arr[i] \% 8 = 1$ & $arr[j] \% 8 = 7$

$arr[i] \% 8 = 2$ & $arr[j] \% 8 = 6$

$arr[i] \% 8 = 3$ & $arr[j] \% 8 = 5$

$arr[i] \% 8 = 4$ & $arr[j] \% 8 = 4$

$arr[] = [13, 14, 22, 3, 32, 19, 16, 12, 34]$, $m = 4$

	0	1	2	3	4	5	6	7	8
arr[] =	13	14	22	3	32	19	16	12	34
	↓% 4	↓% 4	↓% 4	↓% 4	↓	↓	↓	↓	↓
	1	2	2	3	0	3	0	0	2

map:

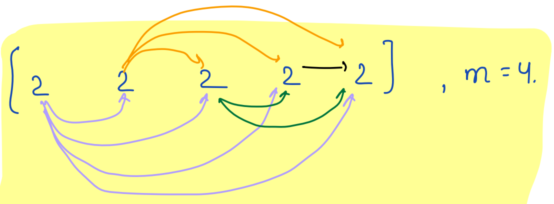
key [rem]	value [cnt]
0	3
1	1
2	3
3	2

m = 4:

arr[i] % m   arr[j] % m

1 ↑ cnt1   3 ↑ cnt2   ⇒ cnt1 * cnt2 ⇐ pairs

2 ↑ cnt1   2 ↑ cnt1   ⇒ $[cnt1 * (cnt1 - 1)] / 2$



, m = 4.

$5_{C_2} = \frac{5 * 4}{2} = 10$

0 ↑ cnt1   0 ↑ cnt1 $[cnt1 * (cnt1 - 1)] / 2$

---

m = 6

Rem: 0
1
2
3
4
5

Pairs

1 [cnt1]   5 [cnt2] ⟶ cnt1 * cnt2

2   4 ⟶

3   3

0 ↑ cnt1   0 ↑ cnt1 ⟶ $cnt1_{C_2} = (cnt1 * cnt1-1)/2$

```
int  countPairs(int[] arr, int m) {

    Map<Integer, Integer> freq;

    for( int  val : arr) {

        int rem = val % m;

        if( freq.containskey(rem)) {
            int  prev = freq.get(rem);

            freq.put( rem,  prev+1);

        } else{
            freq.put(rem, 1);

        }
    }

    // If  rem = 0 { arr[i] % m=0 && arr[j] % m==0}

    int  cnt = [freq.get(0) * [freq.get(0) - 1]] / 2;


    int  l = 1;

    int  r = m-1;

    while( l < r) {

        cnt += freq.get(l) * freq.get(r);

        l++;
        r--;
    }
    if (l==r) {

        cnt+= [freq.get(l) * [freq.get(l) -1]] / 2;


    return cnt;
}
```

TC: O(n)
SC: O(n)

Handle the case when rem is not there in hashmap.

Qu 3 Rearrange the array.

Given arr[n]

$0 \leq arr[i] < n$

Rearrange: $arr[i] = arr[arr[i]]$

Example:

	0	1	2	3	4
arr[] =	3	2	4	1	0
updated =	1	4	0	2	3

	0	1	2	3	4	5	6
arr[] =	3	1	4	6	5	0	2
updated:	6	1	5	2	0	3	4

Brute force: Take an extra array, do updation there.

newArr[i] = arr[arr[i]].

TC: O(n)

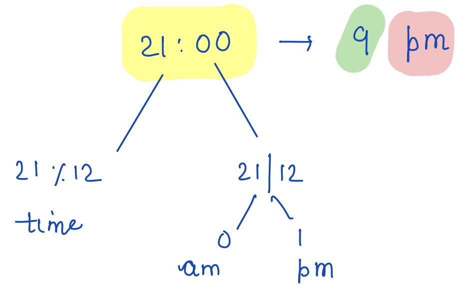
SC: O(n)

Approach2: TC: O(n) SC: O(1)

12 hrs format:

9:00 [am | pm]

24 hrs format



dividend = divisor * quotient + remainder

21 12 1 9

which will help me to store time and morning/evening

Eg: arr[] = [

0	1	2	3	4
~~3~~	~~2~~	~~4~~	~~1~~	~~0~~]
~~1~~	4	2	3	1 → data was lost

swapped ans: 0

arr[i] = arr[arr[i]]

actual no: [1 4 0 2 3]

challenge: have new value | old value at same time.

dividend = divisor * quotient + remainder

dividend → try to make

- old value at that idx
- updated value

divisor → arr.length

Algo:

0 1 2 3 4

arr[] = [3 2 4 1 0] divisor = 5

*5 *5 *5 *5 *5

15 10 20 5 0

old values: [15|5] 10|5 20|5 5|5 0|5

dividend = divisor * quotient + remainder

dividend → try to make → old value at that idx / updated value

divisor → arr.length

actual no. [1 4 0 2 3]

15|5 → updation

old value

15 + getvalue (15|5) th idx

15 + getvalue (3rd) idx

15 + 5 [updated value]

↓ old value

5|5 = 1

15 + 1 = 16

16

old value: $\frac{16}{5} = 3$

expected value: 16 % 5 = 1

10|5 →

10 + getvalue ($\frac{10}{5}$) th idx

10 + getvalue (2nd) idx

10 + 20 [updated value]

↓ old value

10 + $\frac{20}{5}$ = 4.

