Contents

1 Basic 1
- 1.1 Default Code 1
- 1.2 IncreaseStackSize 1
- 1.3 Pragma optimization 2
- 1.4 Debugger 2
- 1.5 Quick Random 2
- 1.6 IO Optimization 2

2 Data Structure 2
- 2.1 Bigint 2
- 2.2 ScientificNotation 3
- 2.3 unordered_map 4
- 2.4 extc_balance_tree 4
- 2.5 extc_heap 4
- 2.6 SkewHeap 4
- 2.7 Disjoint Set 5
- 2.8 Treap 5
- 2.9 SparseTable 5
- 2.10 FenwickTree 6

3 Graph 6
- 3.1 BCC Edge 6
- 3.2 BCC Vertex 6
- 3.3 Strongly Connected Components 7
- 3.4 Bipartie Matching 7
- 3.5 MinimumCostMaximumFlow 7
- 3.6 MaximumFlow 8

4 Math 8
- 4.1 Prime Table 8
- 4.2 ax+by=gcd 8
- 4.3 Pollard Rho 9
- 4.4 Linear Sieve 9
- 4.5 NloglogN Sieve 9
- 4.6 Range Sieve 9
- 4.7 Miller Rabin 9
- 4.8 Inverse Element 9
- 4.9 Euler Phi Function 10
- 4.10 Gauss Elimination 10
- 4.11 Fast Fourier Transform 10
- 4.12 Chinese Remainder 10
- 4.13 NTT 11

5 Geometry 11
- 5.1 Point Class 11
- 5.2 Circle Class 12
- 5.3 Line Class 12
- 5.4 Segment Class 12
- 5.5 Triangle Circumcentre 13
- 5.6 2D Convex Hull 13
- 5.7 SimulateAnnealing 13

6 Stringology 13
- 6.1 Hash 13
- 6.2 Suffix Array 14
- 6.3 KMP 14

1 Basic

1.1 Default Code

```
#include <bits/stdc++.h>
using namespace std;
/* include everything for Kotori~ <3 */

typedef int64_t lld;
typedef uint64_t llu;
typedef long double llf;
typedef pair<int,int> PII;
typedef pair<int,lld> PIL;
typedef pair<lld,int> PLI;
typedef pair<lld,lld> PLL;
template<typename T>
using maxHeap = priority_queue<T,vector<T>,less<T>>;
template<typename T>
using minHeap = priority_queue<T,vector<T>,greater<T>>;
/* define some types for Ruby! */

#define FF first
#define SS second
#define SZ(x) (int)((x).size())
#define ALL(x) begin(x), end(x)
#define PB push_back
#define WC(x) while((x)--)
/* make code shorter for Di~a~ */

template<typename Iter>
ostream& _out(ostream &s, Iter b, Iter e) {
    s<<"[";
    for ( auto it=b; it!=e; it++ ) s<<(it==b?"":" ")<<*
        it;
    s<<"]";
    return s;
}
template<typename A, typename B>
ostream& operator <<( ostream &s, const pair<A,B> &p )
    { return s<<"("<<p.FF<<","<<p.SS<<")"; }
template<typename T>
ostream& operator <<( ostream &s, const vector<T> &c )
    { return _out(s,ALL(c)); }
/* make output easier for Ainyan~n~ */

bool debug = 0;
#define DUMP(x) if(debug) cerr<<__PRETTY_FUNCTION__<<":
    "<<__LINE__<<" - "<<(#x)<<"="<<(x)<<'\n'
template<typename T>
void DEBUG(const T& x){if(debug) cerr<<x;}
template<typename T, typename... Args>
void DEBUG(const T& head,const Args& ...tail){
    if(debug){cerr<<head; DEBUG(tail...);}
}
/* Let's debug with Nico~Nico~Ni */

int main(int argc, char* argv[]){
    if(argc>1 and string(argv[1])=="-D") debug=1;
    if(!debug){ios_base::sync_with_stdio(0);cin.tie(0)
        ;}
    return 0;
}
```

1.2 IncreaseStackSize

```
//stack resize
asm( "mov %0,%%esp\n" ::"g"(mem+10000000) );
//change esp to rsp if 64-bit system

//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
    const rlim_t ks = 64*1024*1024;
    struct rlimit rl;
    int res=getrlimit(RLIMIT_STACK, &rl);
    if(res==0){
        if(rl.rlim_cur<ks){
            rl.rlim_cur=ks;
            res=setrlimit(RLIMIT_STACK, &rl);
        }
    }
}
```

1.3 Pragma optimization

```
#pragma GCC optimize("Ofast,no-stack-protector,no-math-
    errno,unroll-loops")
#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm
    ,mmx,avx,tune=native")
```

1.4 Debugger

```
#! /usr/bin/env python3
import subprocess, platform

os_name = platform.system()
cmd = []
prefix = ""

if os_name == 'Windows':
    cmd = ["cmd", "/C"]
else:
    cmd = ["bash", "-c"]
   prefix = "./"

def GetTestData(exe):
    myout = subprocess.check_output(cmd + ["%s%s"%(
        prefix, exe)])
   return myout.decode("utf8")
def Judge(a,b,testdata):
    f = open("test.in", "w+")
   f.write(testdata)
   f.close()
   myout = subprocess.check_output(cmd + ["%s%s < test.
       in"%(prefix, a)])
   ansout = subprocess.check_output(cmd + ["%s%s < test
       .in"%(prefix, b)])
   if not myout == ansout:
       print("answer: %s"%ansout.decode("utf8"),end="")
      print("output: %s"%myout.decode("utf8"),end="")
      print("WA!")
      return False
  return True
if __name__ == '__main__':
    cnt = 0
   isOK = True
   while isOK:
       cnt += 1
      print(cnt)
      isOK = Judge("1397.exe", "test.exe", GetTestData(
          "gen.exe"))
```

1.5 Quick Random

```
// PRNG {{{
template<class T, T x1, T x2, T x3, int y1, int y2, int
     y3>
struct PRNG {
    using S = typename std::make_signed<T>::type;
    T s;
    PRNG(T _s = 0) : s(_s) {}
    T next() {
        T z = (s += x1);
        z = (z ^ (z >> y1)) * x2;
        z = (z ^ (z >> y2)) * x3;
        return z ^ (z >> y3);
    }
    T next(T n) { return next() % n; }
    S next(S l, S r) { return l + next(r - l + 1); }
    T operator()() { return next(); }
    T operator()(T n) { return next(n); }
    S operator()(S l, S r) { return next(l, r); }
    static T gen(T s) { return PRNG(s)(); }
    template<class U>
      void shuffle(U first, U last) {
          size_t n = last - first;
          for (size_t i = 0; i < n; i++) swap(first[i],
               first[next(i + 1)]);
      }
};

using R32 = PRNG<uint32_t, 0x9E3779B1, 0x85EBCA6B, 0
    xC2B2AE35, 16, 13, 16>;
R32 r32;
using R64 = PRNG<uint64_t, 0x9E3779B97F4A7C15, 0
    xBF58476D1CE4E5B9, 0x94D049BB133111EB, 30, 27, 31>;
R64 r64;
// }}}
```

1.6 IO Optimization

```
static inline int gc() {
    static char buf[1 << 20], *p = buf, *end = buf;
    if (p == end) {
        if ((end = buf + fread(buf, 1, 1 << 20, stdin))
             == buf) return EOF;
        p = buf;
    }
    return *p++;
}

template<typename T>
static inline bool gn(T &_){
    register int c = gc(); register T __ = 1; _ = 0;
    while(!isdigit(c) and c!=EOF and c!='-') c = gc();
    if(c == '-') { __ = -1; c = gc(); }
    if(c == EOF) return false;
    while(isdigit(c)) _ = _ * 10 + c - '0', c = gc();
    _ *= __;
    return true;
}
template <typename T, typename ...Args>
static inline bool gn(T &x, Args& ...args){return gn(x)
     and gn(args...);}
```

2 Data Structure

2.1 Bigint

```
struct Bigint{
    static const int LEN = 60;
    static const int BIGMOD = 10000;

    int s;
    int vl, v[LEN];
    Bigint() : s(1) { vl = 0; }
    Bigint(long long a) {
        s = 1; vl = 0;
        if (a < 0) { s = -1; a = -a; }
        while (a) {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    Bigint(string str) {
        s = 1; vl = 0;
        int stPos = 0, num = 0;
        if (!str.empty() && str[0] == '-') {
            stPos = 1;
            s = -1;
        }
        for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
            num += (str[i] - '0') * q;
            if ((q *= 10) >= BIGMOD) {
                push_back(num);
                num = 0; q = 1;
            }
        }
        if (num) push_back(num);
        n();
    }

    int len() const {
        return vl;
    }
    bool empty() const { return len() == 0; }
    void push_back(int x) {
        v[vl++] = x;
    }
    void pop_back() {
        vl--;
    }
    int back() const {
```

```
        return v[vl-1];
    }
    void n() {
        while (!empty() && !back()) pop_back();
    }
    void resize(int nl) {
        vl = nl;
        fill(v, v+vl, 0);
    }

    void print() const {
        if (empty()) { putchar('0'); return; }
        if (s == -1) putchar('-');
        printf("%d", back());
        for (int i=len()-2; i>=0; i--) printf("%.4d",v[
            i]);
    }
    friend std::ostream& operator << (std::ostream& out
        , const Bigint &a) {
        if (a.empty()) { out << "0"; return out; }
        if (a.s == -1) out << "-";
        out << a.back();
        for (int i=a.len()-2; i>=0; i--) {
            char str[10];
            snprintf(str, 5, "%.4d", a.v[i]);
            out << str;
        }
        return out;
    }

    int cp3(const Bigint &b)const {
        if (s != b.s) return s - b.s;
        if (s == -1) return -(-*this).cp3(-b);
        if (len() != b.len()) return len()-b.len();//
            int
        for (int i=len()-1; i>=0; i--)
            if (v[i]!=b.v[i]) return v[i]-b.v[i];
        return 0;
    }

    bool operator < (const Bigint &b)const{ return cp3(
        b)<0; }
    bool operator <= (const Bigint &b)const{ return cp3
        (b)<=0; }
    bool operator == (const Bigint &b)const{ return cp3
        (b)==0; }
    bool operator != (const Bigint &b)const{ return cp3
        (b)!=0; }
    bool operator > (const Bigint &b)const{ return cp3(
        b)>0; }
    bool operator >= (const Bigint &b)const{ return cp3
        (b)>=0; }

    Bigint operator - () const {
        Bigint r = (*this);
        r.s = -r.s;
        return r;
    }
    Bigint operator + (const Bigint &b) const {
        if (s == -1) return -(-(*this)+(-b));
        if (b.s == -1) return (*this)-(-b);
        Bigint r;
        int nl = max(len(), b.len());
        r.resize(nl + 1);
        for (int i=0; i<nl; i++) {
            if (i < len()) r.v[i] += v[i];
            if (i < b.len()) r.v[i] += b.v[i];
            if(r.v[i] >= BIGMOD) {
                r.v[i+1] += r.v[i] / BIGMOD;
                r.v[i] %= BIGMOD;
            }
        }
        r.n();
        return r;
    }
    Bigint operator - (const Bigint &b) const {
        if (s == -1) return -(-(*this)-(-b));
        if (b.s == -1) return (*this)+(-b);
        if ((*this) < b) return -(b-(*this));
        Bigint r;
        r.resize(len());
        for (int i=0; i<len(); i++) {
            r.v[i] += v[i];
            if (i < b.len()) r.v[i] -= b.v[i];
            if (r.v[i] < 0) {
                r.v[i] += BIGMOD;
                r.v[i+1]--;
            }
        }
        r.n();
        return r;
    }
    Bigint operator * (const Bigint &b) {
        Bigint r;
        r.resize(len() + b.len() + 1);
        r.s = s * b.s;
        for (int i=0; i<len(); i++) {
            for (int j=0; j<b.len(); j++) {
                r.v[i+j] += v[i] * b.v[j];
                if(r.v[i+j] >= BIGMOD) {
                    r.v[i+j+1] += r.v[i+j] / BIGMOD;
                    r.v[i+j] %= BIGMOD;
                }
            }
        }
        r.n();
        return r;
    }
    Bigint operator / (const Bigint &b) {
        Bigint r;
        r.resize(max(1, len()-b.len()+1));
        int oriS = s;
        Bigint b2 = b; // b2 = abs(b)
        s = b2.s = r.s = 1;
        for (int i=r.len()-1; i>=0; i--) {
            int d=0, u=BIGMOD-1;
            while(d<u) {
                int m = (d+u+1)>>1;
                r.v[i] = m;
                if((r*b2) > (*this)) u = m-1;
                else d = m;
            }
            r.v[i] = d;
        }
        s = oriS;
        r.s = s * b.s;
        r.n();
        return r;
    }
    Bigint operator % (const Bigint &b) {
        return (*this)-(*this)/b*b;
    }
};
```

2.2 ScientificNotation

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

struct SciFi{
    typedef double base_t;
    base_t x; int p;
    SciFi(){x=0;p=0;}
    SciFi(base_t k){
        p = floor(log10(k));
        x = k / pow((base_t)10, p);
    }
    SciFi(base_t a, int b){
        x=a;p=b;
    }
    SciFi operator=(base_t k){
        p = floor(log10(k));
        x = k / pow((base_t)10, p);
        return *this;
    }
    SciFi operator*(SciFi k)const{
        int nP = p+k.p;
        base_t nX = x*k.x;
        int tp = floor(log10(nX));
        return SciFi(nX/pow((base_t)10, tp), nP+tp);
    }
    SciFi operator*=(SciFi k){
        p+=k.p;
        x*=k.x;
        int tp = floor(log10(x));
        p+=tp;
        x/=pow((base_t)10, tp);
        return *this;
```

```
    }
    SciFi operator+(SciFi k)const{
        int newP = std::min(k.p, p);
        base_t x1 = x*pow((base_t)10, p-newP);
        base_t x2 = k.x*pow((base_t)10, k.p-newP);
        x1+=x2;
        int tp = floor(log10(x1));
        newP+=tp;
        x1 /= pow((base_t)10, tp);
        return SciFi(x1, newP);
    }
    SciFi operator+=(SciFi k){
        int newP = std::min(k.p, p);
        base_t x1 = x*pow((base_t)10, p-newP);
        base_t x2 = k.x*pow((base_t)10, k.p-newP);
        x1+=x2;
        int tp = floor(log10(x1));
        newP+=tp;
        x1 /= pow((base_t)10, tp);
        x=x1;p=newP;
        return *this;
    }
    bool operator<(SciFi a)const{
        if(p == a.p) return x<a.x;
        return p<a.p;
    }
    bool operator>(SciFi a)const{
        if(p == a.p) return x>a.x;
        return p>a.p;
    }
    bool operator==(SciFi a)const{
        return p==a.p and x==a.x;
    }
};

int main(){
    double a; scanf("%lf",&a);
    SciFi aa=a, x;
    x = aa*SciFi(2);
    printf("%.2lfe%c%03d\n", x.x, "+-"[x.p<0], abs(x.p)
        );
    return 0;
}
```

2.3 unordered_map

```
#include <ext/pb_ds/assoc_container.hpp>
using __gnu_pbds::cc_hash_table;
using __gnu_pbds::gp_hash_table;
template<typename A, typename B> using hTable1 =
    cc_hash_table<A,B>;
template<typename A, typename B> using hTable2 =
    gp_hash_table<A,B>;
```

2.4 extc_balance_tree

```
#include <functional>
#include <ext/pb_ds/assoc_container.hpp>
using std::less;
using std::greater;
using __gnu_pbds::tree;
using __gnu_pbds::rb_tree_tag;
using __gnu_pbds::ov_tree_tag;
using __gnu_pbds::splay_tree_tag;
using __gnu_pbds::null_type;
using __gnu_pbds::tree_order_statistics_node_update;

template<typename T>
using ordered_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;

template<typename A, B>
using ordered_map = tree<A, B, less<A>, rb_tree_tag,
    tree_order_statistics_node_update>;

int main(){
    ordered_set<int> ss;
    ordered_map<int,int> mm;
    ss.insert(1);
    ss.insert(5);
    assert(*ss.find_by_order(0)==1);
    assert(ss.order_of_key(-1)==0);
    assert(ss.order_of_key(87)==2);
    return 0;
}
```

2.5 extc_heap

```
#include <functional>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/priority_queue.hpp>
using std::less;
using std::greater;
using __gnu_pbds::priority_queue;
using __gnu_pbds::pairing_heap_tag;
using __gnu_pbds::binary_heap_tag;
using __gnu_pbds::binomial_heap_tag;
using __gnu_pbds::rc_binomial_heap_tag;
using __gnu_pbds::thin_heap_tag;

int main(){
    priority_queue<int,less<int>,pairing_heap_tag> pq1,
         pq2;
    pq1.push(1);
    pq2.push(2);
    pq1.join(pq2);
    assert(pq2.size()==0);
    auto it = pq1.push(87);
    pq1.modify(it, 19);
    while(!pq1.empty()){
        pq1.top();
        pq1.pop();
    }
    return 0;
}
```

2.6 SkewHeap

```
#include <functional>
using std::less;

template<typename T, typename cmp=less<T> >
class SkewHeap{
private:
    struct SkewNode{
        T x;
        SkewNode *lc, *rc;
        SkewNode(T a=0):x(a), lc(nullptr), rc(nullptr)
            {}
    } *root;
    cmp CMP_;
    size_t count;
    SkewNode* Merge(SkewNode* a, SkewNode* b){
        if(!a or !b) return a?a:b;
        if(CMP_(a->x, b->x)) swap(a, b);
        a->rc = Merge(a->rc, b);
        swap(a->lc, a->rc);
        return a;
    }
    void clear(SkewNode*& a){
        if(!a) return;
        clear(a->lc); clear(a->rc);
        delete a; a = nullptr;
    }
public:
    SkewHeap(): root(nullptr), count(0){}
    bool empty(){return count==0;}
    size_t size(){return count;}
    T top(){return root->x;}
    void clear(){clear(root);count = 0;}
    void push(const T& x){
        SkewNode* a = new SkewNode(x);
        count += 1;
        root = Merge(root, a);
    }
    void join(SkewHeap& a){
        count += a.count; a.count = 0;
        root = Merge(root, a.root);
    }
    void pop(){
        count -= 1;
        SkewNode* rt = Merge(root->lc, root->rc);
        delete root; root = rt;
    }
```

```
    friend void swap(SkewHeap& a, SkewHeap& b){
        swap(a.root, b.root);
    }
};
```

2.7 Disjoint Set

```
class DJS{
private:
    vector<int> fa, sz, sv;
    vector<pair<int*, int>> opt;
    inline void assign(int *k, int v){
        opt.emplace_back(k, *k);
        *k = v;
    }
public:
    inline void init(int n){
        fa.resize(n); iota(fa.begin(), fa.end(), 0);
        sz.resize(n); fill(sz.begin(), sz.end(), 1);
        opt.clear();
    }
    int query(int x){
        if(fa[x] == x) return x;
        return query(fa[x]);
    }
    inline void merge(int a, int b){
        int af = query(a), bf = query(b);
        if(af == bf) return;
        if(sz[af] < sz[bf]) swap(af, bf);
        assign(&fa[bf], fa[af]);
        assign(&sz[af], sz[af]+sz[bf]);
    }
    inline void save(){sv.push_back((int)opt.size());}
    inline void undo(){
        int ls = sv.back(); sv.pop_back();
        while((int)opt.size() > ls){
            pair<int*, int> cur=opt.back();
            cur->first = cur.second;
            opt.pop_back();
        }
    }
}
```

2.8 Treap

```
class Treap{
private:
    const int MEM = 500000 + 5;
    unsigned seed;
    inline unsigned myrand(){
        static unsigned seed = time(NULL);
        seed = seed*seed*127 + seed*227 + 2147483587;
        seed ^= seed*97;
        seed /= 7123;
        return seed;
    }
    struct node{
        node *lc, *rc;
        int pri, size, val;
        node(){}
        node(int x):
            lc(nullptr),
            rc(nullptr),
            pri(myrand()),
            size(1),
            val(x)
        {}
        inline void pull(){
            size = 1;
            if(lc) size += lc->size;
            if(rc) size += rc->size;
        }
    } *root, pool[MEM];
    int mem_;
    inline node* new_node(int x){
        static int mem_ = 0;
        assert(mem_ < MEM);
        pool[mem_]=node(x);
        return &pool[mem_++];
    }
    inline int sz(node* x){return x?x->size:0;}
    node* merge(node *a, node *b){
        if(!a or !b) return a?a:b;
        if(a->pri > b->pri){
            a->rc = merge(a->rc, b);
            a->pull();
            return a;
        }else{
            b->lc = merge(a, b->lc);
            b->pull();
            return b;
        }
    }
    void split(Treap* t, int k, Treap*& a, Treap*& b){
        if(!t) a=b=nullptr;
        else if(sz(t->lc) < k){
            a = t;
            split(t->rc, k - sz(t->lc) - 1 , a->rc, b);
            a->pull();
        }else{
            b = t;
            split(t->lc, k, a, b->lc);
            b->pull();
        }
    }
    int oOk(node* rr, int x){
        if(rr==NULL)return 0;
        if((rr->val) < x)return gSize(rr->l)+oOk(rr->r,
            x)+1;
        else return oOk(rr->l, x);
    }
public:
    Treap(){root=nullptr;seed=time(NULL);mem_=0;}
    void do_something_at(int l, int r){
        // 1-base [l, r]
        split(root, l-1, tl, root);
        split(root, r-l+1, root, tr);
        root = merge(tl, merge(root, tr));
    }
    void insert(int x){
        node *a, *b;
        split(root, x, a, b);
        root = merge(merge(a, new node(x)), b);
        root->size = gSize(root->l)+gSize(root->r)+1;
    }
    void remove(int x){
        //need debug may contain bugs
        node *a, *b, *c, *d;
        split(root, x, a, b);
        a->size = gSize(a->l)+gSize(a->r);
        split(a, x-1, c, d);
        root = merge(b, c);
        root->size = gSize(root->l)+gSize(root->r);
        delete d;
    }
    int order_of_key(int x){return oOk(root,x);}
};
```

2.9 SparseTable

```
template<typename T, typename Cmp_=std::less<T>>
class SparseTable{
private:
    vector<vector<T>> table;
    vector<int> lg;
    T cmp_(T a, T b){
        return Cmp_()(a, b)?a:b;
    }
public:
    void init(T arr[], int n){
        // 0-base
        lg.resize(n+1);
        lg[0] = -1, lg[1] = 0;
        for(int i=2;i<=n;i++) lg[i] = lg[i>>1]+1;
        table.resize(lg[n]+1);
        table[0].resize(n);
        for(int i=0;i<n;i++) table[0][i] = arr[i];
        for(int i=1;i<=lg[n];i++){
            int len = 1<<(i-1), sz = 1<<i;
            table[i].resize(n-sz+1);
            for(int j=0;j<=n-sz;j++){
                table[i][j] = cmp_(table[i-1][j], table
                    [i-1][j+len]);
            }
        }
    }
```

```
    T query(int l, int r){
        // 0-base [l, r)
        int wh = lg[r-l], len=1<<wh;
        return cmp_(table[wh][l], table[wh][r-len]);
    }
};
```

2.10 FenwickTree

```
#include <vector>
using std::vector;

template<typename T>
class BIT{
#define ALL(x) begin(x), end(x)
private:
    vector<T> arr;
    int n;
    inline int lowbit(int x){return x & (-x);}
    T query(int x){
        T ret = 0;
        while(x > 0){
            ret += arr[x];
            x -= lowbit(x);
        }
        return ret;
    }
public:
    void init(int n_){
        n = n_;
        arr.resize(n);
        fill(arr.begin(), arr.end(), 0);
    }
    void modify(int pos, T v){
        while(pos < n){
            arr[pos] += v;
            pos += lowbit(pos);
        }
    }
    T query(int l, int r){
        // 1-base (l, r]
        return query(r) - query(l);
    }
#undef ALL
};

template<typename T>
class BIT{
#define ALL(x) begin(x), end(x)
private:
    vector<T> arr;
    int n;
    inline int lowbit(int x){return x & (-x);}
    void add(int s, int v){
        while(s){
            arr[s]+=v;
            s-=lowbit(s);
        }
    }
public:
    void init(int n_){
        n = n_;
        arr.resize(n);
        fill(ALL(arr), 0);
    }
    void add(int l, int r, T v){
        //1-base (l, r]
        add(l, -v);
        add(r, v);
    }
    T query(int x){
        T r=0;
        while(x<size){
            r+=arr[x];
            x+=lowbit(x);
        }
        return r;
    }
#undef ALL
};
```

3 Graph

3.1 BCC Edge

