Shift Operators

a : 0 0 0 1 1 0 0 1 (25)

1 2	1 4	1 2 8
1 6	3 2	6 4
3 2	6 4	8
5 0	1 0 0	2 0 0

a<<1 : a>>1 : 0 0 0 0 1 1 0 0 (12)

a<<2 : a>>2 : 0 0 0 0 0 1 1 0, 0 0 0 0 0 0 1 1 (6)

a<<3 : a>>3 : 0 0 0 0 0 0 1 1, 0 0 0 0 0 0 0 0 (3)

a<<i : a>>i : 0 0 0 0 0 0 0 0 (0) $a/2^i$

a<<1 : 0 0 1 1 0 0 1 0 (50)

a<<2 : 0 1 1 0 0 1 0 0 (100)

a<<3 : 1 1 0 0 1 0 0 0 (200)

a<<i : 2^i $a*2^i$

a = 0 0 0 0 0 1 0 1

a<<1 : 0 0 0 0 1 0 1 0

a<<2 : 0 0 0 1 0 1 0 0

a<<3 : 0 0 1 0 1 0 0 0

a<<i ;

```
int pow2(int N){
int n = 1<<N;
}
```

```
long pow2(int N){
    return 1<<N;
}
```

~~for~~

N = 25 ✓
N = 40 ✗
as 1 is int so it is 32 bit,

we have not type cast 1 as long then it will work
for N = 63, unsigned long will work.

Now, for N = 76, 255 : 2^N (how will it work)

~~3 e 9 + 7~~

Calculate a^N.

```
int pow(int a, int N){
    for (int i = 1; i <= N; i++)
        ans = ans * a;
    return ans;
}
```

for $\begin{matrix} a=3 \\ N=40 \end{matrix}$ | $\begin{matrix} a=253 \\ N=177 \end{matrix}$ | $\begin{matrix} a=2 \\ N=100 \end{matrix}$ | $\begin{matrix} a=73 \\ N=7025 \end{matrix}$

then it will not work as value will be ~~100~~ large then question will mention to use mod

ans %. M = [0, M-1]
%. 5 = [0, 4]

M = 1e9+7 = 10^9+7

int x = 30005 → 3e 4+5

double x = .001 → 1e-3

= .00003 → 3e-5

Modulo Arithematic

$(a+b) \% M = (a \% M + b \% M) \% M$

$18 \% 5 = 3$

$10 \% 5 + 8 \% 5$

$(a \times b) \% M = ((a \% M) \times (b \% M)) \% M$

$(3 \times 3) \% 5 = 4$

$3 \% 5 = 3 = 9$

$3 \% 5 = 3$

$(2 * 9) \% 5 = 18 \% 5 = 3$

$2 * 4 = 8 \% 5 = 3$

$(9 * 2) \% 5 = 3$

$19 \% 5 = 4 = 8 \% 5 = 3$

$2 \% 5 = 2$

$5 * 5 \% 5 = 0$

$2 * 3 \% 5 = 1$

$2 * 3 = 6$

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

$(18 - 2) \% 5$

$= 16 \% 5 = 1$

$3 - 2 = 1$

$(9 - 9) \% 5$

$= 0$

$(25 - 23) \% 5 = 2$

$0 - 3 = -3 + 5 = 2$

$4 - 3 = 1$

$(a / b) \% M$

$\frac{6}{2} \% 3 \rightarrow 0$ ~~→~~

$\frac{24}{8} \% 5 = 3$

$\frac{4}{3} = \frac{4}{3}$

$$(a/b) \,\%\, m = \left(a \times \frac{1}{b}\right) \%\, m = (a \times b^{-1}) \%\, m$$

$$= ((a \%\, m) \times (b^{-1} \%\, m)) \%\, m$$

inverse modulo

$b^{-1} \%\, m$ does not work in normal way, we calculate it by using Euclid's Extended algo.

To understand Euclid Extended algo we will first go for Euclid algo. (GCD)

for a^N

```
ans = 1 ; M = 1e9+7
i = 1 to N
ans = (ans %a M * a %b M) %c M

return ans %d M
```

Here mod at a, b & d are unnecessary we can avoid them only c is required.

