代码库

Blazar

2017 年 11 月 19 日

目录

1 数论

1.1 快速求逆元

返回结果：

$$x^{-1}(mod)$$

使用条件：$x \in [0, mod)$ 并且 x 与 mod 互质

```
LL inv(LL a, LL p) {
    LL d, x, y;
    exgcd(a, p, d, x, y);
    return d == 1 ? (x + p) % p : -1;
}
```

1.2 莫比乌斯反演

```
#include<cstdio>
#include<string>
#include<cstring>
#include<algorithm>
using namespace std;
int mu[100001],prime[100001];
bool check[100001];
int tot;
inline void findmu()
{
     memset(check,false,sizeof(check));
     mu[1]=1;
     int i,j;
     for(i=2;i<=100000;i++)
     {
          if(!check[i])
          {
               tot++;
               prime[tot]=i;
               mu[i]=-1;
          }
          for(j=1;j<=tot;j++)
          {
               if(i*prime[j]>100000)
                    break;
               check[i*prime[j]]=true;
               if(i%prime[j]==0)
               {
                    mu[i*prime[j]]=0;
                    break;
               }
               else
                    mu[i*prime[j]]=-mu[i];
          }
     }
}
int sum[100001];
//找 [1,n],[1,m] 内互质的数的对数
inline long long solve(int n,int m)
{
     long long ans=0;
     if(n>m)
          swap(n,m);
     int i,la=0;
     for(i=1;i<=n;i=la+1)
     {
          la=min(n/(n/i),m/(m/i));
          ans+=(long long)(sum[la]-sum[i-1])*(n/i)*(m/i);
     }
     return ans;
}
int main()
{
     //freopen("b.in","r",stdin);
    // freopen("b.out","w",stdout);
     findmu();
     sum[0]=0;
     int i;
     for(i=1;i<=100000;i++)
          sum[i]=sum[i-1]+mu[i];
     int a,b,c,d,k;
     int T;
     scanf("%d",&T);
     while(T--)
     {
          scanf("%d%d%d%d%d",&a,&b,&c,&d,&k);
          long long ans=0;
     ans=solve(b/k,d/k)-solve((a-1)/k,d/k)-solve(b/k,(c-1)/k)+sol
          printf("%lld\n",ans);
     }
     return 0;
}
```

1.3 扩展欧几里德算法

返回结果：

$$ax + by = gcd(a, b)$$

时间复杂度：$\mathcal{O}(nlogn)$

```
LL exgcd(LL a, LL b, LL &x, LL &y) {
    if(!b) {
        x = 1;
        y = 0;
        return a;
    } else {
        LL d = exgcd(b, a % b, x, y);
        LL t = x;
        x = y;
        y = t - a / b * y;
        return d;
    }
}
```

1.4 中国剩余定理

返回结果：

$$x \equiv r_i (mod\ p_i)\ \ (0 \le i < n)$$

使用条件：p_i 需两两互质

```
LL china(int n, int *a, int *m) {
    LL M = 1, d, x = 0, y;
    for(int i = 0; i < n; i++)
        M *= m[i];
    for(int i = 0; i < n; i++) {
        LL w = M / m[i];
        d = exgcd(m[i], w, d, y);
        y = (y % M + M) % M;
        x = (x + y * w % M * a[i]) % M;
    }
    while(x < 0)x += M;
    return x;
}
```

1.5 中国剩余定理 2

```
//merge Ax=B and ax=b to A'x=B'
void merge(LL &A,LL &B,LL a,LL b){
    LL x,y;
    sol(A,-a,b-B,x,y);
    A=lcm(A,a);
    B=(a*y+b)%A;
    B=(B+A)%A;
}
```

1.6 组合数取模

```
LL prod = 1, P;
pair<LL, LL> comput(LL n, LL p, LL k) {
    if(n <= 1) return make_pair(0, 1);
    LL ans = 1, cnt = 0;
    ans = pow(prod, n / P, P);
    cnt = n / p;
    pair<LL, LL> res = comput(n / p, p, k);
    cnt += res.first;
    ans = ans * res.second % P;
    for(int i = n - n % P + 1; i <= n; i++)
    if(i % p)
            ans = ans * i % P;
    return make_pair(cnt, ans);
}
pair<LL, LL> calc(LL n, LL p, LL k) {
    prod = 1;
    P = pow(p, k, 1e18);
    for(int i = 1; i < P; i++)
    if(i % p)
      prod = prod * i % P;
    pair<LL, LL> res = comput(n, p, k);
    return res;
}
LL calc(LL n, LL m, LL p, LL k) {
    pair<LL, LL>A, B, C;
    LL P = pow(p, k, 1e18);
    A = calc(n, p, k);
    B = calc(m, p, k);
    C = calc(n - m, p, k);
    LL ans = 1;
    ans = pow(p, A.first - B.first - C.first, P);
    ans = ans * A.second % P * inv(B.second, P) %
 ↪   P * inv(C.second, P) % P;
    return ans;
}
```

1.7 卢卡斯定理

```
LL Lucas(LL n, LL m, LL p) {
    LL ans = 1;
    while(n && m) {
        LL a = n % p, b = m % p;
        if(a < b) return 0;
        ans = (ans * C(a, b, p)) % p;
        n /= p;
        m /= p;
    }
    return ans % p;
```

```
}
```

1.8 小步大步

返回结果：

$$a^x = b \ (mod \ p)$$

使用条件：p 为质数
时间复杂度：$\mathcal{O}(\sqrt{n})$

```
LL BSGS(LL a, LL b, LL p) {
    LL m = sqrt(p) + .5, v = inv(pw(a, m, p), p),
↪    e = 1;
    map<LL, LL> hash;
    hash[1] = 0;
    for(int i = 1; i < m; i++)
        e = e * a % p, hash[e] = i;
    for(int i = 0; i <= m; i++) {
        if(hash.count(b))
      return i * m + hash[b];
        b = b * v % p;
    }
    return -1;
}
```

1.9 Miller Rabin 素数测试

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19,
↪    23, 29, 31, 37};
bool check(long long n, int base) {
    long long n2 = n - 1, res;
    int s = 0;
    while(n2 % 2 == 0) n2 >>= 1, s++;
    res = pw(base, n2, n);
    if((res == 1) || (res == n - 1)) return 1;
    while(s--) {
        res = mul(res, res, n);
        if(res == n - 1) return 1;
    }
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n == 2)
        return true;
    if(n < 2 || n % 2 == 0)
        return false;
    for(int i = 0; i < 12 && BASE[i] < n; i++) {
        if(!check(n, BASE[i]))
            return false;
    }
    return true;
}
```

1.10 Pollard Rho 大数分解

时间复杂度：$\mathcal{O}(n^{1/4})$

```
LL prho(LL n, LL c) {
    LL i = 1, k = 2, x = rand() % (n - 1) + 1, y
↪    = x;
    while(1) {
        i++;
        x = (x * x % n + c) % n;
        LL d = __gcd((y - x + n) % n, n);
        if(d > 1 && d < n)return d;
        if(y == x)return n;
        if(i == k)y = x, k <<= 1;
    }
}
void factor(LL n, vector<LL>&fat) {
    if(n == 1)return;
    if(isprime(n)) {
        fat.push_back(n);
        return;
    }
    LL p = n;
    while(p >= n)p = prho(p, rand() % (n - 1) +
↪    1);
    factor(p, fat);
    factor(n / p, fat);
}
```

1.11 快速数论变换 (zky)

返回结果：

$$c_i = \sum_{0 \le j \le i} a_j \cdot b_{i-j} (mod) \ \ (0 \le i < n)$$

使用说明：$magic$ 是 mod 的原根
时间复杂度：$\mathcal{O}(nlogn)$

```
/*
{(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
      ,(377487361,7),(998244353,3),(1224736769,3)
      ,(1300234241,3),(1484783617,5)}
*/
int mo = 998244353, G = 3;
```

```
void NTT(int a[], int n, int f) {
    for(register int i = 0; i < n; i++)
        if(i < rev[i])
            swap(a[i], a[rev[i]]);
    for (register int i = 2; i <= n; i <<= 1) {
        static int exp[maxn];
        exp[0] = 1;
        exp[1] = pw(G, (mo - 1) / i);
        if(f == -1)exp[1] = pw(exp[1], mo - 2);
        for(register int k = 2; k < (i >> 1); k++)
            exp[k] = 1LL * exp[k - 1] * exp[1] % mo;
        for(register int j = 0; j < n; j += i) {
            for(register int k = 0; k < (i >> 1); k++) {
                register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
                register int A = pA, B = 1LL * pB * exp[k] % mo;
                pA = (A + B) % mo;
                pB = (A - B + mo) % mo;
            }
        }
    }
    if(f == -1) {
        int rv = pw(n, mo - 2) % mo;
        for(int i = 0; i < n; i++)
            a[i] = 1LL * a[i] * rv % mo;
    }
}
void mul(int m, int a[], int b[], int c[]) {
    int n = 1, len = 0;
    while(n < m)n <<= 1, len++;
    for (int i = 1; i < n; i++)
        rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
    NTT(a, n, 1);
    NTT(b, n, 1);
    for(int i = 0; i < n; i++)
        c[i] = 1LL * a[i] * b[i] % mo;
    NTT(c, n, -1);
}
```

1.12 原根

```
vector<LL>fct;
bool check(LL x, LL g) {
    for(int i = 0; i < fct.size(); i++)
        if(pw(g, (x - 1) / fct[i], x) == 1)
            return 0;
    return 1;
}
LL findrt(LL x) {
    LL tmp = x - 1;
    for(int i = 2; i * i <= tmp; i++) {
        if(tmp % i == 0) {
            fct.push_back(i);
            while(tmp % i == 0)tmp /= i;
        }
    }
    if(tmp > 1) fct.push_back(tmp);
    // x is 1,2,4,p^n,2p^n
    // x has phi(phi(x)) primitive roots
    for(int i = 2; i < int(1e9); i++)
    if(check(x, i))
            return i;
    return -1;
}
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n, int base) {
    long long n2 = n - 1, res;
    int s = 0;
    while(n2 % 2 == 0) n2 >>= 1, s++;
    res = pw(base, n2, n);
    if((res == 1) || (res == n - 1)) return 1;
    while(s--) {
        res = mul(res, res, n);
        if(res == n - 1) return 1;
    }
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n == 2)
        return true;
    if(n < 2 || n % 2 == 0)
        return false;
    for(int i = 0; i < 12 && BASE[i] < n; i++) {
        if(!check(n, BASE[i]))
            return false;
    }
    return true;
}
```

1.13 线性递推

```
//已知 a_0, a_1, ..., a_{m-1}\\
a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}\\
        求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}\\

void linear_recurrence(long long n, int m, int a[], int c[], int p) {
    long long v[M] = {1 % p}, u[M << 1], msk = !!n;
    for(long long i(n); i > 1; i >>= 1) {
        msk <<= 1;
    }
    for(long long x(0); msk; msk >>= 1, x <<= 1) {
        fill_n(u, m << 1, 0);
        int b(!!(n & msk));
        x |= b;
        if(x < m) {
            u[x] = 1 % p;
        } else {
            for(int i(0); i < m; i++) {
                for(int j(0), t(i + b); j < m; j++, t++) {
                    u[t] = (u[t] + v[i] * v[j]) % p;
                }
            }
            for(int i((m << 1) - 1); i >= m; i--) {
                for(int j(0), t(i - m); j < m; j++, t++) {
                    u[t] = (u[t] + c[j] * u[i]) % p;
                }
            }
        }
        copy(u, u + m, v);
    } //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1]
    for(int i(m); i < 2 * m; i++) {
        a[i] = 0;
        for(int j(0); j < m; j++) {
            a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        b[j] = 0;
        for(int i(0); i < m; i++) {
            b[j] = (b[j] + v[i] * a[i + j]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        a[j] = b[j];
    }
}
```

1.14 线性筛

```
void sieve() {
    f[1] = mu[1] = phi[1] = 1;
    for(int i = 2; i < maxn; i++) {
        if(!minp[i]) {
            minp[i] = i;
            minpw[i] = i;
            mu[i] = -1;
            phi[i] = i - 1;
            f[i] = i - 1;
            p[++p[0]] = i; // Case 1 prime
        }
        for(int j = 1; j <= p[0] && (LL)i * p[j] < maxn; j++) {
            minp[i * p[j]] = p[j];
            if(i % p[j] == 0) {
                // Case 2 not coprime
                minpw[i * p[j]] = minpw[i] * p[j];
                phi[i * p[j]] = phi[i] * p[j];
                mu[i * p[j]] = 0;
                if(i == minpw[i]) {
                    f[i * p[j]] = i * p[j] - i; // Special Case for f(p^k)
                } else {
                    f[i * p[j]] = f[i / minpw[i]] * f[minpw[i] * p[j]];
                }
                break;
            } else {
                // Case 3 coprime
                minpw[i * p[j]] = p[j];
                f[i * p[j]] = f[i] * f[p[j]];
                phi[i * p[j]] = phi[i] * (p[j] - 1);
                mu[i * p[j]] = -mu[i];
            }
        }
    }
```

```
}
```

1.15 直线下整点个数

返回结果：

$$\sum_{0\le i<n} \lfloor \frac{a+b\cdot i}{m} \rfloor$$

使用条件：$n,m>0$，$a,b\ge 0$

时间复杂度：$\mathcal{O}(nlogn)$

```
//calc \sum_{i=0}^{n-1} [(a+bi)/m]
// n,a,b,m > 0
LL solve(LL n, LL a, LL b, LL m) {
    if(b == 0)
        return n * (a / m);
    if(a >= m || b >= m)
        return n * (a / m) + (n - 1) * n / 2 * (b
    ↪  / m) + solve(n, a % m, b % m, m);
    return solve((a + b * n) / m, (a + b * n) %
    ↪  m, m, b);
}
```

2 数值

2.1 高斯消元

```
void Gauss(){
  int r,k;
  for(int i=0;i<n;i++){
    r=i;
    for(int j=i+1;j<n;j++)
      if(fabs(A[j][i])>fabs(A[r][i]))r=j;
    if(r!=i)for(int
  ↪  j=0;j<=n;j++)swap(A[i][j],A[r][j]);
    for(int k=i+1;k<n;k++){
      double f=A[k][i]/A[i][i];
      for(int j=i;j<=n;j++)A[k][j]-=f*A[i][j];
    }
  }
  for(int i=n-1;i>=0;i--){
    for(int j=i+1;j<n;j++)
      A[i][n]-=A[j][n]*A[i][j];
    A[i][n]/=A[i][i];
  }
  for(int i=0;i<n-1;i++)
    cout<<fixed<<setprecision(3)<<A[i][n]<<" ";
  cout<<fixed<<setprecision(3)<<A[n-1][n];
}
bool Gauss(){
  for(int i=1;i<=n;i++){
    int r=0;
    for(int j=i;j<=m;j++)
    if(a[j][i]){r=j;break;}
    if(!r)return 0;
    ans=max(ans,r);
    swap(a[i],a[r]);
    for(int j=i+1;j<=m;j++)
    if(a[j][i])a[j]^=a[i];
  }for(int i=n;i>=1;i--){
    for(int j=i+1;j<=n;j++)if(a[i][j])
    a[i][n+1]=a[i][n+1]^a[j][n+1];
  }return 1;
}
LL Gauss(){
  for(int i=0;i<n;i++)for(int
↪  j=0;j<n;j++)A[i][j]%=m;
  for(int i=0;i<n;i++)for(int
↪  j=0;j<n;j++)A[i][j]=(A[i][j]+m)%m;
  LL ans=n%2?-1:1;
  for(int i=0;i<n;i++){
    for(int j=i+1;j<n;j++){
      while(A[j][i]){
        LL t=A[i][i]/A[j][i];
        for(int k=0;k<n;k++)
        A[i][k]=(A[i][k]-A[j][k]*t%m+m)%m;
        swap(A[i],A[j]);
        ans=-ans;
      }
    }ans=ans*A[i][i]%m;
  }return (ans%m+m)%m;
}
int Gauss(){//求秩
  int r,now=-1;
  int ans=0;
  for(int i = 0; i <n; i++){
    r = now + 1;
    for(int j = now + 1; j < m; j++)
      if(fabs(A[j][i]) > fabs(A[r][i]))
        r = j;
    if (!sgn(A[r][i])) continue;
    ans++;
    now++;
    if(r != now)
      for(int j = 0; j < n; j++)
        swap(A[r][j], A[now][j]);
```

```
    for(int k = now + 1; k < m; k++){
      double t = A[k][i] / A[now][i];
      for(int j = 0; j < n; j++){
        A[k][j] -= t * A[now][j];
      }
    }
  }
  return ans;
}
```

2.2 线性基

```
const int N = 65;

LL bin[N], bas[N];
int pos[N], num;

void add(long long x, int m)
{
  for(int j = m; j >= 0; j--)
    if((x & bin[j]) && pos[j])
      x ^= bas[pos[j]];
  if(x == 0)
    return;
  for(int j = m; j >= 0; j--)
    if(x & bin[j])
    {
      pos[j] = ++num;
      bas[num] = x;
      break;
    }
}

int work(long long *a, int n, int m)
{
  num = 0;
  memset(pos, 0, sizeof(pos));
  for(int i = 1; i <= n; i++)
    add(a[i], m);
  return num;
}

typedef complex<double> cp;
const double pi = acos(-1);
void FFT(vector<cp>&num,int len,int ty){
    for(int i=1,j=0;i<len-1;i++){
        for(int k=len;j^=k>>=1,~j&k;);
        if(i<j)
            swap(num[i],num[j]);
    }
    for(int h=0;(1<<h)<len;h++){
        int step=1<<h,step2=step<<1;
        cp
↪   w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
        for(int i=0;i<len;i+=step2){
            cp w(1,0);
            for(int j=0;j<step;j++){
                cp &x=num[i+j+step];
                cp &y=num[i+j];
                cp d=w*x;
                x=y-d;
                y=y+d;
                w=w*w0;
            }
        }
    }
    if(ty==-1)
        for(int i=0;i<len;i++)