14

old val: $\frac{14}{5} = 2$

new value: 14 % 5 = 4

```
void  rearrange (int[] arr) {

    for(i=0; i<n; i++) {

        arr[i] = arr[i] * n;
                   ↑        ↑
                quotient  divisor
               [old value]
    }

    for(i=0; i<n; i++) {

        int idx = arr[i] / n;

        int old-value = arr[idx] / n;

        arr [i] += old-value;
                   rem
    }

    for(i=0; i<n; i++){

        arr[i] = arr[i] % n;
    }

}
```

TC: O(n)

SC: O(1)

* Inverse modulo

$$\left(\frac{a}{b}\right) \% m = \left(\frac{a \% m}{b \% m}\right) \% m$$ [wrong]

can be 0

eg: $a = 5$ $b = 7$, $m = 7$ $\left[\left(\frac{5 \% 7}{7 \% 7}\right) \% 7\right] = \frac{5}{0} \% 7$

$$(a | b) \% m = (a * b^{-1}) \% m$$

$$= [a \% m * b^{-1} \% m] \% m$$

Inverse Modulo

Congruency: x and y are said to be congruent w.r.t n if -

$$x \% n = y \% n$$

$\updownarrow$

$$x \equiv y \ (\text{mod } n)$$

fermat little theorem { Bonus }

a , $p \rightarrow$ prime no. , $a < p$

$$a^{p-1} \% p = 1$$

$$a^{p-1} \% p = 1 \% p$$

$$a^{p-1} \equiv 1 \ (\text{mod } p)$$

$$a^{p-1} * a^{-1} \equiv a^{-1} * 1 \ (\text{mod } p)$$

$$a^{p-2} \equiv a^{-1} \text{ mod } p$$

$$a^{p-2} \% p = a^{-1} \% p$$

[if m is non-prime: Extended Euclid algo]

$$= [a \% m * b^{-1} \% m] \% m$$

$(b^{m-2} \% m)$ if m is prime.

friday : 2nd June

Contest → Arrays

Bit Manipulation } h/w assignments.

Thankyou 🙂

Doubt

$(x + y) \% \underset{8}{m} = 0$

$x+y <$
- $0 \; [x=0 \;\&\; y=0]$
- 8: $x=1 \;\; y=7$; $x=2 \;\; y=6$; $x=3 \;\; y=5$; $x=4 \;\; y=4$

$(arr[i] + arr[j]) \% m = 0$

$[\underset{x}{\underset{\uparrow}{arr[i] \% m}} + \underset{y}{\underset{\uparrow}{arr[j] \% m}}] \% m = 0$

arr[] = [__ __ $\frac{15}{\%8} = 7$ __ $\frac{21}{\%8} = 7$ __ __ __]

m = 8

map:

key rem	value
0	4
1	②
2	③
3	①
4	③
5	②
6	⑥
7	④

0{ (0, 0) → sum=0 → $\frac{4(4-1)}{2}$

⑧{
- (1, 7) → ⑧
- (2, 6) → ⑱
- (3, 5) → ②
- (4, 4) → $\frac{3(3-1)}{2}$

pairs = 20 36 52 ④

$\underbrace{20 \quad 36 \quad 52}$

$\underset{n}{{}^{3}C_{2}} = \frac{3(3-1)}{1}$

Qu1 Given 2 numbers A and B [A > B]

find count of possible M such that

A % m = B % m [m > 1]

a = 13 , b = 7.

13 % 2 = 1 7 % 2 = 1

13 % 3 = 1 7 % 3 = 1

13 % 6 = 1 7 % 6 = 1

a = 12 b = 18

12 % 2 = 0 18 % 2 = 0

12 % 3 = 0 18 % 3 = 0

12 % 6 = 0 18 % 6 = 0

factors (18 - 12)

factors (6)

$$(a - b) \% m = (a \% m - b \% m + m) \% m$$

$$a \% m = b \% m$$

$$\boxed{a \% m - b \% m} = 0$$

$$a \% m - b \% m + m = m$$

$$(a \% m - b \% m + m) \% m = \underbrace{m \% m}_{0}$$

$$(a - b) \% m = 0$$

m is multiple of a - b.

factors (a - b) ← f1, f2, f3 } multiples

return factors - 1.

6 → ~~1~~, 2, 3, 6

ans = 0

$\overline{31}$ $\overline{30}$ $\overline{29}$ $\overline{28}$ — — $\overline{2}$ $\overline{1}$ $\overline{0}$

→ 1 ans = 2^31 + 2^30 + 2^29 — —

. . — — $2^2 + 2^1 + 2^0$ ←