```
class BCC{
private:
    int low[N], dfn[N], cnt;
    bool bcc[N];
    vector<PII> G[N];
    void dfs(int w, int f){
        dfn[w] = cnt++;
        low[w] = dfn[w];
        for(auto i: G[w]){
            int u = i.FF, t = i.SS;
            if(u == f) continue;
            if(dfn[u]!=0){
                low[w] = min(low[w], dfn[u]);
            }else{
                dfs(u, w);
                low[w] = min(low[w], low[u]);
                if(low[u] > dfn[w]) bcc[t] = true;
            }
        }
    }
public:
    void init(int n, int m){
        for(int i=0;i<n;i++) G[i].clear();
        fill(bcc, bcc+m, false);
        cnt = 0;
    }
    void add_edge(int u, int v){
        G[u].PB({v, cnt});
        G[v].PB({u, cnt});
        cnt++;
    }
    void solve(){cnt = 1;dfs(0, 0);}
    // the id will be same as insert order, 0-base
    bool is_bcc(int x){return bcc[x];}
} bcc;
```

3.2 BCC Vertex

```
class BCC{
private:
    vector<vector<pair<int,int>>> G;
    vector<int> dfn, low, id, sz;
    vector<bool> vis, ap;
    int n, ecnt, bcnt;
    void tarjan(int u, int f, int d){
        vis[u] = true;
        dfn[u] = low[u] = d;
        int child = 0;
        for(auto e: G[u]) if(e.first != f){
            int v = e.first;
            if(vis[v]){
                low[u] = min(low[u], dfn[v]);
            }else{
                tarjan(v, u, d+1);
                if(low[v] >= dfn[u]) ap[u] = true;
                low[u] = min(low[u], low[v]);
                child += 1;
            }
        }
        if(dfn[u]==0 and child <= 1) ap[u] = false;
    }
    void bfs_bcc(int x){
        // not sure
        queue<int> bfs;
        bfs.push(x); vis[x] = true;
        while(!bfs.empty()){
            int u = bfs.front(); bfs.pop();
            for(auto e: G[u]){
                id[e.second] = bcnt;
                if(ap[e.first] or vis[e.first])
                    continue;
                bfs.push(e.first); vis[e.first] = true;
                sz[bcnt] += 1;
            }
        }
    }
public:
    void init(int n_){
```

```
        n = n_; G.clear(); G.resize(n);
        dfn.resize(n); low.resize(n);
        vis.clear(); vis.resize(n);
        ap.clear(); ap.resize(n);
        ecnt = 0, bcnt = 0;
    }
    void add_edge(int u, int v){
        assert(0 <= u and u < n);
        assert(0 <= v and v < n);
        G[u].emplace_back(v, ecnt);
        G[v].emplace_back(u, ecnt);
        ecnt += 1;
    }
    void solve(){
        for(int i=0;i<n;i++) if(!vis[i]) {
            tarjan(i, i, 0);
        }
        id.resize(ecnt);
        vis.clear(); vis.resize(n);
        sz.clear(); sz.resize(n);
        for(int i=0;i<n;i++) if(ap[i]){
            bfs_bcc(i); bcnt += 1;
        }
    }
    bool isAP(int x){return ap[x];}
    int count(){return bcnt;}
    // bcc_id of edges by insert order (0-base)
    int get_id(int x){return id[x];}
    // bcc size by bcc_id
    int get_size(int x){return sz[x];}
} bcc;
```

3.3 Strongly Connected Components

```
class SCC{
private:
    int n, num_;
    vector<int> G[N], rG[N], ord, num;
    bool vis[N];
    void dfs(int u){
        if(vis[u]) return;
        vis[u]=1;
        for(auto v: G[u]) dfs(v);
        ord.PB(u);
    }
    void rdfs(int u){
        if(vis[u]) return;
        num[u] = num_;
        vis[u] = 1;
        for(auto v: rG[u]) rdfs(v);
    }
public:
    inline void init(int n_){
        n=n_, num_=0;
        num.resize(n);
        for(int i=0;i<n;i++) G[i].clear();
        for(int i=0;i<n;i++) rG[i].clear();
    }
    inline void add_edge(int st, int ed){
        G[st].PB(ed);
        rG[ed].PB(st);
    }
    void solve(){
        memset(vis, 0, sizeof(vis));
        for(int i=0;i<n;i++){
            if(!vis[i]) dfs(i);
        }
        reverse(ALL(ord));
        memset(vis, 0, sizeof(vis));
        for(auto i: ord){
            if(!vis[i]){
                rdfs(i);
                num_++;
            }
        }
    }
    inline int get_id(int x){return num[x];}
    inline int count(){return num_;}
} scc;
```

3.4 Bipartie Matching

```
#include <bits/stdc++.h>
using namespace std;
#define N 500

class BipartieMatching{
  private:
    vector<int> X[N], Y[N];
    int fX[N], fY[N], n;
    bitset<N> walked;
    bool dfs(int x){
      for(auto i:X[x]){
        if(walked[i])continue;
        walked[i]=1;
        if(fY[i]==-1||dfs(fY[i])){
          fY[i]=x;fX[x]=i;
          return 1;
        }
      }
      return 0;
    }
  public:
    void init(int _n){
      n=_n;
      for(int i=0;i<n;i++){
        X[i].clear();
        Y[i].clear());
        fX[i]=fY[i]=-1;
      }
      walked.reset();
    }
    void AddEdge(int x, int y){
      X[x].push_back(y);
      Y[y].push_back(y);
    }
    int solve(){
      int cnt = 0;
      for(int i=0;i<n;i++){
        walked.reset();
        if(dfs(i)) cnt++;
      }
      // return how many pair matched
      return cnt;
    }
};
```

3.5 MinimumCostMaximumFlow

```
class MiniCostMaxiFlow{
    typedef int CapT;
    typedef lld WeiT;
    typedef pair<CapT, WeiT> PCW;
    const CapT INF_CAP = 1<<30;
    const WeiT INF_WEI = 1LL<<60;
    const int MAXV = N;
private:
    struct Edge{
        int to, back;
        WeiT wei;
        CapT cap;
        Edge(){}
        Edge(int a, int b, WeiT c, CapT d): to(a), back
            (b), wei(c), cap(d) {}
    };
    int ori, edd, V;
    vector<Edge> G[MAXV];
    int fa[MAXV], wh[MAXV];
    bool inq[MAXV];
    WeiT dis[MAXV];
    PCW SPFA(){
        for(int i=0;i<V;i++) inq[i]=0;
        for(int i=0;i<V;i++) dis[i]=INF_WEI;
        queue<int> qq;
        qq.push(ori);
        dis[ori]=0;
        while(!qq.empty()){
            int u = qq.front(); qq.pop();
            inq[u]=0;
            for(int i=0;i<SZ(G[u]);i++){
                Edge e = G[u][i];
                int v = e.to;
                WeiT d = e.wei;
                if(e.cap > 0 and dis[v] > dis[u]+d){
                    dis[v]=dis[u]+d;
                    fa[v]=u;
```

```
                    wh[v] = i;
                    if(inq[v]) continue;
                    qq.push(v);
                    inq[v]=1;
                }
            }
        }
        if(dis[edd]==INF_WEI) return {-1, -1};
        CapT mw=INF_CAP;
        for(int i=edd;i!=ori;i=fa[i]){
            mw = min(mw, G[fa[i]][wh[i]].cap);
        }
        for(int i=edd;i!=ori;i=fa[i]){
            auto &eg = G[fa[i]][wh[i]];
            eg.cap -= mw;
            G[eg.to][eg.back].cap += mw;
        }
        return {mw, dis[edd]};
    }
public:
    void init(int a, int b, int n=MAXV){
        V=n;
        ori = a;
        edd = b;
        for(int i=0;i<n;i++) G[i].clear();
    }
    void addEdge(int st, int ed, WeiT w, CapT c){
        G[st].PB(Edge(ed, SZ(G[ed]), w, c));
        G[ed].PB(Edge(st, SZ(G[st])-1, -w, 0));
    }
    PCW solve(){
        CapT cc=0; WeiT ww=0;
        while(true){
            PCW ret = SPFA();
            if(ret.FF==-1) break;
            cc += ret.FF;
            ww += ret.SS;
        }
        return {cc, ww};
    }
} mcmf;
```

3.6 MaximumFlow