↪   num[i]=cp(num[i].real()/(double)len,num[i].imag());
}
vector<cp> mul(vector<cp>a,vector<cp>b){
    int len=a.size()+b.size();
    while((len&-len)!=len)len++;
    while(a.size()<len)a.push_back(cp(0,0));
    while(b.size()<len)b.push_back(cp(0,0));
    FFT(a,len,1);
    FFT(b,len,1);
    vector<cp>ans(len);
    for(int i=0;i<len;i++)
        ans[i]=a[i]*b[i];
    FFT(ans,len,-1);
    return ans;
}
```

2.3 1e9+7 FFT

```
// double 精度对 10^9 + 7 取模最多可以做到 2^20
const int MOD = 1000003;
const double PI = acos(-1);
typedef complex<double> Complex;
const int N = 65536, L = 15, MASK = (1 << L) - 1;
Complex w[N];
void FFTInit() {
  for (int i = 0; i < N; ++i)
    w[i] = Complex(cos(2 * i * PI / N), sin(2 * i
↪   * PI / N));
}
void FFT(Complex p[], int n) {
```

```
  for (int i = 1, j = 0; i < n - 1; ++i) {
    for (int s = n; j ^= s >>= 1, ~j & s;);
    if (i < j) swap(p[i], p[j]);
  }
  for (int d = 0; (1 << d) < n; ++d) {
    int m = 1 << d, m2 = m * 2, rm = n >> (d +
↪  1);
    for (int i = 0; i < n; i += m2) {
      for (int j = 0; j < m; ++j) {
        Complex &p1 = p[i + j + m], &p2 = p[i +
↪  j];
        Complex t = w[rm * j] * p1;
        p1 = p2 - t, p2 = p2 + t;
      } } }
}
Complex A[N], B[N], C[N], D[N];
void mul(int a[N], int b[N]) {
  for (int i = 0; i < N; ++i) {
    A[i] = Complex(a[i] >> L, a[i] & MASK);
    B[i] = Complex(b[i] >> L, b[i] & MASK);
  }
  FFT(A, N), FFT(B, N);
  for (int i = 0; i < N; ++i) {
    int j = (N - i) % N;
    Complex da = (A[i] - conj(A[j])) * Complex(0,
↪  -0.5),
        db = (A[i] + conj(A[j])) * Complex(0.5,
↪  0),
        dc = (B[i] - conj(B[j])) * Complex(0,
↪  -0.5),
        dd = (B[i] + conj(B[j])) * Complex(0.5,
↪  0);
    C[j] = da * dd + da * dc * Complex(0, 1);
    D[j] = db * dd + db * dc * Complex(0, 1);
  }
  FFT(C, N), FFT(D, N);
  for (int i = 0; i < N; ++i) {
    long long da = (long long)(C[i].imag() / N +
↪  0.5) % MOD,
          db = (long long)(C[i].real() / N + 0.5)
↪  % MOD,
          dc = (long long)(D[i].imag() / N + 0.5)
↪  % MOD,
          dd = (long long)(D[i].real() / N + 0.5)
↪  % MOD;
    a[i] = ((dd << (L * 2)) + ((db + dc) << L) +
↪  da) % MOD;
  }
}
```

2.4 单纯形法求解线性规划

返回结果：

$$max\{c_{1\times m}\cdot x_{m\times 1} \mid x_{m\times 1} \geq 0_{m\times 1}, a_{n\times m}\cdot x_{m\times 1} \leq b_{n\times 1}\}$$

```
namespace LP{
  const int maxn=233;
  double a[maxn][maxn];
  int Ans[maxn],pt[maxn];
  int n,m;
  void pivot(int l,int i){
    double t;
    swap(Ans[l+n],Ans[i]);
    t=-a[l][i];
    a[l][i]=-1;
    for(int j=0;j<=n;j++)a[l][j]/=t;
    for(int j=0;j<=m;j++){
      if(a[j][i]&&j!=l){
        t=a[j][i];
        a[j][i]=0;
        for(int k=0;k<=n;k++)a[j][k]+=t*a[l][k];
      }
    }
  }
  vector<double> solve(vector<vector<double>
↪  >A,vector<double>B,vector<double>C){
    n=C.size();
    m=B.size();
    for(int i=0;i<C.size();i++)
      a[0][i+1]=C[i];
    for(int i=0;i<B.size();i++)
      a[i+1][0]=B[i];

    for(int i=0;i<m;i++)
      for(int j=0;j<n;j++)
        a[i+1][j+1]=-A[i][j];

    for(int i=1;i<=n;i++)Ans[i]=i;

    double t;
    for(;;){
      int l=0;t=-eps;
      for(int
↪  j=1;j<=m;j++)if(a[j][0]<t)t=a[l=j][0];
      if(!l)break;
      int i=0;
      for(int
↪  j=1;j<=n;j++)if(a[l][j]>eps){i=j;break;}
      if(!i){
```

```
        puts("Infeasible");
        return vector<double>();
      }
      pivot(l,i);
    }
    for(;;){
      int i=0;t=eps;
      for(int
↪   j=1;j<=n;j++)if(a[0][j]>t)t=a[0][i=j];
      if(!i)break;
      int l=0;
      t=1e30;
      for(int j=1;j<=m;j++)if(a[j][i]<-eps){
        double tmp;
        tmp=-a[j][0]/a[j][i];
        if(t>tmp)t=tmp,l=j;
      }
      if(!l){
        puts("Unbounded");
        return vector<double>();
      }
      pivot(l,i);
    }
    vector<double>x;
    for(int i=n+1;i<=n+m;i++)pt[Ans[i]]=i-n;
    for(int
↪   i=1;i<=n;i++)x.push_back(pt[i]?a[pt[i]][0]:0);
    return x;
  }
}
```

2.5 自适应辛普森

```
double area(const double &left, const double
↪   &right) {
    double mid = (left + right) / 2;
    return (right - left) * (calc(left) + 4 *
↪   calc(mid) + calc(right)) / 6;
}

double simpson(const double &left, const double
↪   &right,
               const double &eps, const double
↪   &area_sum) {
    double mid = (left + right) / 2;
    double area_left = area(left, mid);
    double area_right = area(mid, right);
    double area_total = area_left + area_right;
    if (std::abs(area_total - area_sum) < 15 *
↪   eps) {
        return area_total + (area_total -
↪   area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2,
↪   area_right);
}

double simpson(const double &left, const double
↪   &right, const double &eps) {
    return simpson(left, right, eps, area(left,
↪   right));
}
```

2.6 多项式求根

```
const double eps=1e-12;
double a[10][10];
typedef vector<double> vd;
int sgn(double x) { return x < -eps ? -1 : x >
↪   eps; }
double mypow(double x,int num){
  double ans=1.0;
  for(int i=1;i<=num;++i)ans*=x;
  return ans;
}
double f(int n,double x){
  double ans=0;
  for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
  return ans;
}
double getRoot(int n,double l,double r){
  if(sgn(f(n,l))==0)return l;
  if(sgn(f(n,r))==0)return r;
  double temp;
  if(sgn(f(n,l))>0)temp=-1;else temp=1;
  double m;
  for(int i=1;i<=10000;++i){
    m=(l+r)/2;
    double mid=f(n,m);
    if(sgn(mid)==0){
      return m;
    }
    if(mid*temp<0)l=m;else r=m;
  }
  return (l+r)/2;
}
```

```
vd did(int n){
  vd ret;
  if(n==1){
    ret.push_back(-1e10);
    ret.push_back(-a[n][0]/a[n][1]);
    ret.push_back(1e10);
    return ret;
  }
  vd mid=did(n-1);
  ret.push_back(-1e10);
  for(int i=0;i+1<mid.size();++i){
    int
↪   t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
    if(t1*t2>0)continue;
    ret.push_back(getRoot(n,mid[i],mid[i+1]));
  }
  ret.push_back(1e10);
  return ret;
}
int main(){
  int n; scanf("%d",&n);
  for(int i=n;i>=0;--i){
    scanf("%lf",&a[n][i]);
  }
  for(int i=n-1;i>=0;--i)
    for(int
↪   j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
  vd ans=did(n);
  sort(ans.begin(),ans.end());
  for(int
↪   i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
  return 0;
}
```

2.7 快速求逆

```
long long inverse(const long long &x, const long
↪   long &mod) {
    if (x == 1) {
        return 1;
    } else {
        return (mod - mod / x) * inverse(mod % x,
↪   mod) % mod;
    }
}
```

2.8 魔幻多项式

多项式求逆

原理： 令 $G(x) = x * A - 1$（其中 A 是一个多项式系数），根据牛顿迭代法有：

$$F_{t+1}(x) \equiv F_t(x) - \frac{F_t(x) * A(x) - 1}{A(x)}$$
$$\equiv 2F_t(x) - F_t(x)^2 * A(x) \pmod{x^{2t}}$$

注意事项：

1. $F(x)$ 的常数项系数必然不为 0，否则没有逆元；
2. 复杂度是 $O(n \log n)$ 但是常数比较大（10^5 大概需要 0.3 秒左右）；
3. 传入的两个数组必须不同，但传入的次数界没有必要是 2 的次幂；

```
void getInv(int *a, int *b, int n) {
  static int tmp[MAXN];
  b[0] = fpm(a[0], MOD - 2, MOD);
  for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
    for (; M <= 3 * (c - 1); M <<= 1);
    meminit(b, c, M);
    meminit(tmp, c, M);
    memcopy(tmp, a, 0, c);
    DFT(tmp, M, 0);
    DFT(b, M, 0);
    for (int i = 0; i < M; i++) {
      b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] *
↪   b[i] % MOD + MOD) % MOD;
    }
    DFT(b, M, 1);
    meminit(b, c, M);
  }
}
```

多项式除法

作用： 给出两个多项式 $A(x)$ 和 $B(x)$，求两个多项式 $D(x)$ 和 $R(x)$ 满足：

$$A(x) \equiv D(x)B(x) + R(x) \pmod{x^n}$$

注意事项：

1. 常数比较大概为 6 倍 FFT 的时间，即大约 10^5 的数据 0.07s 左右；

2. 传入两个多项式的次数界，没有必要是 2 的次幂，但是要保证除数多项式不为 0。

```
void divide(int n, int m, int *a, int *b, int *d,
↪   int *r) {
↪   // n、m 分别为多项式 A（被除数）和 B（除数）的次数界
  static int M, tA[MAXN], tB[MAXN], inv[MAXN],
↪   tD[MAXN];
  for (; n > 0 && a[n - 1] == 0; n--);
  for (; m > 0 && b[m - 1] == 0; m--);
  for (int i = 0; i < n; i++) tA[i] = a[n - i -
↪   1];
  for (int i = 0; i < m; i++) tB[i] = b[m - i -
↪   1];
  for (M = 1; M <= n - m + 1; M <<= 1);
  meminit(tB, m, M);
  getInv(tB, inv, M);
  for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
  meminit(inv, n - m + 1, M);
  meminit(tA, n - m + 1, M);
  DFT(inv, M, 0);
  DFT(tA, M, 0);
  for (int i = 0; i < M; i++) {
    d[i] = 1ll * inv[i] * tA[i] % MOD;
  }
  DFT(d, M, 1);
  std::reverse(d, d + n - m + 1);
  for (M = 1; M <= n; M <<= 1);
  memcopy(tB, b, 0, m); meminit(tB, m, M);
  memcopy(tD, d, 0, n - m + 1); meminit(tD, n - m
↪   + 1, M);
  DFT(tD, M, 0);
  DFT(tB, M, 0);
  for (int i = 0; i < M; i++) {
    r[i] = 1ll * tD[i] * tB[i] % MOD;
  }
  DFT(r, M, 1);
  meminit(r, n, M);
  for (int i = 0; i < n; i++) {
    r[i] = (a[i] - r[i] + MOD) % MOD;
  }
}
```

3 数据结构

3.1 lct

```
struct LCT
{
  int fa[N], c[N][2], rev[N], sz[N];

  void update(int o) {
    sz[o] = sz[c[o][0]] + sz[c[o][1]] + 1;
  }
  void pushdown(int o) {
    if(rev[o]) {
      rev[o] = 0;
      rev[c[o][0]] ^= 1;
      rev[c[o][1]] ^= 1;
      swap(c[o][0], c[o][1]);
    }
  }
  bool ch(int o) {
    return o == c[fa[o]][1];
  }
  bool isroot(int o) {
    return c[fa[o]][0] != o && c[fa[o]][1] != o;
  }
  void setc(int x, int y, bool d) {
    if(x) fa[x] = y;
    if(y) c[y][d] = x;
  }
  void rotate(int x) {
    if(isroot(x)) return;
    int p = fa[x], d = ch(x);
    if(isroot(p)) fa[x] = fa[p];
    else setc(x, fa(p), ch(p));
    setc(c[x][d^1], p, d);
    setc(p, x, d^1);
    update(p);
    update(x);
  }
  void splay(int x) {
    static int q[N], top;
    int y = q[top = 1] = x;
    while(!isroot(y)) q[++top] = y = fa[y];
    while(top) pushdown(q[top--]);
    while(!isroot(x)) {
      if(!isroot(fa[x]))
        rotate(ch(fa[x]) == ch(x) ? fa[x] : x);
      rotate(x);
    }
  }
```

```
  void access(int x) {
    for(int y = 0; x; y = x, x = fa[x])
      splay(x), c[x][1] = y, update(x);
  }
  void makeroot(int x) {
    access(x), splay(x), rev(x) ^= 1;
  }
  void link(int x, int y) {
    makeroot(x), fa[x] = y, splay(x);
  }
  void cut(int x, int y) {
    makeroot(x);
    access(y);
    splay(y);
    c[y][0] = fa[x] = 0;
  }
};
```

3.2 可持久化 Trie

```
int Pre[N];
int n, q, Len, cnt, Lstans;
char s[N];
int First[N], Last[N];
int Root[N];
int Trie_tot;
struct node{
    int To[30];
    int Lst;
}Trie[N];
int tot;
struct node1{
    int L, R, Lson, Rson, Sum;
}tree[N * 25];
int Build(int L, int R){
    ++tot;
    tree[tot].L = L;
    tree[tot].R = R;
    tree[tot].Lson = tree[tot].Rson =
↪  tree[tot].Sum = 0;
    if (L == R) return tot;
    int s = tot;
    int mid = (L + R) >> 1;
    tree[s].Lson = Build(L, mid);
    tree[s].Rson = Build(mid + 1, R);
    return s;
}
int Same(int x){
    ++tot;
    tree[tot] = tree[x];
    return tot;
}
int Add(int Lst, int pos){
    int s = Same(Lst);
    tree[s].Sum++;
    if (tree[s].L == tree[s].R) return s;
    int Mid = (tree[s].L + tree[s].R) >> 1;
    if (pos <= Mid) tree[s].Lson =
↪  Add(tree[Lst].Lson, pos);
    else tree[s].Rson = Add(tree[Lst].Rson, pos);
    return s;
}

int Ask(int Lst, int Cur, int L, int R, int pos){
    if (L >= pos) return 0;
    if (R < pos) return tree[Cur].Sum -
↪  tree[Lst].Sum;
    int Mid = (L + R) >> 1;
    int Ret = Ask(tree[Lst].Lson, tree[Cur].Lson,
↪  L, Mid, pos);
    Ret += Ask(tree[Lst].Rson, tree[Cur].Rson,
↪  Mid + 1, R, pos);
    return Ret;
}

int main(){
    while (scanf("%d", &n) == 1){
        for (int i = 1; i <= Trie_tot; i++){
            for (int j = 1; j <= 26; j++)
                Trie[i].To[j] = 0;
            Trie[i].Lst = 0;
        }
        Trie_tot = 1;
        cnt = 0;
        for (int ii = 1; ii <= n; ii++){
            scanf("%s", s + 1);
            Len = strlen(s + 1);
            int Cur = 1;
            First[ii] = cnt + 1;
            for (int i = 1; i <= Len; i++){
                int ch = s[i] - 'a' + 1;
                if (Trie[Cur].To[ch] == 0){
                    ++Trie_tot;
                    Trie[Cur].To[ch] = Trie_tot;
                }
                Cur = Trie[Cur].To[ch];
                Pre[++cnt] = Trie[Cur].Lst;
                Trie[Cur].Lst = ii;
```

```
            }
            Last[ii] = cnt;
        }
        tot = 0;
        Root[0] = Build(0, n);
        for (int i = 1; i <= cnt; i++){
            Root[i] = Add(Root[i - 1], Pre[i]);
        }
        Lstans = 0;
        scanf("%d", &q);
        for (int ii = 1; ii <= q; ii++){
            int L, R;
            scanf("%d%d", &L, &R);
            L = (L + Lstans) % n + 1;
            R = (R + Lstans) % n + 1;
            if (L > R) swap(L, R);
            int Ret = Ask(Root[First[L] - 1],
↪   Root[Last[R]], 0, n, L);
            printf("%d\n", Ret);
            Lstans = Ret;
        }
    }
    return 0;
}
```

3.3 k-d 树

```
struct Point{
  int data[MAXK], id;
}p[MAXN];

struct KdNode{
  int l, r;
  Point p, dmin, dmax;
  KdNode() {}
  KdNode(const Point &rhs) : l(0), r(0), p(rhs),
↪   dmin(rhs), dmax(rhs) {}
  inline void merge(const KdNode &rhs) {
    for (register int i = 0; i < k; i++) {
      dmin.data[i] = std::min(dmin.data[i],
↪   rhs.dmin.data[i]);
      dmax.data[i] = std::max(dmax.data[i],
↪   rhs.dmax.data[i]);
    }
  }
  inline long long getMinDist(const Point
↪   &rhs)const {
    register long long ret = 0;
    for (register int i = 0; i < k; i++) {
      if (dmin.data[i] <= rhs.data[i] &&
↪   rhs.data[i] <= dmax.data[i]) continue;
      ret += std::min(1ll * (dmin.data[i] -
↪   rhs.data[i]) * (dmin.data[i] - rhs.data[i]),
        1ll * (dmax.data[i] - rhs.data[i]) *
↪   (dmax.data[i] - rhs.data[i]));
    }
    return ret;
  }
  long long getMaxDist(const Point &rhs) {
    long long ret = 0;
    for (register int i = 0; i < k; i++) {
      int tmp = std::max(std::abs(dmin.data[i] -
↪   rhs.data[i]),
          std::abs(dmax.data[i] - rhs.data[i]));
      ret += 1ll * tmp * tmp;
    }
    return ret;
  }
}tree[MAXN * 4];

struct Result{
  long long dist;
  Point d;
  Result() {}
  Result(const long long &dist, const Point &d) :
↪   dist(dist), d(d) {}
  bool operator >(const Result &rhs)const {
    return dist > rhs.dist || (dist == rhs.dist
↪   && d.id < rhs.d.id);
  }
  bool operator <(const Result &rhs)const {
    return dist < rhs.dist || (dist == rhs.dist
↪   && d.id > rhs.d.id);
  }
};

inline long long sqrdist(const Point &a, const
↪   Point &b) {
  register long long ret = 0;
  for (register int i = 0; i < k; i++) {
    ret += 1ll * (a.data[i] - b.data[i]) *
↪   (a.data[i] - b.data[i]);
  }
  return ret;
}

inline int alloc() {
  size++;
```

```cpp
  tree[size].l = tree[size].r = 0;
  return size;
}

void build(const int &depth, int &rt, const int &l, const int &r) {
  if (l > r) return;
  register int middle = l + r >> 1;
  std::nth_element(p + l, p + middle, p + r + 1,
    [=](const Point &a, const Point &b){return a.data[depth] < b.data[depth];});
  tree[rt = alloc()] = KdNode(p[middle]);
  if (l == r) return;
  build((depth + 1) % k, tree[rt].l, l, middle - 1);
  build((depth + 1) % k, tree[rt].r, middle + 1, r);
  if (tree[rt].l)
    tree[rt].merge(tree[tree[rt].l]);
  if (tree[rt].r)
    tree[rt].merge(tree[tree[rt].r]);
}

std::priority_queue<Result, std::vector<Result>, std::greater<Result> > heap;

void getMinKth(const int &depth, const int &rt, const int &m, const Point &d) { // 求 K 近点
  Result tmp = Result(sqrdist(tree[rt].p, d), tree[rt].p);
  if ((int)heap.size() < m) {
    heap.push(tmp);
  } else if (tmp < heap.top()) {
    heap.pop();
    heap.push(tmp);
  }
  int x = tree[rt].l, y = tree[rt].r;
  if (x != 0 && y != 0 && sqrdist(d, tree[x].p) > sqrdist(d, tree[y].p)) std::swap(x, y);
  if (x != 0 && ((int)heap.size() < m || tree[x].getMinDist(d) < heap.top().dist)) {
    getMinKth((depth + 1) % k, x, m, d);
  }
  if (y != 0 && ((int)heap.size() < m || tree[y].getMinDist(d) < heap.top().dist)) {
    getMinKth((depth + 1) % k, y, m, d);
  }
}

void getMaxKth(const int &depth, const int &rt, const int &m, const Point &d) { // 求 K 远点
  Result tmp = Result(sqrdist(tree[rt].p, d), tree[rt].p);
  if ((int)heap.size() < m) {
    heap.push(tmp);
  } else if (tmp > heap.top()) {
    heap.pop();
    heap.push(tmp);
  }
  int x = tree[rt].l, y = tree[rt].r;
  if (x != 0 && y != 0 && sqrdist(d, tree[x].p) < sqrdist(d, tree[y].p)) std::swap(x, y);
  if (x != 0 && ((int)heap.size() < m || tree[x].getMaxDist(d) >= heap.top().dist)) { // 这里的 >= 是因为在距离相等的时候需要按照 id 排序
    getMaxKth((depth + 1) % k, x, m, d);
  }
  if (y != 0 && ((int)heap.size() < m || tree[y].getMaxDist(d) >= heap.top().dist)) {
    getMaxKth((depth + 1) % k, y, m, d);
  }
}
```

3.4 树上莫队

```cpp
#include <bits/stdc++.h>

using namespace std;

const int N = 40005;
const int M = 100005;
const int LOGN = 17;

int n, m;
int w[N];
vector<int> g[N];
int bid[N << 1];

struct Query
{
  int l, r, extra, i;
  friend bool operator < (const Query &a, const Query &b)
  {
    if(bid[a.l] != bid[b.l])
      return bid[a.l] < bid[b.l];
    return a.r < b.r;
  }
```

```
} q[M];

void input()
{
  vector<int> vs;
  scanf("%d%d", &n, &m);
  for(int i = 1; i <= n; i++)
  {
    scanf("%d", &w[i]);
    vs.push_back(w[i]);
  }
  sort(vs.begin(), vs.end());
  vs.resize(unique(vs.begin(), vs.end()) - vs.begin());
  for(int i = 1; i <= n; i++)
    w[i] = lower_bound(vs.begin(), vs.end(), w[i]) - vs.begin() + 1;
  for(int i = 2; i <= n; i++)
  {
    int a, b;
    scanf("%d%d", &a, &b);
    g[a].push_back(b);
    g[b].push_back(a);
  }
  for(int i = 1; i <= m; i++)
  {
    scanf("%d%d", &q[i].l, &q[i].r);
    q[i].i = i;
  }
}

int dfs_clock;
int st[N], ed[N];
int fa[N][LOGN], dep[N];
int col[N << 1];
int id[N << 1];

void dfs(int x, int p)
{
  col[st[x] = ++dfs_clock] = w[x];
  id[st[x]] = x;
  fa[x][0] = p; dep[x] = dep[p] + 1;
  for(int i = 0; fa[x][i]; i++)
    fa[x][i + 1] = fa[fa[x][i]][i];
  for(auto y: g[x])
    if(y != p)
      dfs(y, x);
  col[ed[x] = ++dfs_clock] = w[x];
  id[ed[x]] = x;
}

int lca(int x, int y)
{
  if(dep[x] < dep[y]) swap(x, y);
  for(int i = LOGN - 1; i >= 0; i--)
    if(dep[fa[x][i]] >= dep[y])
      x = fa[x][i];
  if(x == y) return x;
  for(int i = LOGN - 1; i >= 0; i--)
    if(fa[x][i] != fa[y][i])
      x = fa[x][i], y = fa[y][i];
  return fa[x][0];
}

void prepare()
{
  dfs_clock = 0;
  dfs(1, 0);
  int BS = (int)sqrt(dfs_clock + 0.5);
  for(int i = 1; i <= dfs_clock; i++)
    bid[i] = (i + BS - 1) / BS;
  for(int i = 1; i <= m; i++)
  {
    int a = q[i].l;
    int b = q[i].r;
    int c = lca(a, b);
    if(st[a] > st[b]) swap(a, b);
    if(c == a)
    {
      q[i].l = st[a];
      q[i].r = st[b];
      q[i].extra = 0;
    }
    else
    {
      q[i].l = ed[a];
      q[i].r = st[b];
      q[i].extra = c;
    }
  }
  sort(q + 1, q + m + 1);
}

int curans;
int ans[M];
int cnt[N];
bool state[N];
```

```
void rev(int x)
{
  int &c = cnt[col[x]];
  curans -= !!c;
  c += (state[id[x]] ^= 1) ? 1 : -1;
  curans += !!c;
}

void solve()
{
  prepare();
  curans = 0;
  memset(cnt, 0, sizeof(cnt));
  memset(state, 0, sizeof(state));

  int l = 1, r = 0;
  for(int i = 1; i <= m; i++)
  {
    while(l < q[i].l) rev(l++);
    while(l > q[i].l) rev(--l);
    while(r < q[i].r) rev(++r);
    while(r > q[i].r) rev(r--);
    if(q[i].extra) rev(st[q[i].extra]);
    ans[q[i].i] = curans;
    if(q[i].extra) rev(st[q[i].extra]);
  }
  for(int i = 1; i <= m; i++)
    printf("%d\n", ans[i]);
}

int main()
{
  input();
  solve();
  return 0;
}
```

3.5 树状数组 kth

```
int find(int k){
    int cnt=0,ans=0;
    for(int i=22;i>=0;i--){
        ans+=(1<<i);
        if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
        else cnt+=d[ans];
    }
    return ans+1;
}
```

3.6 虚树

```
int a[maxn*2],sta[maxn*2];
int top=0,k;
void build(){
    top=0;
    sort(a,a+k,bydfn);
    k=unique(a,a+k)-a;
    sta[top++]=1;_n=k;
    for(int i=0;i<k;i++){
        int LCA=lca(a[i],sta[top-1]);
        while(dep[LCA]<dep[sta[top-1]]){
            if(dep[LCA]>=dep[sta[top-2]]){
                add_edge(LCA,sta[--top]);

↪   if(sta[top-1]!=LCA)sta[top++]=LCA;
                break;

↪   }add_edge(sta[top-2],sta[top-1]);top--;
        }if(sta[top-1]!=a[i])sta[top++]=a[i];
    }
    while(top>1)
        add_edge(sta[top-2],sta[top-1]),top--;
  for(int i=0;i<k;i++)inr[a[i]]=1;
}
```

4 图论

4.1 点双连通分量 (lyx)

```
#define SZ(x) ((int)x.size())

const int N = 400005;
↪   // N 开 2 倍点数，因为新树会加入最多 n 个新点
const int M = 200005;

vector<int> g[N];

int bccno[N], bcc_cnt;
vector<int> bcc[N];
bool iscut[N];

struct Edge {
  int u, v;
} stk[M << 2];
int top; // 注意栈大小为边数 4 倍
int dfn[N], low[N], dfs_clock;

void dfs(int x, int fa)
```

```cpp
{
  low[x] = dfn[x] = ++dfs_clock;
  int child = 0;
  for(int i = 0; i < SZ(g[x]); i++) {
    int y = g[x][i];
    if(!dfn[y]) {
      child++;
      stk[++top] = (Edge){x, y};
      dfs(y, x);
      low[x] = min(low[x], low[y]);
      if(low[y] >= dfn[x]) {
        iscut[x] = true;
        bcc[++bcc_cnt].clear();
        for(;;) {
          Edge e = stk[top--];
          if(bccno[e.u] != bcc_cnt) {
↪   bcc[bcc_cnt].push_back(e.u); bccno[e.u] =
↪   bcc_cnt; }
          if(bccno[e.v] != bcc_cnt) {
↪   bcc[bcc_cnt].push_back(e.v); bccno[e.v] =
↪   bcc_cnt; }
          if(e.u == x && e.v == y) break;
        }
      }
    } else if(y != fa && dfn[y] < dfn[x]) {
      stk[++top] = (Edge){x, y};
      low[x] = min(low[x], dfn[y]);
    }
  }
  if(fa == 0 && child == 1) iscut[x] = false;
}

void find_bcc()
↪   // 求点双联通分量，需要时手动 1 到 n 清空，1-based
{
  memset(dfn, 0, sizeof(dfn));
  memset(iscut, 0, sizeof(iscut));
  memset(bccno, 0, sizeof(bccno));
  dfs_clock = bcc_cnt = 0;
  for(int i = 1; i <= n; i++)
    if(!dfn[i])
      dfs(i, 0);
}

vector<int> G[N];

void prepare() { // 建出缩点后的树
  for(int i = 1; i <= n + bcc_cnt; i++)
    G[i].clear();
  for(int i = 1; i <= bcc_cnt; i++) {
    int x = i + n;
    for(int j = 0; j < SZ(bcc[i]); j++) {
      int y = bcc[i][j];
      G[x].push_back(y);
      G[y].push_back(x);
    }
  }
}
```

4.2 Hopcoft-Karp 求最大匹配

```cpp
int matchx[N], matchy[N], level[N];

bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i)
↪   {
        int y = edge[x][i];
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w]
↪   && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}

int solve() {
    std::fill(matchx, matchx + n, -1);
    std::fill(matchy, matchy + m, -1);
    for (int answer = 0; ; ) {
        std::vector<int> queue;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                queue.push_back(i);
            } else {
                level[i] = -1;
            }
        }
        for (int head = 0; head <
↪   (int)queue.size(); ++head) {
            int x = queue[head];
            for (int i = 0; i <
↪   (int)edge[x].size(); ++i) {
                int y = edge[x][i];
```

```
                int w = matchy[y];
                if (w != -1 && level[w] < 0) {
                    level[w] = level[x] + 1;
                    queue.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1 && dfs(i)) {
                delta++;
            }
        }
        if (delta == 0) {
            return answer;
        } else {
            answer += delta;
        }
    }
}
```

4.3 KM 带权匹配

注意事项：最小权完美匹配，复杂度为 $\mathcal{O}(|V|^3)$。

```
int DFS(int x){
    visx[x] = 1;
    for (int y = 1;y <= ny;y ++){
        if (visy[y]) continue;
        int t = lx[x] + ly[y] - w[x][y];
        if (t == 0) {
            visy[y] = 1;
            if (link[y] == -1||DFS(link[y])){
                link[y] = x;
                return 1;
            }
        }
        else slack[y] = min(slack[y],t);
    }
    return 0;
}
int KM(){
    int i,j;
    memset(link,-1,sizeof(link));
    memset(ly,0,sizeof(ly));
    for (i = 1; i <= nx; i++)
        for (j = 1, lx[i] = -inf; j <= ny; j++)
          lx[i] = max(lx[i],w[i][j]);
    for (int x = 1; x <= nx; x++){
        for (i = 1; i <= ny; i++) slack[i] = inf;
        while (true) {
            memset(visx, 0, sizeof(visx));
            memset(visy, 0, sizeof(visy));
            if (DFS(x)) break;
            int d = inf;
            for (i = 1; i <= ny;i++)
                if (!visy[i] && d > slack[i]) d =
↪  slack[i];
            for (i = 1; i <= nx; i++)
                if (visx[i]) lx[i] -= d;
            for (i = 1; i <= ny; i++)
                if (visy[i]) ly[i] += d;
                else slack[i] -= d;
        }
    }
    int res = 0;
    for (i = 1;i <= ny;i ++)
        if (link[i] > -1) res += w[link[i]][i];
    return res;
}
```

4.4 2-SAT 问题

```
int stamp, comps, top;
int dfn[N], low[N], comp[N], stack[N];

void add(int x, int a, int y, int b) {
    edge[x << 1 | a].push_back(y << 1 | b);
}

void tarjan(int x) {
    dfn[x] = low[x] = ++stamp;
    stack[top++] = x;
    for (int i = 0; i < (int)edge[x].size(); ++i)
↪  {
        int y = edge[x][i];
        if (!dfn[y]) {
            tarjan(y);
            low[x] = std::min(low[x], low[y]);
        } else if (!comp[y]) {
            low[x] = std::min(low[x], dfn[y]);
        }
    }
    if (low[x] == dfn[x]) {
        comps++;
        do {
            int y = stack[--top];
            comp[y] = comps;
        } while (stack[top] != x);
```

```
    }
}

bool solve() {
    int counter = n + n + 1;
    stamp = top = comps = 0;
    std::fill(dfn, dfn + counter, 0);
    std::fill(comp, comp + counter, 0);
    for (int i = 0; i < counter; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
    for (int i = 0; i < n; ++i) {
        if (comp[i << 1] == comp[i << 1 | 1]) {
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
    }
    return true;
}
```

4.5 有根树的同构

```
const unsigned long long MAGIC = 4423;

unsigned long long magic[N];
std::pair<unsigned long long, int> hash[N];

void solve(int root) {
    magic[0] = 1;
    for (int i = 1; i <= n; ++i) {
        magic[i] = magic[i - 1] * MAGIC;
    }
    std::vector<int> queue;
    queue.push_back(root);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            queue.push_back(y);
        }
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

        std::vector<std::pair<unsigned long long, int> > value;
        for (int i = 0; i < (int)son[x].size(); ++i) {
            int y = son[x][i];
            value.push_back(hash[y]);
        }
        std::sort(value.begin(), value.end());

        hash[x].first = hash[x].first * magic[1] + 37;
        hash[x].second++;
        for (int i = 0; i < (int)value.size(); ++i) {
            hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
            hash[x].second += value[i].second;
        }
        hash[x].first = hash[x].first * magic[1] + 41;
        hash[x].second++;
    }
}
```

4.6 Dominator Tree

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

const int MAXN = 50101;
const int MAXM = 110101;

class Edge
{public:
  int size;
  int begin[MAXN], dest[MAXM], next[MAXM];
  void clear(int n)
  {
    size = 0;
    fill(begin, begin + n, -1);
  }
  Edge(int n = MAXN)
  {
    clear(n);
  }
  void add_edge(int u, int v)
  {
```

```
    dest[size] = v;
    next[size] = begin[u];
    begin[u] = size++;
  }
};

class dominator
{public:
  int dfn[MAXN], sdom[MAXN], idom[MAXN], id[MAXN], f[MAXN], fa[MAXN], smin[MAXN], stamp;

  void predfs(int x, const Edge &succ)
  {
    id[dfn[x] = stamp++] = x;
    for(int i = succ.begin[x]; ~i; i = succ.next[i])
    {
      int y = succ.dest[i];
      if(dfn[y] < 0)
      {
        f[y] = x;
        predfs(y, succ);
      }
    }
  }
  int getfa(int x)
  {
    if(fa[x] == x)
      return x;
    int ret = getfa(fa[x]);
    if(dfn[sdom[smin[fa[x]]]] < dfn[sdom[smin[x]]])
      smin[x] = smin[fa[x]];
    return fa[x] = ret;
  }
  void solve(int s, int n, const Edge &succ)
  {
    fill(dfn, dfn + n, -1);
    fill(idom, idom + n, - 1);
    static Edge pred, tmp;
    pred.clear(n);
    for(int i = 0; i < n; ++i)
      for(int j = succ.begin[i]; ~j; j = succ.next[j])
        pred.add_edge(succ.dest[j], i);
    stamp = 0;
    tmp.clear(n);
    predfs(s, succ);
    for(int i = 0; i < stamp; ++i)
      fa[id[i]] = smin[id[i]] = id[i];
    for(int o = stamp - 1; o >= 0; --o)
    {
      int x = id[o];
      if(o)
      {
        sdom[x] = f[x];
        for(int i = pred.begin[x]; ~i; i = pred.next[i])
        {
          int p = pred.dest[i];
          if(dfn[p] < 0)
            continue;
          if(dfn[p] > dfn[x])
          {
            getfa(p);
            p = sdom[smin[p]];
          }
          if(dfn[sdom[x]] > dfn[p])
            sdom[x] = p;
        }
        tmp.add_edge(sdom[x], x);
      }
      while(~tmp.begin[x])
      {
        int y = tmp.dest[tmp.begin[x]];
        tmp.begin[x] = tmp.next[tmp.begin[x]];
        getfa(y);
        if(x != sdom[smin[y]])
          idom[y] = smin[y];
        else
          idom[y] = x;
      }
      for(int i = succ.begin[x]; ~i; i = succ.next[i])
        if(f[succ.dest[i]] == x)
          fa[succ.dest[i]] = x;
    }
    idom[s] = s;
    for(int i = 1; i < stamp; ++i)
    {
      int x = id[i];
      if(idom[x] != sdom[x])
        idom[x] = idom[idom[x]];
    }
  }
};
```

```
int ans[MAXN];

Edge e;
dominator dom1;

int dfs(int x)
{
  if(dom1.idom[x] <= 0)
    return 0;
  if(ans[x] > 0)
    return ans[x];
  if(dom1.idom[x] == x)
    return ans[x] = x;
  return ans[x] = x + dfs(dom1.idom[x]);
}

int main()
{
  int n, m;
  while(scanf("%d%d", &n, &m) == 2)
  {
    e.clear(n + 1);
    fill(ans, ans + n + 1, 0);
    for(int i = 0; i < m; ++i)
    {
      int u, v;
      scanf("%d%d", &u, &v);
      e.add_edge(u, v);
    }
    dom1.solve(n, n + 1, e);
    for(int i = 1; i <= n; ++i)
      printf("%d%c", dfs(i), " \n"[i == n]);
  }
  return 0;
}
```

4.7 哈密尔顿回路（ORE 性质的图）

```
int left[N], right[N], next[N], last[N];

void cover(int x) {
    left[right[x]] = left[x];
    right[left[x]] = right[x];
}

int adjacent(int x) {
    for (int i = right[0]; i <= n; i = right[i])
↪   {
        if (graph[x][i]) {
            return i;
        }
    }
    return 0;
}

std::vector<int> solve() {
    for (int i = 1; i <= n; ++i) {
        left[i] = i - 1;
        right[i] = i + 1;
    }
    int head, tail;
    for (int i = 2; i <= n; ++i) {
        if (graph[1][i]) {
            head = 1;
            tail = i;
            cover(head);
            cover(tail);
            next[head] = tail;
            break;
        }
    }
    while (true) {
        int x;
        while (x = adjacent(head)) {
            next[x] = head;
            head = x;
            cover(head);
        }
        while (x = adjacent(tail)) {
            next[tail] = x;
            tail = x;
            cover(tail);
        }
        if (!graph[head][tail]) {
            for (int i = head, j; i != tail; i =
↪   next[i]) {
                if (graph[head][next[i]] &&
↪   graph[tail][i]) {
                    for (j = head; j != i; j =
↪   next[j]) {
                        last[next[j]] = j;
                    }
                    j = next[head];
                    next[head] = next[i];
                    next[tail] = i;
                    tail = j;
                    for (j = i; j != head; j =
↪   last[j]) {
```

```
                        next[j] = last[j];
                    }
                    break;
                }
            }
        }
        next[tail] = head;
        if (right[0] > n) {
            break;
        }
        for (int i = head; i != tail; i =
    ↪   next[i]) {
            if (adjacent(i)) {
                head = next[i];
                tail = i;
                next[tail] = 0;
                break;
            }
        }
    }
    std::vector<int> answer;
    for (int i = head; ; i = next[i]) {
        if (i == 1) {
            answer.push_back(i);
            for (int j = next[i]; j != i; j =
    ↪   next[j]) {
                answer.push_back(j);
            }
            answer.push_back(i);
            break;
        }
        if (i == tail) {
            break;
        }
    }
    return answer;
}
```

4.8 无向图最小割

```
int node[N], dist[N];
bool visit[N];

int solve(int n) {
    int answer = INT_MAX;
    for (int i = 0; i < n; ++i) {
        node[i] = i;
    }
    while (n > 1) {
        int max = 1;
        for (int i = 0; i < n; ++i) {
            dist[node[i]] =
    ↪   graph[node[0]][node[i]];
            if (dist[node[i]] > dist[node[max]])
    ↪   {
                max = i;
            }
        }
        int prev = 0;
        memset(visit, 0, sizeof(visit));
        visit[node[0]] = true;
        for (int i = 1; i < n; ++i) {
            if (i == n - 1) {
                answer = std::min(answer,
    ↪   dist[node[max]]);
                for (int k = 0; k < n; ++k) {
                    graph[node[k]][node[prev]] =

    ↪   (graph[node[prev]][node[k]] +=
    ↪   graph[node[k]][node[max]]);
                }
                node[max] = node[--n];
            }
            visit[node[max]] = true;
            prev = max;
            max = -1;
            for (int j = 1; j < n; ++j) {
                if (!visit[node[j]]) {
                    dist[node[j]] +=
    ↪   graph[node[prev]][node[j]];
                    if (max == -1 ||
    ↪   dist[node[max]] < dist[node[j]]) {
                        max = j;
                    }
                }
            }
        }
    }
    return answer;
}
```

4.9 带花树

```
int match[N], belong[N], next[N], mark[N],
    ↪   visit[N];
std::vector<int> queue;

int find(int x) {
```

```
    if (belong[x] != x) {
        belong[x] = find(belong[x]);
    }
    return belong[x];
}

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x != y) {
        belong[x] = y;
    }
}

int lca(int x, int y) {
    static int stamp = 0;
    stamp++;
    while (true) {
        if (x != -1) {
            x = find(x);
            if (visit[x] == stamp) {
                return x;
            }
            visit[x] = stamp;
            if (match[x] != -1) {
                x = next[match[x]];
            } else {
                x = -1;
            }
        }
        std::swap(x, y);
    }
}

void group(int a, int p) {
    while (a != p) {
        int b = match[a], c = next[b];
        if (find(c) != p) {
            next[c] = b;
        }
        if (mark[b] == 2) {
            mark[b] = 1;
            queue.push_back(b);
        }
        if (mark[c] == 2) {
            mark[c] = 1;
            queue.push_back(c);
        }
        merge(a, b);
        merge(b, c);
        a = c;
    }
}

void augment(int source) {
    queue.clear();
    for (int i = 0; i < n; ++i) {
        next[i] = visit[i] = -1;
        belong[i] = i;
        mark[i] = 0;
    }
    mark[source] = 1;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size()
     && match[source] == -1; ++head) {
        int x = queue[head];
        for (int i = 0; i < (int)edge[x].size();
     ++i) {
            int y = edge[x][i];
            if (match[x] == y || find(x) ==
     find(y) || mark[y] == 2) {
                continue;
            }
            if (mark[y] == 1) {
                int r = lca(x, y);
                if (find(x) != r) {
                    next[x] = y;
                }
                if (find(y) != r) {
                    next[y] = x;
                }
                group(x, r);
                group(y, r);
            } else if (match[y] == -1) {
                next[y] = x;
                for (int u = y; u != -1; ) {
                    int v = next[u];
                    int mv = match[v];
                    match[v] = u;
                    match[u] = v;
                    u = mv;
                }
                break;
            } else {
                next[y] = x;
                mark[y] = 2;
                mark[match[y]] = 1;
                queue.push_back(match[y]);
```

```
            }
        }
    }
}

int solve() {
    std::fill(match, match + n, -1);
    for (int i = 0; i < n; ++i) {
        if (match[i] == -1) {
            augment(i);
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += (match[i] != -1);
    }
    return answer;
}
```

5 字符串

5.1 KMP 算法

```
void getnex(char *s, int *nex)
{
  int n = strlen(s + 1);
  for(int j = 0, i = 2; i <= n; i++)
  {
    while(j && s[j + 1] != s[i])
      j = nex[j];
    if(s[i] == s[j + 1])
      j++;
    nex[i] = j;
  }
}
int main()
{
  const int N = 1e6 + 10;
  static char s[N];
  static int nex[N];
  scanf("%s", s + 1);
  getnex(s, nex);
}
```

5.2 扩展 KMP 算法

```
#include<bits/stdc++.h>
#define next NEXT
//next[i] 表示 s 和其后缀 s[i, n] 的 lcp 的长度
void getnext(char s[], int n, int next[])
{
  next[1] = n;
  int &t = next[2] = 0;
  for(; t + 2 <= n && s[1 + t] == s[2 + t]; t++);
  int pos = 2;
  for(int i = 3; i <= n; i++)
  {
    if(i + next[i - pos + 1] < pos + next[pos])
      next[i] = next[i - pos + 1];
    else
    {
      int j = max(0, next[pos] + pos - i);
      for(;i + j <= n && s[i + j] == s[j + 1]; j++);
      next[i] = j;
      pos = i;
    }
  }
}

//extend[i] 表示 s2 和 s1 后缀 s1[i, n] 的 lcp 的长度
void getextend(char s1[], char s2[], int extend[])
{
  int n = strlen(s1 + 1), m = strlen(s2 + 1);
  getnext(s2, m, next);
  int &t = extend[1] = 0;
  for(; t < n && t < m && s1[1 + t] == s2[1 + t]; t++);
  int pos = 1;
  for(int i = 2; i <= n; i++)
  {
    if(i + next[i - pos + 1] < pos + extend[pos])
      extend[i] = next[i - pos + 1];
    else
    {
      int j = max(0, extend[pos] + pos - i);
      for(; i + j <= n && j < m && s1[i + j] == s2[j + 1]; j++);
      extend[i] = j;
      pos = i;
    }
  }
}
```

5.3 AC 自动机

```
const int C = 26;
const int L = 1e5 + 5;
const int N = 5e5+10;
int n, root, cnt, fail[N], son[N][26], num[N];
char s[L];

inline int newNode()
{
  cnt++;
  memset(son[cnt], 0, sizeof(son[cnt]));
  fail[cnt] = num[cnt] = 0;
  return cnt;
}

void insert(char *s)
{
  int n = strlen(s + 1);
  int now = 1;
  for(int i = 1; i <= n; i++)
  {
    int c = s[i] - 'a';
    if(!son[now][c])
      son[now][c] = newNode();
    now = son[now][c];
  }
  num[now]++;
}


void getfail(){
  static queue<int> Q;
  fail[root] = 0;
  Q.push(root);
  while(!Q.empty())
  {
    int now = Q.front();
    Q.pop();
    for(int i = 0; i < C; i++)
      if(son[now][i])
      {
        Q.push(son[now][i]);
        int p = fail[now];
        while(!son[p][i])
          p = fail[p];
        fail[son[now][i]] = son[p][i];
      }
      else
        son[now][i] = son[fail[now]][i];
  }
}

int main()
{
  cnt = 0;
  root = newNode();
  scanf("%d", &n);
  for(int i = 0; i < C; i++)
    son[0][i] = 1;
  for(int i = 1; i <= n; i++)
  {
    scanf("%s", s + 1);
    insert(s);
  }
  getfail();
  return 0;
}
```

5.4 后缀自动机

5.4.1 广义后缀自动机（多串）

注意事项：空间是插入字符串总长度的 2 倍并请注意字符集大小。

```
const int N = 251010;
const int C = 26;

int tot, las, root;
struct Node
{
  int son[C], len, par;
  void clear()
  {
    memset(son, 0, sizeof(son));
    par = len = 0;
  }
}node[N << 1];

inline int newNode()
{
  node[++tot].clear();
  return tot;
}

void extend(int c)
{
  int p = las;
  if (node[p].son[c]) {
```

```
    int q = node[p].son[c];
    if (node[p].len + 1 == node[q].len)
      las = q;
    else
    {
      int nq = newNode();
      las = nq;
      node[nq] = node[q];
      node[nq].len = node[p].len + 1;
      node[q].par = nq;
      for (; p && node[p].son[c] == q; p =
  ↪   node[p].par)
        node[p].son[c] = nq;
    }
  }
  else // Naive Suffix Automaton
  {
    int np = newNode();
    las = np;
    node[np].len = node[p].len + 1;
    for (; p && !node[p].son[c]; p = node[p].par)
      node[p].son[c] = np;
    if (!p)
      node[np].par = root;
    else
    {
      int q = node[p].son[c];
      if (node[p].len + 1 == node[q].len)
        node[np].par = q;
      else
      {
        int nq = newNode();
        node[nq] = node[q];
        node[nq].len = node[p].len + 1;
        node[q].par = node[np].par = nq;
        for (; p && node[p].son[c] == q; p =
  ↪   node[p].par)
          node[p].son[c] = nq;
      }
    }
  }
}

void add(char *s)
{
  int len = strlen(s + 1);
  las = root;
  for(int i = 1; i <= len; i++)
    extend(s[i] - 'a');
}
```

5.4.2 sam-ypm

sam-nsubstr

```
//SAM 利用后缀树进行计算，由儿子向 parert 更新
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef pair<int, int> pii;
const int inf = 1e9;
const int N = 251010;
const int C = 26;

int tot, las, root;
struct Node
{
  int son[C], len, par, count;
  void clear()
  {
    memset(son, 0, sizeof(son));
    par = count = len = 0;
  }
}node[N << 1];


inline int newNode()
{
  node[++tot].clear();
  return tot;
}

void extend(int
 ↪  c)//传入转化为数字之后的字符，从 0 开始
{
  int p = las, np = newNode();
  las = np;
  node[np].len = node[p].len + 1;
  for(;p && !node[p].son[c]; p = node[p].par)
    node[p].son[c] = np;
  if(p == 0)
    node[np].par = root;
  else
  {
    int q = node[p].son[c];
    if(node[p].len + 1 == node[q].len)
      node[np].par = q;
    else
    {
```

```
      int nq = newNode();
      node[nq] = node[q];
      node[nq].len = node[p].len + 1;
      node[q].par = node[np].par = nq;
      for(;p && node[p].son[c] == q; p =
↪   node[p].par)
        node[p].son[c] = nq;
    }
  }
}


int main(){
  static char s[N];
  while(scanf("%s", s + 1) == 1)
  {
    tot = 0;
    root = las = newNode();
    int n = strlen(s + 1);
    for(int i = 1;i <= n; i++)
      extend(s[i] - 'a');

    static int cnt[N], order[N << 1];
    memset(cnt, 0, sizeof(*cnt) * (n + 5));
    for(int  i = 1; i <= tot; i++)
      cnt[node[i].len]++;
    for(int i = 1; i <= n; i++)
      cnt[i] += cnt[i - 1];
    for(int i = tot; i; i--)
      order[ cnt[node[i].len]-- ] = i;

    static int
↪   dp[N];//dp[i] 为长度为 i 的子串中出现次数最多的串的出现次数
    memset(dp, 0, sizeof(dp));
    for(int now = root, i = 1;  i <= n; i++)
    {
      now = node[now].son[s[i] - 'a'];
      node[now].count++;
    }
    for(int i = tot; i; i--)
    {
      Node &now = node[order[i]];
      dp[now.len] = max(dp[now.len], now.count);
      node[now.par].count += now.count;
    }
    for(int i = n - 1; i; i--)
      dp[i] = max(dp[i], dp[i + 1]);
    for(int i = 1; i <= n; i++)
      printf("%d\n", dp[i]);
  }
}
```

sam-lcs

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef pair<int, int> pii;
const int inf = 1e9;
const int N = 101010;
const int C = 26;

int tot, las, root;
struct Node
{
  int son[C], len, par, count;
  void clear()
  {
    memset(son, 0, sizeof(son));
    par = count = len = 0;
  }
}node[N << 1];


inline int newNode()
{
  node[++tot].clear();
  return tot;
}

void extend(int
↪   c)//c 为转化为数字之后的字符，从 0 开始
{
  int p = las, np = newNode();
  las = np;
  node[np].len = node[p].len + 1;
  for(;p && !node[p].son[c]; p = node[p].par)
    node[p].son[c] = np;
  if(p == 0)
    node[np].par = root;
  else
  {
    int q = node[p].son[c];
    if(node[p].len + 1 == node[q].len)
      node[np].par = q;
    else
    {
      int nq = newNode();
      node[nq] = node[q];
```

```
      node[nq].len = node[p].len + 1;
      node[q].par = node[np].par = nq;
      for(;p && node[p].son[c] == q; p =
↪   node[p].par)
        node[p].son[c] = nq;
    }
  }
}


int main(){
  static char s[N];
  scanf("%s", s + 1);
  tot = 0;
  root = las = newNode();
  int n = strlen(s + 1);
  for(int i = 1;i <= n; i++)
    extend(s[i] - 'a');

  static int cnt[N], order[N << 1];
  memset(cnt, 0, sizeof(*cnt) * (n + 5));
  for(int i = 1; i <= tot; i++)
    cnt[node[i].len]++;
  for(int i = 1; i <= n; i++)
    cnt[i] += cnt[i - 1];
  for(int i = tot; i; i--)
    order[ cnt[node[i].len]-- ] = i;

  static int ANS[N << 1], dp[N << 1];
  memset(dp, 0, sizeof(*dp) * (tot + 5));
  for(int i = 1; i <= tot; i++)
    ANS[i] = node[i].len;
  while(scanf("%s", s + 1) == 1)
  {
    n = strlen(s + 1);
    for(int now = root, len = 0, i = 1; i <= n;
↪   i++)
    {
      int c = s[i] - 'a';
      while(now != root && !node[now].son[c])
        now = node[now].par;
      if(node[now].son[c])
      {
        len = min(len, node[now].len) + 1;
        now = node[now].son[c];
      }
      else
        len = 0;
      dp[now] = max(dp[now], len);
    }
    for(int i = tot; i; i--)
    {
      int now = order[i];
      dp[node[now].par] = max(dp[node[now].par],
↪   dp[now]);
      ANS[now] = min(ANS[now], dp[now]);
      dp[now] = 0;
    }
  }
  int ans = 0;
  for(int i = 1; i<= tot; i++)
    ans = max(ans, ANS[i]);
  printf("%d\n", ans);
}
```

5.5 后缀数组

注意事项： $\mathcal{O}(n \log n)$ 倍增构造。

```
#define ws wws
const int MAXN = 201010;
int wa[MAXN], wb[MAXN], wv[MAXN], ws[MAXN];
int sa[MAXN], rk[MAXN], height[MAXN];
char s[MAXN];

inline bool cmp(int *r, int a, int b, int l)
{
  return r[a] == r[b] && r[a + l] == r[b + l];
}

void SA(char *r, int *sa, int n, int m)
{
  int *x = wa, *y = wb;
  for(int i = 1; i <= m; i++)ws[i] = 0;
  for(int i = 1; i <= n; i++)ws[x[i] = r[i]]++;
  for(int i = 1; i <= m; i++)ws[i] += ws[i - 1];
  for(int i = n; i > 0; i--)sa[ ws[x[i]]-- ] = i;
  for(int j = 1, p = 0; p < n; j <<= 1, m = p)
  {
    p = 0;
    for(int i = n - j + 1; i <= n; i++)y[++p] =
↪   i;
    for(int i = 1; i <= n; i++)if(sa[i] > j)
↪   y[++p] = sa[i] - j;
    for(int i = 1; i <= n; i++)wv[i] = x[y[i]];
    for(int i = 1; i <= m; i++)ws[i] = 0;
    for(int i = 1; i <= n; i++)ws[wv[i]]++;
    for(int i = 1; i <= m; i++)ws[i] += ws[i -
↪   1];
```

```cpp
    for(int i = n; i > 0; i--)sa[ ws[wv[i]]-- ] = y[i];
    swap(x, y);
    x[sa[1]] = p = 1;
    for(int i = 2; i <= n; i++)
      x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p : ++p;
  }
}

void getheight(char *r, int *sa, int *rk, int *h, int n)
{
  for(int i = 1; i <= n; i++)
    rk[sa[i]] = i;
  for(int i = 1, p = 0; i <= n; i++, p ? p-- : 0)
  {
    int j = sa[rk[i] - 1];
    while(r[i + p] == r[j + p])
      p++;
    h[rk[i]] = p;
  }
}
```

注意: $\mathcal{O}(n)$ 线性构造，常数大，约为倍增的 0.5-0.6 倍

```cpp
//dc3, 1-based
//r 数组开 0~n, n + 1 个元素, 其中 0~n - 1 存字符串的 ascii 码(>0), r[n] = 0;
//执行完后 sa[0] 舍弃不用, sa[1~n] 是从 0 开始的 sa 数组, 将 sa[i]++ 后为正常 1-based 的 sa 数组
#include <bits/stdc++.h>
#define rank RANK
#define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : tb))
#define G(x) ((x) < tb ? (x) * 3 + 1 : ((x) - tb) * 3 + 2)
using namespace std;
const int N = 101010;

int wa[N], wb[N], wv[N], wss[N];
int r[N * 3], sa[N * 3], rank[N], height[N];
char s[N];

bool c0(int *r, int a, int b)
{
  return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
}

int c12(int k, int *r, int a, int b)
{
  if(k == 2)
    return r[a] < r[b] || (r[a] == r[b] && c12(1, r, a + 1, b + 1));
  else
    return r[a] < r[b] || (r[a] == r[b] && wv[a + 1] < wv[b + 1]);
}

void sort(int *r, int *a, int *b, int n, int m)
{
  memset(wss, 0, sizeof(*wss) * (m + 2));
  for(int i = 0; i < n; i++) wss[wv[i] = r[a[i]]]++;
  for(int i = 1; i < m; i++) wss[i] += wss[i - 1];
  for(int i = n - 1; i >= 0; i--) b[ --wss[wv[i]] ] = a[i];
}

void dc3(int *r, int *sa, int n, int m)
{
  int *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
  r[n] = r[n + 1] = 0;
  for(int i = 0; i < n; i++)
    if(i % 3 != 0)
      wa[tbc++] = i;
  sort(r + 2, wa, wb, tbc, m);
  sort(r + 1, wb, wa, tbc, m);
  sort(r, wa, wb, tbc, m);
  rn[F(wb[0])] = 0;
  p = 1;
  for(int i = 1; i < tbc; i++)
    rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
  if(p < tbc)
    dc3(rn, san, tbc, p);
  else
    for(int i = 0; i < tbc; i++)
      san[rn[i]] = i;
  for(int i = 0; i < tbc; i++)
    if(san[i] < tb)
      wb[ta++] = san[i] * 3;

  if(n % 3 == 1)
    wb[ta++] = n - 1;
  sort(r, wb, wa, ta, m);
  for(int i = 0; i < tbc; i++)
    wv[wb[i] = G(san[i])] = i;
```

```
  p = 0;
  int i = 0, j = 0;
  for(;i < ta && j < tbc; p++)
    sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
  for(; i < ta; p++)
    sa[p] = wa[i++];
  for(; j < tbc; p++)
    sa[p] = wb[j++];
}

void getheight(char s[], int sa[], int n)
{
  for(int i = 1; i <= n; i++)
    rank[sa[i]] = i;
  for(int p = 0, i = 1; i <= n; i++, p = (p) ? p - 1 : p)
    if(rank[i] > 1)
    {
      int j = sa[rank[i] - 1];
      while(s[i + p] == s[j + p])
        p++;
      height[rank[i]] = p;
    }
}


int main()
{
  scanf("%s", s + 1);
  int n = strlen(s + 1);
  for(int i = 0; i <= n; i++)// <= n !!!
    r[i] = s[i + 1];
  dc3(r, sa, n + 1, 255);//now the value of sa is from 0 to n - 1;
  for(int i = n; i ; i--)//after this operation, the value of sa is from 1 to n
    sa[i]++;
  getheight(s, sa, n);
  for(int i = 1;i <= n; i++)
    printf("%d ", sa[i]);
  puts("");
  for(int i = 2; i <= n; i++)
    printf("%d ", height[i]);
  puts("");
}
```

5.6 回文自动机

注意事项：请注意字符集大小。

```
const int C = 26;
const int N = 301010;

char s[N];
int cnt, last;

struct Node
{
  int son[C], fail, size, len;
  void newNode(int l)
  {
    memset(son, 0, sizeof(son));
    fail = size = 0;
    len = l;
  }
}node[N];

void init()
{
  cnt = 2;
  node[1].newNode(0);//Even root
  node[2].newNode(-1);//Odd root
  last = 1;
  node[1].fail = 2;
  node[2].fail = 1;
}

void add(int c, int L)
{
  int p = last;
  while(s[L - node[p].len - 1] != s[L])
    p = node[p].fail;
  if(!node[p].son[c])
  {
    int q = ++cnt, &fq = node[q].fail;
    node[q].newNode(node[p].len + 2);
    fq = node[p].fail;
    while(s[L - node[fq].len - 1] != s[L])
      fq = node[fq].fail;
    fq = max(1, node[fq].son[c]);
    node[p].son[c] = q;
  }
  last = node[p].son[c];
  node[last].size++;
}
```

```cpp
void calc()
{
  for(int i = cnt; i; i--)
    node[node[i].fail].size += node[i].size;
}

int main()
{
  scanf("%s", s + 1)
  int n = strlen(s + 1);
  s[0] = '$';
  init();
  for(int i = 1; i <= n; i++)
    add(s[i] - 'a', i);
  calc();
}
```

5.7 Manacher

注意事项：1-based 算法，请注意下标。

```cpp
int manacher(char *st)
{
  const int N = 1e6+10;
  static char s[N << 1];
  static int p[N << 1];
  int n = strlen(st + 1);
  s[0] = '$';
  s[1] = '#';
  for(int i = 1; i <= n; i++)
  {
    s[i << 1] = st[i];
    s[(i << 1) + 1] = '#';
  }
  n = n * 2 + 1;
  s[n + 1] = 0;
  int pos, mx = 0, res = 0;
  for(int i = 1; i <= n; i++)
  {
    p[i] = (mx > i) ? min(p[pos * 2 - i], mx - i) : 1;
    while(s[i + p[i]] == s[i - p[i]])
      p[i]++;
    if(p[i] + i - 1 > mx)
    {
      mx = p[i] + i - 1;
      pos = i;
    }
    res = max(p[i], res);
  }
  return res - 1;
}
```

5.8 循环串的最小表示

注意事项：0-Based 算法，请注意下标。

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 100100;
char s[N];
/*
int work1(int *a, int n){//输出最靠左的最小表示
  for(int i = 0; i < n; i++)
    a[i + n] = a[i];
  int pos = 0;
  for(int i = 1, k; i < n;){
    for(k = 0; k < n && a[pos + k] == a[i + k]; k++);
    if(k < n && a[i + k] < a[pos + k]){
      int t = pos;
      pos = i;
      i = max(i + 1, t + k + 1);
    }
    else{
      i += k + 1;
    }
  }
  return pos;
}

int work2(int *a, int n){//输出最靠右的最小表示，待验，谨慎使用
  for(int i = 0; i < n; i++)
    a[i + n] = a[i];
  int pos = 0;
  for(int i = 1, k; i < n;){
    for(k = 0; k < n && a[pos + k] == a[i + k]; k++);
    if(k == n){
      pos = i;
      i++;
      continue;
    }
    if(k < n && a[i + k] < a[pos + k]){
      int t = pos;
      pos = i;
      i = max(i + 1, t + k + 1);
    }
    else{
      i += k + 1;
    }
  }
```

```
  return pos;
}
*/
int getmin(char *s, int n){// 0-base
  int i = 0, j = 1, k = 0;
  while(i < n && j < n && k < n){
    int x = i + k;
    int y = j + k;
    if(x >= n) x -= n;
    if(y >= n) y -= n;
    if(s[x] == s[y])
      k++;
    else{
      if(s[x] > s[y])
        i += k + 1;
      else
        j += k + 1;
      if(i == j)
        j ++;
      k = 0;
    }
  }
  return min(i ,j);
}

int main(){
  int T;
  scanf("%d", &T);
  while(T--){
    int n;
    scanf("%d", &n);
    scanf("%s", s);
    printf("%d\n", getmin(s, n));
  }
}
```

5.9 后缀树

注意事项:

1. 边上的字符区间是左闭右开区间;

2. 如果要建立关于多个串的后缀树，请用不同的分隔符，并且对于每个叶子结点，去掉和它父亲的连边上出现的第一个分隔符之后的所有字符;

```
const int MAXL = 100001;
↪  // The length of the string being inserted into the ST.
const int MAXD = 27;
↪  // The size of the alphabet.

struct SuffixTree{
  int size, length, pCur, dCur, lCur, lBuf,
↪  text[MAXL];
  std::pair<int, int> suffix[MAXL];

  struct Node{
    int left, right, sLink, next[MAXD];
  }tree[MAXL * 2];

  int getLength(const int &rhs) {
    return tree[rhs].right ? tree[rhs].right -
↪  tree[rhs].left : length + 1 -
↪  tree[rhs].left;
  }
  void addLink(int &last, int node) {
    if (last != 0) tree[last].sLink = node;
    last = node;
  }
  int alloc(int left, int right = 0) {
    size++;
    memset(&tree[size], 0, sizeof(tree[size]));
    tree[size].left = left;
    tree[size].right = right;
    tree[size].sLink = 1;
    return size;
  }
  bool move(int node) {
    int length = getLength(node);
    if (lCur >= length) {
      lCur -= length;
      dCur += length;
      pCur = node;
      return true;
    }
    return false;
  }
  void init() {
    size = length = 0;
    lCur = dCur = lBuf = 0;
    pCur = alloc(0);
  }
  void extend(int x) {
    text[++length] = x;
    lBuf++;
```

```
    for (int last = 0; lBuf > 0; ) {
      if (lCur == 0) dCur = length;
      if (!tree[pCur].next[text[dCur]]) {
        int newleaf = alloc(length);
        tree[pCur].next[text[dCur]] = newleaf;
        suffix[length + 1 - lBuf] =
↪   std::make_pair(pCur, newleaf);
        addLink(last, pCur);
      } else {
        int nownode =
↪   tree[pCur].next[text[dCur]];
        if (move(nownode)) continue;
        if (text[tree[nownode].left + lCur] == x)
↪   {
          lCur++;
          addLink(last, pCur);
          break;
        }
        int newleaf = alloc(length), newnode =
↪   alloc(tree[nownode].left, tree[nownode].left
↪   + lCur);
        tree[nownode].left += lCur;
        tree[pCur].next[text[dCur]] = newnode;
        tree[newnode].next[x] = newleaf;
        tree[newnode].next[text[tree[nownode].left]]
↪   = nownode;
        suffix[length + 1 - lBuf] =
↪   std::make_pair(newnode, newleaf);
        addLink(last, newnode);
      }
      lBuf--;
      if (pCur == 1 && lCur > 0) lCur--, dCur++;
      else pCur = tree[pCur].sLink;
    }
  }
};
```

6 计算几何

6.1 二维几何

```
// 求圆与直线的交点
bool isCL(Circle a, Line l, P &p1, P &p2) {
  D x = (l.s - a.o) % l.d,
    y = l.d.sqrlen(),
    d = x * x - y * ((l.s - a.o).sqrlen() - a.r *
↪   a.r);
  if (sign(d) < 0) return false;
  P p = l.s - x / y * l.d, delta =
↪   sqrt(max((D)0., d)) / y * l.d;
  p1 = p + delta, p2 = p - delta;
  return true;
}
// 求圆与圆的交面积
D areaCC(const Circle &c1, const Circle &c2) {
  D d = (c1.o - c2.o).len();
  if (sign(d - (c1.r + c2.r)) >= 0) {
    return 0;
  }
  if (sign(d - abs(c1.r - c2.r)) <= 0) {
    D r = min(c1.r, c2.r);
    return r * r * pi;
  }
  D x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2
↪   * d),
      t1 = acos(min(1., max(-1., x / c1.r))), t2
↪   = acos(min(1., max(-1., (d - x) / c2.r)));
  return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d
↪   * c1.r * sin(t1);
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool isCC(Circle a, Circle b, P &p1, P &p2) {
  D s1 = (a.o - b.o).len();
  if (sign(s1 - a.r - b.r) > 0 || sign(s1 -
↪   abs(a.r - b.r)) < 0) return false;
  D s2 = (a.r * a.r - b.r * b.r) / s1;
  D aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
  P o = aa / (aa + bb) * (b.o - a.o) + a.o;
  P delta = sqrt(max(0., a.r * a.r - aa * aa)) *
↪   (b.o - a.o).zoom(1).rev();
  p1 = o + delta, p2 = o - delta;
  return true;
}
// 求点到圆的切点，按关于点的顺时针方向返回两个点，rev 必须是 (-y
bool tanCP(const Circle &c, const P &p0, P &p1, P
↪   &p2) {
  D x = (p0 - c.o).sqrlen(), d = x - c.r * c.r;
  if (d < eps) return false;
↪   // 点在圆上认为没有切点
  P p = c.r * c.r / x * (p0 - c.o);
  P delta = (-c.r * sqrt(d) / x * (p0 -
↪   c.o)).rev();
  p1 = c.o + p + delta;
  p2 = c.o + p - delta;
  return true;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线，rev 必
```

```cpp
vector<Line> extanCC(const Circle &c1, const Circle &c2) {
  vector<Line> ret;
  if (sign(c1.r - c2.r) == 0) {
    P dir = c2.o - c1.o;
    dir = (c1.r / dir.len() * dir).rev();
    ret.push_back(Line(c1.o + dir, c2.o - c1.o));
    ret.push_back(Line(c1.o - dir, c2.o - c1.o));
  } else {
    P p = 1. / (c1.r - c2.r) * (-c2.r * c1.o + c1.r * c2.o);
    P p1, p2, q1, q2;
    if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) {
      if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
      ret.push_back(Line(p1, q1 - p1));
      ret.push_back(Line(p2, q2 - p2));
    }
  }
  return ret;
}
// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向返回两条线，rev 必须是 (-y, x)
vector<Line> intanCC(const Circle &c1, const Circle &c2) {
  vector<Line> ret;
  P p = 1. / (c1.r + c2.r) * (c2.r * c1.o + c1.r * c2.o);
  P p1, p2, q1, q2;
  if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2)) { // 两圆相切认为没有切线
    ret.push_back(Line(p1, q1 - p1));
    ret.push_back(Line(p2, q2 - p2));
  }
  return ret;
}
bool contain(vector<P> poly, P p) { // 判断点 p 是否被多边形包含，包括落在边界上
  int ret = 0, n = poly.size();
  for(int i = 0; i < n; ++ i) {
    P u = poly[i], v = poly[(i + 1) % n];
    if (onSeg(p, u, v)) return true; // 在边界上
    if (sign(u.y - v.y) <= 0) swap(u, v);
    if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <= 0) continue;
    ret += sign((v - p) * (u - p)) > 0;
  }
  return ret & 1;
}
```

```cpp
vector<P> convexCut(const vector<P>&ps, Line l) { // 用半平面 (s,d) 的逆时针方向去切凸多边形
  vector<P> qs;
  int n = ps.size();
  for (int i = 0; i < n; ++i) {
    Point p1 = ps[i], p2 = ps[(i + 1) % n];
    int d1 = sign(l.d * (p1 - l.s)), d2 = sign(l.d * (p2 - l.s));
    if (d1 >= 0) qs.push_back(p1);
    if (d1 * d2 < 0) qs.push_back(isLL(Line(p1, p2 - p1), l));
  }
  return qs;
}
```

6.2 凸包

```cpp
inline bool turn_left(const Point &a, const Point &b, const Point &c) {
  return sgn(det(b - a, c - a)) >= 0;
}
void convex_hull(vector<Data> p, vector<Data> &res) {
  int n = (int)p.size(), cnt = 0;
  sort(p.begin(), p.end(), [&](const Data &a, const Data &b) {
      if(fabs(a.p.x - b.p.x) < eps) return a.p.y > b.p.y;
      return a.p.x < b.p.x; });
  res.clear();
  for(int i = 0; i < n; i++) {
    while(cnt > 1 && turn_left(res[cnt - 2].p, p[i].p, res[cnt - 1].p)) {
      cnt--;
      res.pop_back();
    }
    res.push_back(p[i]);
    ++cnt;
  }
  int fixed = cnt;
  for(int i = n - 2; i >= 0; i--) {
    while(cnt > fixed && turn_left(res[cnt - 2].p, p[i].p, res[cnt - 1].p)) {
      --cnt;
      res.pop_back();
    }
    res.push_back(p[i]);
    ++cnt;
```

```
  }
}
```

6.3 阿波罗尼茨圆

硬币问题：易知两两相切的圆半径为 r1, r2, r3, 求与他们都相切的圆的半径 r4

分母取负号，答案再取绝对值，为外切圆半径

分母取正号为内切圆半径

// $r_4^{\pm} = \frac{r_1 r_2 r_3}{r_1 r_2 + r_1 r_3 + r_2 r_3 \pm 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$

6.4 最小覆盖球

```
// 注意，无法处理小于四点的退化情况
struct P;
P a[33];
P intersect(const Plane & a, const Plane & b, const Plane & c) {
  P c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z, c.nor.z), c4(a.m, b.m, c.m);
  return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
}
bool in(const P & a, const Circle & b) {
  return sign((a - b.o).len() - b.r) <= 0;
}
vector<P> vec;
Circle calc() {
  if (vec.empty()) {
    return Circle(Point(0, 0, 0), 0);
  } else if(1 == (int)vec.size()) {
    return Circle(vec[0], 0);
  } else if(2 == (int)vec.size()) {
    return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
  } else if(3 == (int)vec.size()) {
    double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 /
       fabs(((vec[0] - vec[2]) * (vec[1] - vec[2])).len()));
    return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
           Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
         Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
  } else {
    P o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
         Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),
         Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
    return Circle(o, (o - vec[0]).len());
  }
}
Circle miniBall(int n) {
  Circle res(calc());
  for(int i(0); i < n; i++) {
    if(!in(a[i], res)) {
      vec.push_back(a[i]);
      res = miniBall(i);
      vec.pop_back();
      if (i) { Point tmp(a[i]); memmove(a + 1, a, sizeof(Point) * i); a[0] = tmp; }
    }
  }
  return res;
}
int main() {
  for(int i(0); i < n; i++) a[i].scan();
  sort(a, a + n);
  n = unique(a, a + n) - a;
  vec.clear();
  random_shuffle(a, a + n);
  printf("%.10f\n", miniBall(n).r);
}
```

6.5 三角形与圆交

```
// 反三角函数要在 [-1, 1] 中，sqrt 要与 0 取 max 别忘了取正负
// 改成周长请用注释，res1 为直线长度，res2 为弧线长度
// 多边形与圆求交时，相切精度比较差
D areaCT(P pa, P pb, D r) { //, D & res1, D & res2) {
  if (pa.len() < pb.len()) swap(pa, pb);
  if (sign(pb.len()) == 0) return 0;
  // if (sign(pb.len()) == 0) { res1 += min(r, pa.len()); retu
  D a = pb.len(), b = pa.len(), c = (pb - pa).len();
  D sinB = fabs(pb * (pb - pa)), cosB = pb % (pb - pa), area = fabs(pa * pb);
  D S, B = atan2(sinB, cosB), C = atan2(area, pa % pb);
  sinB /= a * c; cosB /= a * c;
  if (a > r) {
```

```
        S = C / 2 * r * r; D h = area / c;//res2 += -1 * sgn * C * r; D h = area / c;
        if (h < r && B < pi / 2) {
            //res2 -= -1 * sgn * 2 * acos(max((D)-1., min((D)1., h / r))) * r;
            //res1 += 2 * sqrt(max((D)0., r * r - h * h));
            S -= (acos(max((D)-1., min((D)1., h / r))) * r * r - h * sqrt(max((D)0. ,r * r - h * h)));
        }
    } else if (b > r) {
        D theta = pi - B - asin(max((D)-1., min((D)1., sinB / r * a)));
        S = a * r * sin(theta) / 2 + (C - theta) / 2 * r * r;
        //res2 += -1 * sgn * (C - theta) * r;
        //res1 += sqrt(max((D)0., r * r + a * a - 2 * r * a * cos(theta)));
    } else S = area / 2;
    //res1 += (pb - pa).len();
    return S;
}
```

6.6 圆并

```
struct Event {
  P p; D ang; int delta;
  Event (P p = Point(0, 0), D ang = 0, int delta = 0) : p(p), ang(ang), delta(delta) {}
};
bool operator < (const Event &a, const Event &b) { return a.ang < b.ang; }
void addEvent(const Circle &a, const Circle &b, vector<Event> &evt, int &cnt) {
  D d2 = (a.o - b.o).sqrlen(), dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1) / 2,
    pRatio = sqrt(max((D)0., -(d2 - sqr(a.r - b.r)) * (d2 - sqr(a.r + b.r)) / (d2 * d2 * 4)));
  P d = b.o - a.o, p = d.rot(pi / 2),
    q0 = a.o + d * dRatio + p * pRatio,
    q1 = a.o + d * dRatio - p * pRatio;
  D ang0 = (q0 - a.o).ang(), ang1 = (q1 - a.o).ang();
  evt.emplace_back(q1, ang1, 1); evt.emplace_back(q0, ang0, -1);
  cnt += ang1 > ang0;
}
bool issame(const Circle &a, const Circle &b) { return sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) == 0; }
bool overlap(const Circle &a, const Circle &b) { return sign(a.r - b.r - (a.o - b.o).len()) >= 0; }
bool intersect(const Circle &a, const Circle &b) { return sign((a.o - b.o).len() - a.r - b.r) < 0; }
int C;
Circle c[N];
double area[N];
void solve() { // 返回覆盖至少 k 次的面积
  memset(area, 0, sizeof(D) * (C + 1));
  for (int i = 0; i < C; ++i) {
    int cnt = 1;
    vector<Event> evt;
    for (int j = 0; j < i; ++j) if (issame(c[i], c[j])) ++cnt;
    for (int j = 0; j < C; ++j)
      if (j != i && !issame(c[i], c[j]) && overlap(c[j], c[i]))
        ++cnt;
    for (int j = 0; j < C; ++j)
      if (j != i && !overlap(c[j], c[i]) && !overlap(c[i], c[j]) && intersect(c[i], c[j]))
        addEvent(c[i], c[j], evt, cnt);
    if (evt.empty()) area[cnt] += PI * c[i].r * c[i].r;
    else {
      sort(evt.begin(), evt.end());
      evt.push_back(evt.front());
      for (int j = 0; j + 1 < (int)evt.size(); ++j) {
        cnt += evt[j].delta;
        area[cnt] += det(evt[j].p, evt[j + 1].p) / 2;
        D ang = evt[j + 1].ang - evt[j].ang;
        if (ang < 0) ang += PI * 2;
        area[cnt] += ang * c[i].r * c[i].r / 2 - sin(ang) * c[i].r * c[i].r / 2;
} } } }
```

6.7 整数半平面交

```
typedef __int128 J; // 坐标 |1e9| 就要用 int128 来判断
struct Line {
```

```cpp
  bool include(P a) const { return (a - s) * d >= 0; } // 严格去掉 =
  bool include(Line a, Line b) const {
    J l1(a.d * b.d);
    if(!l1)  return true;
    J x(l1 * (a.s.x - s.x)), y(l1 * (a.s.y - s.y));
    J l2((b.s - a.s) * b.d);
    x += l2 * a.d.x; y += l2 * a.d.y;
    J res(x * d.y - y * d.x);
    return l1 > 0 ? res >= 0 : res <= 0; // 严格去掉 =
  }
};
bool HPI(vector<Line> v) { // 返回 v 中每个射线的右侧的交是否非空
  sort(v.begin(), v.end());// 按方向排极角序
  { // 同方向取最严格的一个
    vector<Line> t; int n(v.size());
    for(int i(0), j; i < n; i = j) {
      LL mx(-9e18); int mxi;
      for(j = i; j < n && v[i].d * v[j].d == 0; j++) {
        LL tmp(v[j].s * v[i].d);
        if(tmp > mx)
          mx = tmp, mxi = j;
      }
      t.push_back(v[mxi]);
    }
    swap(v, t);
  }
  deque<Line> res;
  bool emp(false);
  for(auto i : v) {
    if(res.size() == 1) {
      if(res[0].d * i.d == 0 && !i.include(res[0].s))  {
        res.pop_back();
        emp = true;
      }
    } else if(res.size() >= 2) {
      while(res.size() >= 2u && !i.include(res.back(), res[res.size() - 2])) {
        if(i.d * res[res.size() - 2].d == 0 || !res.back().include(i, res[res.size() - 2])) {
          emp = true;
          break;
        }
        res.pop_back();
      }
      while(res.size() >= 2u && !i.include(res[0], res[1])) res.pop_front();
    }
    if(emp) break;
    res.push_back(i);
  }
  while (res.size() > 2u && !res[0].include(res.back(), res[res.size() - 2])) res.pop_back();
  return !emp;// emp: 是否为空，res 按顺序即为半平面交
}
```

6.8 三角形

```cpp
P fermat(const P& a, const P& b, const P& c) {
  D ab((b - a).len()), bc((b - c).len()), ca((c - a).len());
  D cosa((b - a) % (c - a) / ab / ca);
  D cosb((a - b) % (c - b) / ab / bc);
  D cosc((b - c) % (a - c) / ca / bc);
  P mid; D sq3(sqrt(3) / 2);
  if(sign((b - a) * (c - a)) < 0) swap(b, c);
  if(sign(cosa + 0.5) < 0) mid = a;
  else if(sign(cosb + 0.5) < 0) mid = b;
  else if(sign(cosc + 0.5) < 0) mid = c;
  else mid = intersection(Line(a, c + (b - c).rot(sq3) - a), Line(c, b + (a - b).rot(sq3) - c));
  return mid;
  // mid 为三角形 abc 费马点，要求 abc 非退化
  length = (mid - a).len() + (mid - b).len() + (mid - c).len();
  // 以下求法仅在三角形三个角均小于 120 度时，可以求出 ans 为费马
  length = (a - c - (b - c).rot(sq3)).len();
}
P inCenter(const P & A, const P & B, const P & C) { // 内心
  D a = (B - C).len(), b = (C - A).len(), c = (A - B).len(),
    s = abs((B - A) * (C - A)),
    r = s / (a + b + c); // 内接圆半径
  return 1. / (a + b + c) * (A * a + B * b + C * c); // 偏心则将对应点前两个加号改为减号
}
```

```cpp
P circumCenter(const P & a, const P & b, const P & c) { // 外心
  P bb = b - a, cc = c - a;
  // 半径为 a * b * c / 4 / S, a, b, c 为边长, S 为面积
  D db = bb.sqrlen(), dc = cc.sqrlen(), d = 2 * (bb * cc);
  return a - 1. / d * P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc);
}
P othroCenter(const P & a, const P & b, const P & c) { // 垂心
  P ba = b - a, ca = c - a, bc = b - c;
  D Y = ba.y * ca.y * bc.y,
       A = ca.x * ba.y - ba.x * ca.y,
       x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
       y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
  return P(x0, y0);
}
```

6.9 经纬度求球面最短距离

```cpp
double sphereDis(double lon1, double lat1, double lon2, double lat2, double R) {
  return R * acos(cos(lat1) * cos(lat2) * cos(lon1 - lon2) + sin(lat1) * sin(lat2));
}
```

6.10 长方体表面两点最短距离

```cpp
int r;
void turn(int i, int j, int x, int y, int z,int x0, int y0, int L, int W, int H) {
  if (z==0) { int R = x*x+y*y; if (R<r) r=R;
  } else {
    if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x, x0+L, y0, H, W, L);
    if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y, x0, y0+W, L, H, W);
    if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0, x0-H, y0, H, W, L);
    if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0, x0, y0-H, L, H, W);
  }
}
int main(){
  int L, H, W, x1, y1, z1, x2, y2, z2;
  cin >> L >> W >> H >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
  if (z1!=0 && z1!=H) if (y1==0 || y1==W)
       swap(y1,z1), std::swap(y2,z2), std::swap(W,H);
  else swap(x1,z1), std::swap(x2,z2), std::swap(L,H);
  if (z1==H) z1=0, z2=H-z2;
  r=0x3fffffff;
  turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  cout<<r<<endl;
}
```

6.11 点到凸包切线

```cpp
P lb(P x, vector<P> & v, int le, int ri, int sg) {
    if (le > ri) le = ri;
    int s(le), t(ri);
    while (le != ri) {
        int mid((le + ri) / 2);
        if (sign((v[mid] - x) * (v[mid + 1] - v[mid])) == sg)
            le = mid + 1; else ri = mid;
    }
    return x - v[le]; // le 即为下标, 按需返回
}
// v[0] 为顺时针上凸壳, v[1] 为顺时针下凸壳, 均允许起始两个点横坐
// 返回值为真代表严格在凸包外, 顺时针旋转在 d1 方向先碰到凸包
bool getTan(P x, vector<P> * v, P & d1, P & d2) {
    if (x.x < v[0][0].x) {
        d1 = lb(x, v[0], 0, sz(v[0]) - 1, 1);
        d2 = lb(x, v[1], 0, sz(v[1]) - 1, -1);
        return true;
    } else if(x.x > v[0].back().x) {
        d1 = lb(x, v[1], 0, sz(v[1]) - 1, 1);
        d2 = lb(x, v[0], 0, sz(v[0]) - 1, -1);
        return true;
    } else {
        for(int d(0); d < 2; d++) {
            int id(lower_bound(v[d].begin(), v[d].end(), x,
            [&](const P & a, const P & b) {
                return d == 0 ? a < b : b < a;
            }) - v[d].begin());
            if (id && (id == sz(v[d]) || (v[d][id - 1] - x) * (v[d][id] - x) > 0)) {
                d1 = lb(x, v[d], id, sz(v[d]) - 1, 1);
                d2 = lb(x, v[d], 0, id, -1);
                return true;
```

```
                }
            }
        }
        return false;
    }
```

