All Competitions > World CodeSprint 10 > Maximal AND Subsequences

Maximal AND Subsequences

locked

by nabila_ahmed

Problem Submissions Leaderboard Discussions Editorial

Editorial by nabila_ahmed

Basic knowledge:

- Every number can be represented as a binary number. Largesity of a number depends on its most significant bits. Therefore to make the end result maximum we have to make its most significant bits one.
- And operation on a bit only gives one if all of the numbers in the operation has one on that particular bit.

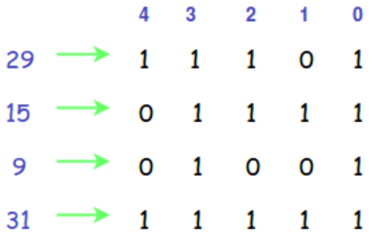
I will explain the problem with an example.

Lets the input is:

4 3

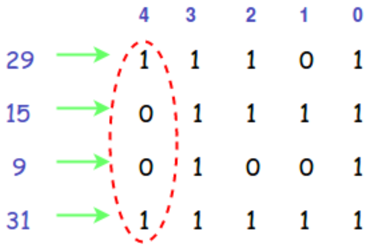
29 15 9 31

The binary representation of the four numbers are as follows:



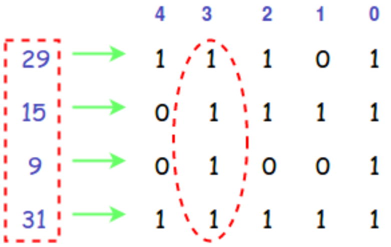
	4	3	2	1	0
29 →	1	1	1	0	1
15 →	0	1	1	1	1
9 →	0	1	0	0	1
31 →	1	1	1	1	1

We start iterating from the most significant bit which in this case is 4. On standing bit-4 we check all the numbers and count how many of them has 1 in this bit.



	4	3	2	1	0
29 →	1	1	1	0	1
15 →	0	1	1	1	1
9 →	0	1	0	0	1
31 →	1	1	1	1	1

We found two, which is less than k that means and result always has 0 in bit-4 for all possible combination of k numbers. Now move to the next bit.



	4	3	2	1	0
29 →	1	1	1	0	1
15 →	0	1	1	1	1
9 →	0	1	0	0	1
31 →	1	1	1	1	1

Statistics

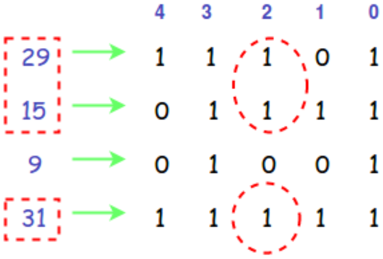
Difficulty: Medium

Time Complexity: $O(n \cdot logA_i)$

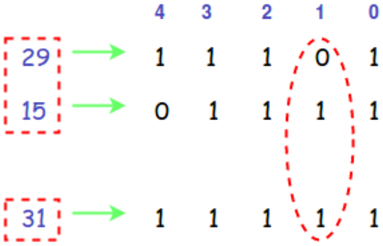
Required Knowledge: Bitwise operation, loop, vector

Publish Date: Mar 02 2017

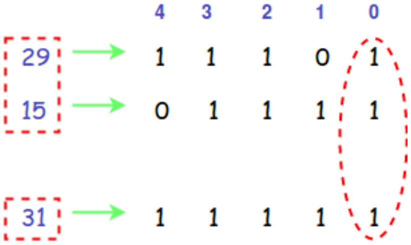
Here all the four number has 1 in bit-3. Therefore and result always has 1 in bit-3 for all possible combination of k numbers. Now move to bit-2.



Here 29, 15 and 31 has 1 in bit-2 and 9 has 0. Now if we consider all the combinations we can observe that only one combination $(29, 15, 31)$ has 1 in bit-2 and all other combinations { $(29, 15, 9)$, $(29, 9, 31)$, $(15, 9, 31)$} has 0 in this bit. We can also observe that the combinations have 0 due to the presence of 9 because 9 has 0 in bit-2. So we delete 9, and move to bit 1.



Here we get 1 only in two numbers which is less than k that means and result always has 0 in bit-1 for all possible combination of k numbers. Now move to the next bit.