```
class Dinic{
private:
    using CapT = int64_t;
    struct Edge{
        int to, rev;
        CapT cap;
    };
    int n, st, ed;
    vector<vector<Edge>> G;
    vector<int> lv;
    bool BFS(){
        fill(lv.begin(), lv.end(), -1);
        queue<int> bfs;
        bfs.push(st);
        lv[st] = 0;
        while(!bfs.empty()){
            int u = bfs.front(); bfs.pop();
            for(auto e: G[u]){
                if(e.cap <= 0 or lv[e.to]!=-1) continue
                    ;
                lv[e.to] = lv[u] + 1;
                bfs.push(e.to);
            }
        }
        return (lv[ed]!=-1);
    }
    CapT DFS(int u, CapT f){
        if(u == ed) return f;
        CapT ret = 0;
        for(auto& e: G[u]){
            if(e.cap <= 0 or lv[e.to]!=lv[u]+1)
                continue;
            CapT nf = DFS(e.to, min(f, e.cap));
            ret += nf; e.cap -= nf; f -= nf;
            G[e.to][e.rev].cap += nf;
            if(f == 0) return ret;
        }
        if(ret == 0) lv[u] = -1;
        return ret;
    }
public:
    void init(int n_, int st_, int ed_){
        n = n_, st = st_, ed = ed_;
        G.resize(n); lv.resize(n);
        fill(G.begin(), G.end(), vector<Edge>());
    }
    void add_edge(int u, int v, CapT c){
        G[u].push_back({v, (int)(G[v].size()), c});
        G[v].push_back({u, (int)(G[u].size())-1, 0});
    }
    CapT max_flow(){
        CapT ret = 0;
        while(BFS()){
            CapT f = DFS(st, numeric_limits<CapT>::max
                ());
            ret += f;
            if(f == 0) break;
        }
        return ret;
    }
} flow;
```

4 Math

4.1 Prime Table

```
// 1000000000 < primes < 2147483647
1002939109, 1020288887, 1028798297, 1038684299,
1041211027, 1051762951, 1058585963, 1063020809,
1094763083, 1106384353, 1120154459, 1140593173,
1147930723, 1172520109, 1183835981, 1187659051,
1241251303, 1247184097, 1255940849, 1272759031,
1287027493, 1288511629, 1294632499, 1312650799,
1314753281, 1320080669, 1321970357, 1333133947,
1337684419, 1353508067, 1358715989, 1364961029,
1366046831, 1376536367, 1381705499, 1410637769,
1411311571, 1422795043, 1437499801, 1495803851,
1511764363, 1526710979, 1538018089, 1542373769,
1545326953, 1549429633, 1556212739, 1575971759,
1586465261, 1608336427, 1609783001, 1620728569,
1643267081, 1652401603, 1656717203, 1660920671,
1666858577, 1669260361, 1670240317, 1678791131,
1685583143, 1725964619, 1734856421, 1743134179,
1761537223, 1774260193, 1778872889, 1781930609,
1803000149, 1814256623, 1834876331, 1839154463,
1840044389, 1843241713, 1856039431, 1868564531,
1868732623, 1884198443, 1884616807, 1885059541,
1909942399, 1914471137, 1923951707, 1925453197,
1937719153, 1954649041, 1958915237, 1970709803,
1979612177, 1980446837, 1989761941, 2007826547,
2008033571, 2011186739, 2039465081, 2039728567,
2093735719, 2116097521, 2123852629, 2140170259

// 2147483647 < primes < 4000000000

3148478261, 3153064147, 3176351071, 3187523093,
3196772239, 3201312913, 3203063977, 3204840059,
3210224309, 3213032591, 3217689851, 3218469083,
3219857533, 3231880427, 3235951699, 3273767923,
3276188869, 3277183181, 3282463507, 3285553889,
3319309027, 3327005333, 3327574903, 3341387953,
3373293941, 3380077549, 3380892997, 3381118801,
3384716479, 3386991323
```

4.2 ax+by=gcd

```
// By Adrien1018 (not knowing how to use.
// ax+ny = 1, ax+ny == ax == 1 (mod n)
tuple<int, int, int> extended_gcd(int a, int b) {
    if (!b) return make_tuple(a, 1, 0);
    int d, x, y;
    tie(d, x, y) = extended_gcd(b, a % b);
    return make_tuple(d, y, x - (a / b) * y);
}
// ax+by = gcd (by Eddy1021
PII gcd(int a, int b){
    if(b == 0) return {1, 0};
    PII q = gcd(b, a % b);
    return {q.second, q.first - q.second * (a / b)};
}
```

4.3 Pollard Rho

```
// coded by hanhanW
// does not work when n is prime
long long modit(long long x,long long mod) {
    if(x>=mod) x-=mod;
    //if(x<0) x+=mod;
    return x;
}
long long mult(long long x,long long y,long long mod) {
    long long s=0,m=x%mod;
    while(y) {
        if(y&1) s=modit(s+m,mod);
        y>>=1;
        m=modit(m+m,mod);
    }
    return s;
}
long long f(long long x,long long mod) {
    return modit(mult(x,x,mod)+1,mod);
}
long long pollard_rho(long long n) {
    if(!(n&1)) return 2;
    while (true) {
        long long y=2, x=rand()%(n-1)+1, res=1;
        for (int sz=2; res==1; sz*=2) {
            for (int i=0; i<sz && res<=1; i++) {
                x = f(x, n);
                res = __gcd(abs(x-y), n);
            }
            y = x;
        }
        if (res!=0 && res!=n) return res;
    }
}
```

4.4 Linear Sieve

```
const int N = 20000000;
bool sieve[N];

void linear_sieve(){
    vector<int> prime;
    for (int i=2; i<N; i++){
        if (!sieve[i]) prime.push_back(i);
        for (int j=0; i*prime[j]<N; j++)
        {
            sieve[i*prime[j]] = true;
            if (i % prime[j] == 0) break;
        }
    }
}
```

4.5 NloglogN Sieve

```
bol notprime[N];
vector<int> primes;

void Sieve(int n){
    // reverse true false for quicker
    for(int i=2;i<=n;i++){
        if(!notprime[i]){
            primes.push_back(i);
            for(int j=i*i;j<=n;j+=i) notprime[i]=true;
        }
    }
}
```

4.6 Range Sieve

```
#include <algorithm>
typedef long long lld;
const int MAX_SQRT_B = 50000;
const int MAX_L = 200000 + 5;

bool is_prime_small[MAX_SQRT_B];
bool is_prime[MAX_L];
void sieve(lld,lld);
```

```
void sieve(lld l, lld r){
    // [l, r)
    for(lld i=2;i*i<r;i++) is_prime_small[i] = true;
    for(lld i=l;i<r;i++) is_prime[i-l] = true;
    if(l==1) is_prime[0] = false;
    for(lld i=2;i*i<r;i++){
        if(!is_prime_small[i]) continue;
        for(lld j=i*i;j*j<r;j+=i) is_prime_small[j]=
            false;
        for(lld j=std::max(2LL, (l+i-1)/i)*i;j<r;j+=i)
            is_prime[j-l]=false;
    }
}
```

4.7 Miller Rabin

```
lld modu(lld a, lld m){
    while(a >= m) a -= m;
    return a;
}
lld mul(lld a, lld b, lld m){
    if(a < b) swap(a, b);
    lld ret = 0;
    while(b){
        if(b & 1) ret = modu(ret+a, m);
        a = modu(a+a, m);
        b >>= 1;
    }
    return ret;
}
lld qPow(lld a, lld k, lld m){
    lld ret = 1;
    a %= m;
    while(k){
        if(k & 1) ret = mul(ret, a, m);
        a = mul(a, a, m);
        k >>= 1;
    }
    return modu(ret, m);
}
bool witness(lld a, lld s, int t, lld n){
    lld b = qPow(a, s, n);
    if(b == 0) return false;
    while(t--){
        lld bb = mul(b, b, n);
        if(bb == 1 and b != 1 and b != n-1) return true
            ;
        b = bb;
    }
    return b != 1;
}
bool miller_rabin(lld n){
    if(n < 2) return false;
    if(!(n & 1)) return (n==2);
    lld x = n-1; int t = 0;
    while(!(x&1)) x >>= 1, t++;
    lld sprp[] =
        {2,325,9375,28178,450775,9780504,1795265022};
    for(int i=0;i<7;i++){
        if(witness(sprp[i]%n, x, t, n)) return false;
    }
    return true;
}
```

4.8 Inverse Element

```
// x's inverse mod k
// if k is prime
long long GetInv(long long x, long long k){
    return qPow(x, k-2);
}

// x's inverse mod k
// if k is not prime
long long GetInv(long long x, long long k){
    return qPow(x, Euler(k)-1);
}
// or extended_gcd(x, k).second
// if you need [1, x] (most use: [1, k-1]
void solve(int x, long long k){
    inv[1] = 1;
```

```
    for(int i=2;i<x;i++)
        inv[i] = ((long long)(k - k/i) * inv[k % i]) %
            k;
}
```

4.9 Euler Phi Function

```
/*
   extended euler:
   a^b mod p
   if gcd(a, p)==1: a^(b%phi(p))
   elif b < phi(p): a^b mod p
   else a^(b%phi(p) + phi(p))
 */
inline int64_t Euler(int x){
    int64_t r=1;
    for(int i=2;i*i<=x;++i){
        if(x%i==0){
            x/=i;
            r*=(i-1);
            while(x%i==0){
                x/=i;
                r*=i;
            }
        }
    }
    if(x>1) r*=x-1;
    return r;
}

vector<int> primes;
bool notprime[N];
int64_t phi[N];
inline void euler_sieve(int n){
    for(int i=2;i<n;i++){
        if(!notprime[i]){
            primes.push_back(i);
            phi[i] = i-1;
        }
        for(auto j: primes){
            if(i*j >= n) break;
            notprime[i*j] = true;
            phi[i*j] = phi[i] * phi[j];
            if(i % j == 0){
                phi[i*j] = phi[i] * j;
                break;
            }
        }
    }
}
```

4.10 Gauss Elimination

```
typedef long double llf;
const int N = 300;
const llf EPS = 1e-8;

// make m[i][i] = x, m[i][j] = 0
// v is for solving equation:
// for(int i=0;i<n;i++) ans[pos[i]] = val[i]/mtx[i][pos
   [i]];
// for(int i=0;i<n;i++) cout << ans[i] << '\n';
bool Gauss(llf m[N][N], llf v[N], int n, int pos[N]){
    for(int i=0;i<n;i++){
        int x=-1, y=-1; llf m = 0;
        for(int j=i;j<n;j++) for(int k=i;k<n;k++){
            if(fabs(m[j][pos[k]])>m){
                m = fabs(m[j][pos[k]]);
                x = j, y = k;
            }
        }
        if(x==-1 or y==-1) return false;
        swap(m[x], m[i]);
        swap(v[x], v[i]);
        swap(pos[y], pos[i]);
        for(int j=i+1;j<n;j++){
            llf xi = m[j][pos[i]]/m[i][pos[i]];
            for(int k=0;k<n;k++) m[j][pos[k]] -= xi*m[i
                ][pos[k]];
            v[j] -= xi*v[i];
        }
    }
    for(int i=n-1;i>=0;i--){
        for(int j=i-1;j>=0;j--){
            llf xi = m[j][pos[i]]/m[i][pos[i]];
            for(int k=0;k<n;k++) m[j][pos[k]] -= xi*m[i
                ][pos[k]];
            v[j] -= xi*v[i];
        }
    }
    return true;
}
```

4.11 Fast Fourier Transform

```
/*
   polynomial multiply:
   FFT(a, N, true);
   FFT(b, N, true);
   for(int i=0;i<MAXN;i++) c[i] = a[i]*b[i];
   FFT(c, N, false);
   yeah~ go result in c
   (N must be 2^k and >= len(a)+len(b))
 */
typedef long double llf;
typedef complex<llf> cplx;
const int MAXN = 262144;
const llf PI = acos((llf)-1);

cplx A[MAXN], B[MAXN], C[MAXN], omega[MAXN+1];

void init_omega(){
    const cplx I = {0, 1};
    for(int i=0;i<=MAXN;i++) omega[i] = exp(i*2*PI/MAXN
        *I);
}

void FFT(cplx arr[], int n, bool ori){
    // n must be 2^k
    int theta = MAXN / n;
    for(int len=n;len>=2;len>>=1){
        int tot = len>>1;
        for(int i=0;i<tot;i++){
            cplx omg = omega[ori?i*theta%MAXN:MAXN-(i*
                theta%MAXN)];
            for(int j=i;j<n;j+=len){
                int k = j+tot;
                cplx x = arr[j] - arr[k];
                arr[j] += arr[k];
                arr[k] = omg * x;
            }
        }
        theta = (theta * 2) % MAXN;
    }
    int i = 0;
    for(int j=1;j<n-1;j++){
        for(int k=n>>1;k>(i^=k);k>>=1);
        if(j < i) swap(arr[j], arr[i]);
    }
    if(ori) return;
    for(int i=0;i<n;i++) arr[i] /= n;
}
```

4.12 Chinese Remainder

```
// ax+ny = 1, ax+ny == ax == 1 (mod n)
pair<lld,lld> gcd(lld a, lld b){
    if(b == 0) return {1, 0};
    pair<lld,lld> q = gcd(b, a % b);
    return {q.second, q.first - q.second * (a / b)};
}

lld crt(lld ans[], lld pri[], int n){
    lld M = 1;
    for(int i=0;i<n;i++) M *= pri[i];
    lld ret = 0;
    for(int i=0;i<n;i++){
        lld inv = (gcd(M/pri[i], pri[i]).first + pri[i
            ])%pri[i];
        ret += (ans[i]*(M/pri[i])%M * inv)%M;
        ret %= M;
    }
    return ret;
}
```

4.13 NTT

```
typedef long long LL;
// Remember coefficient are mod P
/* p=a*2^n+1
   n     2^n         p           a    root
   5     32          97          3    5
   6     64          193         3    5
   7     128         257         2    3
   8     256         257         1    3
   9     512         7681        15   17
   10    1024        12289       12   11
   11    2048        12289       6    11
   12    4096        12289       3    11
   13    8192        40961       5    3
   14    16384       65537       4    3
   15    32768       65537       2    3
   16    65536       65537       1    3
   17    131072      786433      6    10
   18    262144      786433      3    10 (605028353,
       2308, 3)
   19    524288      5767169     11   3
   20    1048576     7340033     7    3
   21    2097152     23068673    11   3
   22    4194304     104857601   25   3
   23    8388608     167772161   20   3
   24    16777216    167772161   10   3
   25    33554432    167772161   5    3 (1107296257, 33,
       10)
   26    67108864    469762049   7    3
   27    134217728   2013265921 15    31 */
// (must be 2^k)
// To implement poly. multiply:
// NTT<P, root, MAXN> ntt;
// ntt( n , a ); // or ntt.tran( n , a );
// ntt( n , b );
// for( int i = 0 ; i < n ; i++ )
//   c[ i ] = a[ i ] * b[ i ];
// ntt( n , c , 1 );
//
// then you have the result in c :: [LL]

template<LL P, LL root, int MAXN>
struct NTT{
    static LL bigmod(LL a, LL b) {
        LL res = 1;
        for (LL bs = a; b; b >>= 1, bs = (bs * bs) % P)
            {
            if(b&1) res=(res*bs)%P;
        }
        return res;
    }
    static LL inv(LL a, LL b) {
        if(a==1)return 1;
        return (((LL)(a-inv(b%a,a))*b+1)/a)%b;
    }
    LL omega[MAXN+1];
    NTT() {
        omega[0] = 1;
        LL r = bigmod(root, (P-1)/MAXN);
        for (int i=1; i<=MAXN; i++)
            omega[i] = (omega[i-1]*r)%P;
    }
    // n must be 2^k
    void tran(int n, LL a[], bool inv_ntt=false){
        int basic = MAXN / n;
        int theta = basic;
        for (int m = n; m >= 2; m >>= 1) {
            int mh = m >> 1;
            for (int i = 0; i < mh; i++) {
                LL w = omega[i*theta%MAXN];
                for (int j = i; j < n; j += m) {
                    int k = j + mh;
                    LL x = a[j] - a[k];
                    if (x < 0) x += P;
                    a[j] += a[k];
                    if (a[j] > P) a[j] -= P;
                    a[k] = (w * x) % P;
                }
            }
            theta = (theta * 2) % MAXN;
        }
        int i = 0;
        for (int j = 1; j < n - 1; j++) {
            for (int k = n >> 1; k > (i ^= k); k >>= 1)
                ;
            if (j < i) swap(a[i], a[j]);
        }
        if (inv_ntt) {
            LL ni = inv(n,P);
            reverse( a+1 , a+n );
            for (i = 0; i < n; i++)
                a[i] = (a[i] * ni) % P;
        }
    }
    void operator()(int n, LL a[], bool inv_ntt=false)
        {
        tran(n, a, inv_ntt);
    }
};

const LL P=2013265921,root=31;
const int MAXN=4194304;
NTT<P, root, MAXN> ntt;
```

5 Geometry

5.1 Point Class

```
template<typename T>
struct Point{
    typedef long double llf;
    static constexpr llf EPS = 1e-8;
    T x, y;
    Point(): x(0), y(0){}
    Point(T _, T __): x(_), y(__){}
    template<typename T2>
      Point(const Point<T2>& a): x(a.x), y(a.y){}
    inline llf theta() const {
        return atan2((llf)y, (llf)x);
    }
    inline llf dis() const {
        return hypot((llf)x, (llf)y);
    }
    inline llf dis(const Point& o) const {
        return hypot((llf)(x-o.x), (llf)(y-o.y));
    }
    Point operator-(const Point& o) const {
        return Point(x-o.x, y-o.y);
    }
    Point operator-=(const Point& o){
        x-=o.x, y-=o.y;
        return *this;
    }
    Point operator+(const Point& o) const {
        return Point(x+o.x, y+o.y);
    }
    Point operator+=(const Point& o){
        x+=o.x, y+=o.y;
        return *this;
    }
    Point operator*(const T& k) const {
        return Point(x*k, y*k);
    }
    Point operator*=(const T& k){
        x*=k, y*=k;
        return *this;
    }
    Point operator/(const T& k) const {
        return Point(x/k, y/k);
    }
    Point operator/=(const T& k){
        x/=k, y/=k;
        return *this;
    }
    Point operator-() const {
        return Point(-x, -y);
    }
    Point rot90() const {
        return Point(-y, x);
    }
    bool equal(const Point& o, true_type) const {
        return fabs(x-o.x) < EPS and fabs(y-o.y) < EPS;
    }
    bool equal(const Point& o, false_type) const {
        return x==o.x and y==o.y;
    }
    bool operator==(const Point& o) const {
```

```
        return equal(o, is_floating_point<T>());
    }
    bool operator!=(const Point& o) const {
        return !(*this == o);
    }
    bool operator<(const Point& o) const {
        return theta() < o.theta();
        // sort like what pairs did
        // return fabs(x-o.x)<EPS?y<o.y:x<o.x;
    }
    friend inline T cross(const Point& a, const Point&
        b){
        return a.x*b.y - b.x*a.y;
    }
    friend inline T dot(const Point& a, const Point &b)
        {
        return a.x*b.x + a.y*b.y;
    }
    friend ostream& operator<<(ostream& ss, const Point
        & o){
        ss<<"("<<o.x<<", "<<o.y<<")";
        return ss;
    }
};
```

5.2 Circle Class

```
template<typename T>
struct Circle{
    Point<T> o;
    T r;
    vector<Point<llf>> operator&(const Circle& aa)
        const {
        // https://www.cnblogs.com/wangzming/p/8338142.
            html
        llf d=o.dis(aa.o);
        if(d > r+aa.r+EPS or d < fabs(r-aa.r)-EPS)
            return {};
        llf dt = (r*r - aa.r*aa.r)/d, dl = (d+dt)/2;
        Point<llf> dir = (aa.o-o); dir /= d;
        Point<llf> pcrs = dir*dl + o;
        dt=sqrt(max(0.0L, r*r - dl*dl)), dir=dir.rot90
            ();
        return {pcrs + dir*dt, pcrs - dir*dt};
    }
};
```

5.3 Line Class

```
const Point<long double> INF_P(-1e20, 1e20);
const Point<long double> NOT_EXIST(1e20, 1e-20);
template<typename T>
struct Line{
    static constexpr long double EPS = 1e-8;
    // ax+by+c = 0
    T a, b, c;
    Line(): a(0), b(1), c(0){}
    Line(T _, T __, T ___): a(_), b(__), c(___){
        assert(fabs(a)>EPS or fabs(b)>EPS);
    }
    template<typename T2>
      Line(const Line<T2>& x): a(x.a), b(x.b), c(x.c){}
    typedef Point<long double> Pt;
    bool equal(const Line& o, true_type) const {
        return fabs(a-o.a) < EPS and fabs(b-o.b) < EPS
            and fabs(c-o.b) < EPS;
    }
    bool euqal(const Line& o, false_type) const {
        return a==o.a and b==o.b and c==o.c;
    }
    bool operator==(const Line& o) const {
        return euqal(o, is_floating_point<T>());
    }
    bool operator!=(const Line& o) const {
        return !(*this == o);
    }
    friend inline bool on_line__(const Point<T>& p,
        const Line& l, true_type){
        return fabs(l.a*p.x + l.b*p.y + l.c) < EPS;
    }
    friend inline bool on_line__(const Point<T>& p,
        const Line& l, false_type){
        return l.a*p.x + l.b*p.y + l.c == 0;
    }
    friend inline bool on_line(const Point<T>&p const
        Line& l){
        return on_line__(p, l, is_floating_point<T>());
    }
    friend inline bool is_parallel__(const Line& x,
        const Line& y, true_type){
        return fabs(x.a*y.b - x.b*y.a) < EPS;
    }
    friend inline bool is_parallel__(const Line& x,
        const Line& y, false_type){
        return x.a*y.b == x.b*y.a;
    }
    friend inline bool is_parallel(const Line& x, const
         Line& y){
        return is_parallel__(x, y, is_floating_point<T
            >());
    }
    friend inline Pt get_inter(const Line& x, const
        Line& y){
        typedef long double llf;
        if(x==y) return INF_P;
        if(is_parallel(x, y)) return NOT_EXIST;
        llf delta = x.a*y.b - x.b*y.a;
        llf delta_x = x.b*y.c - x.c*y.b;
        llf delta_y = x.c*y.a - x.a*y.c;
        return Pt(delta_x / delta, delta_y / delta);
    }
    friend ostream& operator<<(ostream& ss, const Line&
         o){
        ss<<o.a<<"x+"<<o.b<<"y+"<<o.c<<"=0";
        return ss;
    }
};
template<typename T>
inline Line<T> get_line(const Point<T>& a, const Point<
    T>& b){
    return Line<T>(a.y-b.y, b.x-a.x, (b.y-a.y)*a.x-(b.x
        -a.x)*a.y);
}
```

5.4 Segment Class

```
const long double EPS = 1e-8;
template<typename T>
struct Segment{
    // p1.x < p2.x
    Line<T> base;
    Point<T> p1, p2;
    Segment(): base(Line<T>()), p1(Point<T>()), p2(
        Point<T>()){
        assert(on_line(p1, base) and on_line(p2, base))
            ;
    }
    Segment(Line<T> _, Point<T> __, Point<T> ___): base
        (_), p1(__), p2(___){
        assert(on_line(p1, base) and on_line(p2, base))
            ;
    }
    template<typename T2>
      Segment(const Segment<T2>& _): base(_.base), p1(_
        .p1), p2(_.p2) {}
    typedef Point<long double> Pt;
    friend bool on_segment(const Point<T>& p, const
        Segment& l){
        if(on_line(p, l.base))
            return (l.p1.x-p.x)*(p.x-l.p2.x)>=0 and (l.
                p1.y-p.y)*(p.y-l.p2.y)>=0;
        return false;
    }
    friend bool have_inter(const Segment& a, const
        Segment& b){
        if(is_parallel(a.base, b.base)){
            return on_segment(a.p1, b) or on_segment(a.
                p2, b) or on_segment(b.p1, a) or
                on_segment(b.p2, a);
        }
        Pt inter = get_inter(a.base, b.base);
        return on_segment(inter, a) and on_segment(
            inter, b);
    }
    friend inline Pt get_inter(const Segment& a, const
        Segment& b){
```