6.12 直线与凸包的交点

```
// a 是顺时针凸包, i1 为 x 最小的点, j1 为 x 最大的点 需保证 j1 > i1
// n 是凸包上的点数, a 需复制多份或写循环数组类
int lowerBound(int le, int ri, const P & dir) {
  while (le < ri) {
    int mid((le + ri) / 2);
    if (sign((a[mid + 1] - a[mid]) * dir) <= 0) {
      le = mid + 1;
    } else ri = mid;
  }
  return le;
}
int boundLower(int le, int ri, const P & s, const P & t) {
  while (le < ri) {
    int mid((le + ri + 1) / 2);
    if (sign((a[mid] - s) * (t - s)) <= 0) {
      le = mid;
    } else ri = mid - 1;
  }
  return le;
}

void calc(P s, P t) {
  if(t < s) swap(t, s);
  int i3(lowerBound(i1, j1, t - s)); // 和上凸包的切点
  int j3(lowerBound(j1, i1 + n, s - t)); // 和下凸包的切点
  int i4(boundLower(i3, j3, s, t)); // 如果有交则是右侧的交点, 与 a[i4]~a[i4+1] 相交 要判断是否有交的话 就手动 check 一下
  int j4(boundLower(j3, i3 + n, t, s)); // 如果有交左侧的交点, 与 a[j4]~a[j4+1] 相交
  // 返回的下标不一定在 [0 ~ n-1] 内
}
```

6.13 平面最近点对

```
// Create: 2017-10-22 20:15:34
#include <bits/stdc++.h>

using namespace std;

const int N = 100005;

struct Data {
  double x, y;
};

double sqr(double x) {
  return x * x;
}
double dis(Data a, Data b) {
  return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
}

int n;
Data p[N], q[N];

double solve(int l, int r) {
  if(l == r) return 1e18;
  if(l + 1 == r) return dis(p[l], p[r]);
  int m = (l + r) / 2;
  double d = min(solve(l, m), solve(m + 1, r));
  int qt = 0;
  for(int i = l; i <= r; i++) {
    if(fabs(p[m].x - p[i].x) <= d) {
      q[++qt] = p[i];
    }
  }
  sort(q + 1, q + qt + 1, [&](const Data &a, const Data &b) {
      return a.y < b.y; });
  for(int i = 1; i <= qt; i++) {
    for(int j = i + 1; j <= qt; j++) {
      if(q[j].y - q[i].y >= d) break;
      d = min(d, dis(q[i], q[j]));
    }
  }
  return d;
}

int main()
{
  while(scanf("%d", &n) == 1 && n) {
    for(int i = 1; i <= n; i++) {
      scanf("%lf%lf", &p[i].x, &p[i].y);
    }
    sort(p + 1, p + n + 1, [&](const Data &a, const Data &b) {
        return a.x < b.x || (a.x == b.x && a.y < b.y); });
```

```
    double ans = solve(1, n);
    printf("%.2f\n", ans / 2);
  }
  return 0;
}
```

7 其他

7.1 斯坦纳树

```
priority_queue<pair<int, int> > Q;

// m is key point
// n is all point

for (int s = 0; s < (1 << m); s++){
  for (int i = 1; i <= n; i++){
    for (int s0 = (s&(s-1)); s0 ; s0=(s&(s0-1))){
        f[s][i] = min(f[s][i], f[s0][i] + f[s -
↪   s0][i]);
      }
  }
  for (int i = 1; i <= n; i++) vis[i] = 0;
    while (!Q.empty()) Q.pop();
  for (int i = 1; i <= n; i++){
    Q.push(mp(-f[s][i], i));
  }
  while (!Q.empty()){
    while (!Q.empty() && Q.top().first !=
↪    -f[s][Q.top().second]) Q.pop();
      if (Q.empty()) break;
      int Cur = Q.top().second; Q.pop();
      for (int p = g[Cur]; p; p = nxt[p]){
        int y = adj[p];
        if ( f[s][y] > f[s][Cur] + 1){
          f[s][y] = f[s][Cur] + 1;
          Q.push(mp(-f[s][y], y));
        }
      }
  }
}
```

7.2 无敌的读入优化

```
namespace Reader {
  const int L = (1 << 20) + 5;
  char buffer[L], *S, *T;
  __inline bool getchar(char &ch) {
    if (S == T) {
      T = (S = buffer) + fread(buffer, 1, L,
↪   stdin);
      if (S == T) {
        ch = EOF;
        return false;
      }
    }
    ch = *S ++;
    return true;
  }
  __inline bool getint(int &x) {
    char ch;
    for (; getchar(ch) && (ch < '0' || ch > '9');
↪   );
    if (ch == EOF) return false;
    x = ch - '0';
    for (; getchar(ch), ch >= '0' && ch <= '9'; )
      x = x * 10 + ch - '0';
    return true;
  }
}
Reader::getint(x);
Reader::getint(y);
```

7.3 最小树形图

```
const int maxn=1100;

int n,m , g[maxn][maxn] , used[maxn] , pass[maxn]
↪   , eg[maxn] , more , queue[maxn];

void combine (int id , int &sum ) {
  int tot = 0 , from , i , j , k ;
  for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
    queue[tot++]=id ; pass[id]=1;
  }
  for ( from=0; from<tot && queue[from]!=id ;
↪   from++);
  if  ( from==tot ) return ;
  more = 1 ;
  for ( i=from ; i<tot ; i++) {
    sum+=g[eg[queue[i]]][queue[i]] ;
    if ( i!=from ) {
      used[queue[i]]=1;
      for ( j = 1 ; j <= n ; j++) if ( !used[j] )
        if ( g[queue[i]][j]<g[id][j] )
↪   g[id][j]=g[queue[i]][j] ;
    }
  }
```

```
  for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
    for ( j=from ; j<tot ; j++){
      k=queue[j];
      if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
    }
  }
}

int mdst( int root ) { // return the total length of MDST
  int i , j , k , sum = 0 ;
  memset ( used , 0 , sizeof ( used ) ) ;
  for ( more =1; more ; ) {
    more = 0 ;
    memset (eg,0,sizeof(eg)) ;
    for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
      for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
        if ( k==0 || g[j][i] < g[k][i] ) k=j ;
      eg[i] = k ;
    }
    memset(pass,0,sizeof(pass));
    for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine ( i , sum ) ;
  }
  for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
  return sum ;
}
```

7.4 DLX

```
int n,m,K;
struct DLX{
  int L[maxn],R[maxn],U[maxn],D[maxn];
  int sz,col[maxn],row[maxn],s[maxn],H[maxn];
  bool vis[233];
  int ans[maxn],cnt;
  void init(int m){
    for(int i=0;i<=m;i++){
      L[i]=i-1;R[i]=i+1;
      U[i]=D[i]=i;s[i]=0;
    }
    memset(H,-1,sizeof H);
    L[0]=m;R[m]=0;sz=m+1;
  }
void Link(int r,int c){
  U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
  if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
  else{
    L[sz]=H[r];R[sz]=R[H[r]];
    L[R[H[r]]]=sz;R[H[r]]=sz;
  }
  s[c]++;col[sz]=c;row[sz]=r;sz++;
}
void remove(int c){
  for(int i=D[c];i!=c;i=D[i])
    L[R[i]]=L[i],R[L[i]]=R[i];
}
void resume(int c){
  for(int i=U[c];i!=c;i=U[i])
    L[R[i]]=R[L[i]]=i;
}
int A(){
  int res=0;
  memset(vis,0,sizeof vis);
  for(int i=R[0];i;i=R[i])if(!vis[i]){
    vis[i]=1;res++;
    for(int j=D[i];j!=i;j=D[j])
      for(int k=R[j];k!=j;k=R[k])
        vis[col[k]]=1;
  }
  return res;
}
void dfs(int d,int &ans){
  if(R[0]==0){ans=min(ans,d);return;}
  if(d+A()>=ans)return;
  int tmp=23333,c;
  for(int i=R[0];i;i=R[i])
    if(tmp>s[i])tmp=s[i],c=i;
  for(int i=D[c];i!=c;i=D[i]){
    remove(i);
    for(int j=R[i];j!=i;j=R[j])remove(j);
    dfs(d+1,ans);
    for(int j=L[i];j!=i;j=L[j])resume(j);
    resume(i);
  }
}
void del(int c){//exactly cover
      L[R[c]]=L[c];R[L[c]]=R[c];
  for(int i=D[c];i!=c;i=D[i])
    for(int j=R[i];j!=i;j=R[j])
      U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
  }
  void add(int c){  //exactly cover
```

```
        R[L[c]]=L[R[c]]=c;
    for(int i=U[c];i!=c;i=U[i])
      for(int j=L[i];j!=i;j=L[j])
        ++s[col[U[D[j]]=D[U[j]]=j]];
    }
  bool dfs2(int k){//exactly cover
        if(!R[0]){
            cnt=k;return 1;
        }
        int c=R[0];
    for(int i=R[0];i;i=R[i])
      if(s[c]>s[i])c=i;
        del(c);
    for(int i=D[c];i!=c;i=D[i]){
      for(int j=R[i];j!=i;j=R[j])
        del(col[j]);
            ans[k]=row[i];if(dfs2(k+1))return
 ↪  true;
      for(int j=L[i];j!=i;j=L[j])
        add(col[j]);
        }
        add(c);
    return 0;
  }
}dlx;
int main(){
  dlx.init(n);
  for(int i=1;i<=m;i++)
    for(int j=1;j<=n;j++)
      if(dis(station[i],city[j])<mid-eps)
        dlx.Link(i,j);
      dlx.dfs(0,ans);
}
```

7.5 插头 DP

```
int n,m,l;
struct L{
    int d[11];
    int& operator[](int x){return d[x];}
    int mc(int x){
        int an=1;
        if(d[x]==1){
            for(x++;x<l;x++)if(d[x]){
                an=an+(d[x]==1?1:-1);
                if(!an)return x;
            }
        }else{
            for(x--;x>=0;x--)if(d[x]){
                an=an+(d[x]==2?1:-1);
                if(!an)return x;
            }
        }
    }
    int h(){int an=0;for(int
 ↪  i=l-1;i>=0;i--)an=an*3+d[i];return an;}
    L s(int x,int y){
        L S=*this;
        S[x]=y;return S;
    }
    L operator>>(int _){
        L S=*this;
        for(int i=l-1;i>=1;i--)S[i]=S[i-1];
        S[0]=0;return S;
    }
};
struct Int{
    int len;
    int a[40];
    Int(){len=1;memset(a,0,sizeof a);}
    Int operator+=(const Int &o){
        int l=max(len,o.len);
        for(int i=0;i<l;i++)
            a[i]=a[i]+o.a[i];
        for(int i=0;i<l;i++)
            a[i+1]+=a[i]/10,a[i]%=10;
        if(a[l])l++;len=l;
        return *this;
    }
    void print(){
        for(int i=len-1;i>=0;i--)
            printf("%d",a[i]);
        puts("");
    }
};
struct hashtab{
    int sz;
    int tab[177147];
    Int w[177147];
    L s[177147];
    hashtab(){memset(tab,-1,sizeof tab);}
    void cl(){
        for(int i=0;i<sz;i++)tab[s[i].h()]=-1;
        sz=0;
    }
    Int& operator[](L S){
        int h=S.h();

 ↪  if(tab[h]==-1)tab[h]=sz,s[sz]=S,w[sz]=Int(),sz++;
```

```
        return w[tab[h]];
    }
}f[2];
bool check(L S){
    int cn1=0,cn2=0;
    for(int i=0;i<l;i++){
        cn1+=S[i]==1;
        cn2+=S[i]==2;
    }return cn1==1&&cn2==1;
}
int main(){
    Int One;One.a[0]=1;
    scanf("%d%d",&n,&m);if(n<m)swap(n,m);l=m+1;
    if(n==1||m==1){puts("1");return 0;}
    int cur=0;f[cur].cl();
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            if(i==1&&j==1){
                f[cur][L().s(0,1).s(1,2)]+=One;
                continue;
            }
            cur^=1;f[cur].cl();
            for(int k=0;k<f[!cur].sz;k++){
                L S=f[!cur].s[k];Int w=f[!cur][S];
                int d1=S[j-1],d2=S[j];
                if(d1==0&&d2==0){
                    if(i!=n&&j!=m)f[cur][S.s(j-1,1).s(j,2)]+=w;
                }else
                if(d1==0||d2==0){
                    if(i!=n)f[cur][S.s(j-1,d1|d2).s(j,0)]+=w;
                    if(j!=m)f[cur][S.s(j-1,0).s(j,d1|d2)]+=w;
                }else
                if(d1==1&&d2==2){
                    if(i==n&&j==m&&check(S))
                        (w+=w).print();
                }else
                if(d1==2&&d2==1){
                    f[cur][S.s(j-1,0).s(j,0)]+=w;
                }else
                if((d1==1&&d2==1)||(d1==2&&d2==2)){
                    int m1=S.mc(j),m2=S.mc(j-1);
                    f[cur][S.s(j-1,0).s(j,0).s(m1,1).s(m2,2)]+=w;
                }
            }
        }
        cur^=1;f[cur].cl();
        for(int k=0;k<f[!cur].sz;k++){
            L S=f[!cur].s[k];Int w=f[!cur][S];
            f[cur][S>>1]=w;
        }
    }
    return 0;
}
```

7.6 某年某月某日是星期几

```
int solve(int year, int month, int day) {
    int answer;
    if (month == 1 || month == 2) {
        month += 12;
        year--;
    }
    if ((year < 1752) || (year == 1752 && month < 9) ||
        (year == 1752 && month == 9 && day < 3)) {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4 + 5) % 7;
    } else {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4
                 - year / 100 + year / 400) % 7;
    }
    return answer;
}
```

7.7 枚举大小为 k 的子集

使用条件：$k > 0$

```
void solve(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        // ...
        int x = comb & -comb, y = comb + x;
        comb = (((comb & ~y) / x) >> 1) | y;
    }
}
```

7.8 环状最长公共子串

```
int n, a[N << 1], b[N << 1];
```

```cpp
bool has(int i, int j) {
    return a[(i - 1) % n] == b[(j - 1) % n];
}

const int DELTA[3][2] = {{0, -1}, {-1, -1}, {-1, 0}};

int from[N][N];

int solve() {
    memset(from, 0, sizeof(from));
    int ret = 0;
    for (int i = 1; i <= 2 * n; ++i) {
        from[i][0] = 2;
        int left = 0, up = 0;
        for (int j = 1; j <= n; ++j) {
            int upleft = up + 1 + !!from[i - 1][j];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
            int count = 0;
            for (int x = i, y = n; y; ) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
            for (; x <= i; ++x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                        y++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

7.9 LLMOD

```cpp
LL multiplyMod(LL a, LL b, LL P) { // `需要保证 a 和 b 非负`
  LL t = (a * b - LL((long double)a / P * b + 1e-3) * P) % P;
  return t < 0 : t + P : t;
}
```

7.10 STL 内存清空

```cpp
template <typename T>
__inline void clear(T& container) {
  container.clear(); // 或者删除了一堆元素
  T(container).swap(container);
}
```

7.11 开栈

```cpp
register char *_sp __asm__("rsp");
int main() {
  const int size = 400 << 20;//400MB
  static char *sys, *mine(new char[size] + size - 4096);
  sys = _sp; _sp = mine; _main(); _sp = sys;
}
```

7.12 32-bit/64-bit 随机素数

32-bit	64-bit
73550053	1249292846855685773
148898719	1701750434419805569
189560747	3605499878424114901
459874703	5648316673387803781
1202316001	6125342570814357977
1431183547	6215155308775851301
1438011109	6294606778040623451
1538762023	6347330550446020547
1557944263	7429632924303725207
1981315913	8524720079480389849

8 vimrc

