杂七杂八

```
#define _USE_MATH_DEFINES
M_PI_2
ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
priority_queue
```

素数筛

```
vector<long long> sieve(long long n){
    vector<char> isp(n + 1, 1);
    vector<long long> p(1, 2);
    long long i, j, r = sqrt(n);
    // isp[0]=0;isp[1]=0;
    for (i = 4; i <= n; i += 2) isp[i] = 0;
    for (i = 3; i <= r; i += 2){
        if (isp[i] == 1){
            p.push_back(i);
            j = n / i;
            if (j % 2 == 0) j--;
            for (; j >= i; j -= 2) if (isp[j] == 1) isp[i * j] = 0;
        }
    }
    if (++r % 2 == 0) r++;
    for (; r <= n; r += 2) if (isp[r] == 1) p.push_back(r);
    return p;
}

vector<int> pri;
bool not_prime[N];
void pre(int n)
{
    for (int i = 2; i <= n; ++i)
    {
        if (!not_prime[i]) pri.push_back(i);
        for (int pri_j : pri)
        {
            if (i * pri_j > n) break;
            not_prime[i * pri_j] = true;
            if (i % pri_j == 0) break;
        }
    }
}
```

快速幂

```

```

```
long long binpow(long long base, long long exp)
{
    long long ans = 1;
    while (exp)
    {
        if (exp & 1LL) ans *= base;
        base *= base;
        exp >>= 1;
    }
    return ans;
}

long long binpow(long long base, long long exp, long long mod)
{
    long long ans = 1;
    base %= mod;
    while (exp)
    {
        if (exp & 1LL) ans = ans * base % mod;
        base = base * base % mod;
        exp >>= 1;
    }
    return ans;
}
```

gcd

```
long long gcd(long long x, long long y)
{
    long long z;
    x = abs(x);
    y = abs(y);
    if (x > y) swap(x, y);
    if (x == 0) return y;
    while (y != 0)
    {
        z = x % y;
        x = y;
        y = z;
    }
    return x;
}
```

ST表

```
template <typename T>
class SparseTable {
  using VT = vector<T>;
  using VVT = vector<VT>;
```

```
  using func_type = function<T(const T &, const T &)>;
  VVT ST;
  static T default_func(const T &t1, const T &t2) { return max(t1, t2); }
  func_type op;
 public:
  SparseTable(const vector<T> &v, func_type _func = default_func) {
    op = _func;
    int len = v.size(), l1 = ceil(log2(len)) + 1;
    ST.assign(len, VT(l1, 0));
    for (int i = 0; i < len; ++i) {
      ST[i][0] = v[i];
    }
    for (int j = 1; j < l1; ++j) {
      int pj = (1 << (j - 1));
      for (int i = 0; i + pj < len; ++i) {
        ST[i][j] = op(ST[i][j - 1], ST[i + (1 << (j - 1))][j - 1]);
      }
    }
  }
  T query(int l, int r) {
    int lt = r - l + 1;
    int q = floor(log2(lt));
    return op(ST[l][q], ST[r - (1 << q) + 1][q]);
  }
};
```