Mod is costly operator. Avoid unnecessary use.

```
int solve (int x, int y) {
    return 1<<x | 1 <<y;
}
```

$0 \le x, y \le 25$

1, 4 → 10010 → 10000

1<<1 → 00010

1<<4 → 10000

10010

```
bool checkBit (int N, int i) {
    int n = 1<<i;
    if (N & n > 0) {
        true
    }
}
```

$1 \le N \le 10^9$

$0 \le i \le 30$

1010

1

set

10010

1010

0010

0010

1000

0010

0000

---

all three works

```
return (N>>i) % 2
return (N>>i)&1;
return (N & (1<<i))!=0
```

```
int Bits (int N) {
    for (int i=0; i<31; i++) {
        if (N&1 == 1)
            cnt++;
        N = N>>1;
    }
}
```

$0 \le N \le 10^9$

0101

0&01

0001

0001

0010

0001

N	N-1	N & (N-1)
5 (101)	4 (100)	100 (4)
18 (10010)	17 (10001)	10000 (16)
25 (11001)	24 (11000)	11000 (24)
44 (101100)	43 (101011)	101000 (40)
16 (10000)	15 (01111)	00000 (0)

```
int CountBits (int n){
  While (N != 0){
      Count++;
    N = N & N-1;
    }
return cnt;
}
```

Complexity analysis of algo

Count divisor of N;

N -> 15 → 1, 3, 5, 15 (4)

12 → 1, 2, 3, 4, 6, 12 (6)

```
int divisor (int N){
  for (int i = 1; i <= sqrt(N); i++){
      if (N % i == 0)
        Count++;
    }
return Count * 2 + Count - 1;
}
```

$1\ GHz = 10^9$ (ins) cc/sec

$N = 10^9$ → 10^{18}

10^9 iterations — 10^{18} iterations

10^9 instructions — 10^{18} instructions

Time = 1 sec — $10^9 \times 10^9$ ~~sec~~ ~~sec~~ 10^9 sec

$1 \rightarrow 10^9$

$2 \rightarrow 2 \times 10^9$

$10 \text{ sec} \rightarrow 10^9 \times 10^9$

$$\frac{10^9}{60 \times 60 \times 24 \times 365} = 31.7 \text{ yrs}$$

100

1	x	100
2	x	50
4	x	25
5	x	20
10	x	10
20	x	5
25	x	4
50	x	2
100	x	1

$8

1	x	18
2	x	9
3	x	6
9	x	2
18	x	1

Complexity analysis of algo

Is time & space

```
int fn(int ar[], int N){
    int x = N * N;
    int y = N+1;
    int z = x*y + x-y;
    return x * y + 3*z + ...;
}
```

15 instruction

Whatever be the value of N, instruction executed are 15, so that is constant time.

Big O(1)

```
for(i = 0 to N) {  → repeated N times
    x = ...
    y = ...
    z = ...
}
```

O(1)

total = O(N)

```
for (i = 0 to N)
    for (j = 0 to N)
        ...
```

$O(N^2)$

10^6 / 10^3 = 10^3

10^3 / 10^6 = 10^{-3}

10^{-8} / 10^8 = 10^{-16}

$10^{-4} \times 100$

0.01

While Calculating time complexity we consider iterations & not instructions as instruction time depends on architecture of system ; it might change, but iteration time will be same.

```
for(int i=0; i<N; i++)
    for(int j= i; j<N; j++)
```

0 , 1 , 2 , 3 , 4 , 5

0-10, 1-10, 2-10, 3-10, 4-10, 5-10

$\frac{n(n+1)}{2} = \frac{N^2}{2} + \frac{N}{2} = N^2$

~~N-[1+2+3+...N)~~

i	j	total
0	0-N	N
1	1-N	N-1
2	2-N	N-2
		N-3
		N-4
		⋮
N-1	N-1 → N	1

```
for(int i=0; i<=N; i*=2)   → infinite loop as i is 0
```

```
for(int i=1; i<=N; i*=2)
```

i	total
1	1
2	1
4	1
8	1
16	1
32	1
64	

$2^k \leq N$

$\log_2 2^k = \log_2 N$

$k = \log_2 N$

$2^0 \Rightarrow 1$

$2^1 \Rightarrow 2$

$2^2 \Rightarrow 3$

$2^3 \Rightarrow 4$

$2^k \Rightarrow k+1$

```
bool fn(int ar[], int N, int k){
    for(int i=0; i<N; i++)
        if(ar[i]==k)
            return T;
    return F;
}
```

3

Linear Search

best case : 1
worst case : not found, take N iteration
average case = somewhere in b/w

P-1

Sol-1 : $5N^3 + 10N^2 + 100$ | $O(N^3)$
Sol-2 : $40N^3 + N^2 + 4000$ | $O(N^3)$

as per big O both are same but sol 1 is better as coefficient of N^3 in sol-1 is less.