```
set ruler
set number
set smartindent
set autoindent
set tabstop=4
set softtabstop=4
set shiftwidth=4
set hlsearch
set incsearch
set autoread
set backspace=2
set mouse=a

syntax on

nmap <C-A> ggVG
vmap <C-C> "+y

filetype plugin indent on

autocmd FileType cpp set cindent
autocmd FileType cpp map <F9> :w <CR> :!g++ % -o
↪   %< -g -std=c++11 -Wall -Wextra -Wconversion
↪   && size %< <CR>
autocmd FileType cpp map <C-F9> :!g++ % -o %<
↪   -std=c++11 -O2 && size %< <CR>
autocmd FileType cpp map <F8> :!time ./%< < %<.in
↪   <CR>
autocmd FileType cpp map <F5> :!time ./%< <CR>

map <F3> :vnew %<.in <CR>
map <F4> :!gedit % <CR>
```

9 常用结论

9.1 上下界网络流

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，$F(u,v)$ 表示边 (u,v) 的流量。设 $G(u,v)=F(u,v)-B(u,v)$，显然有

$$0 \le G(u,v) \le C(u,v)-B(u,v)$$

无源汇的上下界可行流

建立超级源点 S^* 和超级汇点 T^*，对于原图每条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

有源汇的上下界可行流

从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照**无源汇的上下界可行流**一样做即可，流量即为 $T \to S$ 边上的流量。

有源汇的上下界最大流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下届为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最大流。
2. 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 和超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使

得新网络存在**无源汇的上下界可行流**即为原图的最小流。

2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*, T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

9.2 上下界费用流

来源：BZOJ 3876 设汇 t，源 s，超级源 S，超级汇 T，本质是每条边的下界为 1，上界为 MAX，跑一遍有源汇的上下界最小费用最小流。（因为上界无穷大，所以只要满足所有下界的最小费用最小流）

1. 对每个点 x：从 x 到 t 连一条费用为 0，流量为 MAX 的边，表示可以任意停止当前的剧情（接下来的剧情从更优的路径去走，画个样例就知道了）
2. 对于每一条边权为 `z` 的边 `x->y`：
   - 从 `S` 到 `y` 连一条流量为 1，费用为 `z` 的边，代表这条边至少要被走一次。
   - 从 `x` 到 `y` 连一条流量为 MAX，费用为 `z` 的边，代表这条边除了至少走的一次之外还可以随便走。
   - 从 `x` 到 `T` 连一条流量为 1，费用为 0 的边。（注意是每一条 `x->y` 的边都连，或者你可以记下 `x` 的出边数 `Kx`，连一次流量为 `Kx`，费用为 0 的边）。

建完图后从 `S` 到 `T` 跑一遍费用流，即可。（当前跑出来的就是满足上下界的最小费用最小流了）

9.3 弦图相关

1. 团数 $\leq$ 色数，弦图团数 = 色数
2. 设 $next(v)$ 表示 $N(v)$ 中最前的点．令 `w*` 表示所有满足 $A \in B$ 的 `w` 中最后的一个点，判断 $v \cup N(v)$ 是否为极大团，只需判断是否存在一个 `w`，满足 $Next(w) = v$ 且 $|N(v)| + 1 \leq |N(w)|$ 即可．
3. 最小染色：完美消除序列从后往前依次给每个点染色，给每个点染上可以染的最小的颜色
4. 最大独立集：完美消除序列从前往后能选就选
5. 弦图最大独立集数 = 最小团覆盖数，最小团覆盖：设最大独立集为 $\{p_1, p_2, \ldots, p_t\}$，则 $\{p_1 \cup N(p_1), \ldots, p_t \cup N(p_t)\}$ 为最小团覆盖

9.4 Bernoulli 数

1. 初始化：$B_0(n) = 1$
2. 递推公式：

$$B_m(n) = n^m - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k(n)}{m-k+1}$$

3. 应用：

$$\sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} n^{m+1-k}$$

10 常见错误

1. 数组或者变量类型开错，例如将 `double` 开成 `int`；
2. 函数忘记返回返回值；
3. 初始化数组没有初始化完全；
4. 对空间限制判断不足导致 MLE；
5. 对于重边未注意，
6. 对于 0、1base 未弄清楚，用混
7. `map` 的赋值问题（`dis[] = find(dis[])`）
8. 输出格式

11 测试列表

1. 检测评测机是否开 O2;
2. 检测 __int128 以及 __float128 是否能够使用;
3. 检测是否能够使用 C++11;
4. 检测是否能够使用 Ext Lib;
5. 检测程序运行所能使用的内存大小;
6. 检测程序运行所能使用的栈大小;
7. 检测是否有代码长度限制;
8. 检测是否能够正常返 Runtime Error (assertion、return 1、空指针);
9. 查清楚厕所方位和打印机方位;

12 Java

12.1 Java Hints

```java
import java.util.*;
import java.math.*;
import java.io.*;

public class Main{
  static class Task{
    void solve(int testId, InputReader cin,
↪   PrintWriter cout) {
      // Write down the code you want
    }
  };

  public static void main(String args[]) {
    InputStream inputStream = System.in;
    OutputStream outputStream = System.out;
    InputReader in = new
↪   InputReader(inputStream);
    PrintWriter out = new
↪   PrintWriter(outputStream);
    TaskA solver = new TaskA();
    solver.solve(1, in, out);
    out.close();
  }

  static class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream) {
      reader = new BufferedReader(new
↪   InputStreamReader(stream), 32768);
      tokenizer = null;
    }

    public String next() {
      while (tokenizer == null ||
↪   !tokenizer.hasMoreTokens()) {
        try {
          tokenizer = new
↪   StringTokenizer(reader.readLine());
        } catch (IOException e) {
          throw new RuntimeException(e);
        }
      }
      return tokenizer.nextToken();
    }

    public int nextInt() {
      return Integer.parseInt(next());
    }

  }
};
// Arrays
int a[];
.fill(a[, int fromIndex, int toIndex],val); |
↪  .sort(a[, int fromIndex, int toIndex])
// String
String s;
.charAt(int i); | compareTo(String) |
↪  compareToIgnoreCase () | contains(String) |
length () | substring(int l, int len)
// BigInteger
.abs() | .add() | bitLength () | subtract () |
↪  divide () | remainder () |
↪  divideAndRemainder () | modPow(b, c) |
pow(int) | multiply () | compareTo () |
gcd() | intValue () | longValue () |
↪  isProbablePrime(int c) (1 - 1/2^c) |
nextProbablePrime () | shiftLeft(int) | valueOf
↪  ()
// BigDecimal
.ROUND_CEILING | ROUND_DOWN_FLOOR |
↪  ROUND_HALF_DOWN | ROUND_HALF_EVEN |
↪  ROUND_HALF_UP | ROUND_UP
```

```
.divide(BigDecimal b, int scale , int round_mode)
↪   | doubleValue () | movePointLeft(int) |
↪   pow(int) |
setScale(int scale , int round_mode) |
↪   stripTrailingZeros ()
 BigDecimal.setScale()方法用于格式化小数点
setScale(1)表示保留一位小数，默认用四舍五入方式
setScale(1,BigDecimal.ROUND_DOWN)直接删除多余的小数位，如 2.35会变成 2.3
setScale(1,BigDecimal.ROUND_UP)进位处理，2.35变成 2.4
setScale(1,BigDecimal.ROUND_HALF_UP)四舍五入，2.35变成 2.4
setScaler(1,BigDecimal.ROUND_HALF_DOWN)四舍五入，2.35变成 2.3，如果是 5 则向下舍
setScaler(1,BigDecimal.ROUND_CEILING)接近正无穷大的舍入
setScaler(1,BigDecimal.ROUND_FLOOR)接近负无穷大的舍入，数字>0和 ROUND_UP 作用一样，数字<0和 ROUND_DOWN 作用一样
setScaler(1,BigDecimal.ROUND_HALF_EVEN)向最接近的数字舍入，如果与两个相邻数字的距离相等，则向相邻的偶数舍入。
// StringBuilder
StringBuilder sb = new StringBuilder ();
sb.append(elem) | out.println(sb)
// TODO Java STL 的使用方法以及上面这些方法的检验
```

13 数学

13.1 常用数学公式

13.1.1 求和公式

1. $\sum_{k=1}^{n}(2k-1)^2=\frac{n(4n^2-1)}{3}$
2. $\sum_{k=1}^{n}k^3=[\frac{n(n+1)}{2}]^2$
3. $\sum_{k=1}^{n}(2k-1)^3=n^2(2n^2-1)$
4. $\sum_{k=1}^{n}k^4=\frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
5. $\sum_{k=1}^{n}k^5=\frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
6. $\sum_{k=1}^{n}k(k+1)=\frac{n(n+1)(n+2)}{3}$
7. $\sum_{k=1}^{n}k(k+1)(k+2)=\frac{n(n+1)(n+2)(n+3)}{4}$
8. $\sum_{k=1}^{n}k(k+1)(k+2)(k+3)=\frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

13.1.2 斐波那契数列

1. $fib_0=0, fib_1=1, fib_n=fib_{n-1}+fib_{n-2}$
2. $fib_{n+2}\cdot fib_n-fib_{n+1}^2=(-1)^{n+1}$
3. $fib_{-n}=(-1)^{n-1}fib_n$
4. $fib_{n+k}=fib_k\cdot fib_{n+1}+fib_{k-1}\cdot fib_n$
5. $gcd(fib_m, fib_n)=fib_{gcd(m,n)}$
6. $fib_m|fib_n^2\Leftrightarrow nfib_n|m$

13.1.3 错排公式

1. $D_n=(n-1)(D_{n-2}-D_{n-1})$
2. $D_n=n!\cdot(1-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\ldots+\frac{(-1)^n}{n!})$

13.1.4 莫比乌斯函数

$$\mu(n)=\begin{cases}1 & 若n=1\\ (-1)^k & 若n无平方数因子，且n=p_1p_2\ldots p_k\\ 0 & 若n有大于1的平方数因数\end{cases}$$

$$\sum_{d|n}\mu(d)=\begin{cases}1 & 若n=1\\ 0 & 其他情况\end{cases}$$

$$g(n)=\sum_{d|n}f(d)\Leftrightarrow f(n)=\sum_{d|n}\mu(d)g(\frac{n}{d})$$

$$g(x)=\sum_{n=1}^{[x]}f(\frac{x}{n})\Leftrightarrow f(x)=\sum_{n=1}^{[x]}\mu(n)g(\frac{x}{n})$$

13.1.5 伯恩赛德引理

设 G 是一个有限群，作用在集合 X 上。对每个 g 属于 G，令 X^g 表示 X 中在 g 作用下的不动元素，轨道数（记作 $|X/G|$）由如下公式给出：

$$|X/G|=\frac{1}{|G|}\sum_{g\in G}|X^g|.$$

13.1.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

$$p(n)=\sum_{k\in\mathbb{Z}\setminus\{0\}}(-1)^{k-1}p\left(n-\frac{k(3k-1)}{2}\right)$$

13.1.7 树的计数

1. 有根树计数：$n+1$ 个结点的有根树的个数为

$$a_{n+1}=\frac{\sum_{j=1}^{n}j\cdot a_j\cdot S_{n,j}}{n}$$

其中，

$$S_{n,j}=\sum_{i=1}^{n/j}a_{n+1-ij}=S_{n-j,j}+a_{n+1-j}$$

2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为

$$a_n-\sum_{i=1}^{n/2}a_ia_{n-i}$$

当 n 为偶数时，n 个结点的无根树的个数为

$$a_n-\sum_{i=1}^{n/2}a_ia_{n-i}+\frac{1}{2}a_{\frac{n}{2}}(a_{\frac{n}{2}}+1)$$

3. n 个结点的完全图的生成树个数为

$$n^{n-2}$$

4. 矩阵 - 树定理：图 G 由 n 个结点构成，设 $\boldsymbol{A}[G]$ 为图 G 的邻接矩阵、$\boldsymbol{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\boldsymbol{C}[G]=\boldsymbol{D}[G]-\boldsymbol{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

13.1.8 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系：

$$V-E+F=C+1$$

其中，V 是顶点的数目，E 是边的数目，F 是面的数目，C 是组成图形的连通部分的数目。当图是单连通图的时候，公式简化为：

$$V-E+F=2$$

13.1.9 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i、边上格点数目 b 的关系：

$$A=i+\frac{b}{2}-1$$

13.1.10 牛顿恒等式

设

$$\prod_{i=1}^{n}(x-x_i)=a_n+a_{n-1}x+\cdots+a_1x^{n-1}+a_0x^n$$

$$p_k=\sum_{i=1}^{n}x_i^k$$

则

$$a_0p_k+a_1p_{k-1}+\cdots+a_{k-1}p_1+ka_k=0$$

特别地，对于

$$|\boldsymbol{A}-\lambda\boldsymbol{E}|=(-1)^n(a_n+a_{n-1}\lambda+\cdots+a_1\lambda^{n-1}+a_0\lambda^n)$$

有

$$p_k=Tr(\boldsymbol{A}^k)$$

13.2 平面几何公式

13.2.1 三角形

1. 半周长

$$p=\frac{a+b+c}{2}$$

2. 面积

$$S=\frac{a\cdot H_a}{2}=\frac{ab\cdot sinC}{2}=\sqrt{p(p-a)(p-b)(p-c)}$$

3. 中线

$$M_a=\frac{\sqrt{2(b^2+c^2)-a^2}}{2}=\frac{\sqrt{b^2+c^2+2bc\cdot cosA}}{2}$$

4. 角平分线

$$T_a=\frac{\sqrt{bc\cdot[(b+c)^2-a^2]}}{b+c}=\frac{2bc}{b+c}cos\frac{A}{2}$$

5. 高线

$$H_a=bsinC=csinB=\sqrt{b^2-(\frac{a^2+b^2-c^2}{2a})^2}$$

6．内切圆半径

$$r=\frac{S}{p}=\frac{arcsin\frac{B}{2}\cdot sin\frac{C}{2}}{sin\frac{B+C}{2}}=4R\cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2}$$

$$=\sqrt{\frac{(p-a)(p-b)(p-c)}{p}}=p\cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2}$$

7．外接圆半径

$$R=\frac{abc}{4S}=\frac{a}{2sinA}=\frac{b}{2sinB}=\frac{c}{2sinC}$$

13.2.2 四边形

D_1,D_2 为对角线，M 对角线中点连线，A 为对角线夹角，p 为半周长

1．$a^2+b^2+c^2+d^2=D_1^2+D_2^2+4M^2$

2．$S=\frac{1}{2}D_1D_2sinA$

3．对于圆内接四边形

$$ac+bd=D_1D_2$$

4．对于圆内接四边形

$$S=\sqrt{(p-a)(p-b)(p-c)(p-d)}$$

13.2.3 正 n 边形

R 为外接圆半径，r 为内切圆半径

1．中心角

$$A=\frac{2\pi}{n}$$

2．内角

$$C=\frac{n-2}{n}\pi$$

3．边长

$$a=2\sqrt{R^2-r^2}=2R\cdot sin\frac{A}{2}=2r\cdot tan\frac{A}{2}$$

4．面积

$$S=\frac{nar}{2}=nr^2\cdot tan\frac{A}{2}=\frac{nR^2}{2}\cdot sinA=\frac{na^2}{4\cdot tan\frac{A}{2}}$$

13.2.4 圆

1．弧长

$$l=rA$$

2．弦长

$$a=2\sqrt{2hr-h^2}=2r\cdot sin\frac{A}{2}$$

3．弓形高

$$h=r-\sqrt{r^2-\frac{a^2}{4}}=r(1-cos\frac{A}{2})=\frac{1}{2}\cdot arctan\frac{A}{4}$$

4．扇形面积

$$S_1=\frac{rl}{2}=\frac{r^2A}{2}$$

5．弓形面积

$$S_2=\frac{rl-a(r-h)}{2}=\frac{r^2}{2}(A-sinA)$$

13.2.5 棱柱

1．体积

$$V=Ah$$

A 为底面积，h 为高

2．侧面积

$$S=lp$$

l 为棱长，p 为直截面周长

3．全面积

$$T=S+2A$$

13.2.6 棱锥

1．体积

$$V=Ah$$

A 为底面积，h 为高

2．正棱锥侧面积

$$S=lp$$

l 为棱长，p 为直截面周长

3．正棱锥全面积

$$T=S+2A$$

13.2.7 棱台

1. 体积

$$V = (A_1 + A_2 + \sqrt{A_1 A_2}) \cdot \frac{h}{3}$$

A_1, A_2 为上下底面积，h 为高

2. 正棱台侧面积

$$S = \frac{p_1 + p_2}{2} l$$

p_1, p_2 为上下底面周长，l 为斜高

3. 正棱台全面积

$$T = S + A_1 + A_2$$

13.2.8 圆柱

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = 2\pi r(h + r)$$

3. 体积

$$V = \pi r^2 h$$

13.2.9 圆锥

1. 母线

$$l = \sqrt{h^2 + r^2}$$

2. 侧面积

$$S = \pi r l$$

3. 全面积

$$T = \pi r(l + r)$$

4. 体积

$$V = \frac{\pi}{3} r^2 h$$

13.2.10 圆台

1. 母线

$$l = \sqrt{h^2 + (r_1 - r_2)^2}$$

2. 侧面积

$$S = \pi(r_1 + r_2) l$$

3. 全面积

$$T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$$

4. 体积

$$V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1 r_2) h$$

13.2.11 球

1. 全面积

$$T = 4\pi r^2$$

2. 体积

$$V = \frac{4}{3}\pi r^3$$

13.2.12 球台

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

$$T = \pi(2rh + r_1^2 + r_2^2)$$

3. 体积

$$V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$$

13.2.13 球扇形

1. 全面积

$$T = \pi r(2h + r_0)$$

h 为球冠高，r_0 为球冠底面半径

2. 体积

$$V = \frac{2}{3}\pi r^2 h$$

13.3 积分表

$\int \frac{1}{1+x^2} dx = \tan^{-1} x$

$\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a}$

$\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln |a^2 + x^2|$

$\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a}$

$\int \sqrt{x^2 \pm a^2} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right|$

$\int \sqrt{a^2 - x^2} dx = \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2-x^2}}$

$\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right|$

$\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \ln \left| x + \sqrt{x^2 \pm a^2} \right|$

$\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a}$

$\int \frac{x}{\sqrt{x^2 \pm a^2}} dx = \sqrt{x^2 \pm a^2}$

$\int \frac{x}{\sqrt{a^2 - x^2}} dx = -\sqrt{a^2 - x^2}$

$\int \sqrt{ax^2 + bx + c} dx = \frac{b+2ax}{4a} \sqrt{ax^2 + bx + c} + \frac{4ac-b^2}{8a^{3/2}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx^+ c)} \right|$

$\int x^n e^{ax} \, \mathrm{d}x = \dfrac{x^n e^{ax}}{a} - \dfrac{n}{a} \int x^{n-1} e^{ax} \, \mathrm{d}x$

$\int \sin^2 ax dx = \frac{x}{2} - \frac{1}{4a} \sin 2ax$

$\int \sin^3 ax dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a}$

$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$

$\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a}$

$\int \tan ax dx = -\frac{1}{a} \ln \cos ax$

$\int \tan^2 ax dx = -x + \frac{1}{a} \tan ax$

$\int x \cos ax dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax$

$\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax$

$\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2}$

$\int x^2 \sin ax dx = \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2}$