代码库

Blazar

2017 年 10 月 23 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)+solve((a-1)/k,(c-1)/k);
            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：

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

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 扩展小步大步

```
LL exBSGS(LL a, LL b, LL p) {
//  a^x=b (mod p)
    b %= p;
    LL e = 1 % p;
    for(int i = 0; i < 100; i++) {
        if(e == b) return i;
        e = e * a % p;
    }
    int r = 0;
    while(gcd(a, p) != 1) {
        LL d = gcd(a, p);
        if(b % d) return -1;
        p /= d;
        b /= d;
        b = b * inv(a / d, p);
```

```
        r++;
    }
    LL res = BSGS(a, b, p);
    if(res == -1) return -1;
    return res + r;
}
```

1.8 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;
}
```

1.9 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;
}
```

1.10 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;
}
```

1.11 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);
}
```

1.12 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);
```

```
}
```

1.13 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;
}
```

1.14 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];
```

```
    }
}
```

1.15 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];
            }
        }
    }
}
```

1.16 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：$\mathcal{O}(nlogn)$

```cpp
//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

```cpp
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 快速傅立叶变换

返回结果：

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

时间复杂度：$\mathcal{O}(nlogn)$

```cpp
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

```cpp
// 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 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));
}
```

2.5 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;
}
```

3 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;
    }
}
```

4 魔幻多项式

快速傅里叶变换

注意事项：请实现复数类 `Complex`，并注意快速傅里叶变换精度较差，建议使用快速数论变换。

```cpp
int prepare(int n) {
  int len = 1;
  for (; len <= 2 * n; len <<= 1);
  for (int i = 0; i < len; i++) {
    e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
    e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
  }
  return len;
}
void DFT(Complex *a, int n, int f) {
  for (int i = 0, j = 0; i < n; i++) {
    if (i > j) std::swap(a[i], a[j]);
    for (int t = n >> 1; (j ^= t) < t; t >>= 1);
  }
  for (int i = 2; i <= n; i <<= 1)
    for (int j = 0; j < n; j += i)
      for (int k = 0; k < (i >> 1); k++) {
        Complex A = a[j + k];
        Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
        a[j + k] = A + B;
        a[j + k + (i >> 1)] = A - B;
      }
  if (f == 1) {
    for (int i = 0; i < n; i++)
      a[i].a /= n;
  }
}
```

光速数论变换

注意事项： MOD 应该为一个特殊的质数 $2^n + 1$ 且 n 应该要足够大，PRT 为这个质数的原根。

```cpp
// meminit(A, l, r) 是将数组 A 的 [l, r) 清 0。
// memcopy(target, source, l, r) 是将 source 的 [l, r) 复制到 target 的 [l, r)
#define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
#define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
void DFT(int *a, int n, int f) { // 封闭形式，常数小（$10^7$ 跑 2.23 秒）
  for (register int i = 0, j = 0; i < n; i++) {
    if (i > j) std::swap(a[i], a[j]);
    for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
  }
  for (register int i = 2; i <= n; i <<= 1) {
    static int exp[MAXN];
    exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i);
    if (f == 1) exp[1] = fpm(exp[1], MOD - 2);
    for (register int k = 2; k < (i >> 1); k++) {
      exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
    }
    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] % MOD;
        pA = (A + B) % MOD;
        pB = (A - B + MOD) % MOD;
      }
    }
  }
  if (f == 1) {
    register int rev = fpm(n, MOD - 2, MOD);
    for (register int i = 0; i < n; i++) {
      a[i] = 1ll * a[i] * rev % MOD;
    }
  }
}
// 在不写高精度的情况下合并 FFT 所得结果对 MOD 取模后的答案
// 值得注意的是，这个东西不能最后再合并，而是应该每做一次多项式乘法就 CRT 一次
int CRT(int *a) {
  static int x[3];
  for (int i = 0; i < 3; i++) {
    x[i] = a[i];
    for (int j = 0; j < i; j++) {
      int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
      if (t < 0) t += FFT[i] -> MOD;
      x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
    }
  }
  int sum = 1, ret = x[0] % MOD;
  for (int i = 1; i < 3; i ++) {
    sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
    ret += 1LL * x[i] * sum % MOD;
    if(ret >= MOD) ret -= MOD;
  }
  return ret;
}
for (int i = 0; i < 3; i++) // inv 数组的预处理过程，inverse(x, p) 表示求 x 在 p 下逆元
  for (int j = 0; j < 3; j++)
    inv[i][j] = inverse(FFT[i] -> MOD, FFT[j] -> MOD);
```

牛顿迭代法

问题描述：给出多项式 $G(x)$，求解多项式 $F(x)$ 满足：

$$G(F(x)) \equiv 0 \pmod{x^n}$$

答案只需要精确到 $F(x) \bmod x^n$ 即可。

实现原理：考虑倍增，假设有：

$$G(F_t(x)) \equiv 0 \pmod{x^t}$$

对 $G(F_{t+1}(x))$ 在模 x^{2t} 意义下进行 Taylor 展开：

$$G(F_{t+1}(x)) \equiv G(F_t(x)) + \frac{G'(F_t(x))}{1!}(F_{t+1}(x) - F_t(x)) \pmod{x^{2t}}$$

那么就有：

$$F_{t+1}(x) \equiv F_t(x) - \frac{G(F_t(x))}{G'(F_t(x))} \pmod{x^{2t}}$$

注意事项： $G(F(x))$ 的常数项系数必然为 0，这个可以作为求解的初始条件；

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：

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

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);
    }
}
```

多项式取指数和对数

作用： 给出一个多项式 $A(x)$，求一个多项式 $F(x)$ 满足 $e^A(x) - F(x) \equiv 0 \pmod{x^n}$。
原理： 令 $G(x) = \ln x - A$（其中 A 是一个多项式系数），根据牛顿迭代法有：

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

求 $\ln F_t(x)$ 可以用先求导再积分的办法，即：

$$\ln A(x) = \int \frac{F'(x)}{F(x)} \, \mathrm{d}x$$

多项式的求导和积分可以在 $O(n)$ 的时间内完成，因此总复杂度为 $O(n \log n)$。

应用： 加速多项式快速幂。

注意事项：

1. 进行 log 的多项式必须保证常数项系数为 1，否则必须要先求出 $\log a[0]$ 是多少；
2. 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；
3. 常数比较大，10^5 的数据求指数和对数分别需要 0.37s 和 0.85s 左右的时间，注意这里 memset 几乎不占用时。

```
void getDiff(int *a, int *b, int n) { // 多项式取微分
  for (int i = 0; i + 1 < n; i++) {
    b[i] = 1ll * (i + 1) * a[i + 1] % MOD;
  }
  b[n - 1] = 0;
}
void getInt(int *a, int *b, int n) { // 多项式取积分，积分常数为 0
  static int inv[MAXN];
  inv[1] = 1;
  for (int i = 2; i < n; i++) {
    inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
  }
  b[0] = 0;
  for (int i = 1; i < n; i++) {
    b[i] = 1ll * a[i - 1] * inv[i] % MOD;
  }
}
void getLn(int *a, int *b, int n) {
  static int inv[MAXN], d[MAXN];
  int M = 1;
  for (; M <= 2 * (n - 1); M <<= 1);
  getInv(a, inv, n);
  getDiff(a, d, n);
  meminit(d, n, M);
  meminit(inv, n, M);
  DFT(d, M, 0); DFT(inv, M, 0);
  for (int i = 0; i < M; i++) {
    d[i] = 1ll * d[i] * inv[i] % MOD;
  }
  DFT(d, M, 1);
  getInt(d, b, n);
}
void getExp(int *a, int *b, int n) {
  static int ln[MAXN], tmp[MAXN];
  b[0] = 1;
```

```
  for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
    for (; M <= 2 * (c - 1); M <<= 1);
    int bound = std::min(c, n);
    memcopy(tmp, a, 0, bound);
    meminit(tmp, bound, M);
    meminit(b, c, M);
    getLn(b, ln, c);
    meminit(ln, c, M);
    DFT(b, M, 0);
    DFT(tmp, M, 0);
    DFT(ln, M, 0);
    for (int i = 0; i < M; i++) {
      b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + 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;
  }
}
```

5 数据结构

5.1 平衡的二叉查找树

5.1.1 Treap

```
struct Node {
  Node *lc, *rc;
  int r, v, cnt, sz;
  Node() {}
  Node(int _v) {
    v = _v;
    sz = 0;
    cnt = 1;
    r = rand();
    lc = rc = NULL;
  }
  void update();
};

inline int size(Node *o) {
  return o ? o->sz : 0;
}
void Node::update() {
  sz = cnt + size(lc) + size(rc);
}
void rotate_l(Node *&o) {
  Node *k = o->rc;
  o->rc = k->lc;
  k->lc = o;
  o->update();
  k->update();
  o = k;
```

```
}
void rotate_r(Node *&o) {
  Node *k = o->lc;
  o->lc = k->rc;
  k->rc = o;
  o->update();
  k->update();
  o = k;
}
void insert(Node *&o, int v) {
  if(o == NULL) {
    o = new Node(v);
  } else if(v < o->v) {
    insert(o->lc, v);
    if(o->lc->r < o->r)
      rotate_r(o);
  } else if(v > o->v) {
    insert(o->rc, v);
    if(o->rc->r < o->r)
      rotate_l(o);
  } else o->cnt++;
  o->update();
}
void remove(Node *&o, int v) { // 删除前请确保 v 存在
  if(v < o->v) {
    remove(o->lc, v);
  } else if(v > o->v) {
    remove(o->rc, v);
  } else if(o->cnt == 1) {
    if(o->lc == NULL) {
      Node *k = o;
      o = o->rc;
      delete k;
    } else if(o->rc == NULL) {
      Node *k = o;
      o = o->lc;
      delete k;
    } else {
      if(o->lc->r < o->rc->r) {
        rotate_l(o);
        remove(o->lc, v);
      } else {
        rotate_r(o);
        remove(o->rc, v);
      }
    }
  } else o->cnt--;
  if(o) o->update();
}
```

```
void merge(Node *&to, Node *from) {
  if(from == NULL) return;
  merge(to, from->lc);
  for(int i = 0; i < from->cnt; ++i)
    insert(to, from->v);
  merge(to, from->rc);
}
void del_tree(Node *&o) {
  if(o == NULL) return;
  del_tree(o->lc);
  del_tree(o->rc);
  delete o;
  o = NULL;
}
int get_rank(Node *o, int v) { // 查询小于等于 v 的个数
  if(o == NULL) return 0;
  if(v < o->v) return get_rank(o->lc, v);
  else if(v == o->v) return size(o->lc) + o->cnt;
  else return size(o->lc) + o->cnt + get_rank(o->rc, v);
}
```

5.1.2 Splay

```
struct Node {
    int fa, c[2];
    int val, sum, sz;
    int rev, tag, inv;
    int lmx, lmn, rmx, rmn;
} T[N];
int root;

#define fa(o) T[o].fa
#define lc(o) T[o].c[0]
#define rc(o) T[o].c[1]
#define sz(o) T[o].sz
#define val(o) T[o].val
#define sum(o) T[o].sum
#define rev(o) T[o].rev
#define tag(o) T[o].tag
#define inv(o) T[o].inv
#define lmx(o) T[o].lmx
#define rmx(o) T[o].rmx
#define lmn(o) T[o].lmn
#define rmn(o) T[o].rmn
#define ctype(x) (x == rc(fa(x)))

inline void setc(int x, int y, const bool d) { if(x) fa(x) = y; if(y) T[y].c[d] = x; }
void set_tag(int o, int v) {
    if(o == 0) return;
    rev(o) = inv(o) = 0;
```

```
    val(o) = tag(o) = v;
    sum(o) = v * sz(o);
    lmx(o) = lmn(o) = rmx(o) = rmn(o) = 0;
    relax(lmn(o), sum(o)); tense(lmx(o), sum(o));
    relax(rmn(o), sum(o)); tense(rmx(o), sum(o));
}
void set_inv(int o) {
    if(o == 0) return;
    inv(o) ^= 1; sum(o) *= -1; val(o) *= -1;
    swap(lmn(o), lmx(o)); lmn(o) *= -1; lmx(o) *= -1;
    swap(rmn(o), rmx(o)); rmn(o) *= -1; rmx(o) *= -1;
}
void set_rev(int o) {
    if(o == 0) return;
    rev(o) ^= 1;
    swap(lc(o), rc(o));
    swap(lmx(o), rmx(o));
    swap(lmn(o), rmn(o));
}
inline void pushdown(int o) {
    if(tag(o)) set_tag(lc(o), tag(o)), set_tag(rc(o), tag(o)), tag(o) = 0;
    if(rev(o)) set_rev(lc(o)), set_rev(rc(o)), rev(o) = 0;
    if(inv(o)) set_inv(lc(o)), set_inv(rc(o)), inv(o) = 0;
}
inline void update(int o) {
    if(o == 0) return;
    sz(o) = sz(lc(o)) + sz(rc(o)) + 1;
    sum(o) = sum(lc(o)) + sum(rc(o)) + val(o);
    lmx(o) = lmn(o) = rmx(o) = rmn(o) = 0;
    relax(lmn(o), lmn(lc(o))); relax(lmn(o), sum(lc(o)) + val(o) + lmn(rc(o)));
    tense(lmx(o), lmx(lc(o))); tense(lmx(o), sum(lc(o)) + val(o) + lmx(rc(o)));
    relax(rmn(o), rmn(rc(o))); relax(rmn(o), sum(rc(o)) + val(o) + rmn(lc(o)));
    tense(rmx(o), rmx(rc(o))); tense(rmx(o), sum(rc(o)) + val(o) + rmx(lc(o)));
}
inline void rotate(int o) {
    int p = fa(o), mark = ctype(o);
    if(fa(p)) setc(o, fa(p), ctype(p));
    else fa(o) = 0, root = o;
    setc(T[o].c[mark ^ 1], p, mark);
    setc(p, o, mark ^ 1);
    update(p); update(o);
}
void splay(int x, int rt = 0) {
    static int q[N], top;
    q[top = 1] = x;
    for(int i = x; i != rt; i = fa(i))
        q[++top] = fa(i);
    while(top) pushdown(q[top--]);
    while(fa(x) != rt) {
```

```
        if(fa(fa(x)) != rt) {
            if(ctype(x) == ctype(fa(x)))
                rotate(fa(x));
            else rotate(x);
        }
        rotate(x);
    }
}
int find(int rank) {
    int x = root;
    while(true) {
        pushdown(x);
        int s = sz(lc(x)) + 1;
        if(rank < s) x = lc(x);
        else if(rank == s) return x;
        else rank -= s, x = rc(x);
    }
}
void init() {
    for(int i = 1; i <= n; ++i) {
        sz(i) = 1;
        if(src[i] == '(') {
            val(i) = sum(i) = 1;
            lmx(i) = rmx(i) = 1;
            lmn(i) = rmn(i) = 0;
        } else {
            val(i) = sum(i) = -1;
            lmx(i) = rmx(i) = 0;
            lmn(i) = rmn(i) = -1;
        }
    }
    for(int i = 1; i <= n; ++i) {
        setc(i, i + 1, 0);
        update(i + 1);
    }
    root = n + 1;
}
inline int interval(int l, int r) {
    int x, y;
    splay(x = find(r + 1));
    if(l - 1) {
        splay(y = find(l - 1), x);
        return rc(y);
    } else {
    return lc(x);
  }
}
```

5.2 坚固的数据结构

5.2.1 坚固的平衡树

```
#define sz(x) (x?x->siz:0)
struct node{
    int siz,key;
    LL val,sum;
    LL mu,a,d;
    node *c[2],*f;
    void split(int ned,node *&p,node *&q);
    node* rz(){
        sum=val;siz=1;
        if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
        if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
        return this;
    }
    void make(LL _mu,LL _a,LL _d){
        sum=sum*_mu+_a*siz+_d*siz*(siz-1)/2;
        val=val*_mu+_a+_d*sz(c[0]);
        mu*=_mu;a=a*_mu+_a;d=d*_mu+_d;
    }
    void pd(){
        if(mu==1&&a==0&&d==0)return;
        if(c[0])c[0]->make(mu,a,d);
        if(c[1])c[1]->make(mu,a+d+d*sz(c[0]),d);
        mu=1;a=d=0;
    }
    node(){mu=1;}
}nd[maxn*2],*root;
node *merge(node *p,node *q){
    if(!p||!q)return p?p->rz():(q?q->rz():0);
    p->pd();q->pd();
    if(p->key<q->key){
        p->c[1]=merge(p->c[1],q);
        return p->rz();
    }else{
        q->c[0]=merge(p,q->c[0]);
        return q->rz();
    }
}
void node::split(int ned,node *&p,node *&q){
    if(!ned){p=0;q=this;return;}
    if(ned==siz){p=this;q=0;return;}
    pd();
    if(sz(c[0])>=ned){
        c[0]->split(ned,p,q);c[0]=0;rz();
        q=merge(q,this);
    }else{
        c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
```

```
        p=merge(this,p);
    }
}
int main(){
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }
}
```

5.2.2 坚固的左偏树

```
int merge(int a, int b)
{
    if(a == 0) return b;
    if(b == 0) return a;
    if(v[a] < v[b]) swap(a, b);
    rc[a] = merge(rc[a], b);
    if(rc[a]) fa[rc[a]] = a;
    if(d[lc[a]] < d[rc[a]]) swap(lc[a], rc[a]);
    d[a] = rc[a] ? d[rc[a]] + 1 : 0;
    return a;
}
```

5.3 树上的魔术师

5.3.1 轻重树链剖分

```
int n;

vector<int> g[N];

int son[N], top[N];
int dep[N], sz[N], fa[N];
int dfn[N], pos[N], segn;

void dfs(int x, int p) {
  sz[x] = 1;
  fa[x] = p;
  son[x] = -1;
  dep[x] = dep[p] + 1;
  for(auto y: g[x]) {
    if(y == p) continue;
    dfs(y, x);
    sz[x] += sz[y];
    if(son[x] == -1 || sz[son[x]] < sz[y])
      son[x] = y;
  }
```

```
}
void DFS(int x, int tp) {
  top[x] = tp;
  dfn[pos[x] = ++segn] = x;
  if(~son[x]) DFS(son[x], tp);
  for(auto y: g[x])
    if(y != fa[x] && y != son[x])
      DFS(y, y);
}
void modify(int x, int y, int c)
{
  qv = c;
  while(top[x] != top[y])
  {
    if(dep[top[x]] < dep[top[y]])
      swap(x, y);
    ql = pos[top[x]]; qr = pos[x];
    seg_modify(1, 1, segn);
    x = fa[top[x]];
  }
  if(dep[x] > dep[y]) swap(x, y);
  ql = pos[x]; qr = pos[y];
  seg_modify(1, 1, segn);
}
bool flag_res;
void seg_query(int o, int l, int r) {
  if(ql <= l && r <= qr) {
    if(flag_res)
      res = res + val[o];
    else {
      res = val[o];
      flag_res = true;
    }
    return;
  }
  pushdown(o, l, r);
  if(ql <= mid) seg_query(Lc);
  if(mid < qr) seg_query(Rc);
}
int query(int x, int y) // 点操作版
  // 边操作每个点对应其父边（根除外）
{
  bool flag_pre = false;
  bool flag_suf = false;
  while(top[x] != top[y])
  {
    flag_res = false;
    if(dep[top[x]] > dep[top[y]])
    {
```

```
      ql = pos[top[x]]; qr = pos[x];
      seg_query(1, 1, segn);
      if(flag_pre)
        pre = pre + reverse(res);
      else
      {
        pre = reverse(res);
        flag_pre = true;
      }
      x = fa[top[x]];
    }
    else
    {
      ql = pos[top[y]]; qr = pos[y];
      seg_query(1, 1, segn);
      if(flag_suf)
        suf = res + suf;
      else
      {
        suf = res;
        flag_suf = true;
      }
      y = fa[top[y]];
    }
  }
  flag_res = false;
  if(dep[x] < dep[y])
  {
    ql = pos[x]; qr = pos[y];
    seg_query(1, 1, segn);
  }
  else
  {
    ql = pos[y]; qr = pos[x];
    seg_query(1, 1, segn);
    res = reverse(res);
  }
  if(flag_pre) res = pre + res;
  if(flag_suf) res = res + suf;
  return res.mx;
}
```

5.3.2 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;
  }
};
```

5.4 RMQ

```
for (int i = 1; i <= n; i++)
  Log[i] = int(log2(i));

for (int i = 1; i <= n; i++)
  Rmq[i][0] = i;

for (int k = 1; (1 << k) <= n; k++)
  for (int i = 1; i + (1 << k) - 1 <= n; i++){
    int x = Rmq[i][k - 1], y = Rmq[i + (1 << (k - 1))][k - 1];
    if (a[x] < a[y])
      Rmq[i][k] = x;
    else
      Rmq[i][k] = y;
  }

int Smallest(int l, int r){
  int k = Log[r - l + 1];

  int x = Rmq[l][k];
  int y = Rmq[r - (1 << k) + 1][k];

  if (a[x] < a[y]) return x;
  else return y;
}
```

5.5 可持久化线段树

```
struct node1 {
  int L, R, Lson, Rson, Sum;
} tree[N * 40];
int root[N], a[N], b[N];
int tot, n, m;
int Real[N];
int Same(int x) {
  ++tot;
  tree[tot] = tree[x];
  return tot;
}
```

```
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 Ask(int Lst, int Cur, int L, int R, int k) {
  if (L == R) return L;
  int Mid = (L + R) >> 1;
  int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
  if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
  k -= Left;
  return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
}
int Add(int Lst, int pos) {
  int root = Same(Lst);
  tree[root].Sum++;
  if (tree[root].L == tree[root].R) return root;
  int mid = (tree[root].L + tree[root].R) >> 1;
  if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
  else tree[root].Rson = Add(tree[root].Rson, pos);
  return root;
}
int main() {
  scanf("%d%d", &n, &m);
  int up = 0;
  for (int i = 1; i <= n; i++){
    scanf("%d", &a[i]);
    b[i] = a[i];
  }
  sort(b + 1, b + n + 1);
  up = unique(b + 1, b + n + 1) - b - 1;
  for (int i = 1; i <= n; i++){
    int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
    Real[tmp] = a[i];
    a[i] = tmp;
  }
  tot = 0;
  root[0] = build(1, up);
  for (int i = 1; i <= n; i++){
    root[i] = Add(root[i - 1], a[i]);
  }
  for (int i = 1; i <= m; i++){
```

```
    int u, v, w;
    scanf("%d%d%d", &u, &v, &w);
    printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
  }
  return 0;
}
```

5.6 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;
}
```

```
41
42  int Ask(int Lst, int Cur, int L, int R, int pos){
43      if (L >= pos) return 0;
44      if (R < pos) return tree[Cur].Sum - tree[Lst].Sum;
45      int Mid = (L + R) >> 1;
46      int Ret = Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, pos);
47      Ret += Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, pos);
48      return Ret;
49  }
50
51  int main(){
52      while (scanf("%d", &n) == 1){
53          for (int i = 1; i <= Trie_tot; i++){
54              for (int j = 1; j <= 26; j++)
55                  Trie[i].To[j] = 0;
56              Trie[i].Lst = 0;
57          }
58          Trie_tot = 1;
59          cnt = 0;
60          for (int ii = 1; ii <= n; ii++){
61              scanf("%s", s + 1);
62              Len = strlen(s + 1);
63              int Cur = 1;
64              First[ii] = cnt + 1;
65              for (int i = 1; i <= Len; i++){
66                  int ch = s[i] - 'a' + 1;
67                  if (Trie[Cur].To[ch] == 0){
68                      ++Trie_tot;
69                      Trie[Cur].To[ch] = Trie_tot;
70                  }
71                  Cur = Trie[Cur].To[ch];
72                  Pre[++cnt] = Trie[Cur].Lst;
73                  Trie[Cur].Lst = ii;
74              }
75              Last[ii] = cnt;
76          }
77          tot = 0;
78          Root[0] = Build(0, n);
79          for (int i = 1; i <= cnt; i++){
80              Root[i] = Add(Root[i - 1], Pre[i]);
81          }
82          Lstans = 0;
83          scanf("%d", &q);
84          for (int ii = 1; ii <= q; ii++){
85              int L, R;
86              scanf("%d%d", &L, &R);
87              L = (L + Lstans) % n + 1;
88              R = (R + Lstans) % n + 1;
89              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;
}
```

5.7 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() {}
```

```cpp
  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++;
  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);
  }
}
```

5.8 莫队算法

```
// uva12345
// single testcase
#include <bits/stdc++.h>

using namespace std;

const int N = 50005;
const int MAXV = 1e6 + 5;

int n, m;
int a[N];
int bid[N];
int qn, modn;

struct Query
{
  int l, r, last, i;

  friend bool operator < (const Query &a, const Query &b)
  {
    if(bid[a.l] != bid[b.l])
```

```
      return bid[a.l] < bid[b.l];
    if(bid[a.r] != bid[b.r])
      return bid[a.r] < bid[b.r];
    return a.last < b.last;
  }
} q[N];

struct Modify
{
  int x, to, from;
} mod[N];

void input()
{
  scanf("%d%d", &n, &m);
  for(int i = 1; i <= n; i++)
    scanf("%d", &a[i]);
  for(int i = 1; i <= m; i++)
  {
    int x, y;
    static char op[2];
    scanf("%s%d%d", op, &x, &y);
    x++;
    if(op[0] == 'Q')
    {
      ++qn;
      q[qn] = (Query){x, y, modn, qn};
    }
    else if(op[0] == 'M')
    {
      mod[++modn] = (Modify){x, y, a[x]};
      a[x] = y;
    }
    else assert(false);
  }
  int BS = (int)pow(n + 0.5, 2.0 / 3.0);
  for(int i = 1; i <= n; i++)
    bid[i] = (i + BS - 1) / BS;
}

int ans[N];
int curans;
int cnt[MAXV];
bool state[N];

void modify(int x)
{
  int &c = cnt[a[x]];
  curans -= !!c;
```

```
  c += (state[x] ^= 1) ? 1 : -1;
  curans += !!c;
}

void change(int i, bool type)
{
  int x = mod[i].x;
  int newv = type ? mod[i].to : mod[i].from;
  if(state[x])
  {
    modify(x);
    a[x] = newv;
    modify(x);
  }
  else
  {
    a[x] = newv;
  }
}

void solve()
{
  sort(q + 1, q + qn + 1);
  int l = 1, r = 0, now = modn;
  for(int i = 1; i <= qn; i++)
  {
    while(q[i].last < now) change(now--, 0);
    while(q[i].last > now) change(++now, 1);
    while(q[i].l < l) modify(--l);
    while(q[i].l > l) modify(l++);
    while(q[i].r < r) modify(r--);
    while(q[i].r > r) modify(++r);
    ans[q[i].i] = curans;
  }
  for(int i = 1; i <= qn; i++)
    printf("%d\n", ans[i]);
}

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

5.9 树上莫队

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

```

```cpp
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];
```

```
101         q[i].r = st[b];
102         q[i].extra = 0;
103       }
104       else
105       {
106         q[i].l = ed[a];
107         q[i].r = st[b];
108         q[i].extra = c;
109       }
110     }
111     sort(q + 1, q + m + 1);
112   }
113
114   int curans;
115   int ans[M];
116   int cnt[N];
117   bool state[N];
118
119   void rev(int x)
120   {
121     int &c = cnt[col[x]];
122     curans -= !!c;
123     c += (state[id[x]] ^= 1) ? 1 : -1;
124     curans += !!c;
125   }
126
127   void solve()
128   {
129     prepare();
130     curans = 0;
131     memset(cnt, 0, sizeof(cnt));
132     memset(state, 0, sizeof(state));
133
134     int l = 1, r = 0;
135     for(int i = 1; i <= m; i++)
136     {
137       while(l < q[i].l) rev(l++);
138       while(l > q[i].l) rev(--l);
139       while(r < q[i].r) rev(++r);
140       while(r > q[i].r) rev(r--);
141       if(q[i].extra) rev(st[q[i].extra]);
142       ans[q[i].i] = curans;
143       if(q[i].extra) rev(st[q[i].extra]);
144     }
145     for(int i = 1; i <= m; i++)
146       printf("%d\n", ans[i]);
147   }
148
149   int main()
```

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

5.10 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;
}
```

5.11 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;
}
```

5.12 点分治（zky）

```
int siz[maxn],f[maxn],dep[maxn],cant[maxn],root,All,d[maxn];
void makert(int u,int fa){
    siz[u]=1;f[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
```

```
        if(e.v!=fa&&!cant[e.v]){
            dep[e.v]=dep[u]+1;
            makert(e.v,u);
            siz[u]+=siz[e.v];
            f[u]=max(f[u],siz[e.v]);
        }
    }f[u]=max(f[u],All-f[u]);
    if(f[root]>f[u])root=u;
}
void dfs(int u,int fa){
  //Gain data
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(e.v==fa||cant[e.v])continue;
        d[e.v]=d[u]+e.w;
        dfs(e.v,u);
    }
}
void calc(int u){
  d[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        d[e.v]=e.w;
        dfs(e.v,u);

    }
}
void solve(int u){
    calc(u);cant[u]=1;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        All=siz[e.v];
        f[root=0]=n+1;
        makert(e.v,0);
        solve(root);
    }
}
All=n
f[root=0]=n+1;
makert(1,1);
solve(root);
```

6 图论

6.1 点双连通分量

```
// 坚固的点双连通分量
int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N];
vector<int> a[N];
pair<int, int> stack[M];
void tarjan(int x, int p) {
  dfn[x] = low[x] = ++ind2;
  for (int i = 0; i < a[x].size(); ++i)
    if (dfn[x] > dfn[a[x][i]] && a[x][i] != p){
      stack[++size] = make_pair(x, a[x][i]);
      if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
        if (!dfn[a[x][i]]){
          tarjan(a[x][i], x);
          low[x] = min(low[x], low[a[x][i]]);
          if (low[a[x][i]] >= dfn[x]){
            tot1 = tot2 = 0;
            ++flag;
            for (; ; ){
              if (block[stack[size].first] != flag) {
                ++tot1;
                block[stack[size].first] = flag;
              }
              if (block[stack[size].second] != flag) {
                ++tot1;
                block[stack[size].second] = flag;
              }
              if (stack[size].first == x && stack[size].second == a[x][i])
                break;
              ++tot2;
              --size;
            }
            for (; stack[size].first == x && stack[size].second == a[x][i]; --size)
              ++tot2;
            if (tot2 < tot1)
              ans1 += tot2;
            if (tot2 > tot1)
              ans2 += tot2;
          }
        }
        else
          low[x] = min(low[x], dfn[a[x][i]]);
    }
}
int main(){
  for (; ; ){
    scanf("%d%d", &n, &m);
    if (n == 0 && m == 0) return 0;
```

```
    for (int i = 1; i <= n; ++i) {
      a[i].clear();
      dfn[i] = 0;
    }
    for (int i = 1; i <= m; ++i){
      scanf("%d%d",&x, &y);
      ++x, ++y;
      a[x].push_back(y);
      a[y].push_back(x);
    }
    for (int i = 1; i <= n; ++i)
      sort(a[i].begin(), a[i].end());
    ans1 = ans2 = ind2 = 0;
    for (int i = 1; i <= n; ++i)
      if (!dfn[i]) {
        size = 0;
        tarjan(i, 0);
      }
    printf("%d %d\n", ans1, ans2);
  }
  return 0;
}
// 朴素的点双连通分量
void tarjan(int x){
  dfn[x] = low[x] = ++ind2;
  v[x] = 1;
  for (int i = nt[x]; pt[i]; i = nt[i])
    if (!dfn[pt[i]]){
      tarjan(pt[i]);
      low[x] = min(low[x], low[pt[i]]);
      if (dfn[x] <= low[pt[i]])
        ++v[x];
    }
    else
      low[x] = min(low[x], dfn[pt[i]]);
}
int main(){
  for (; ; ){
    scanf("%d%d", &n, &m);
    if (n == 0 && m == 0)
      return 0;
    for (int i = 1; i <= ind; ++i)
      nt[i] = pt[i] = 0;
    ind = n;
    for (int i = 1; i <= ind; ++i)
      last[i] = i;
    for (int i = 1; i <= m; ++i){
      scanf("%d%d", &x, &y);
      ++x, ++y;
```

```
96          edge(x, y), edge(y, x);
97        }
98        memset(dfn, 0, sizeof(dfn));
99        memset(v, 0, sizeof(v));
100       ans = num = ind2 = 0;
101       for (int i = 1; i <= n; ++i)
102         if (!dfn[i]){
103           root = i;
104           size = 0;
105           ++num;
106           tarjan(i);
107           --v[root];
108         }
109       for (int i = 1; i <= n; ++i)
110         if (v[i] + num - 1 > ans)
111           ans = v[i] + num - 1;
112       printf("%d\n",ans);
113     }
114     return 0;
115 }
```

6.2 点双连通分量（lyx）

```
1  #define SZ(x) ((int)x.size())
2
3  const int N = 400005; // N 开 2 倍点数，因为新树会加入最多 n 个新点
4  const int M = 200005;
5
6  vector<int> g[N];
7
8  int bccno[N], bcc_cnt;
9  vector<int> bcc[N];
10 bool iscut[N];
11
12 struct Edge {
13   int u, v;
14 } stk[M << 2];
15 int top; // 注意栈大小为边数 4 倍
16 int dfn[N], low[N], dfs_clock;
17
18 void dfs(int x, int fa)
19 {
20   low[x] = dfn[x] = ++dfs_clock;
21   int child = 0;
22   for(int i = 0; i < SZ(g[x]); i++) {
23     int y = g[x][i];
24     if(!dfn[y]) {
25       child++;
26       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);
    }
  }
}
```

6.3 Hungary 求最大匹配

```cpp
int n, m, stamp;
int match[N], visit[N];

bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
        if (visit[y] != stamp) {
            visit[y] = stamp;
            if (match[y] == -1 || dfs(match[y])) {
                match[y] = x;
                return true;
            }
        }
    }
    return false;
}

int solve() {
    std::fill(match, match + m, -1);
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        stamp++;
        answer += dfs(i);
    }
    return answer;
}
```

6.4 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;
        }
    }
}
```

6.5 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;
}
```

6.6 稀疏图最大流

注意事项：适用于比较稀疏的一般图。

```
int Maxflow_Isap(int s,int t,int n) {
  std::fill(pre + 1, pre + n + 1, 0);
  std::fill(d + 1, d + n + 1, 0);
  std::fill(gap + 1, gap + n + 1, 0);
  for (int i = 1; i <= n; i++) cur[i] = h[i];
  gap[0] = n;
  int u = pre[s] = s, v, maxflow = 0;
  while (d[s] < n) {
    v = n + 1;
    for (int i = cur[u]; i; i = e[i].next)
    if (e[i].flow && d[u] == d[e[i].node] + 1) {
      v = e[i].node; cur[u]=i; break;
```

```
    }
    if (v <= n) {
      pre[v] = u; u = v;
      if (v == t) {
        int dflow = INF, p = t; u = s;
        while (p != s) {
          p = pre[p];
          dflow = std::min(dflow, e[cur[p]].flow);
        }
        maxflow += dflow; p = t;
        while (p != s) {
          p = pre[p];
          e[cur[p]].flow -= dflow;
          e[e[cur[p]].opp].flow += dflow;
        }
      }
    }
    else{
      int mindist = n + 1;
      for (int i = h[u]; i; i = e[i].next)
        if (e[i].flow && mindist > d[e[i].node]) {
          mindist = d[e[i].node]; cur[u] = i;
        }
      if (!--gap[d[u]]) return maxflow;
      gap[d[u] = mindist + 1]++; u = pre[u];
    }
  }
  return maxflow;
}
```

6.7 稠密图最大流

注意事项：适用于二分图以及一些比较稠密的、增广路径比较短的图。

```
int bfs(){
  for (int i = 1;i <= t;i ++) d[i] = -1;
  int l,r;
  q[l = r = 0] = s, d[s] = 0;
  for (;l <= r;l ++)
    for (int k = h[q[l]]; k > -1; k = nxt[k])
      if (d[p[k]] == -1 && c[k] > 0) d[p[k]] = d[q[l]] + 1, q[++ r] = p[k];
  return d[t] > -1 ? 1 : 0;
}
int dfs(int u,int ext){
  if (u == t) return ext;
  int k = w[u],ret = 0;
  for (; k > -1; k = nxt[k], w[u] = k){      //w 数组为当前弧
    if (ext == 0) break;
    if (d[p[k]] == d[u] + 1 && c[k] > 0){
      int flow = dfs(p[k], min(c[k], ext));
```

```
      if (flow > 0){
        c[k] -= flow, c[k ^ 1] += flow;
        ret += flow, ext -= flow;     //ret 累计增广量，ext 记录还可增广的量
      }
    }
  }
  if (k == -1) d[u] = -1;
  return ret;
}
void dinic() {
  while (bfs()) {
    for (int i = 1; i <= t;i ++) w[i] = h[i];
    dfs(s, inf);
  }
}
```

6.8 稀疏图费用流

```
struct EdgeList {
    int size;
    int last[N];
    int succ[M], other[M], flow[M], cost[M];
    void clear(int n) {
        size = 0;
        std::fill(last, last + n, -1);
    }
    void add(int x, int y, int c, int w) {
        succ[size] = last[x];
        last[x] = size;
        other[size] = y;
        flow[size] = c;
        cost[size++] = w;
    }
} e;

int n, source, target;
int prev[N];

void add(int x, int y, int c, int w) {
    e.add(x, y, c, w);
    e.add(y, x, 0, -w);
}

bool augment() {
    static int dist[N], occur[N];
    std::vector<int> queue;
    std::fill(dist, dist + n, INT_MAX);
    std::fill(occur, occur + n, 0);
    dist[source] = 0;
```

```
    occur[source] = true;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = e.last[x]; ~i; i = e.succ[i]) {
            int y = e.other[i];
            if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
                dist[y] = dist[x] + e.cost[i];
                prev[y] = i;
                if (!occur[y]) {
                    occur[y] = true;
                    queue.push_back(y);
                }
            }
        }
        occur[x] = false;
    }
    return dist[target] < INT_MAX;
}

std::pair<int, int> solve() {
    std::pair<int, int> answer = std::make_pair(0, 0);
    while (augment()) {
        int number = INT_MAX;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            number = std::min(number, e.flow[prev[i]]);
        }
        answer.first += number;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            e.flow[prev[i]] -= number;
            e.flow[prev[i] ^ 1] += number;
            answer.second += number * e.cost[prev[i]];
        }
    }
    return answer;
}
```

6.9 稠密图费用流

```
int aug(int no,int res) {
    if(no == t) return cost += pi1 * res,res;
    v[no] = true;
    int flow = 0;
    for(int i = h[no]; ~ i ;i = nxt[i])
    if(cap[i] && !expense[i] && !v[p[i]]) {
      int d = aug(p[i],min(res,cap[i]));
      cap[i] -= d,cap[i ^ 1] += d,flow += d,res -= d;
      if( !res ) return flow;
    }
```

```
    return flow;
}
bool modlabel() {
    int d = maxint;
    for(int i = 1;i <= t;++ i)
    if(v[i]) {
      for(int j = h[i]; ~ j ;j = nxt[j])
        if(cap[j] && !v[p[j]] && expense[j] < d) d = expense[j];
    }
    if(d == maxint)return false;
    for(int i = 1;i <= t;++ i)
    if(v[i]) {
      for(int j = h[i];~ j;j = nxt[j])
        expense[j] -= d,expense[j ^ 1] += d;
    }
    pi1 += d;
    return true;
}
void minimum_cost_flow_zkw() {
  cost = 0;
  do{
    do{
      memset(v,false,sizeof v);
    }while (aug(s,maxint));
  }while (modlabel());
}
```

6.10 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;
}
```

6.11 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++;
    }
}
```

6.12 Dominator Tree

```
#define foreach(A, x, it) for (int it = A.h[x]; it; it = A.e[it].next)

const int MAXN = 200001;
const int MAXM = 400001;

template<class T, int MAXN, int MAXM>
struct AdjList{
  struct Edge{
    T data;
    int next;
  }e[MAXM];
  int h[MAXN], t;
  void add(int x, const T &data) {
    t++;
    e[t].data = data;
    e[t].next = h[x];
    h[x] = t;
  }
  void drop(int x) {
    h[x] = e[h[x]].next;
  }
  T & operator [](const int &index) {
    return e[index].data;
  }
  void clear(int n) {
    std::fill(h + 1, h + n + 1, t = 0);
```

```
  }
};
// fa 是并查集的父亲，f 是树的父亲，sdom 是半必经点，idom 是必经点，smin 是带权并查集的权值
// pred 是前驱链表，succ 是后继链表
int dfn[MAXN], sdom[MAXN], idom[MAXN], id[MAXN], f[MAXN], fa[MAXN], smin[MAXN];
AdjList<int, MAXN, MAXM> pred, succ;
long long answer[MAXN];

void predfs(int x) {
  id[dfn[x] = ++stamp] = x;
  foreach(succ, x, i) {
    int y = succ[i];
    if (!dfn[y]) {
      f[y] = x;
      predfs(y);
    }
  }
}

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 tarjan(int s) {
  static AdjList<int, MAXN, MAXN> tmp;
  stamp = tmp.t = 0;
  predfs(s);
  for (int i = 1; i <= stamp; i++) {
    fa[id[i]] = smin[id[i]] = id[i];
    tmp.h[id[i]] = idom[id[i]] = 0;
  }
  for (int o = stamp; o >= 1; o--) {
    int x = id[o];
    if (o != 1) {
      sdom[x] = f[x];
      foreach(pred, x, i) {
        int p = pred[i];
        if (!dfn[p]) continue;
        if (dfn[p] > dfn[x]) {
          getfa(p);
          p = sdom[smin[p]];
        }
        if (dfn[sdom[x]] > dfn[p]) {
          sdom[x] = p;
```

```
        }
      }
      tmp.add(sdom[x], x);
    }
    while (tmp.h[x] != 0) {
      int y = tmp[tmp.h[x]];
      tmp.drop(x);
      getfa(y);
      if (x != sdom[smin[y]]) {
        idom[y] = smin[y];
      } else {
        idom[y] = x;
      }
    }
    foreach(succ, x, i) {
      if (f[succ[i]] == x) {
        fa[succ[i]] = x;
      }
    }
  }
  idom[s] = s;
  for (int i = 2; i <= stamp; i++) {
    int x = id[i];
    if (idom[x] != sdom[x]) {
      idom[x] = idom[idom[x]];
    }
  }
}
```

6.13 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;
}
```

6.14 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;
}
```

6.15 弦图判定

```
int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L[1001], R[1001],
v[1001], idx[1001], pos[1001];
bool b[1001][1001];

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

bool cmp(const int &x, const int &y){
    return(idx[x] < idx[y]);
}

int main(){
    for (;;)
    {
        n = read(); m = read();
        if (!n && !m) return 0;
        memset(first, 0, sizeof(first)); l = 0;
        memset(b, false, sizeof(b));
        for (int i = 1; i <= m; i++)
        {
            int x = read(), y = read();
            if (x != y && !b[x][y])
            {
```

```
            b[x][y] = true; b[y][x] = true;
            makelist(x, y); makelist(y, x);
        }
    }
    memset(f, 0, sizeof(f));
    memset(L, 0, sizeof(L));
    memset(R, 255, sizeof(R));
    L[0] = 1; R[0] = n;
    for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
    memset(idx, 0, sizeof(idx));
    memset(v, 0, sizeof(v));
    for (int i = n; i; --i)
    {
        int now = c[i];
        R[f[now]]--;
        if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
        idx[now] = i; v[i] = now;
        for (int x = first[now]; x; x = next[x])
            if (!idx[where[x]])
            {
                swap(c[pos[where[x]]], c[R[f[where[x]]]]);
                pos[c[pos[where[x]]]] = pos[where[x]];
                pos[where[x]] = R[f[where[x]]];
                L[f[where[x]] + 1] = R[f[where[x]]]--;
                if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                if (R[f[where[x]] + 1] == -1)
                    R[f[where[x]] + 1] = L[f[where[x]] + 1];
                ++f[where[x]];
            }
    }
    bool ok = true;
    //v 是完美消除序列.
    for (int i = 1; i <= n && ok; i++)
    {
        int cnt = 0;
        for (int x = first[v[i]]; x; x = next[x])
            if (idx[where[x]] > i) c[++cnt] = where[x];
        sort(c + 1, c + cnt + 1, cmp);
        bool can = true;
        for (int j = 2; j <= cnt; j++)
            if (!b[c[1]][c[j]])
            {
                ok = false;
                break;
            }
    }
    if (ok) printf("Perfect\n");
    else printf("Imperfect\n");
    printf("\n");
```

```
    }
}
```

6.16 弦图求最大团

```
int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c[100001],
↪  f[100001],
pos[100001], idx[100001], v[100001], ans;

inline void makelist(int x, int y){
    where[++l] = y;
    next[l] = first[x];
    first[x] = l;
}

int read(){
    char ch;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
    int cnt = 0;
    for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
    return(cnt);
}

int main(){
    //freopen("1006.in", "r", stdin);
    //freopen("1006.out", "w", stdout);
    memset(first, 0, sizeof(first)); l = 0;
    n = read(); m = read();
    for (int i = 1; i <= m; i++)
    {
        int x, y;
        x = read(); y = read();
        makelist(x, y); makelist(y, x);
    }
    memset(L, 0, sizeof(L));
    memset(R, 255, sizeof(R));
    memset(f, 0, sizeof(f));
    memset(idx, 0, sizeof(idx));
    for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
    L[0] = 1; R[0] = n; ans = 0;
    for (int i = n; i; --i)
    {
        int now = c[i], cnt = 1;
        idx[now] = i; v[i] = now;
        if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
        for (int x = first[now]; x; x = next[x])
            if (!idx[where[x]])
            {
                swap(c[pos[where[x]]], c[R[f[where[x]]]]);
```

```
                pos[c[pos[where[x]]]] = pos[where[x]];
                pos[where[x]] = R[f[where[x]]];
                L[f[where[x]] + 1] = R[f[where[x]]]--;
                if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
                if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
                ++f[where[x]];
            }
            else ++cnt;
        ans = max(ans, cnt);
    }
    printf("%d\n", ans);
}
```

6.17 最大团搜索

```
// mc[i] 代表只用 i-n 号点的答案
// g 代表连通性
void dfs(int size) {
  int i, j, k;
  if (len[size] == 0) {
    if (size > ans) {
      ans = size;
      found = true;
    }
    return;
  }
  for (k = 0; k < len[size] && !found; k ++) {
    if (size + len[size] - k <= ans) break;
    i = list[size][k];
    if (size + mc[i] <= ans) break;
    for (j = k + 1, len[size + 1] = 0; j < len[size]; j ++)
      if (g[i][list[size][j]]) list[size + 1][len[size + 1] ++] = list[size][j];
    dfs(size + 1);
    if (found) {
      prin[size + 1] = i;
    }
  }
}
void work() {
  int i, j;
  mc[n] = ans = 1;
  ansi = 1;
  for (i = n - 1; i; i --) {
    found = false;
    len[1] = 0;
    for (j = i + 1; j <= n; j ++) if (g[i][j]) list[1][len[1]++] = j;
    dfs(1);
    mc[i] = ans;
    if (found) prin[1] = i;
```

```
  }
}
void print() {
  printf("%d\n", ans);
  for (int i = 1; i < ans; i ++) printf("%d ", prin[i]);
  printf("%d\n", prin[ans]);
}
```

6.18 极大团计数

```
bool g[N][N];
int ne[N], ce[N], list[N][N], ans;
void dfs(int size) {
  if (ans > 1000) return;
  int i, j, k, t, cnt, best = 0;
  bool bb;
  if (ne[size] == ce[size]) {
    if (ce[size] == 0) ++ans;
    return;
  }
  for (t = 0, i = 1; i <= ne[size]; ++i) {
    for (cnt = 0, j = ne[size] + 1; j <= ce[size]; ++j)
      if (!g[list[size][i]][list[size][j]]) ++cnt;
    if (t == 0 || cnt < best) t = i, best = cnt;
  }
  if (t && best <= 0) return;
  for (k = ne[size] + 1; k <= ce[size]; ++k) {
    if (t > 0) {
      for (i = k; i <= ce[size]; ++i)
        if (!g[list[size][t]][list[size][i]]) break;
      swap(list[size][k], list[size][i]);
    }
    i = list[size][k];
    ne[size + 1] = ce[size + 1] = 0;
    for (j = 1; j < k; ++j)
      if (g[i][list[size][j]])
        list[size + 1][++ne[size + 1]] = list[size][j];
    for (ce[size + 1] = ne[size + 1], j = k + 1; j <= ce[size]; ++j)
      if (g[i][list[size][j]]) list[size + 1][++ce[size + 1]] = list[size][j];
    dfs(size + 1);
    ++ne[size];
    --best;
    for (j = k + 1, cnt = 0; j <= ce[size]; ++j)
      if (!g[i][list[size][j]]) ++cnt;
    if (t == 0 || cnt < best) t = k, best = cnt;
    if (t && best <= 0) break;
  }
}
int main(){
```

```
  int n, m;
  while (scanf("%d%d", &n, &m) == 2) {
    for (int i = 1; i <= n; ++i)
      for (int j = 1; j <= n; ++j)
        g[i][j] = false;
    while (m--) {
      int x, y;
      scanf("%d%d", &x, &y);
      g[x][y] = g[y][x] = true;
    }
    ne[0] = 0;
    ce[0] = 0;
    for (int i = 1; i <= n; ++i)
      list[0][++ce[0]] = i;
    ans = 0;
    dfs(0);
    if (ans > 1000) puts("Too many maximal sets of friends.");
    else printf("%d\n", ans);
  }
  return 0;
}
```

6.19 最小树形图

```
int n, m, used[N], pass[N], eg[N], more, queue[N];
double g[N][N];

void combine(int id, double &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];
    }
  }
}

double mdst(int root) {
  int i, j, k;
  double 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;
}
```

6.20 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;
}
```

6.21 度限制生成树

```
int n, m, S, K, ans , cnt , Best[N], fa[N], FE[N];
int f[N], p[M], t[M], c[M], o, Cost[N];
bool u[M], d[M];
pair<int, int> MinCost[N];
struct Edge {
  int a, b, c;
  bool operator < (const Edge & E) const { return c < E.c; }
}E[M];
vector<int> SE;
inline int F(int x) {
  return fa[x] == x ? x : fa[x] = F(fa[x]);
}
inline void AddEdge(int a, int b, int C) {
  p[++o] = b; c[o] = C;
  t[o] = f[a]; f[a] = o;
}
void dfs(int i, int father) {
  fa[i] = father;
  if (father == S) Best[i] = -1;
  else {
    Best[i] = i;
    if (~Best[father] && Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
  }
  for (int j = f[i]; j; j = t[j])
  if (!d[j] && p[j] != father) {
    Cost[p[j]] = c[j];
    FE[p[j]] = j;
    dfs(p[j], i);
  }
}
inline bool Kruskal() {
  cnt = n - 1,ans = 0; o = 1;
  for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
  sort(E + 1, E + m + 1);
  for (int i = 1; i <= m; i++) {
    if (E[i].b == S) swap(E[i].a, E[i].b);
    if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
      fa[F(E[i].a)] = F(E[i].b);
      ans += E[i].c;
      cnt --;
      u[i] = true;
      AddEdge(E[i].a, E[i].b, E[i].c);
      AddEdge(E[i].b, E[i].a, E[i].c);
    }
  }
  for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
  for (int i = 1; i <= m; i++)
  if (E[i].a == S) {
```

```
      SE.push_back(i);
      MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
    }
    int dif = 0;
    for (int i = 1; i <= n; i++)
    if (i != S && fa[i] == i) {
      if (MinCost[i].second == INF) return false;
      if (++ dif > K) return false;
      dfs(E[MinCost[i].second].b, S);
      u[MinCost[i].second] = true;
      ans += MinCost[i].first;
    }
    return true;
}
bool Solve() {
    memset(d,false,sizeof d);
    memset(u,false,sizeof u);
    if (!Kruskal()) return false;
    for (int i = cnt + 1; i <= K && i <= n; i++) {
      int MinD = INF, MinID = -1;
      for (int j = (int) SE.size() - 1; j >= 0; j--)
      if (u[SE[j]])
        SE.erase(SE.begin() + j);
      for (int j = 0; j < (int) SE.size(); j++) {
        int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
        if (tmp < MinD) {
          MinD = tmp;
          MinID= SE[j];
        }
      }
      if (MinID == -1) return true;
      if (MinD >= 0) break;
      ans += MinD;
      u[MinID] = true;
      d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
      dfs(E[MinID].b, S);
    }
    return true;
}
int main(){
    Solve();
    return 0;
}
```

7 字符串

7.1 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);
}
```

7.2 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;
    }
  }
}
```

7.3 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;
}
```

7.4 后缀自动机

7.4.1 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。

```cpp
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');
}
```

7.4.2 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)//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];
  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);
}
```

7.5 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];
    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

```
//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;


```

```
66
67    p = 0;
68    int i = 0, j = 0;
69    for(;i < ta && j < tbc; p++)
70      sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
71    for(; i < ta; p++)
72      sa[p] = wa[i++];
73    for(; j < tbc; p++)
74      sa[p] = wb[j++];
75  }
76
77  void getheight(char s[], int sa[], int n)
78  {
79    for(int i = 1; i <= n; i++)
80      rank[sa[i]] = i;
81    for(int p = 0, i = 1; i <= n; i++, p = (p) ? p - 1 : p)
82      if(rank[i] > 1)
83      {
84        int j = sa[rank[i] - 1];
85        while(s[i + p] == s[j + p])
86          p++;
87        height[rank[i]] = p;
88      }
89  }
90
91
92  int main()
93  {
94    scanf("%s", s + 1);
95    int n = strlen(s + 1);
96    for(int i = 0; i <= n; i++)// <= n !!!
97      r[i] = s[i + 1];
98    dc3(r, sa, n + 1, 255);//now the value of sa is from 0 to n - 1;
99    for(int i = n; i ; i--)//after this operation, the value of sa is from 1 to n
100     sa[i]++;
101   getheight(s, sa, n);
102   for(int i = 1;i <= n; i++)
103     printf("%d ", sa[i]);
104   puts("");
105   for(int i = 2; i <= n; i++)
106     printf("%d ", height[i]);
107   puts("");
108 }
```

7.6 回文自动机

注意事项：请注意字符集大小。

```
1  struct Palindromic_Tree{
2    int nTree, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
```

```
  int allocate(int len) {
    l[nTree] = len;
    r[nTree] = 0;
    fail[nTree] = 0;
    memset(c[nTree], 0, sizeof(c[nTree]));
    return nTree++;
  }
  void init() {
    nTree = nStr = 0;
    int newEven = allocate(0);
    int newOdd = allocate(-1);
    last = newEven;
    fail[newEven] = newOdd;
    fail[newOdd] = newEven;
    s[0] = -1;
  }
  void add(int x) {
    s[++nStr] = x;
    int nownode = last;
    while (s[nStr - l[nownode] - 1] != s[nStr]) nownode = fail[nownode];
    if (!c[nownode][x]) {
      int newnode = allocate(l[nownode] + 2), &newfail = fail[newnode];
      newfail = fail[nownode];
      while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
      newfail = c[newfail][x];
      c[nownode][x] = newnode;
    }
    last = c[nownode][x];
    r[last]++;
  }
  void count() {
    for (int i = nTree - 1; i >= 0; i--) {
      r[fail[i]] += r[i];
    }
  }
}
```

7.7 Manacher

注意事项：1-based 算法，请注意下标。

```
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;
  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;
}
```

7.8 循环串的最小表示

注意事项： 0-Based 算法，请注意下标。

```
#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));
  }
}
```

7.9 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;
    }
  }
};
```

8 计算几何

8.1 三维几何

```
/* 大拇指指向 x 轴正方向时，4 指弯曲由 y 轴正方向指向 z 轴正方向
   大拇指沿着原点到点 (x, y, z) 的向量，4 指弯曲方向旋转 w 度 */
/* (x, y, z) * A = (x_new, y_new, z_new)，行向量右乘转移矩阵 */
void calc(D x, D y, D z, D w) {
  w = w * pi / 180;
  memset(a, 0, sizeof(a));
  s1 = x * x + y * y + z * z;
```

```
  a[0][0] = ((y*y+z*z)*cos(w)+x*x)/s1; a[0][1] = x*y*(1-cos(w))/s1+z*sin(w)/sqrt(s1); a[0][2] = x*z*(1-cos(w))/s1-y*sin(w)/sqrt(s1);
  a[1][0] = x*y*(1-cos(w))/s1-z*sin(w)/sqrt(s1); a[1][1] = ((x*x+z*z)*cos(w)+y*y)/s1; a[1][2] = y*z*(1-cos(w))/s1+x*sin(w)/sqrt(s1);
  a[2][0] = x*z*(1-cos(w))/s1+y*sin(w)/sqrt(s1); a[2][1] = y*z*(1-cos(w))/s1-x*sin(w)/sqrt(s1); a[2][2] = ((x*x+y*y)*cos(w)+z*z)/s1;
}
// 求平面和直线的交点
Point3D intersection(const Point3D &a, const Point3D &b, const Point3D &c, const Point3D &l0, const Point3D &l1) {
  Point3D p = pVec(a, b, c); // 平面法向量
  double t = (p.x * (a.x - l0.x) + p.y * (a.y - l0.y) + p.z * (a.z - l0.z)) / (p.x * (l1.x - l0.x) + p.y * (l1.y - l0.y) + p.z * (l1.z - l0.z));
  return l0 + (l1 - l0) * t;
}
```

8.2 三维凸包

```
int mark[N][N], cnt;
D mix(const Point & a, const Point & b, const Point & c) { return a.dot(b.cross(c)); }
double volume(int a, int b, int c, int d) { return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]); }
typedef array<int, 3> Face; vector<Face> face;
inline void insert(int a, int b, int c) { face.push_back({a, b, c}); }
void add(int v) {
  vector<Face> tmp; int a, b, c; cnt++;
  for(auto f : face)
    if(sign(volume(v, f[0], f[1], f[2])) < 0)
      for(int i : f) for(int j : f) mark[i][j] = cnt;
    else tmp.push_back(f);
  face = tmp;
  for(int i(0); i < (int)tmp.size(); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if(mark[a][b] == cnt) insert(b, a, v);
    if(mark[b][c] == cnt) insert(c, b, v);
    if(mark[c][a] == cnt) insert(a, c, v);
  }
}
int Find(int n) {
  for(int i(2); i < n; i++) {
    Point ndir = (info[0] - info[i]).cross(info[1] - info[i]);
    if(ndir == Point(0, 0, 0)) continue; swap(info[i], info[2]);
    for(int j = i + 1; j < n; j++) if(sign(volume(0, 1, 2, j)) != 0) {
      swap(info[j], info[3]); insert(0, 1, 2), insert(0, 2, 1); return 1;
    }
  }
int main() {
  int n; scanf("%d", &n);
  for(int i(0); i < n; i++) info[i].scan();
```

```
  random_shuffle(info, info + n);
  Find(n);
  for(int i = 3; i < n; i++) add(i);
}
```

8.3 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^{±} = r1r2r3 / (r1r2+r1r3+r2r3 ±2√(r1r2r3(r1+r2+r3)))
```

8.4 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);
}
```

8.5 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()); return; }
    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;
}
```

8.6 圆并

```cpp
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;
} } } }
```

8.7 Delaunay 三角剖分

```
/*
Delaunay Triangulation 随机增量算法 :
节点数至少为点数的 6 倍, 空间消耗较大注意计算内存使用
建图的过程在 build 中, 注意初始化内存池和初始三角形的坐标范围 (Triangulation::LOTS)
Triangulation::find 返回包含某点的三角形
Triangulation::add_point 将某点加入三角剖分
某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
如果要找到三角形 u 的邻域, 则枚举它的所有 u.edge[i].tri, 该条边的两个点为 u.p[(i+1)%3], u.p[(i+2)%3]
*/
const int N = 100000 + 5, MAX_TRIS = N * 6;
const double EPSILON = 1e-6, PI = acos(-1.0);
struct Point {
  double x,y; Point():x(0),y(0){}
    Point(double x, double y):x(x),y(y){}
  bool operator ==(Point const& that)const {return x==that.x&&y==that.y;}
};
inline double sqr(double x) { return x*x; }
double dist_sqr(Point const& a, Point const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const& p4) {
  double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
  double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
  double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
  double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
  double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
  double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
  return det > EPSILON;
}
double side(Point const& a, Point const& b, Point const& p) { return (b.x-a.x)*(p.y-a.y) -
 ↪  (b.y-a.y)*(p.x-a.x);}
typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
struct Edge {
  TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
  Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
};
struct Triangle {
  Point p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
  Triangle(Point const& p0, Point const& p1, Point const& p2) {
    p[0] = p0; p[1] = p1; p[2] = p2;
        children[0] = children[1] = children[2] = 0;
  }
  bool has_children() const { return children[0] != 0; }
  int num_children() const {
    return children[0] == 0 ? 0
      : children[1] == 0 ? 1
      : children[2] == 0 ? 2 : 3;
  }
  bool contains(Point const& q) const {
    double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
```

```
48      return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
49    }
50  } triange_pool[MAX_TRIS], *tot_triangles;
51  void set_edge(Edge a, Edge b) {
52    if (a.tri) a.tri->edge[a.side] = b;
53    if (b.tri) b.tri->edge[b.side] = a;
54  }
55  class Triangulation {
56    public:
57      Triangulation() {
58        const double LOTS = 1e6;
59        the_root = new(tot_triangles++)
    ↪   Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
60      }
61      TriangleRef find(Point p) const { return find(the_root,p); }
62      void add_point(Point const& p) { add_point(find(the_root,p),p); }
63    private:
64      TriangleRef the_root;
65      static TriangleRef find(TriangleRef root, Point const& p) {
66        for( ; ; ) {
67          if (!root->has_children()) return root;
68          else for (int i = 0; i < 3 && root->children[i] ; ++i)
69              if (root->children[i]->contains(p))
70                {root = root->children[i]; break;}
71        }
72      }
73      void add_point(TriangleRef root, Point const& p) {
74        TriangleRef tab,tbc,tca;
75        tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
76        tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
77        tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
78        set_edge(Edge(tab,0),Edge(tbc,1)); set_edge(Edge(tbc,0),Edge(tca,1));
79        set_edge(Edge(tca,0),Edge(tab,1)); set_edge(Edge(tab,2),root->edge[2]);
80        set_edge(Edge(tbc,2),root->edge[0]); set_edge(Edge(tca,2),root->edge[1]);
81        root->children[0]=tab; root->children[1]=tbc; root->children[2]=tca;
82        flip(tab,2); flip(tbc,2); flip(tca,2);
83      }
84      void flip(TriangleRef tri, SideRef pi) {
85        TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
86        if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
87        TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
88        TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
89        set_edge(Edge(trk,0), Edge(trl,0));
90        set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
91        set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
92        tri->children[0]=trk; tri->children[1]=trl; tri->children[2]=0;
93        trj->children[0]=trk; trj->children[1]=trl; trj->children[2]=0;
94        flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
95      }
```

```
};
int n; Point ps[N];
void build(){
  tot_triangles = triange_pool; cin >> n;
  for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
  random_shuffle(ps, ps + n); Triangulation tri;
  for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
}
```

8.8 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, x)
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 必须是 (-y, x)
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;
}
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;
}
```

8.9 凸包

```
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;
  }
}
```

8.10 整数半平面交

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

8.11 三角形

```
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 为费马点到 abc 三点距离和
  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
}
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);
}
```

8.12 经纬度求球面最短距离

```
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));
}
```

8.13 长方体表面两点最短距离

```
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;
}
```

8.14 点到凸包切线

```
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标相同
// 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;
}
```

8.15 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
}
```

8.16 平面最近点对

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

using namespace std;
```

```cpp

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;
}
```

9 其他

9.1 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));
        }
      }
  }
}
```

9.2 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);
```

9.3 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 ;
}
```

9.4 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);
}
```

9.5 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;
}
```

9.6 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;
}
```

9.7 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;
    }
}
```

9.8 环状最长公共子串

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

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;
}
```

9.9 LLMOD

```
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;
}
```

9.10 STL 内存清空

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

9.11 开栈

```
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;
}
```

9.12 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

10 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>
```

11 常用结论

11.1 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。

11.2 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）

11.3 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

11.4 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：

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

12 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

13 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；

14 Java

14.1 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
```

14.2 样例代码

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

public static class edge implements Comparable<edge>{
  public int u,v,w;
  public int compareTo(edge e){
    return w-e.w;
  }
}
public static class cmp implements Comparator<edge>{
  public int compare(edge a,edge b){
    if(a.w<b.w)return 1;
    if(a.w>b.w)return -1;
    return 0;
  }
}

public class Main{
  public static long max(long a,long b){
    if(a>b)return a;
    return b;
  }
  public static long Calc(long A, int x){
    long Ret = (A / (1L << x)) * (1L << (x - 1));

    Ret += max(A % (1L << x) - (1L << (x - 1)) + 1, 0L);
```

```
28
29        return Ret;
30      }
31      private static class InputReader {
32
33          public BufferedReader rea;
34          public StringTokenizer tok;
35
36          public InputReader(InputStream stream) {
37              rea = new BufferedReader(new InputStreamReader(stream), 32768);
38              tok = null;
39          }
40
41          public String next() {
42              while (tok == null || !tok.hasMoreTokens()) {
43                  try {
44                      tok = new StringTokenizer(rea.readLine());
45                  } catch (IOException e) {
46                      throw new RuntimeException(e);
47                  }
48              }
49              return tok.nextToken();
50          }
51
52          public int nextInt() {
53              return Integer.parseInt(next());
54          }
55
56          public long nextLong() {
57              return Long.parseLong(next());
58          }
59      }
60
61      public static void main(String arg[]){
62        InputReader cin = new InputReader(System.in);
63        //Scanner cin = new Scanner(System.in);
64        int N = 70;
65
66        long k[] = new long[N];
67        int n;
68
69        while(true){
70          n=cin.nextInt();
71          if (n == 0) break;
72
73          for (int i = 1; i <= n; i ++){
74            k[i]=cin.nextLong();
75            //System.out.println(k[i]);
76          }
```

```java

        long Len;
        BigInteger Sum;
        long A, B;
        long AnsA = -1, AnsB = -1;
        int Ans = 0;

        for (int i = -1; i <= 1; i++){
          Len = k[1] * 2 + i;

          if(Len<=0)continue;

          Sum = BigInteger.ZERO;
          for (int j = 1; j <= n; j++){
            Sum = Sum.add(BigInteger.valueOf(1L << (j - 1)) .multiply(BigInteger.valueOf(k[j])));
          }
          //System.out.println(Sum);

          long x = Len;

          if ((Sum.multiply(BigInteger.valueOf(2))) .mod
    ↪  (BigInteger.valueOf(Len)).compareTo(BigInteger.ZERO) > 0) continue;

          long y = Sum.multiply(BigInteger.valueOf(2)).divide(BigInteger.valueOf(Len)).longValue();
          if ((y - x + 1) % 2 > 0) continue;
          A = (y - x + 1) / 2;
          if ((x + y - 1) % 2 > 0) continue;
          B = (x + y - 1) / 2;

          if (A < 1 || A > (long)1e18) continue;
          if (B < 1 || B > (long)1e18) continue;

          int flag = 1;

          long Cnt;
          long Cnt_B;
          long Cnt_A;

          for (int j = 1; j <= n; j++){
            Cnt_B = Calc(B, j);
            Cnt_A = Calc(A - 1, j);

            if (Cnt_B - Cnt_A != k[j]){
              flag = 0;
              break;
            }
          }

          if (flag==1){
```

```
            Ans++;
            //printf("%lld %lld\n", A, B);
            //System.out.println(A+" "+B);
            AnsA = A;
            AnsB = B;
          }
        }

        if (Ans == 0) System.out.println("None");//puts("None");
        else
        if (Ans == 1)
          System.out.println(AnsA+" "+AnsB);//cout << AnsA << ' ' << AnsB << endl;
        else
          System.out.println("Many");//puts("Many");
      }

    }
}
```

compact1, compact2, compact3

java.math

Class BigInteger

java.lang.Object
    java.lang.Number
        java.math.BigInteger

All Implemented Interfaces:

Serializable, Comparable<BigInteger>

```
public class BigInteger
extends Number
implements Comparable<BigInteger>
```

Immutable arbitrary-precision integers. All operations behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types). BigInteger provides analogues to all of Java's primitive integer operators, and all relevant methods from java.lang.Math. Additionally, BigInteger provides operations for modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.

Semantics of arithmetic operations exactly mimic those of Java's integer arithmetic operators, as defined in *The Java Language Specification*. For example, division by zero throws an `ArithmeticException`, and division of a negative by a positive yields a negative (or zero) remainder. All of the details in the Spec concerning overflow are ignored, as BigIntegers are made as large as necessary to accommodate the results of an operation.

Semantics of shift operations extend those of Java's shift operators to allow for negative shift distances. A right-shift with a negative shift distance results in a left shift, and vice-versa. The unsigned right shift operator (>>>) is omitted, as this operation makes little sense in combination with the "infinite word size" abstraction provided by this class.

Semantics of bitwise logical operations exactly mimic those of Java's bitwise integer operators. The binary operators (`and`, `or`, `xor`) implicitly perform sign extension on the shorter of the two operands prior to performing the operation.

Comparison operations perform signed integer comparisons, analogous to those performed by Java's relational and equality operators.

Modular arithmetic operations are provided to compute residues, perform exponentiation, and compute multiplicative inverses. These methods always return a non-negative result, between `0` and `(modulus - 1)`, inclusive.

Bit operations operate on a single bit of the two's-complement representation of their operand. If necessary, the operand is sign- extended so that it contains the designated bit. None of the single-bit operations can produce a BigInteger with a different sign from the BigInteger being operated on, as they affect only a single bit, and the "infinite word size" abstraction provided by this class ensures that there are infinitely many "virtual sign bits"

preceding each BigInteger.

For the sake of brevity and clarity, pseudo-code is used throughout the descriptions of BigInteger methods. The pseudo-code expression `(i + j)` is shorthand for "a BigInteger whose value is that of the BigInteger `i` plus that of the BigInteger `j`." The pseudo-code expression `(i == j)` is shorthand for "`true` if and only if the BigInteger `i` represents the same value as the BigInteger `j`." Other pseudo-code expressions are interpreted similarly.

All methods and constructors in this class throw `NullPointerException` when passed a null object reference for any input parameter. BigInteger must support values in the range $-2^{\texttt{Integer.MAX\_VALUE}}$ (exclusive) to $+2^{\texttt{Integer.MAX\_VALUE}}$ (exclusive) and may support values outside of that range. The range of probable prime values is limited and may be less than the full supported positive range of `BigInteger`. The range must be at least 1 to $2^{500000000}$.

Implementation Note:

`BigInteger constructors and operations throw ArithmeticException when the result is out of the supported range of` $-2^{\texttt{Integer.MAX\_VALUE}}$ `(exclusive) to` $+2^{\texttt{Integer.MAX\_VALUE}}$ `(exclusive).`

Since:

`JDK1.1`

See Also:

`BigDecimal`, `Serialized Form`

Field Summary

Fields

Modifier and Type	Field and Description
`static BigInteger`	**`ONE`** The BigInteger constant one.
`static BigInteger`	**`TEN`** The BigInteger constant ten.
`static BigInteger`	**`ZERO`** The BigInteger constant zero.

Constructor Summary

Constructors

Constructor and Description
`BigInteger``(byte[] val)` Translates a byte array containing the two's-complement binary representation of a BigInteger into a BigInteger.
`BigInteger``(int signum, byte[] magnitude)` Translates the sign-magnitude representation of a BigInteger into a BigInteger.

BigInteger(int bitLength, int certainty, **Random** rnd)
Constructs a randomly generated positive BigInteger that is probably prime, with the specified bitLength.

BigInteger(int numBits, **Random** rnd)
Constructs a randomly generated BigInteger, uniformly distributed over the range 0 to ($2^{numBits}$ - 1), inclusive.

BigInteger(**String** val)
Translates the decimal String representation of a BigInteger into a BigInteger.

BigInteger(**String** val, int radix)
Translates the String representation of a BigInteger in the specified radix into a BigInteger.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
BigInteger	**abs**() Returns a BigInteger whose value is the absolute value of this BigInteger.
BigInteger	**add**(**BigInteger** val) Returns a BigInteger whose value is (this + val).
BigInteger	**and**(**BigInteger** val) Returns a BigInteger whose value is (this & val).
BigInteger	**andNot**(**BigInteger** val) Returns a BigInteger whose value is (this & ~val).
int	**bitCount**() Returns the number of bits in the two's complement representation of this BigInteger that differ from its sign bit.
int	**bitLength**() Returns the number of bits in the minimal two's-complement representation of this BigInteger, *excluding* a sign bit.
byte	**byteValueExact**() Converts this BigInteger to a byte, checking for lost information.
BigInteger	**clearBit**(int n) Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit cleared.
int	**compareTo**(**BigInteger** val) Compares this BigInteger with the specified BigInteger.
BigInteger	**divide**(**BigInteger** val)

	Returns a BigInteger whose value is `(this / val)`.
`BigInteger[]`	**`divideAndRemainder(BigInteger val)`** Returns an array of two BigIntegers containing `(this / val)` followed by `(this % val)`.
`double`	**`doubleValue()`** Converts this BigInteger to a `double`.
`boolean`	**`equals(Object x)`** Compares this BigInteger with the specified Object for equality.
`BigInteger`	**`flipBit(int n)`** Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit flipped.
`float`	**`floatValue()`** Converts this BigInteger to a `float`.
`BigInteger`	**`gcd(BigInteger val)`** Returns a BigInteger whose value is the greatest common divisor of `abs(this)` and `abs(val)`.
`int`	**`getLowestSetBit()`** Returns the index of the rightmost (lowest-order) one bit in this BigInteger (the number of zero bits to the right of the rightmost one bit).
`int`	**`hashCode()`** Returns the hash code for this BigInteger.
`int`	**`intValue()`** Converts this BigInteger to an `int`.
`int`	**`intValueExact()`** Converts this `BigInteger` to an `int`, checking for lost information.
`boolean`	**`isProbablePrime(int certainty)`** Returns `true` if this BigInteger is probably prime, `false` if it's definitely composite.
`long`	**`longValue()`** Converts this BigInteger to a `long`.
`long`	**`longValueExact()`** Converts this `BigInteger` to a `long`, checking for lost information.
`BigInteger`	**`max(BigInteger val)`** Returns the maximum of this BigInteger and `val`.
`BigInteger`	**`min(BigInteger val)`** Returns the minimum of this BigInteger and `val`.
`BigInteger`	**`mod(BigInteger m)`** Returns a BigInteger whose value is `(this mod m)`.

BigInteger	**modInverse**(**BigInteger** m) Returns a BigInteger whose value is $(this^{-1} \bmod m)$.
BigInteger	**modPow**(**BigInteger** exponent, **BigInteger** m) Returns a BigInteger whose value is $(this^{exponent} \bmod m)$.
BigInteger	**multiply**(**BigInteger** val) Returns a BigInteger whose value is (this * val).
BigInteger	**negate**() Returns a BigInteger whose value is (-this).
BigInteger	**nextProbablePrime**() Returns the first integer greater than this BigInteger that is probably prime.
BigInteger	**not**() Returns a BigInteger whose value is (~this).
BigInteger	**or**(**BigInteger** val) Returns a BigInteger whose value is (this | val).
BigInteger	**pow**(int exponent) Returns a BigInteger whose value is $(this^{exponent})$.
static **BigInteger**	**probablePrime**(int bitLength, **Random** rnd) Returns a positive BigInteger that is probably prime, with the specified bitLength.
BigInteger	**remainder**(**BigInteger** val) Returns a BigInteger whose value is (this % val).
BigInteger	**setBit**(int n) Returns a BigInteger whose value is equivalent to this BigInteger with the designated bit set.
BigInteger	**shiftLeft**(int n) Returns a BigInteger whose value is (this << n).
BigInteger	**shiftRight**(int n) Returns a BigInteger whose value is (this >> n).
short	**shortValueExact**() Converts this BigInteger to a short, checking for lost information.
int	**signum**() Returns the signum function of this BigInteger.
BigInteger	**subtract**(**BigInteger** val) Returns a BigInteger whose value is (this - val).
boolean	**testBit**(int n) Returns true if and only if the designated bit is set.
byte[]	**toByteArray**() Returns a byte array containing the two's-complement

	Returns a byte array containing the two's-complement representation of this BigInteger.
String	**toString**() Returns the decimal String representation of this BigInteger.
String	**toString**(int radix) Returns the String representation of this BigInteger in the given radix.
static **BigInteger**	**valueOf**(long val) Returns a BigInteger whose value is equal to that of the specified long.
BigInteger	**xor**(**BigInteger** val) Returns a BigInteger whose value is (this ^ val).

Methods inherited from class java.lang.Number

byteValue, shortValue

Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

ZERO

```
public static final BigInteger ZERO
```

The BigInteger constant zero.

Since:
1.2

ONE

```
public static final BigInteger ONE
```

The BigInteger constant one.

Since:
1.2

TEN

```
public static final BigInteger TEN
```

The BigInteger constant ten.

compact1, compact2, compact3

java.util

Class TreeMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.TreeMap<K,V>

Type Parameters:

`K` - the type of keys maintained by this map

`V` - the type of mapped values

All Implemented Interfaces:

`Serializable`, `Cloneable`, `Map<K,V>`, `NavigableMap<K,V>`, `SortedMap<K,V>`

```
public class TreeMap<K,V>
extends AbstractMap<K,V>
implements NavigableMap<K,V>, Cloneable, Serializable
```

A Red-Black tree based `NavigableMap` implementation. The map is sorted according to the natural ordering of its keys, or by a `Comparator` provided at map creation time, depending on which constructor is used.

This implementation provides guaranteed log(n) time cost for the `containsKey`, `get`, `put` and `remove` operations. Algorithms are adaptations of those in Cormen, Leiserson, and Rivest's *Introduction to Algorithms*.

Note that the ordering maintained by a tree map, like any sorted map, and whether or not an explicit comparator is provided, must be *consistent with `equals`* if this sorted map is to correctly implement the `Map` interface. (See `Comparable` or `Comparator` for a precise definition of *consistent with equals*.) This is so because the `Map` interface is defined in terms of the `equals` operation, but a sorted map performs all key comparisons using its `compareTo` (or `compare`) method, so two keys that are deemed equal by this method are, from the standpoint of the sorted map, equal. The behavior of a sorted map *is* well-defined even if its ordering is inconsistent with `equals`; it just fails to obey the general contract of the `Map` interface.

Note that this implementation is not synchronized. If multiple threads access a map concurrently, and at least one of the threads modifies the map structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more mappings; merely changing the value associated with an existing key is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the map. If no such object exists, the map should be "wrapped" using the `Collections.synchronizedSortedMap` method. This is best done at creation time, to prevent accidental unsynchronized access to the map:

```
SortedMap m = Collections.synchronizedSortedMap(new TreeMap(...));
```

The iterators returned by the `iterator` method of the collections returned by all of this

class's "collection view methods" are *fail-fast*: if the map is structurally modified at any time after the iterator is created, in any way except through the iterator's own `remove` method, the iterator will throw a `ConcurrentModificationException`. Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw `ConcurrentModificationException` on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

All `Map.Entry` pairs returned by methods in this class and its views represent snapshots of mappings at the time they were produced. They do **not** support the `Entry.setValue` method. (Note however that it is possible to change mappings in the associated map using `put`.)

This class is a member of the Java Collections Framework.

Since:

1.2

See Also:

`Map`, `HashMap`, `Hashtable`, `Comparable`, `Comparator`, `Collection`, `Serialized Form`

Nested Class Summary

Nested classes/interfaces inherited from class java.util.AbstractMap

`AbstractMap.SimpleEntry<K,V>`, `AbstractMap.SimpleImmutableEntry<K,V>`

Constructor Summary

Constructors

Constructor and Description

`TreeMap()`
Constructs a new, empty tree map, using the natural ordering of its keys.

`TreeMap(Comparator<? super K> comparator)`
Constructs a new, empty tree map, ordered according to the given comparator.

`TreeMap(Map<? extends K,? extends V> m)`
Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.

`TreeMap(SortedMap<K,? extends V> m)`
Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`Map.Entry<K,V>`	**`ceilingEntry(K key)`** Returns a key-value mapping associated with the least key greater than or equal to the given key, or `null` if there is no such key.
`K`	**`ceilingKey(K key)`** Returns the least key greater than or equal to the given key, or `null` if there is no such key.
`void`	**`clear()`** Removes all of the mappings from this map.
`Object`	**`clone()`** Returns a shallow copy of this `TreeMap` instance.
`Comparator<? super K>`	**`comparator()`** Returns the comparator used to order the keys in this map, or `null` if this map uses the **natural ordering** of its keys.
`boolean`	**`containsKey(Object key)`** Returns `true` if this map contains a mapping for the specified key.
`boolean`	**`containsValue(Object value)`** Returns `true` if this map maps one or more keys to the specified value.
`NavigableSet<K>`	**`descendingKeySet()`** Returns a reverse order **`NavigableSet`** view of the keys contained in this map.
`NavigableMap<K,V>`	**`descendingMap()`** Returns a reverse order view of the mappings contained in this map.
`Set<Map.Entry<K,V>>`	**`entrySet()`** Returns a **`Set`** view of the mappings contained in this map.
`Map.Entry<K,V>`	**`firstEntry()`** Returns a key-value mapping associated with the least key in this map, or `null` if the map is empty.
`K`	**`firstKey()`** Returns the first (lowest) key currently in this map.
`Map.Entry<K,V>`	**`floorEntry(K key)`** Returns a key-value mapping associated with the greatest key less than or equal to the given key, or `null` if there is no such key.
`K`	**`floorKey(K key)`** Returns the greatest key less than or equal to the given key, or `null` if there is no such key.

	or null if there is no such key.
void	**forEach**(**BiConsumer**<? super **K**,? super **V**> action) Performs the given action for each entry in this map until all entries have been processed or the action throws an exception.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.
SortedMap<**K**,**V**>	**headMap**(**K** toKey) Returns a view of the portion of this map whose keys are strictly less than `toKey`.
NavigableMap<**K**,**V**>	**headMap**(**K** toKey, boolean inclusive) Returns a view of the portion of this map whose keys are less than (or equal to, if `inclusive` is true) `toKey`.
Map.Entry<**K**,**V**>	**higherEntry**(**K** key) Returns a key-value mapping associated with the least key strictly greater than the given key, or `null` if there is no such key.
K	**higherKey**(**K** key) Returns the least key strictly greater than the given key, or `null` if there is no such key.
Set<**K**>	**keySet**() Returns a **Set** view of the keys contained in this map.
Map.Entry<**K**,**V**>	**lastEntry**() Returns a key-value mapping associated with the greatest key in this map, or `null` if the map is empty.
K	**lastKey**() Returns the last (highest) key currently in this map.
Map.Entry<**K**,**V**>	**lowerEntry**(**K** key) Returns a key-value mapping associated with the greatest key strictly less than the given key, or `null` if there is no such key.
K	**lowerKey**(**K** key) Returns the greatest key strictly less than the given key, or `null` if there is no such key.
NavigableSet<**K**>	**navigableKeySet**() Returns a **NavigableSet** view of the keys contained in this map.
Map.Entry<**K**,**V**>	**pollFirstEntry**() Removes and returns a key-value mapping associated with the least key in this map, or `null` if the map is empty.
Map.Entry<**K**,**V**>	**pollLastEntry**() Removes and returns a key-value mapping associated with the greatest key in this map, or `null` if the map is empty

	the greatest key in this map, or null if the map is empty.
V	**put**(K key, V value) Associates the specified value with the specified key in this map.
void	**putAll**(Map<? extends K,? extends V> map) Copies all of the mappings from the specified map to this map.
V	**remove**(Object key) Removes the mapping for this key from this TreeMap if present.
V	**replace**(K key, V value) Replaces the entry for the specified key only if it is currently mapped to some value.
boolean	**replace**(K key, V oldValue, V newValue) Replaces the entry for the specified key only if currently mapped to the specified value.
void	**replaceAll**(BiFunction<? super K,? super V,? extends V> function) Replaces each entry's value with the result of invoking the given function on that entry until all entries have been processed or the function throws an exception.
int	**size**() Returns the number of key-value mappings in this map.
NavigableMap<K,V>	**subMap**(K fromKey, boolean fromInclusive, K toKey, boolean toInclusive) Returns a view of the portion of this map whose keys range from fromKey to toKey.
SortedMap<K,V>	**subMap**(K fromKey, K toKey) Returns a view of the portion of this map whose keys range from fromKey, inclusive, to toKey, exclusive.
SortedMap<K,V>	**tailMap**(K fromKey) Returns a view of the portion of this map whose keys are greater than or equal to fromKey.
NavigableMap<K,V>	**tailMap**(K fromKey, boolean inclusive) Returns a view of the portion of this map whose keys are greater than (or equal to, if inclusive is true) fromKey.
Collection<V>	**values**() Returns a **Collection** view of the values contained in this map.

Methods inherited from class java.util.AbstractMap

equals, hashCode, isEmpty, toString

15 gedit

```
Compile:
#!/bin/sh
full=$GEDIT_CURRENT_DOCUMENT_NAME
name=`echo $full | cut -d. -f1`
g++ $full -o $name -g -Wall

Debug:
#!/bin/bash
name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
gnome-terminal -x bash -c "gdb ./$name"

Run:
#!/bin/bash
name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
gnome-terminal -x bash -c "time ./$name;echo 'Press any key to continue'; read"
```

16 数学

16.1 常用数学公式

16.1.1 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. $\sum_{k=1}^{n}k(k+1)(k+2)(k+3)=\frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

16.1.2 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. $fib_m|fib_n^2\Leftrightarrow nfib_n|m$

16.1.3 错排公式

1. $D_n = (n-1)(D_{n-2} - D_{n-1})$

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

16.1.4 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & 若n = 1 \\ (-1)^k & 若n无平方数因子，且n = p_1 p_2 \ldots p_k \\ 0 & 若n有大于1的平方数因数 \end{cases}$$

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

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

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

16.1.5 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：

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

16.1.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

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

16.1.7 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

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

当 n 为偶数时，n 个结点的无根树的个数为

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

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。

16.1.8 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：

$$V - E + F = 2$$

16.1.9 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：

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

16.1.10 牛顿恒等式

设

$$\prod_{i=1}^{n} (x - x_i) = a_n + a_{n-1} x + \cdots + a_1 x^{n-1} + a_0 x^n$$

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

则

$$a_0 p_k + a_1 p_{k-1} + \cdots + a_{k-1} p_1 + k a_k = 0$$

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

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

16.2 平面几何公式

16.2.1 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

$$\begin{aligned}r&=\frac{S}{p}=\frac{arcsin\frac{B}{2}\cdot sin\frac{C}{2}}{sin\frac{B+C}{2}}=4R\cdot sin\frac{A}{2}sin\frac{B}{2}sin\frac{C}{2}\\&=\sqrt{\frac{(p-a)(p-b)(p-c)}{p}}=p\cdot tan\frac{A}{2}tan\frac{B}{2}tan\frac{C}{2}\end{aligned}$$

7. 外接圆半径

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

16.2.2 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

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

16.2.3 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

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

16.2.4 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

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

5. 弓形面积

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

16.2.5 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

$$T = S + 2A$$

16.2.6 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

$$T = S + 2A$$

16.2.7 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

$$T = S + A_1 + A_2$$

16.2.8 圆柱

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

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

3. 体积

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

16.2.9 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

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

16.2.10 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

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

16.2.11 球

1. 全面积

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

2. 体积

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

16.2.12 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

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

16.2.13 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

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

16.3 积分表

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

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

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

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

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

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

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

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

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

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

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

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

$\int x^n e^{ax}\,dx = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,dx$

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

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

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

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

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

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

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

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

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

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

16.4 立体几何公式

16.4.1 球面三角公式

设 a,b,c 是边长，A,B,C 是所对的二面角，有余弦定理

$$cosa = cosb \cdot cosc + sinb \cdot sinc \cdot cosA$$

正弦定理

$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$

三角形面积是 $A+B+C-\pi$

16.4.2 四面体体积公式

U,V,W,u,v,w 是四面体的 6 条棱，U,V,W 构成三角形，$(U,u),(V,v),(W,w)$ 互为对棱，则

$$V=\frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中

$$\begin{cases} a &= \sqrt{xYZ}, \\ b &= \sqrt{yZX}, \\ c &= \sqrt{zXY}, \\ d &= \sqrt{xyz}, \\ s &= a+b+c+d, \\ X &= (w-U+v)(U+v+w), \\ x &= (U-v+w)(v-w+U), \\ Y &= (u-V+w)(V+w+u), \\ y &= (V-w+u)(w-u+V), \\ Z &= (v-W+u)(W+u+v), \\ z &= (W-u+v)(u-v+W) \end{cases}$$

16.5 博弈游戏

16.6 巴什博奕

1. 只有一堆 `n` 个物品，两个人轮流从这堆物品中取物，规定每次至少取一个，最多取 `m` 个。最后取光者得胜。

2. 显然，如果 $n=m+1$，那么由于一次最多只能取 m 个，所以，无论先取者拿走多少个，后取者都能够一次拿走剩余的物品，后者取胜。因此我们发现了如何取胜的法则：如果 $n=(m+1)r+s$，（`r` 为任意自然数，$s\le m$），那么先取者要拿走 s 个物品，如果后取者拿走 $k(k\le m)$ 个，那么先取者再拿走 $m+1-k$ 个，结果剩下 $(m+1)(r-1)$ 个，以后保持这样的取法，那么先取者肯定获胜。总之，要保持给对手留下 $(m+1)$ 的倍数，就能最后获胜。

16.7 威佐夫博弈

1. 有两堆各若干个物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。

2. 判断一个局势 (a,b) 为奇异局势（必败态）的方法：

$$a_k=[k(1+\sqrt{5})/2],b_k=a_k+k$$

16.8 阶梯博奕

1. 博弈在一列阶梯上进行，每个阶梯上放着自然数个点，两个人进行阶梯博弈，每一步则是将一个阶梯上的若干个点（至少一个）移到前面去，最后没有点可以移动的人输。

2. 解决方法：把所有奇数阶梯看成 `N` 堆石子，做 `NIM`。（把石子从奇数堆移动到偶数堆可以理解为拿走石子，就相当于几个奇数堆的石子在做 `Nim`）

16.9 图上删边游戏

16.9.1 链的删边游戏

1. 游戏规则：对于一条链，其中一个端点是根，两人轮流删边，脱离根的部分也算被删去，最后没边可删的人输。

2. 做法：$sg[i] = n - dist(i) - 1$（其中 n 表示总点数，$dist(i)$ 表示离根的距离）

16.9.2 树的删边游戏

1. 游戏规则：对于一棵有根树，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。

2. 做法：叶子结点的 $sg = 0$，其他节点的 sg 等于儿子结点的 $sg + 1$ 的异或和。

16.9.3 局部连通图的删边游戏

1. 游戏规则：在一个局部连通图上，两人轮流删边，脱离根的部分也算被删去，没边可删的人输。局部连通图的构图规则是，在一棵基础树上加边得到，所有形成的环保证不共用边，且只与基础树有一个公共点。

2. 做法：去掉所有的偶环，将所有的奇环变为长度为 `1` 的链，然后做树的删边游戏。

16.10 常用数学公式

16.11 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. $\sum_{k=1}^{n}k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

16.12 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. $fib_m | fib_n^2 \Leftrightarrow n fib_n | m$

16.13 错排公式

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

16.14 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & 若n = 1 \\ (-1)^k & 若n无平方数因子，且n = p_1 p_2 \ldots p_k \\ 0 & 若n有大于1的平方数因数 \end{cases}$$

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

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

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

16.15 Burnside 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：

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

16.16 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

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

16.17 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

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

当 n 为偶数时，n 个结点的无根树的个数为

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

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。

16.18 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：

$$V-E+F=2$$

16.19 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：

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

16.20 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

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

17 平面几何公式

17.1 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

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

17.2 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

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

17.3 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

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

17.4 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

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

17.5 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

$$T = S + 2A$$

17.6 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

$$T = S + 2A$$

17.7 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

$$T = S + A_1 + A_2$$

17.8 圆柱

1. 侧面积

$$S = 2\pi r h$$

2. 全面积

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

3. 体积

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

17.9 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

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

17.10 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

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

17.11 球

1. 全面积

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

2. 体积

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

17.12 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

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

17.13 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

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

18 立体几何公式

18.1 球面三角公式

设 a, b, c 是边长，A, B, C 是所对的二面角，有余弦定理

$$cosa = cosb \cdot cosc + sinb \cdot sinc \cdot cosA$$

正弦定理

$$\frac{sinA}{sina} = \frac{sinB}{sinb} = \frac{sinC}{sinc}$$

三角形面积是 $A + B + C - \pi$

18.2 四面体体积公式

U, V, W, u, v, w 是四面体的 6 条棱，U, V, W 构成三角形，$(U, u), (V, v), (W, w)$ 互为对棱，则

$$V = \frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw}$$

其中

$$\begin{cases} a &= \sqrt{xYZ}, \\ b &= \sqrt{yZX}, \\ c &= \sqrt{zXY}, \\ d &= \sqrt{xyz}, \\ s &= a+b+c+d, \\ X &= (w-U+v)(U+v+w), \\ x &= (U-v+w)(v-w+U), \\ Y &= (u-V+w)(V+w+u), \\ y &= (V-w+u)(w-u+V), \\ Z &= (v-W+u)(W+u+v), \\ z &= (W-u+v)(u-v+W) \end{cases}$$

19 附录

19.1 NTT 素数及原根列表

Id	Primes	Primitive Root	Id	Primes	Primitive Root	Id	Primes	Primitive Root
1	7340033	3	38	311427073	7	75	786432001	7
2	13631489	15	39	330301441	22	76	799014913	13
3	23068673	3	40	347078657	3	77	800063489	3
4	26214401	3	41	359661569	3	78	802160641	11
5	28311553	5	42	361758721	29	79	818937857	5
6	69206017	5	43	377487361	7	80	824180737	5
7	70254593	3	44	383778817	5	81	833617921	13
8	81788929	7	45	387973121	6	82	850395137	3
9	101711873	3	46	399507457	5	83	862978049	3
10	104857601	3	47	409993217	3	84	880803841	26
11	111149057	3	48	415236097	5	85	883949569	7
12	113246209	7	49	447741953	3	86	897581057	3
13	120586241	6	50	459276289	11	87	899678209	7
14	132120577	5	51	463470593	3	88	907018241	3
15	136314881	3	52	468713473	5	89	913309697	3
16	138412033	5	53	469762049	3	90	918552577	5
17	141557761	26	54	493879297	10	91	919601153	3
18	147849217	5	55	531628033	5	92	924844033	5
19	155189249	6	56	576716801	6	93	925892609	3
20	158334977	3	57	581959681	11	94	935329793	3
21	163577857	23	58	595591169	3	95	938475521	3
22	167772161	3	59	597688321	11	96	940572673	7
23	169869313	5	60	605028353	3	97	943718401	7
24	185597953	5	61	635437057	11	98	950009857	7
25	186646529	3	62	639631361	6	99	957349889	6
26	199229441	3	63	645922817	3	100	962592769	7
27	204472321	19	64	648019969	17	101	972029953	10
28	211812353	3	65	655360001	3	102	975175681	17
29	221249537	3	66	666894337	5	103	976224257	3
30	230686721	6	67	683671553	3	104	985661441	3
31	246415361	3	68	710934529	17	105	998244353	3
32	249561089	3	69	715128833	3	106	1004535809	3
33	257949697	5	70	718274561	3	107	1007681537	3
34	270532609	22	71	740294657	3	108	1012924417	5
35	274726913	3	72	745537537	5	109	1045430273	3
36	290455553	3	73	754974721	11	110	1051721729	6
37	305135617	5	74	770703361	11	111	1053818881	7

19.2 cheat.pdf

Theoretical Computer Science Cheat Sheet

Definitions

$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)\ \forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0\ \forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a\rvert < \epsilon$, $\forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf_{n\to\infty} a_n$	$\lim_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup_{n\to\infty} a_n$	$\lim_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\left[{n \atop k}\right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{{n \atop k}\right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle{n \atop k}\right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

Series

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

$$\sum_{i=1}^{n} i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$$

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

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \quad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n}\frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \quad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\binom{n}{k} = \binom{n}{n-k}$,

4. $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, **5.** $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,

6. $\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, **7.** $\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,

8. $\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$, **9.** $\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,

10. $\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** $\left\{{n \atop 1}\right\} = \left\{{n \atop n}\right\} = 1$,

12. $\left\{{n \atop 2}\right\} = 2^{n-1} - 1$, **13.** $\left\{{n \atop k}\right\} = k\left\{{n-1 \atop k}\right\} + \left\{{n-1 \atop k-1}\right\}$,

14. $\left[{n \atop 1}\right] = (n-1)!$, **15.** $\left[{n \atop 2}\right] = (n-1)!H_{n-1}$, **16.** $\left[{n \atop n}\right] = 1$, **17.** $\left[{n \atop k}\right] \ge \left\{{n \atop k}\right\}$,

18. $\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, **19.** $\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$, **20.** $\sum_{k=0}^{n}\left[{n \atop k}\right] = n!$, **21.** $C_n = \frac{1}{n+1}\binom{2n}{n}$,

22. $\left\langle{n \atop 0}\right\rangle = \left\langle{n \atop n-1}\right\rangle = 1$, **23.** $\left\langle{n \atop k}\right\rangle = \left\langle{n \atop n-1-k}\right\rangle$, **24.** $\left\langle{n \atop k}\right\rangle = (k+1)\left\langle{n-1 \atop k}\right\rangle + (n-k)\left\langle{n-1 \atop k-1}\right\rangle$,

25. $\left\langle{0 \atop k}\right\rangle = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases}$ **26.** $\left\langle{n \atop 1}\right\rangle = 2^n - n - 1$, **27.** $\left\langle{n \atop 2}\right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,

28. $x^n = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{x+k}{n}$, **29.** $\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$, **30.** $m!\left\{{n \atop m}\right\} = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{k}{n-m}$,

31. $\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{n}\left\{{n \atop k}\right\}\binom{n-k}{m}(-1)^{n-k-m}k!$, **32.** $\left\langle\!\left\langle{n \atop 0}\right\rangle\!\right\rangle = 1$, **33.** $\left\langle\!\left\langle{n \atop n}\right\rangle\!\right\rangle = 0$ for $n \ne 0$,

34. $\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle = (k+1)\left\langle\!\left\langle{n-1 \atop k}\right\rangle\!\right\rangle + (2n-1-k)\left\langle\!\left\langle{n-1 \atop k-1}\right\rangle\!\right\rangle$, **35.** $\sum_{k=0}^{n}\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle = \frac{(2n)^{\underline{n}}}{2^n}$,

36. $\left\{{x \atop x-n}\right\} = \sum_{k=0}^{n}\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle\binom{x+n-1-k}{2n}$, **37.** $\left\{{n+1 \atop m+1}\right\} = \sum_{k}\binom{n}{k}\left\{{k \atop m}\right\} = \sum_{k=0}^{n}\left\{{k \atop m}\right\}(m+1)^{n-k}$,

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\begin{bmatrix} n+1 \\ m+1 \end{bmatrix} = \sum_k \begin{bmatrix} n \\ k \end{bmatrix} \binom{k}{m} = \sum_{k=0}^{n} \begin{bmatrix} k \\ m \end{bmatrix} n^{\underline{n-k}} = n! \sum_{k=0}^{n} \frac{1}{k!} \begin{bmatrix} k \\ m \end{bmatrix}$,

39. $\begin{bmatrix} x \\ x-n \end{bmatrix} = \sum_{k=0}^{n} \left\langle\!\left\langle {n \atop k} \right\rangle\!\right\rangle \binom{x+k}{2n}$,

40. $\begin{Bmatrix} n \\ m \end{Bmatrix} = \sum_k \binom{n}{k} \begin{Bmatrix} k+1 \\ m+1 \end{Bmatrix} (-1)^{n-k}$,

41. $\begin{bmatrix} n \\ m \end{bmatrix} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \binom{k}{m} (-1)^{m-k}$,

42. $\begin{Bmatrix} m+n+1 \\ m \end{Bmatrix} = \sum_{k=0}^{m} k \begin{Bmatrix} n+k \\ k \end{Bmatrix}$,

43. $\begin{bmatrix} m+n+1 \\ m \end{bmatrix} = \sum_{k=0}^{m} k(n+k) \begin{bmatrix} n+k \\ k \end{bmatrix}$,

44. $\binom{n}{m} = \sum_k \begin{Bmatrix} n+1 \\ k+1 \end{Bmatrix} \begin{bmatrix} k \\ m \end{bmatrix} (-1)^{m-k}$,

45. $(n-m)! \binom{n}{m} = \sum_k \begin{bmatrix} n+1 \\ k+1 \end{bmatrix} \begin{Bmatrix} k \\ m \end{Bmatrix} (-1)^{m-k}$, for $n \geq m$,

46. $\begin{Bmatrix} n \\ n-m \end{Bmatrix} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \begin{bmatrix} m+k \\ k \end{bmatrix}$,

47. $\begin{bmatrix} n \\ n-m \end{bmatrix} = \sum_k \binom{m-n}{m+k} \binom{m+n}{n+k} \begin{Bmatrix} m+k \\ k \end{Bmatrix}$,

48. $\begin{Bmatrix} n \\ \ell+m \end{Bmatrix} \binom{\ell+m}{\ell} = \sum_k \begin{Bmatrix} k \\ \ell \end{Bmatrix} \begin{Bmatrix} n-k \\ m \end{Bmatrix} \binom{n}{k}$,

49. $\begin{bmatrix} n \\ \ell+m \end{bmatrix} \binom{\ell+m}{\ell} = \sum_k \begin{bmatrix} k \\ \ell \end{bmatrix} \begin{bmatrix} n-k \\ m \end{bmatrix} \binom{n}{k}$.

Trees

Every tree with n vertices has $n-1$ edges.

Kraft inequality: If the depths of the leaves of a binary tree are $d_1, \ldots, d_n$:

$$\sum_{i=1}^{n} 2^{-d_i} \leq 1,$$

and equality holds only if every internal node has 2 sons.

Recurrences

Master method:
$T(n) = aT(n/b) + f(n), \quad a \geq 1, b > 1$

If $\exists \epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$ then

$$T(n) = \Theta(n^{\log_b a}).$$

If $f(n) = \Theta(n^{\log_b a})$ then

$$T(n) = \Theta(n^{\log_b a} \log_2 n).$$

If $\exists \epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and $\exists c < 1$ such that $af(n/b) \leq cf(n)$ for large n, then

$$T(n) = \Theta(f(n)).$$

Substitution (example): Consider the following recurrence

$$T_{i+1} = 2^{2^i} \cdot T_i^2, \quad T_1 = 2.$$

Note that T_i is always a power of two. Let $t_i = \log_2 T_i$. Then we have

$$t_{i+1} = 2^i + 2t_i, \quad t_1 = 1.$$

Let $u_i = t_i / 2^i$. Dividing both sides of the previous equation by 2^{i+1} we get

$$\frac{t_{i+1}}{2^{i+1}} = \frac{2^i}{2^{i+1}} + \frac{t_i}{2^i}.$$

Substituting we find

$$u_{i+1} = \tfrac{1}{2} + u_i, \qquad u_1 = \tfrac{1}{2},$$

which is simply $u_i = i/2$. So we find that T_i has the closed form $T_i = 2^{i2^{i-1}}$.

Summing factors (example): Consider the following recurrence

$$T(n) = 3T(n/2) + n, \quad T(1) = 1.$$

Rewrite so that all terms involving T are on the left side

$$T(n) - 3T(n/2) = n.$$

Now expand the recurrence, and choose a factor which makes the left side "telescope"

$$\begin{aligned} 1\big(T(n) - 3T(n/2) &= n\big) \\ 3\big(T(n/2) - 3T(n/4) &= n/2\big) \\ &\vdots \quad \vdots \quad \vdots \\ 3^{\log_2 n - 1}\big(T(2) - 3T(1) &= 2\big) \end{aligned}$$

Let $m = \log_2 n$. Summing the left side we get $T(n) - 3^m T(1) = T(n) - 3^m = T(n) - n^k$ where $k = \log_2 3 \approx 1.58496$. Summing the right side we get

$$\sum_{i=0}^{m-1} \frac{n}{2^i} 3^i = n \sum_{i=0}^{m-1} \left(\tfrac{3}{2}\right)^i.$$

Let $c = \frac{3}{2}$. Then we have

$$\begin{aligned} n \sum_{i=0}^{m-1} c^i &= n \left(\frac{c^m - 1}{c - 1} \right) \\ &= 2n(c^{\log_2 n} - 1) \\ &= 2n(c^{(k-1)\log_c n} - 1) \\ &= 2n^k - 2n, \end{aligned}$$

and so $T(n) = 3n^k - 2n$. Full history recurrences can often be changed to limited history ones (example): Consider

$$T_i = 1 + \sum_{j=0}^{i-1} T_j, \quad T_0 = 1.$$

Note that

$$T_{i+1} = 1 + \sum_{j=0}^{i} T_j.$$

Subtracting we find

$$\begin{aligned} T_{i+1} - T_i &= 1 + \sum_{j=0}^{i} T_j - 1 - \sum_{j=0}^{i-1} T_j \\ &= T_i. \end{aligned}$$

And so $T_{i+1} = 2T_i = 2^{i+1}$.

Generating functions:

1. Multiply both sides of the equation by x^i.
2. Sum both sides over all i for which the equation is valid.
3. Choose a generating function $G(x)$. Usually $G(x) = \sum_{i=0}^{\infty} x^i g_i$.
3. Rewrite the equation in terms of the generating function $G(x)$.
4. Solve for $G(x)$.
5. The coefficient of x^i in $G(x)$ is g_i.

Example:

$$g_{i+1} = 2g_i + 1, \quad g_0 = 0.$$

Multiply and sum:

$$\sum_{i \geq 0} g_{i+1} x^i = \sum_{i \geq 0} 2g_i x^i + \sum_{i \geq 0} x^i.$$

We choose $G(x) = \sum_{i \geq 0} x^i g_i$. Rewrite in terms of $G(x)$:

$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i \geq 0} x^i.$$

Simplify:

$$\frac{G(x)}{x} = 2G(x) + \frac{1}{1-x}.$$

Solve for $G(x)$:

$$G(x) = \frac{x}{(1-x)(1-2x)}.$$

Expand this using partial fractions:

$$\begin{aligned} G(x) &= x \left(\frac{2}{1-2x} - \frac{1}{1-x} \right) \\ &= x \left(2 \sum_{i \geq 0} 2^i x^i - \sum_{i \geq 0} x^i \right) \\ &= \sum_{i \geq 0} (2^{i+1} - 1) x^{i+1}. \end{aligned}$$

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

$\pi \approx 3.14159$, $e \approx 2.71828$, $\gamma \approx 0.57721$, $\phi = \frac{1+\sqrt{5}}{2} \approx 1.61803$, $\hat{\phi} = \frac{1-\sqrt{5}}{2} \approx -.61803$

i	2^i	p_i
1	2	2
2	4	3
3	8	5
4	16	7
5	32	11
6	64	13
7	128	17
8	256	19
9	512	23
10	1,024	29
11	2,048	31
12	4,096	37
13	8,192	41
14	16,384	43
15	32,768	47
16	65,536	53
17	131,072	59
18	262,144	61
19	524,288	67
20	1,048,576	71
21	2,097,152	73
22	4,194,304	79
23	8,388,608	83
24	16,777,216	89
25	33,554,432	97
26	67,108,864	101
27	134,217,728	103
28	268,435,456	107
29	536,870,912	109
30	1,073,741,824	113
31	2,147,483,648	127
32	4,294,967,296	131

Pascal's Triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36 9 1
1 10 45 120 210 252 210 120 45 10 1

General

Bernoulli Numbers ($B_i = 0$, odd $i \neq 1$):

$B_0 = 1$, $B_1 = -\frac{1}{2}$, $B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}$, $B_6 = \frac{1}{42}$, $B_8 = -\frac{1}{30}$, $B_{10} = \frac{5}{66}$.

Change of base, quadratic formula:

$$\log_b x = \frac{\log_a x}{\log_a b}, \qquad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Euler's number e:

$$e = 1 + \tfrac{1}{2} + \tfrac{1}{6} + \tfrac{1}{24} + \tfrac{1}{120} + \cdots$$

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x.$$

$$\left(1 + \tfrac{1}{n}\right)^n < e < \left(1 + \tfrac{1}{n}\right)^{n+1}.$$

$$\left(1 + \tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

Harmonic numbers:

$1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \ldots$

$$\ln n < H_n < \ln n + 1,$$

$$H_n = \ln n + \gamma + O\left(\frac{1}{n}\right).$$

Factorial, Stirling's approximation:

1, 2, 6, 24, 120, 720, 5040, 40320, 362880, ...

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

Ackermann's function and inverse:

$$a(i,j) = \begin{cases} 2^j & i = 1 \\ a(i-1,2) & j = 1 \\ a(i-1,a(i,j-1)) & i,j \geq 2 \end{cases}$$

$$\alpha(i) = \min\{j \mid a(j,j) \geq i\}.$$

Binomial distribution:

$$\Pr[X = k] = \binom{n}{k} p^k q^{n-k}, \qquad q = 1 - p,$$

$$\mathrm{E}[X] = \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} = np.$$

Poisson distribution:

$$\Pr[X = k] = \frac{e^{-\lambda}\lambda^k}{k!}, \quad \mathrm{E}[X] = \lambda.$$

Normal (Gaussian) distribution:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad \mathrm{E}[X] = \mu.$$

The "coupon collector": We are given a random coupon each day, and there are n different types of coupons. The distribution of coupons is uniform. The expected number of days to pass before we to collect all n types is

$$nH_n.$$

Probability

Continuous distributions: If

$$\Pr[a < X < b] = \int_a^b p(x)\,dx,$$

then p is the probability density function of X. If

$$\Pr[X < a] = P(a),$$

then P is the distribution function of X. If P and p both exist then

$$P(a) = \int_{-\infty}^{a} p(x)\,dx.$$

Expectation: If X is discrete

$$\mathrm{E}[g(X)] = \sum_x g(x) \Pr[X = x].$$

If X continuous then

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x)p(x)\,dx = \int_{-\infty}^{\infty} g(x)\,dP(x).$$

Variance, standard deviation:

$$\mathrm{VAR}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2,$$

$$\sigma = \sqrt{\mathrm{VAR}[X]}.$$

For events A and B:

$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$

$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$

iff A and B are independent.

$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:

$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$

if X and Y are independent.

$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$

$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$

Bayes' theorem:

$$\Pr[A_i|B] = \frac{\Pr[B|A_i]\Pr[A_i]}{\sum_{j=1}^{n} \Pr[A_j]\Pr[B|A_j]}.$$

Inclusion-exclusion:

$$\Pr\left[\bigvee_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \Pr[X_i] + \sum_{k=2}^{n} (-1)^{k+1} \sum_{i_i < \cdots < i_k} \Pr\left[\bigwedge_{j=1}^{k} X_{i_j}\right].$$

Moment inequalities:

$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$

$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

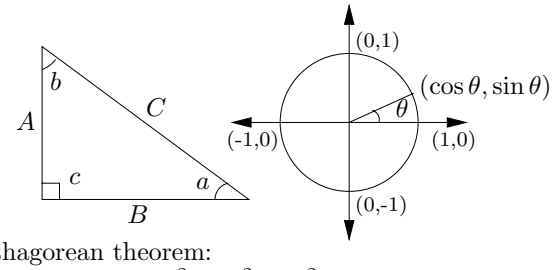
Geometric distribution:

$$\Pr[X = k] = pq^{k-1}, \qquad q = 1 - p,$$

$$\mathrm{E}[X] = \sum_{k=1}^{\infty} kpq^{k-1} = \frac{1}{p}.$$

Theoretical Computer Science Cheat Sheet

Trigonometry



Pythagorean theorem:

$$C^2 = A^2 + B^2.$$

Definitions:

$$\sin a = A/C, \quad \cos a = B/C,$$
$$\csc a = C/A, \quad \sec a = C/B,$$
$$\tan a = \frac{\sin a}{\cos a} = \frac{A}{B}, \quad \cot a = \frac{\cos a}{\sin a} = \frac{B}{A}.$$

Area, radius of inscribed circle:

$$\tfrac{1}{2}AB, \quad \frac{AB}{A+B+C}.$$

Identities:

$$\sin x = \frac{1}{\csc x}, \qquad \cos x = \frac{1}{\sec x},$$
$$\tan x = \frac{1}{\cot x}, \qquad \sin^2 x + \cos^2 x = 1,$$
$$1 + \tan^2 x = \sec^2 x, \qquad 1 + \cot^2 x = \csc^2 x,$$
$$\sin x = \cos\left(\tfrac{\pi}{2} - x\right), \qquad \sin x = \sin(\pi - x),$$
$$\cos x = -\cos(\pi - x), \qquad \tan x = \cot\left(\tfrac{\pi}{2} - x\right),$$
$$\cot x = -\cot(\pi - x), \qquad \csc x = \cot\tfrac{x}{2} - \cot x,$$
$$\sin(x \pm y) = \sin x \cos y \pm \cos x \sin y,$$
$$\cos(x \pm y) = \cos x \cos y \mp \sin x \sin y,$$
$$\tan(x \pm y) = \frac{\tan x \pm \tan y}{1 \mp \tan x \tan y},$$
$$\cot(x \pm y) = \frac{\cot x \cot y \mp 1}{\cot x \pm \cot y},$$
$$\sin 2x = 2\sin x \cos x, \qquad \sin 2x = \frac{2\tan x}{1 + \tan^2 x},$$
$$\cos 2x = \cos^2 x - \sin^2 x, \qquad \cos 2x = 2\cos^2 x - 1,$$
$$\cos 2x = 1 - 2\sin^2 x, \qquad \cos 2x = \frac{1 - \tan^2 x}{1 + \tan^2 x},$$
$$\tan 2x = \frac{2\tan x}{1 - \tan^2 x}, \qquad \cot 2x = \frac{\cot^2 x - 1}{2\cot x},$$
$$\sin(x + y)\sin(x - y) = \sin^2 x - \sin^2 y,$$
$$\cos(x + y)\cos(x - y) = \cos^2 x - \sin^2 y.$$

Euler's equation:

$$e^{ix} = \cos x + i\sin x, \qquad e^{i\pi} = -1.$$

v2.02 ©1994 by Steve Seiden
sseiden@acm.org
http://www.csc.lsu.edu/~seiden

Matrices

Multiplication:

$$C = A \cdot B, \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

Determinants: $\det A \neq 0$ iff A is non-singular.

$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g\begin{vmatrix} b & c \\ e & f \end{vmatrix} - h\begin{vmatrix} a & c \\ d & f \end{vmatrix} + i\begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= aei + bfg + cdh - ceg - fha - ibd.$$

Permanents:

$$\operatorname{perm} A = \sum_{\pi} \prod_{i=1}^{n} a_{i,\pi(i)}.$$

Hyperbolic Functions

Definitions:

$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

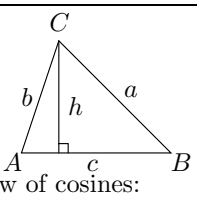
Identities:

$$\cosh^2 x - \sinh^2 x = 1, \qquad \tanh^2 x + \operatorname{sech}^2 x = 1,$$
$$\coth^2 x - \operatorname{csch}^2 x = 1, \qquad \sinh(-x) = -\sinh x,$$
$$\cosh(-x) = \cosh x, \qquad \tanh(-x) = -\tanh x,$$
$$\sinh(x + y) = \sinh x \cosh y + \cosh x \sinh y,$$
$$\cosh(x + y) = \cosh x \cosh y + \sinh x \sinh y,$$
$$\sinh 2x = 2\sinh x \cosh x,$$
$$\cosh 2x = \cosh^2 x + \sinh^2 x,$$
$$\cosh x + \sinh x = e^x, \qquad \cosh x - \sinh x = e^{-x},$$
$$(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$$
$$2\sinh^2 \tfrac{x}{2} = \cosh x - 1, \qquad 2\cosh^2 \tfrac{x}{2} = \cosh x + 1.$$

θ	$\sin\theta$	$\cos\theta$	$\tan\theta$
0	0	1	0
$\frac{\pi}{6}$	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{3}}{3}$
$\frac{\pi}{4}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	1
$\frac{\pi}{3}$	$\frac{\sqrt{3}}{2}$	$\frac{1}{2}$	$\sqrt{3}$
$\frac{\pi}{2}$	1	0	∞

> ... in mathematics you don't understand things, you just get used to them.
> – J. von Neumann

More Trig.