P-2

Sol1: $10N^4 + 5N + 100$ | $O(N^4)$
Sol 2: $15N^3 + 3N^2 + 10000$ | $O(N^3)$

62 Sol 25+100
125+15+25x3 +100000

When we have N value as 5, sol-1 is better but in N=100, sol-2 is better, for any value N>100, sol-2 is better.

Solution with less time complexity may not always perform better, it depends on value of N.

Big - O: Puts an upper bound on the complexity of an algorithm based on input size after a certain threshold.

A)
```
int a = 0, b = 0;
for (int i = 0; i < N; i++) {
    a = a + rand();
}
for (int j = 0; j < M; j++) {
    b = b + rand();
}
```
O(N)
O(M)
O(N+M)

B)
```
int a = 0, b = 0;
for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        a = a + j;
    }
}
for (int k = 0; k < N; k++) {
    b = b + k;
}
```
O(N²)
O(N)
O(N²)

(C)

i	j	total
0	N	N
1	N	N

N²

(D)

i	tot
8	1
4	1
2	1

log(N)

$2^3 \to 1$
$2^2 \to 2$
$2^1 \to 3$
$2^0 \to 4$

(E)
```
for (int i = 1; i <= N; i++) {          N
    int P = pow(i, k);
    for (int j = 1; j <= P; ++j) {      P
        //
    }
}
```

$O(N \times P)$

(D)

i	total
N	1
$\frac{N}{2}$	+1
$\frac{N}{4}$	+1
$\frac{N}{8}$	+1
$\frac{2}{2^k}$	k+1

$\frac{N}{2^k} > 0$

$\frac{N}{2^k} = 1$

$N = 2^k$

$\log N = k$

(F)
```
int cout = 0;
for (int i = N; i > 0; i /= 2) {
    for (int j = 0; j < i; j++) {
        cout += 1;
    }
}
```

i	j	total
N	$0 \to N$	$N \times N$
$\frac{N}{2}$	$0 \to \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$
$\frac{N}{4}$	$0 \to \frac{N}{4}$	$\frac{N}{4} \times \frac{N}{4}$
⋮	⋮	⋮
$\frac{N}{2^k}$	$0 \to \frac{N}{2^k}$	$\frac{N}{2^k} \times \frac{N}{2^k}$

G) $N \log N$

H) N^2

1) D
2) F
3) B
4) 4
5) C
6) E

$N = 10^6$, 1GHz, 1 iter = 1 ist, $10^3 \approx 2^{10}$

Todo

Big-O :	N^3	$N^2 \log N$
iter:	10^{18}	$20 \times (10^{12})$
Tim:	10^9 sec	2×10^4 sec

$(10^6)^3 = 10^{18}$

$10^{3 \times 3} = 2^{30}$

10^{18} $\log_2 10^6 = 20$

10^{12}

$10^3 \to 2^{10}$

$10^6 \to 2^{20}$

	N^2	$N \log N$	N	$\sqrt{N}$
iter	10^{12}	20×10^6	10^6	10^3
Tim	10^3 sec	2×10^{-2} sec	10^{-3}	10^{-6}

	$\log_2 N$	2^N	
		$N=30$	$N=60$
iter	20	$2^{30} = 10^9$	$2^{60} = 10^{18}$
Tim	2×10^{-8}	1 sec	10^9 sec

Read T
loop(T)
Read i/p
process
Print o/p

N → T×N
N2 → T×N²
NlogN → T×NlogN

T × M < 10^9

↓
time to execute our logic

in HR/CC/CF...

the processor is 1GHz, Time limit = 1sec

↓

10^9 inst/sec

in general limit should be T×M < 10^8
assuming 10 instruction per iteration.

Tasks

1. Sheep PDF
2. boring page table
3. for TLE faced calculate Time limit