D. AND, OR and square sum

time limit per test

2 seconds

memory limit per test

512 megabytes

input

standard input

output

standard output

Gottfried learned about binary number representation. He then came up with this task and presented it to you.

You are given a collection of nn non-negative integers a1,…,ana1,…,an. You are allowed to perform the following operation: choose two distinct indices 1≤i,j≤n1≤i,j≤n. If before the operation ai=xai=x, aj=yaj=y, then after the operation ai=x AND yai=x AND y, aj=x OR yaj=x OR y, where ANDAND and OROR are bitwise AND and OR respectively (refer to the Notes section for formal description). The operation may be performed any number of times (possibly zero).

After all operations are done, compute ∑ni=1a2i∑i=1nai2 — the sum of squares of all aiai. What is the largest sum of squares you can achieve?

**Input**

The first line contains a single integer nn (1≤n≤2⋅1051≤n≤2⋅105).

The second line contains nn integers a1,…,ana1,…,an (0≤ai<2200≤ai<220).

**Output**

Print a single integer — the largest possible sum of squares that can be achieved after several (possibly zero) operations.

**Examples**

**input**

**Copy**

1

123

**output**

**Copy**

15129

**input**

**Copy**

3

1 3 5

**output**

**Copy**

51

**input**

**Copy**

2

349525 699050

**output**

**Copy**

1099509530625

**Note**

In the first sample no operation can be made, thus the answer is 12321232.

In the second sample we can obtain the collection 1,1,71,1,7, and 12+12+72=5112+12+72=51.

If xx and yy are represented in binary with equal number of bits (possibly with leading zeros), then each bit of x AND yx AND y is set to 11 if and only if both corresponding bits of xx and yy are set to 11. Similarly, each bit of x OR yx OR y is set to 11 if and only if at least one of the corresponding bits of xx and yy are set to 11. For example, x=3x=3 and y=5y=5 are represented as 01120112 and 10121012 (highest bit first). Then, x AND y=0012=1x AND y=0012=1, and x OR y=1112=7x OR y=1112=7.

题目大意：给定数组，任选两个元素ai,aj，变更为ai&aj,ai|aj,以上操作可以做任意次。经过变更后，计算ai^2的和，要求和值最大。

样例模拟如下

3

1 3 5

51

1(001),3(011),5(101)

1&3,1|3操作如下

(001)&(011)=(001),(001)|(011)=(011)

操作前1,3

210(表示位置)

001

011

第0位上1的数量是2

第1位上1的数量是1

操作后1&3,1|3

210(表示位置)

001

011

第0位上1的数量是2

第1位上1的数量是1

可以看到：操作过后各个位上1的数量竟然不变

1&5,1|5操作如下

(001)&(101)=(001),(001)|(101)=(101)

操作前1,5

210(表示位置)

001

101

第0位上1的数量是2

第2位上1的数量是1

操作后1&5,1|5

210(表示位置)

001

101

第0位上1的数量是2

第2位上1的数量是1

可以看到：操作过后各个位上1的数量竟然不变

3&5,3|5操作如下

(011)&(101)=(001),(011)|(101)=(111)

操作前3,5

210(表示位置)

011

101

第0位上1的数量是2

第1位上1的数量是1

第2位上1的数量是1

操作后3&5,3|5

210(表示位置)

001

111

第0位上1的数量是2

第1位上1的数量是1

第2位上1的数量是1

可以看到：操作过后各个位上1的数量竟然不变

可以考虑，集中优势兵力，将每位上的1先分配到B[1],剩下的分配到B[2],继续剩下的分配到B[3],以此类推。

#include<bits/stdc++.h>

#define ll long long

using namespace std;

ll a[300000];

int G[300];

int main()

{

int n;

scanf("%d",&n); int mxx=0;

for(int i=1;i<=n;i++)

{scanf("%lld",&a[i]);

int cnt=0;

while(a[i])

{

if(a[i]%2==1) {G[cnt]++;}

cnt++;

a[i]/=2;

}

if(cnt>mxx) mxx=cnt;

}

int maxn=0;

for(int i=0;i<mxx;i++)

{

if(G[i]>maxn) {maxn=G[i];}

}

ll ans=0;

for(int i=0;i<maxn;i++)

{ ll now=0;

for(int j=0;j<30;j++)

{

if(G[j])

{ G[j]--;

now+=pow(2,j);

}

}

ans+=now\*now;

}

printf("%lld",ans);

return 0;

}