Fft

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 300010;
const double PI = acos(-1);

int n, m;
struct Complex
{
	double x, y;
	Complex operator+ (const Complex& t) const
	{
		return {x + t.x, y + t.y};
	}
	Complex operator- (const Complex& t) const
	{
		return {x - t.x, y - t.y};
	}
	Complex operator* (const Complex& t) const
	{
		return {x * t.x - y * t.y, x * t.y + y * t.x};
	}
}a[N], b[N];
int rev[N], bit, tot;

void fft(Complex a[], int inv)
{
	for (int i = 0; i < tot; i ++ )
		if (i < rev[i])
			swap(a[i], a[rev[i]]);
	for (int mid = 1; mid < tot; mid <<= 1)
	{
		auto w1 = Complex({cos(PI / mid), inv * sin(PI / mid)});
		for (int i = 0; i < tot; i += mid * 2)
		{
			auto wk = Complex({1, 0});
			for (int j = 0; j < mid; j ++, wk = wk * w1)
			{
				auto x = a[i + j], y = wk * a[i + j + mid];
```

```
                    a[i + j] = x + y, a[i + j + mid] = x - y;
                }
            }
        }
}

int main()
{
    scanf("%d%d", &n, &m);
    for (int i = 0; i <= n; i ++ ) scanf("%lf", &a[i].x);
    for (int i = 0; i <= m; i ++ ) scanf("%lf", &b[i].x);
    while ((1 << bit) < n + m + 1) bit ++;
    tot = 1 << bit;
    for (int i = 0; i < tot; i ++ )
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
    fft(a, 1), fft(b, 1);
    for (int i = 0; i < tot; i ++ ) a[i] = a[i] * b[i];
    fft(a, -1);
    for (int i = 0; i <= n + m; i ++ )
        printf("%d ", (int)(a[i].x / tot + 0.5));

    return 0;
}
```

线性基

选任意个数

求异或第 k 小

```
#include <iostream>
#include <cstring>
#include <algorithm>

using namespace std;

typedef long long LL;
const int N = 10010;

LL a[N];

int main()
{
    int T;
    scanf("%d", &T);
    for (int C = 1; C <= T; C ++ )
    {
```

```
            printf("Case #%d:\n", C);
            int n;
            scanf("%d", &n);
            for (int i = 0; i < n; i ++ ) scanf("%lld", &a[i]);
            int k = 0;
            for (int i = 62; i >= 0; i -- )
            {
                for (int j = k; j < n; j ++ )
                    if (a[j] >> i & 1)
                    {
                        swap(a[j], a[k]);
                        break;
                    }
                if (!(a[k] >> i & 1)) continue;
                for (int j = 0; j < n; j ++ )
                    if (j != k && (a[j] >> i & 1))
                        a[j] ^= a[k];
                k ++ ;
                if (k == n) break;
            }
            reverse(a, a + k);

            int m;
            scanf("%d", &m);
            while (m -- )
            {
                LL x;
                scanf("%lld", &x);
                if (k < n) x -- ;
                if (x >= (1ll << k)) puts("-1");
                else
                {
                    LL res = 0;
                    for (int i = 0; i < k; i ++ )
                        if (x >> i & 1)
                            res ^= a[i];
                    printf("%lld\n", res);
                }
            }
        }
        return 0;
}
```