In bit-0 we find all the three number has one.

And value of all the remaining numbers will be our final result.

Subsequence:

The question basically wants nCk. That is calculate all the possible ways to choose k numbers from n numbers. Here n is the size of our remaining numbers.

Set by nabila_ahmed

Problem Setter's code :

C++

```
#include <bits/stdc++.h>
#include<assert.h>

#define vlong long long
#define mod 1000000007

using namespace std;



inline vlong bigmod ( vlong a, vlong p, vlong m ) {
    vlong res = 1 % m, x = a % m;
    while ( p ) {
        if ( p & 1 ) res = ( res * x ) % m;
        x = ( x * x ) % m; p >>= 1;
    }
    return res;
}

vector<vlong>v;

void solution() {


    int n, k;

    //Taking input
    cin >> n >> k;
    assert(n>1 && n<=1000000);
    assert(k>1 && k<=n);

    for (int i = 0 ; i < n; i++) {
        vlong j;
        cin>>j;
        assert(j>=0 && j<=1000000000000000000LL);
        v.push_back(j);
    }

    //Checking bits
    for (int b = 62, len; b >= 0; b--) {

        vector <vlong> nv;

        len = v.size();

        for (int i=0; i<len; i++) {
            //if i bit is 1
            if (v[i]&(1LL<<b)) {
                nv.push_back(v[i]);
            }
        }
        //Decreasing number of possible values
        if (nv.size() >= k) {
            v = nv;
        }
    }

    //Calculating and value
    vlong ans = v[0];
    for(int i=1; i<k; i++)
    {
       ans = (ans & v[i]);
    }
    cout<<ans<<endl;

    //Claculating nCk
    int len = v.size();
    long long a = 1;
    for(int i=1; i<=len; i++)
    {
       a = (a*i)%(mod);
    }
    long long b = 1;
    for(int i=1; i<=k; i++)
    {
       b = (b*i)%(mod);
    }
    long long c = 1;
    for(int i=1; i<=(len-k); i++)
    {
       c = (c*i)%(mod);
    }
    b = bigmod(b, mod-2, mod);
    c = bigmod(c, mod-2, mod);
    a = (a*b)%mod;
    a = (a*c)%mod;

    cout<<a<<endl;
    return;
}

int main () {

        solution();
    return 0;
}
```

Tested by allleksssssa

Problem Tester's code :

```
#include<bits/stdc++.h>

using namespace std;

const int sz=64;
const int maxi=1e6;
const long long big=1e18;
const long long mo=1e9+7;
int n,k;
long long st[sz+1],a[maxi];
long long f[maxi],fi[maxi];
vector<long long> v[sz+1];

long long step(long long x, long long y, long long mo)
{
    if (y==0) return 1;
      long long  g=step(x,y/2,mo);
        if (y%2) return (((g*g)% mo)*x)%mo; else return (g*g)%mo;
}

int main()
{
    scanf("%d%d",&n,&k);
    assert(n>=2 && n<=100000);
    assert(k>=2 && k<=n);

    f[0]=1;
    for (long long i=1;i<=n;i++)
        f[i]=(f[i-1]*i)%mo;

    for (long long i=0;i<=n;i++)
   fi[i]=step(f[i],mo-2,mo);


    for (int i=0;i<n;i++)
    {
        scanf("%lld",&a[i]);
        assert(a[i]>=0 && a[i]<=big);
        v[sz].push_back(a[i]);
    }

     st[0]=1;
    for (int i=1;i<sz;i++)
        st[i]=st[i-1]*2;

  for (int i=sz-1;i>=0;i--)
  {
      for (int j=0;j<v[i+1].size();j++)
        if (st[i]&v[i+1][j]) v[i].push_back(v[i+1][j]);

        if (v[i].size()<k)
        {
            v[i].clear();

            for (int j=0;j<v[i+1].size();j++)
                v[i].push_back(v[i+1][j]);
        }
  }
 long long ans=v[0][0];
 int sz=v[0].size();
 long long cnt=(((f[sz]*fi[k])%mo)*fi[sz-k])%mo;
  for (int i=1;i<sz;i++)
    ans&=v[0][i];

   printf("%lld\n",ans);
   printf("%lld\n",cnt);
    return 0;
}
```

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature