Contents

Basic

vimrc

```
set nu
set tabstop=4
set softtabstop=4

set autoindent
set shiftwidth=4
set cindent

set smartindent

nmap <F9> :! clear ; g++ -std=c++11 -static  -Wall -O2
     % -o %.out; ./%.out
```

BigInt

```
struct Bigint{
  static const int LEN = 60;
  static const int BIGMOD = 10000;
  int s;
  int vl, v[LEN];
  //  vector<int> v;
  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);
  }
  int len() const { return vl; /* return SZ(v); */ }
  bool empty() const { return len() == 0; }
  void push_back(int x) { v[vl++] = x; /* v.PB(x); */ }
  void pop_back() { vl--; /* v.pop_back(); */ }
  int back() const { return v[vl-1]; /* return v.back()
      ; */ }
  void n() { while (!empty() && !back()) pop_back(); }
  void resize(int nl) {
    vl = nl; fill(v, v+vl, 0);
    //    v.resize(nl); // fill(ALL(v), 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 ? 1 : -1;
  if (s == -1) return -(-*this).cp3(-b);
  if (len() != b.len()) return len()>b.len()?1:-1;
  for (int i=len()-1; i>=0; i--)
    if (v[i]!=b.v[i]) return v[i]>b.v[i]?1:-1;
  return 0;
}
bool operator < (const Bigint &b)const{ return cp3(b)
    ==-1; }
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)
    ==1; }
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;
}
};
```

Mathmatics

Miller Rabin

```
typedef long long LL;

LL bin_pow(LL a, LL n, LL MOD){
  LL re=1;
  while (n>0){
    if (n&1)re = re*a %MOD;
    a = a*a %MOD;
    n>>=1;
  }
  return re;
}
bool is_prime(LL n){
  //static LL sprp[3] = { 2LL, 7LL, 61LL};
  static LL sprp[7] = { 2LL, 325LL, 9375LL,
    28178LL, 450775LL, 9780504LL,
    1795265022LL };
  if (n==1 || (n&1)==0 ) return n==2;
  int u=n-1, t=0;
  while ( (u&1)==0 ) u>>=1, t++;
  for (int i=0; i<7; i++){
    LL x = bin_pow( sprp[i]%n, u, n);
    if (x==0 || x==1 || x==n-1)continue;

    for (int j=1; j<t; j++){
      x=x*x%n;
      if (x==1 || x==n-1)break;
    }
    if (x==n-1)continue;
    return 0;
  }
  return 1;
}
```

ax+by=gcd(a,b)

```
typedef pair<int, int> pii;
pii extgcd(int a, int b){
  if(b == 0) return make_pair(1, 0);
  else{
    int p = a / b;
    pii q = extgcd(b, a % b);
    return make_pair(q.second, q.first - q.second * p);
  }
}
```

FFT

```
const double pi = atan(1.0)*4;
struct Complex {
    double x,y;
```

```
    Complex(double _x=0,double _y=0)
        :x(_x),y(_y) {}
    Complex operator + (Complex &tt) { return Complex(x
        +tt.x,y+tt.y); }
    Complex operator - (Complex &tt) { return Complex(x
        -tt.x,y-tt.y); }
    Complex operator * (Complex &tt) { return Complex(x
        *tt.x-y*tt.y,x*tt.y+y*tt.x); }
};
void fft(Complex *a, int n, int rev) {
    // n是大于等于相乘的两个数组长度的2的幂次
    // 从0开始表示长度，对a进行操作
    // rev==1进行DFT，==-1进行IDFT
    for (int i = 1,j = 0; i < n; ++ i) {
        for (int k = n>>1; k > (j^=k); k >>= 1);
        if (i<j) std::swap(a[i],a[j]);
    }
    for (int m = 2; m <= n; m <<= 1) {
        Complex wm(cos(2*pi*rev/m),sin(2*pi*rev/m));
        for (int i = 0; i < n; i += m) {
            Complex w(1.0,0.0);
            for (int j = i; j < i+m/2; ++ j) {
                Complex t = w*a[j+m/2];
                a[j+m/2] = a[j] - t;
                a[j] = a[j] + t;
                w = w * wm;
            }
        }
    }
    if (rev==-1) {
        for (int i = 0; i < n; ++ i) a[i].x /= n,a[i].y
             /= n;
    }
}
```

Geometry

Flow

Dinic

```
struct Edge{
  int from, to, cap, flow;
};

const int INF = 1<<29;
const int MAXV = 5003;
struct Dinic{ //O(VVE)
  int n, m, s, t;
  vector<Edge> edges;
  vector<int> G[MAXV];
  bool vis[MAXV];
  int d[MAXV];
  int cur[MAXV];

  void AddEdge(int from, int to, int cap){
    edges.push_back( {from,to,cap,0} );
    edges.push_back( {to,from,0,0} );
    m = edges.size();
    G[from].push_back(m-2);
    G[to].push_back(m-1);
  }

  bool dinicBFS(){
    memset(vis,0,sizeof(vis));
    queue<int> que;
    que.push(s); vis[s]=1;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (int ei:G[u]){
        Edge &e = edges[ei];
        if (!vis[e.to] && e.cap>e.flow ){
          vis[e.to]=1;
          d[e.to] = d[u]+1;
          que.push(e.to);
        }
      }
    }
    return vis[t];
  }

  int dinicDFS(int u, int a){
    if (u==t || a==0)return a;
    int flow=0, f;
    for (int &i=cur[u]; i<(int)G[u].size(); i++){
      Edge &e = edges[ G[u][i] ];
      if (d[u]+1!=d[e.to])continue;
      f = dinicDFS(e.to, min(a, e.cap-e.flow) );
      if (f>0){
        e.flow += f;
        edges[ G[u][i]^1 ].flow -=f;
        flow += f;
        a -= f;
        if (a==0)break;
      }
    }
    return flow;
  }

  int maxflow(int s, int t){
    this->s = s, this->t = t;
    int flow=0, mf;
    while ( dinicBFS() ){
      memset(cur,0,sizeof(cur));
      while ( (mf=diniDFS(s,INF)) )flow+=mf;
    }
    return flow;
  }
};
```

Graph

Strongly Connected Component(SCC)

```
#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

struct Scc{
int n, nScc, vst[MXN], bln[MXN];
vector<int> E[MXN], rE[MXN], vec;
void init(int _n){
  n = _n;
  for (int i=0; i<MXN; i++){
    E[i].clear();
    rE[i].clear();
  }
}
void add_edge(int u, int v){
  E[u].PB(v);
  rE[v].PB(u);
}
void DFS(int u){
  vst[u]=1;
  for (auto v : E[u])
    if (!vst[v]) DFS(v);
  vec.PB(u);
}
void rDFS(int u){
  vst[u] = 1;
  bln[u] = nScc;
  for (auto v : rE[u])
    if (!vst[v]) rDFS(v);
}
void solve(){
  nScc = 0;
  vec.clear();
  FZ(vst);
  for (int i=0; i<n; i++)
    if (!vst[i]) DFS(i);
  reverse(vec.begin(),vec.end());
  FZ(vst);
  for (auto v : vec){
    if (!vst[v]){
      rDFS(v);
```

```
      nScc++;
    }
  }
}
};
```

LCA

```
//lv紀錄深度
//father[多少冪次][誰]
//已經建好每個人的父親是誰 (father[0][i]已經建好)
//已經建好深度 (lv[i]已經建好)
void makePP(){
  for(int i = 1; i < 20; i++){
    for(int j = 2; j <= n; j++){
      father[i][j]=father[i-1][ father[i-1][j] ];
    }
  }
}
int find(int a, int b){
  if(lv[a] < lv[b]) swap(a,b);
  int need = lv[a] - lv[b];
  for(int i = 0; need!=0; i++){
    if(need&1) a=father[i][a];
    need >>= 1;
  }
  for(int i = 19 ;i >= 0 ;i--){
    if(father[i][a] != father[i][b]){
      a=father[i][a];
      b=father[i][b];
    }
  }
  return a!=b?father[0][a] : a;
}
```

Data Structure

Disjoint Set

```
struct DisjointSet{
    int n, fa[MAXN];

      void init(int size) {
            for (int i = 0; i <= size; i++) {
                    fa[i] = i;
                        }
              }

        void find(int x) {
              return fa[x] == x ? x : find(fa[x]);
                }

          void unite(int x, int y) {
                p[find(x)] = find(y);
                  }

} djs;
```

Sparse Table

```
const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
  int Sp[MAXN][lgN];
  function<int(int,int)> opt;
  void build(int n, int *a){ // 0 base
    for (int i=0 ;i<n; i++) Sp[i][0]=a[i];

    for (int h=1; h<lgN; h++){
      int len = 1<<(h-1), i=0;
      for (; i+len<n; i++)
        Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
      for (; i<n; i++)
        Sp[i][h] = Sp[i][h-1];
    }
  }
  int query(int l, int r){
    int h = __lg(r-l+1);
    int len = 1<<h;
    return opt( Sp[l][h] , Sp[r-len+1][h] );
  }
};
```

Treap

```
#include<bits/stdc++.h>
using namespace std;
template<class T,unsigned seed>class treap{
  public:
    struct node{
      T data;
      int size;
      node *l,*r;
      node(T d){
        size=1;
        data=d;
        l=r=NULL;
      }
      inline void up(){
        size=1;
        if(l)size+=l->size;
        if(r)size+=r->size;
      }
      inline void down(){
      }
    }*root;
    inline int size(node *p){return p?p->size:0;}
    inline bool ran(node *a,node *b){
      static unsigned x=seed;
      x=0xdefaced*x+1;
      unsigned all=size(a)+size(b);
      return (x%all+all)%all<size(a);
    }
    void clear(node *&p){
      if(p)clear(p->l),clear(p->r),delete p,p=NULL;
    }
    ~treap(){clear(root);}
    void split(node *o,node *&a,node *&b,int k){
      if(!k)a=NULL,b=o;
      else if(size(o)==k)a=o,b=NULL;
      else{
        o->down();
        if(k<=size(o->l)){
          b=o;
          split(o->l,a,b->l,k);
          b->up();
        }else{
          a=o;
          split(o->r,a->r,b,k-size(o->l)-1);
          a->up();
        }
      }
    }
    void merge(node *&o,node *a,node *b){
      if(!a||!b)o=a?a:b;
      else{
        if(ran(a,b)){
          a->down();
          o=a;
          merge(o->r,a->r,b);
        }else{
          b->down();
          o=b;
          merge(o->l,a,b->l);
        }
        o->up();
      }
    }
    void build(node *&p,int l,int r,T *s){
      if(l>r)return;
      int mid=(l+r)>>1;
      p=new node(s[mid]);
      build(p->l,l,mid-1,s);
```

```
      build(p->r,mid+1,r,s);
      p->up();
    }
    inline int rank(T data){
      node *p=root;
      int cnt=0;
      while(p){
        if(data<=p->data)p=p->l;
        else cnt+=size(p->l)+1,p=p->r;
      }
      return cnt;
    }
    inline void insert(node *&p,T data,int k){
      node *a,*b,*now;
      split(p,a,b,k);
      now=new node(data);
      merge(a,a,now);
      merge(p,a,b);
    }
};
treap<int ,20141223>bst;
int n,m,a,b;
int main(){
  //當成二分查找樹用
  while(~scanf("%d",&a))bst.insert(bst.root,a,bst.rank(
      a));
  while(~scanf("%d",&a))printf("%d\n",bst.rank(a));
  bst.clear(bst.root);
  return 0;
}
```

String

KMP

```
template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
  f[0]=-1, f[1]=0;
  for (int i=2; i<=n; i++){
    int w = f[i-1];
    while (w>=0 && s[w+1]!=s[i])w = f[w];
    f[i]=w+1;
  }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
  build_KMP(n,b,f);
  int ans=0;

  for (int i=1, w=0; i<=n; i++){
    while ( w>=0 && b[w+1]!=a[i] )w = f[w];
    w++;
    if (w==m){
      ans++;
      w=f[w];
    }
  }
  return ans;
}
```

Dark Code

輸入優化

```
#include <stdio.h>

char getc(){
  static const int bufsize = 1<<16;
  static char B[bufsize], *S=B, *T=B;
  return (S==T&&(T=(S=B)+fread(B,1,bufsize,stdin),S==T)
      ?0:*S++);
}

template <class T>
bool input(T& a){
  a=(T)0;
  register char p;
  while ((p = getc()) < '-')
    if (p==0 || p==EOF) return false;
  if (p == '-')
    while ((p = getc()) >= '0') a = a*10 - (p^'0');
  else {
    a = p ^ '0';
    while ((p = getc()) >= '0') a = a*10 + (p^'0');
  }
  return true;
}

template <class T, class... U>
bool input(T& a, U&... b){
  if (!input(a)) return false;
  return input(b...);
}
```

Search

Others

Persistence