```cpp
        if(!have_inter(a, b)){
            return NOT_EXIST;
        }else if(is_parallel(a.base, b.base)){
            if(a.p1 == b.p1){
                if(on_segment(a.p2, b) or on_segment(b.
                    p2, a)) return INF_P;
                else return a.p1;
            }else if(a.p1 == b.p2){
                if(on_segment(a.p2, b) or on_segment(b.
                    p1, a)) return INF_P;
                else return a.p1;
            }else if(a.p2 == b.p1){
                if(on_segment(a.p1, b) or on_segment(b.
                    p2, a)) return INF_P;
                else return a.p2;
            }else if(a.p2 == b.p2){
                if(on_segment(a.p1, b) or on_segment(b.
                    p1, a)) return INF_P;
                else return a.p2;
            }
            return INF_P;
        }
        return get_inter(a.base, b.base);
    }
    friend ostream& operator<<(ostream& ss, const
        Segment& o){
        ss<<o.base<<", "<<o.p1<<" ~ "<<o.p2;
        return ss;
    }
};
template<typename T>
inline Segment<T> get_segment(const Point<T>& a, const
    Point<T>& b){
    return Segment<T>(get_line(a, b), a, b);
}
```

5.5 Triangle Circumcentre

```cpp
template<typename T>
Circle<llf> get_circum(const Point<T>& a, const Point<T
    >& b, const Point<T>& c){
    llf a1 = a.x-b.x;
    llf b1 = a.y-b.y;
    llf c1 = (a.x+b.x)/2 * a1 + (a.y+b.y)/2 * b1;

    llf a2 = a.x-c.x;
    llf b2 = a.y-c.y;
    llf c2 = (a.x+c.x)/2 * a2 + (a.y+c.y)/2 * b2;

    Circle<llf> cc;
    cc.o.x = (c1*b2-b1*c2)/(a1*b2-b1*a2);
    cc.o.y = (a1*c2-c1*a2)/(a1*b2-b1*a2);
    cc.r = hypot(cc.o.x-a.x, cc.o.y-a.y);
    return cc;
}
```

5.6 2D Convex Hull

```cpp
template<typename T>
class ConvexHull_2D{
private:
    typedef Point<T> PT;
    vector<PT> dots;
    struct myhash{
        uint64_t operator()(const PT& a) const {
            uint64_t xx=0, yy=0;
            memcpy(&xx, &a.x, sizeof(a.x));
            memcpy(&yy, &a.y, sizeof(a.y));
            uint64_t ret = xx*17+yy*31;
            ret = (ret ^ (ret >> 16))*0x9E3779B1;
            ret = (ret ^ (ret >> 13))*0xC2B2AE35;
            ret = ret ^ xx;
            return (ret ^ (ret << 3)) * yy;
        }
    };
    unordered_set<PT, myhash> in_hull;
public:
    inline void init(){in_hull.clear();dots.clear();}
    void insert(const PT& x){dots.PB(x);}
    void solve(){
        sort(ALL(dots), [](const PT& a, const PT& b){
          return tie(a.x, a.y) < tie(b.x, b.y);
        });
        vector<PT> stk(SZ(dots)<<1);
        int top = 0;
        for(auto p: dots){
            while(top >= 2 and cross(p-stk[top-2], stk[
                top-1]-stk[top-2]) <= 0)
                top --;
            stk[top++] = p;
        }
        for(int i=SZ(dots)-2, t = top+1;i>=0;i--){
            while(top >= t and cross(dots[i]-stk[top
                -2], stk[top-1]-stk[top-2]) <= 0)
                top --;
            stk[top++] = dots[i];
        }
        stk.resize(top-1);
        swap(stk, dots);
        for(auto i: stk) in_hull.insert(i);
    }
    vector<PT> get(){return dots;}
    inline bool in_it(const PT& x){
        return in_hull.find(x)!=in_hull.end();
    }
};
```

5.7 SimulateAnnealing

```cpp
#include <random>
#include <functional>
#include <utility>
#include <algorithm>
using namespace std;

double getY(double);

int main(){
    int rr, ll;
    default_random_engine rEng(time(NULL));
    uniform_real_distribution<double> Range(-1,1);
    uniform_real_distribution<double> expR(0,1);
    auto Random=bind(Range,rEng);
    auto expRand=bind(expR,rEng);
    int step=0;
    double pace=rr-ll, mini=0.95; // need to search for
         it
    double x=max(min(Random()*pace+ll, rr), ll), y=getY
        (x);
    while(pace>=1e-7){
        double newX = max(min(x + Random()*pace, rr),
            ll);
        double newY = getY(newX);
        if(newY < y || expRand() < exp(-step))
            x=newX, y=newY;
        step++;
        pace*=mini;
    }
}

double getY(double x){
    // get y using x
    return x;
}
```

6 Stringology

6.1 Hash

```cpp
#include <string>
typedef long long lld;
const int N = 1000000;
class Hash{
private:
    const lld p = 127, q = 1208220623;
    int sz;
    lld prefix[N], power[N];
public:
    void init(const std::string &x){
        sz = x.size();
        prefix[0]=0;
```

```
        for(int i=1;i<=sz;i++) prefix[i]=((prefix[i-1]*
            p)%q+x[i-1])%q;
        power[0]=1;
        for(int i=1;i<=sz;i++) power[i]=(power[i-1]*p)%
            q;
    }
    lld query(int l, int r){
        // 1-base (l, r]
        return (prefix[r] - (prefix[l]*power[r-l])%q +
            q)%q;
    }
};
```

6.2 Suffix Array

```
//help by http://www.geeksforgeeks.org/suffix-array-set
    -2-a-nlognlogn-algorithm/
#include <bits/stdc++.h>
using namespace std;
#define PB push_back

struct sfx{
    int index;
    int r,nr;
};

char str[N + 10];
int len;

vector<sfx> srs[N + 10];
int mapping[N + 10];
sfx sa[N + 10];

bool cmp(sfx a,sfx b){
    if(a.r==b.r){
        return a.nr<b.nr;
    }else{
        return a.r<b.r;
    }
}

void SA();
void radixSort();

int main(){
    gets(str);
    len = strlen(str);
    SA();
    for(int i=0;i<len;i++){
        printf("%d\n",sa[i].index);
    }
    return 0;
}

void SA(){
    for(int i=0;i<len;i++){
        sa[i].index = i;
        sa[i].r=str[i];
        sa[i].nr=(i+1>=len)?0:str[i+1];
    }
    //sort(sa,sa+len,cmp);
    radixSort();
    for(int j=2;j<=len;j*=2){
        int cnt=1;
        int rr = sa[0].r;
        sa[0].r=cnt;
        mapping[sa[0].index]=0;
        for(int i=1;i<len;i++){
            if(sa[i].r == rr && sa[i].nr == sa[i-1].nr)
                {
                rr=sa[i].r;
                sa[i].r=cnt;
            }else{
                rr=sa[i].r;
                sa[i].r=++cnt;
            }
            mapping[sa[i].index]=i;
        }
        for(int i=0;i<len;i++){
            int nn = sa[i].index+j;
            sa[i].nr = (nn>=len)?0:sa[mapping[nn]].r;
        }
        //sort(sa, sa+len, cmp);
        radixSort();
    }
}

void radixSort(){
    int m = 0;
    for(int i=0;i<len;i++){
        srs[sa[i].nr].PB(sa[i]);
        m=max(m,sa[i].nr);
    }
    int cnt=0;
    for(int i=0;i<=m;i++){
        if(srs[i].empty())continue;
        for(auto j:srs[i]){
            sa[cnt++] = j;
        }
        srs[i].clear();
    }
    m = 0;
    for(int i=0;i<len;i++){
        srs[sa[i].r].PB(sa[i]);
        m=max(m,sa[i].r);
    }
    cnt=0;
    for(int i=0;i<=m;i++){
        if(srs[i].empty())continue;
        for(auto j:srs[i]){
            sa[cnt++] = j;
        }
        srs[i].clear();
    }
}
```

6.3 KMP

```
int F[N<<1];
void KMP(char s1[], char s2[], int n, int m){
    // make F[] for s1+'\0'+s2;
    char ss[N<<1];
    int len = n+m+1;
    for(int i=0;i<n;i++) ss[i] = s1[i];
    ss[n] = '\0';
    for(int i=0;i<m;i++) ss[i+1+n] = s2[i];
    F[0] = F[1] = 0;
    for(int i=1;i<len;i++){
        int j = F[i];
        while(j > 0 and ss[i]!=ss[j]) j = F[j];
        F[i+1] = (ss[i]==ss[j]?j+1:0);
    }
    // just find (F[len2+i] == len2), i from 1 to len+1
        for matching
}
```