Yate's DP with Bitmasks

md.mottakin.chowdhury

July 2018

1 Theory

Given an array A of 2^N integers, we need to calculate for all $mask$,

$F[mask] = \sum_{i \subseteq mask} A[i]$.

$S(mask, i)$ contains all subsets of $mask$ which differ from it only in the first i bits. Then

$$S(mask, i) = \begin{cases} S(mask, i-1) \ if \ i^{th} \ bit \ is \ off \\ S(mask, i-1) \cup S(mask \oplus 2^i, i-1) \ otherwise \end{cases}$$

2 Problems

2.1 Problem 1

You have been given an integer array a on size n. You must report the number of ordered pairs (i, j) such that $a[i] \& a[j]$ is 0.

```
// dp[mask][i] = total number of occurrence of all numbers subset of mask
// such that subsets differ in first i positions.
const int N=(1<<20)+5;
int n, a[MAX+7], cnt[N];
ll dp[N][21];

ll solve()
{
        for(int mask=0; mask<(1<<20); mask++)
        {
            // occurrence count of mask in input
                dp[mask][0]=cnt[mask];

                if(mask&1)
                        dp[mask][0]+=cnt[mask^1];

                for(int i=1; i<=20; i++)
                {
                        dp[mask][i]+=dp[mask][i-1];
```

```
                        if(mask&(1<<i))
                                dp[mask][i]+=dp[mask^(1<<i)][i-1];
                }
        }
        ll ret=0;

        FOR(i,0,n)
        {
        // considering the occurrences of numbers which will be 0 when applied
        // bitwise and with a[i]
                ret+=dp[((1<<20)-1)^a[i]][20];
                cnt[a[i]]=0;
        }
        return ret;
}
```

2.2 Problem 2

You are given an array a of n integers. How many subsets of these integers have 0 when applied bitwise *and* on the subset?

```
const int N=(1<<20);
int n, cnt[N], a;
int dp[N+5];
ll twos[N];

void solve()
{
        for(int mask=0; mask<N; mask++)
                dp[mask]=cnt[mask];
        for(int i=0; i<20; i++)
        {
                for(int mask=0; mask<N; mask++)
                {
                        if(!(mask&(1<<i)))
                                dp[mask]+=dp[mask^(1<<i)];
                }
        }
        // dp[mask] - number of a[i] where a[i]&mask=mask
        twos[0]=1;
        for(int i=1; i<N; i++) twos[i]=(twos[i-1]*2LL)%mod;

        // apply inclusion-exclusion
        ll ans=0;
        for(int i=0; i<N; i++)
        {
                int bits=__builtin_popcount(i);
                ans+=((bits&1) ? (-1) : 1)*twos[dp[i]];
                ans%=mod;
                if(ans<0) ans+=mod;
```

```
        }

        prnt(ans);
}
```

2.3 Problem 3

You are given an array a. For each element of the array, output another element such that their bitwise *and* equals 0, otherwise output -1.

```
// same as problem 1, except we keep out[mask] which denotes an integer such that
// out[mask]&mask=0
const int MAX=(1<<22)+5;
int n, a[MAX], out[MAX], cnt[MAX];
ll dp[MAX];

void solve()
{
        for(int i=0; i<MAX; i++)
        {
                if(cnt[i])
                {
                        dp[i]=cnt[i];
                        out[i]=i;
                }
        }

        for(int i=0; i<=22; i++)
        {
                for(int mask=0; mask<(1<<22); mask++)
                {
                        if(mask&(1<<i))
                        {
                                dp[mask]+=dp[mask^(1<<i)];

                                if(!out[mask])
                                {
                                        if(cnt[mask]) out[mask]=mask;
                                        else out[mask]=out[mask^(1<<i)];
                                }
                        }
                }
        }

        FOR(i,0,n)
        {
                int idx=((1<<22)-1)^a[i];

                if(out[idx]==0) printf("-1\n");
                else printf("%d\n", out[idx]);
```

```
        }
}
```

2.4 Problem 4

You are given an array a of n integers and a vlue k. How many subsets of these integers have a value of k when applied bitwise *or* on the subset?

```
const int N=(1<<20)+5;
int n, m, k, a[MAX], cnt[N];
ll dp[N], twos[N];

void solve()
{
        for(int i=0; i<(1<<20); i++)
                dp[i]=cnt[i]; // count of each number in input

        for(int i=0; i<20; i++)
        {
                for(int mask=0; mask<(1<<20); mask++)
                {
                        if(mask&(1<<i))
                        {
                                dp[mask]+=dp[mask^(1<<i)];
                                dp[mask]%=mod;
                        }
                }
        }

        ll ans=0;
        for(int i=k; i>=0; i--)
        {
                if((i|k)!=k) continue;
                int curr=__builtin_popcount(k^i);
                ans+=(curr%2==0 ? 1 : -1)*(twos[dp[i]]-1);
                ans%=mod;
                if(ans<0) ans+=mod;
                // prnt((1<<curr));
        }
        prnt(ans);
}
```

2.5 Problem 5

You are given an array of n numbers and q queries. In each query, you are asked to add a number, delete a number or answer value of $f(x)$ where $f(x)$ denotes the value of the maximum *bitwise-and* of all the subsequences of length x of the array. The idea is to use Yate's dp along with SQRT Decomposition on queries.

```
const int N=1<<18;
const int bsz=512;
```

```
int cnt[N], a[MAX], n, m, k, dp[N], idx=0;
pii bucket[bsz+5];

void calc()
{
        FOR(i,0,(1<<k))
                dp[i]=cnt[i];

        // dp[mask] = total numbers x such that x&mask=mask
        for(int i=0; i<k; i++)
        {
                for(int mask=0; mask<(1<<k); mask++)
                {
                        if(!(mask&(1<<i)))
                                dp[mask]+=dp[mask^(1<<i)];
                }
        }
}

void add(int x)
{
        cnt[x]++;
        bucket[idx]={x,1};
        idx++;

        if(idx==bsz) // bucket full, recalc dp
        {
                calc();
                idx=0;
        }
}

void rem(int x)
{
        cnt[x]--;
        bucket[idx]={x,-1};
        idx++;
        if(idx==bsz) // bucket full, recalc dp
        {
                calc();
                idx=0;
        }
}

// how many values t are there such that t&curr=curr?
int canget(int curr)
{
        int ret=dp[curr];
        FOR(i,0,idx)
        {
                if((bucket[i].first & curr)==curr)
```

```
                        ret+=bucket[i].second;
        }
        return ret;
}

int getans(int x)
{
        int ret=0;
        // try to ensure the higher bits first
        for(int i=k-1; i>=0; i--)
        {
                int curr=(ret^(1<<i));
                // There are total greater equal x numbers t such
                // that t&curr=curr;
                if(canget(curr)>=x) // We can achieve curr
                        ret|=(1<<i);
        }
        return ret;
}

void handleQueries()
{
        int t, x;
        FOR(i,0,m)
        {
                scanf("%d%d", &t, &x);
                if(t==1) add(x);
                else if(t==2) rem(x);
                else printf("%d\n", getans(x));
        }
}

int main()
{
    scanf("%d%d%d", &n, &m, &k);
    FOR(i,0,n)
    {
        scanf("%d", &a[i]);
        cnt[a[i]]++;
    }
    calc();
    handleQueries();
    return 0;
}
```