Law of cosines:

$$c^2 = a^2 + b^2 - 2ab\cos C.$$

Area:

$$A = \tfrac{1}{2}hc,$$
$$= \tfrac{1}{2}ab\sin C,$$
$$= \frac{c^2 \sin A \sin B}{2\sin C}.$$

Heron's formula:

$$A = \sqrt{s \cdot s_a \cdot s_b \cdot s_c},$$
$$s = \tfrac{1}{2}(a + b + c),$$
$$s_a = s - a,$$
$$s_b = s - b,$$
$$s_c = s - c.$$

More identities:

$$\sin\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{2}},$$
$$\cos\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{2}},$$
$$\tan\tfrac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}} = \frac{1 - \cos x}{\sin x} = \frac{\sin x}{1 + \cos x},$$
$$\cot\tfrac{x}{2} = \sqrt{\frac{1 + \cos x}{1 - \cos x}} = \frac{1 + \cos x}{\sin x} = \frac{\sin x}{1 - \cos x},$$
$$\sin x = \frac{e^{ix} - e^{-ix}}{2i},$$
$$\cos x = \frac{e^{ix} + e^{-ix}}{2},$$
$$\tan x = -i\frac{e^{ix} - e^{-ix}}{e^{ix} + e^{-ix}} = -i\frac{e^{2ix} - 1}{e^{2ix} + 1},$$
$$\sin x = \frac{\sinh ix}{i},$$
$$\cos x = \cosh ix,$$
$$\tan x = \frac{\tanh ix}{i}.$$

Theoretical Computer Science Cheat Sheet

Number Theory

The Chinese remainder theorem: There exists a number C such that:

$$\begin{aligned} C &\equiv r_1 \bmod m_1 \\ &\vdots \quad \vdots \quad \vdots \\ C &\equiv r_n \bmod m_n \end{aligned}$$

if m_i and m_j are relatively prime for $i \neq j$.

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x. If $\prod_{i=1}^{n} p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^{n} p_i^{e_i - 1}(p_i - 1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \bmod b.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \bmod p.$$

The Euclidean algorithm: if $a > b$ are integers then

$$\gcd(a, b) = \gcd(a \bmod b, b).$$

If $\prod_{i=1}^{n} p_i^{e_i}$ is the prime factorization of x then

$$S(x) = \sum_{d|x} d = \prod_{i=1}^{n} \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

Perfect Numbers: x is an even perfect number iff $x = 2^{n-1}(2^n - 1)$ and $2^n - 1$ is prime.

Wilson's theorem: n is a prime iff

$$(n-1)! \equiv -1 \bmod n.$$

Möbius inversion:

$$\mu(i) = \begin{cases} 1 & \text{if } i = 1. \\ 0 & \text{if } i \text{ is not square-free.} \\ (-1)^r & \text{if } i \text{ is the product of } r \text{ distinct primes.} \end{cases}$$

If

$$G(a) = \sum_{d|a} F(d),$$

then

$$F(a) = \sum_{d|a} \mu(d) G\Big(\frac{a}{d}\Big).$$

Prime numbers:

$$p_n = n \ln n + n \ln \ln n - n + n\frac{\ln \ln n}{\ln n} + O\Big(\frac{n}{\ln n}\Big),$$

$$\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3} + O\Big(\frac{n}{(\ln n)^4}\Big).$$

Graph Theory

Definitions:

Loop	An edge connecting a vertex to itself.
Directed	Each edge has a direction.
Simple	Graph with no loops or multi-edges.
Walk	A sequence $v_0 e_1 v_1 \ldots e_\ell v_\ell$.
Trail	A walk with distinct edges.
Path	A trail with distinct vertices.
Connected	A graph where there exists a path between any two vertices.
Component	A maximal connected subgraph.
Tree	A connected acyclic graph.
Free tree	A tree with no root.
DAG	Directed acyclic graph.
Eulerian	Graph with a trail visiting each edge exactly once.
Hamiltonian	Graph with a cycle visiting each vertex exactly once.
Cut	A set of edges whose removal increases the number of components.
Cut-set	A minimal cut.
Cut edge	A size 1 cut.
k-Connected	A graph connected with the removal of any $k-1$ vertices.
k-Tough	$\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G-S) \leq \lvert S \rvert$.
k-Regular	A graph where all vertices have degree k.
k-Factor	A k-regular spanning subgraph.
Matching	A set of edges, no two of which are adjacent.
Clique	A set of vertices, all of which are adjacent.
Ind. set	A set of vertices, none of which are adjacent.
Vertex cover	A set of vertices which cover all edges.
Planar graph	A graph which can be embeded in the plane.
Plane graph	An embedding of a planar graph.

$$\sum_{v \in V} \deg(v) = 2m.$$

If G is planar then $n - m + f = 2$, so

$$f \leq 2n - 4, \quad m \leq 3n - 6.$$

Any planar graph has a vertex with degree ≤ 5.

Notation:

$E(G)$	Edge set
$V(G)$	Vertex set
$c(G)$	Number of components
$G[S]$	Induced subgraph
$\deg(v)$	Degree of v
$\Delta(G)$	Maximum degree
$\delta(G)$	Minimum degree
$\chi(G)$	Chromatic number
$\chi_E(G)$	Edge chromatic number
G^c	Complement graph
K_n	Complete graph
K_{n_1,n_2}	Complete bipartite graph
$\mathrm{r}(k, \ell)$	Ramsey number

Geometry

Projective coordinates: triples (x, y, z), not all x, y and z zero.

$$(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$$

Cartesian	Projective
(x, y)	$(x, y, 1)$
$y = mx + b$	$(m, -1, b)$
$x = c$	$(1, 0, -c)$

Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$

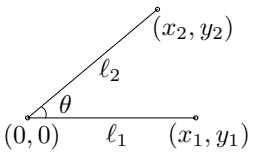
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p},$$

$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p}.$$

Area of triangle (x_0, y_0), (x_1, y_1) and (x_2, y_2):

$$\tfrac{1}{2} \operatorname{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$$

Angle formed by three points:



$$\cos\theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{\ell_1 \ell_2}.$$

Line through two points (x_0, y_0) and (x_1, y_1):

$$\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0.$$

Area of circle, volume of sphere:

$$A = \pi r^2, \qquad V = \tfrac{4}{3}\pi r^3.$$

If I have seen farther than others, it is because I have stood on the shoulders of giants.
– Issac Newton

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:

$$\pi = 2 \cdot \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdot 6 \cdot 6 \cdots}{1 \cdot 3 \cdot 3 \cdot 5 \cdot 5 \cdot 7 \cdots}$$

Brouncker's continued fraction expansion:

$$\frac{\pi}{4} = 1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{7^2}{2 + \cdots}}}}$$

Gregrory's series:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

Newton's series:

$$\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2 \cdot 3 \cdot 2^3} + \frac{1 \cdot 3}{2 \cdot 4 \cdot 5 \cdot 2^5} + \cdots$$

Sharp's series:

$$\frac{\pi}{6} = \frac{1}{\sqrt{3}}\left(1 - \frac{1}{3^1 \cdot 3} + \frac{1}{3^2 \cdot 5} - \frac{1}{3^3 \cdot 7} + \cdots\right)$$

Euler's series:

$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$$
$$\frac{\pi^2}{8} = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \cdots$$
$$\frac{\pi^2}{12} = \frac{1}{1^2} - \frac{1}{2^2} + \frac{1}{3^2} - \frac{1}{4^2} + \frac{1}{5^2} - \cdots$$

Partial Fractions

Let $N(x)$ and $D(x)$ be polynomial functions of x. We can break down $N(x)/D(x)$ using partial fraction expansion. First, if the degree of N is greater than or equal to the degree of D, divide N by D, obtaining

$$\frac{N(x)}{D(x)} = Q(x) + \frac{N'(x)}{D(x)},$$

where the degree of N' is less than that of D. Second, factor $D(x)$. Use the following rules: For a non-repeated factor:

$$\frac{N(x)}{(x-a)D(x)} = \frac{A}{x-a} + \frac{N'(x)}{D(x)},$$

where

$$A = \left[\frac{N(x)}{D(x)}\right]_{x=a}.$$

For a repeated factor:

$$\frac{N(x)}{(x-a)^m D(x)} = \sum_{k=0}^{m-1} \frac{A_k}{(x-a)^{m-k}} + \frac{N'(x)}{D(x)},$$

where

$$A_k = \frac{1}{k!}\left[\frac{d^k}{dx^k}\left(\frac{N(x)}{D(x)}\right)\right]_{x=a}.$$

The reasonable man adapts himself to the world; the unreasonable persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable. – George Bernard Shaw

Calculus

Derivatives:

1. $\frac{d(cu)}{dx} = c\frac{du}{dx}$, **2.** $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$, **3.** $\frac{d(uv)}{dx} = u\frac{dv}{dx} + v\frac{du}{dx}$,

4. $\frac{d(u^n)}{dx} = nu^{n-1}\frac{du}{dx}$, **5.** $\frac{d(u/v)}{dx} = \frac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$, **6.** $\frac{d(e^{cu})}{dx} = ce^{cu}\frac{du}{dx}$,

7. $\frac{d(c^u)}{dx} = (\ln c)c^u\frac{du}{dx}$, **8.** $\frac{d(\ln u)}{dx} = \frac{1}{u}\frac{du}{dx}$,

9. $\frac{d(\sin u)}{dx} = \cos u\frac{du}{dx}$, **10.** $\frac{d(\cos u)}{dx} = -\sin u\frac{du}{dx}$,

11. $\frac{d(\tan u)}{dx} = \sec^2 u\frac{du}{dx}$, **12.** $\frac{d(\cot u)}{dx} = \csc^2 u\frac{du}{dx}$,

13. $\frac{d(\sec u)}{dx} = \tan u\,\sec u\frac{du}{dx}$, **14.** $\frac{d(\csc u)}{dx} = -\cot u\,\csc u\frac{du}{dx}$,

15. $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}}\frac{du}{dx}$, **16.** $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}}\frac{du}{dx}$,

17. $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2}\frac{du}{dx}$, **18.** $\frac{d(\operatorname{arccot} u)}{dx} = \frac{-1}{1+u^2}\frac{du}{dx}$,

19. $\frac{d(\operatorname{arcsec} u)}{dx} = \frac{1}{u\sqrt{1-u^2}}\frac{du}{dx}$, **20.** $\frac{d(\operatorname{arccsc} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$,

21. $\frac{d(\sinh u)}{dx} = \cosh u\frac{du}{dx}$, **22.** $\frac{d(\cosh u)}{dx} = \sinh u\frac{du}{dx}$,

23. $\frac{d(\tanh u)}{dx} = \operatorname{sech}^2 u\frac{du}{dx}$, **24.** $\frac{d(\coth u)}{dx} = -\operatorname{csch}^2 u\frac{du}{dx}$,

25. $\frac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u\,\tanh u\frac{du}{dx}$, **26.** $\frac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u\,\coth u\frac{du}{dx}$,

27. $\frac{d(\operatorname{arcsinh} u)}{dx} = \frac{1}{\sqrt{1+u^2}}\frac{du}{dx}$, **28.** $\frac{d(\operatorname{arccosh} u)}{dx} = \frac{1}{\sqrt{u^2-1}}\frac{du}{dx}$,

29. $\frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2}\frac{du}{dx}$, **30.** $\frac{d(\operatorname{arccoth} u)}{dx} = \frac{1}{u^2-1}\frac{du}{dx}$,

31. $\frac{d(\operatorname{arcsech} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$, **32.** $\frac{d(\operatorname{arccsch} u)}{dx} = \frac{-1}{|u|\sqrt{1+u^2}}\frac{du}{dx}$.

Integrals:

1. $\int cu\,dx = c\int u\,dx$, **2.** $\int (u+v)\,dx = \int u\,dx + \int v\,dx$,

3. $\int x^n\,dx = \frac{1}{n+1}x^{n+1}, \quad n \neq -1$, **4.** $\int \frac{1}{x}dx = \ln x$, **5.** $\int e^x\,dx = e^x$,

6. $\int \frac{dx}{1+x^2} = \arctan x$, **7.** $\int u\frac{dv}{dx}dx = uv - \int v\frac{du}{dx}dx$,

8. $\int \sin x\,dx = -\cos x$, **9.** $\int \cos x\,dx = \sin x$,

10. $\int \tan x\,dx = -\ln|\cos x|$, **11.** $\int \cot x\,dx = \ln|\cos x|$,

12. $\int \sec x\,dx = \ln|\sec x + \tan x|$, **13.** $\int \csc x\,dx = \ln|\csc x + \cot x|$,

14. $\int \arcsin\frac{x}{a}dx = \arcsin\frac{x}{a} + \sqrt{a^2-x^2}, \quad a > 0$,

Theoretical Computer Science Cheat Sheet

Calculus Cont.

15. $\int \arccos\frac{x}{a}dx = \arccos\frac{x}{a} - \sqrt{a^2-x^2}, \quad a>0,$

16. $\int \arctan\frac{x}{a}dx = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2), \quad a>0,$

17. $\int \sin^2(ax)dx = \frac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big),$

18. $\int \cos^2(ax)dx = \frac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big),$

19. $\int \sec^2 x\,dx = \tan x,$

20. $\int \csc^2 x\,dx = -\cot x,$

21. $\int \sin^n x\,dx = -\frac{\sin^{n-1}x\cos x}{n} + \frac{n-1}{n}\int \sin^{n-2}x\,dx,$

22. $\int \cos^n x\,dx = \frac{\cos^{n-1}x\sin x}{n} + \frac{n-1}{n}\int \cos^{n-2}x\,dx,$

23. $\int \tan^n x\,dx = \frac{\tan^{n-1}x}{n-1} - \int \tan^{n-2}x\,dx, \quad n\neq 1,$

24. $\int \cot^n x\,dx = -\frac{\cot^{n-1}x}{n-1} - \int \cot^{n-2}x\,dx, \quad n\neq 1,$

25. $\int \sec^n x\,dx = \frac{\tan x\sec^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \sec^{n-2}x\,dx, \quad n\neq 1,$

26. $\int \csc^n x\,dx = -\frac{\cot x\csc^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \csc^{n-2}x\,dx, \quad n\neq 1,$

27. $\int \sinh x\,dx = \cosh x,$

28. $\int \cosh x\,dx = \sinh x,$

29. $\int \tanh x\,dx = \ln|\cosh x|,$

30. $\int \coth x\,dx = \ln|\sinh x|,$

31. $\int \operatorname{sech} x\,dx = \arctan\sinh x,$

32. $\int \operatorname{csch} x\,dx = \ln\left|\tanh\frac{x}{2}\right|,$

33. $\int \sinh^2 x\,dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x,$

34. $\int \cosh^2 x\,dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x,$

35. $\int \operatorname{sech}^2 x\,dx = \tanh x,$

36. $\int \operatorname{arcsinh}\frac{x}{a}dx = x\operatorname{arcsinh}\frac{x}{a} - \sqrt{x^2+a^2}, \quad a>0,$

37. $\int \operatorname{arctanh}\frac{x}{a}dx = x\operatorname{arctanh}\frac{x}{a} + \frac{a}{2}\ln|a^2-x^2|,$

38. $\int \operatorname{arccosh}\frac{x}{a}dx = \begin{cases} x\operatorname{arccosh}\dfrac{x}{a} - \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a} > 0 \text{ and } a>0, \\ x\operatorname{arccosh}\dfrac{x}{a} + \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a} < 0 \text{ and } a>0, \end{cases}$

39. $\int \frac{dx}{\sqrt{a^2+x^2}} = \ln\left(x+\sqrt{a^2+x^2}\right), \quad a>0,$

40. $\int \frac{dx}{a^2+x^2} = \frac{1}{a}\arctan\frac{x}{a}, \quad a>0,$

41. $\int \sqrt{a^2-x^2}\,dx = \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}, \quad a>0,$

42. $\int (a^2-x^2)^{3/2}dx = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2} + \frac{3a^4}{8}\arcsin\frac{x}{a}, \quad a>0,$

43. $\int \frac{dx}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a}, \quad a>0,$

44. $\int \frac{dx}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right|,$

45. $\int \frac{dx}{(a^2-x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2-x^2}},$

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

47. $\int \frac{dx}{\sqrt{x^2-a^2}} = \ln\left|x+\sqrt{x^2-a^2}\right|, \quad a>0,$

48. $\int \frac{dx}{ax^2+bx} = \frac{1}{a}\ln\left|\frac{x}{a+bx}\right|,$

49. $\int x\sqrt{a+bx}\,dx = \frac{2(3bx-2a)(a+bx)^{3/2}}{15b^2},$

50. $\int \frac{\sqrt{a+bx}}{x}\,dx = 2\sqrt{a+bx} + a\int \frac{1}{x\sqrt{a+bx}}dx,$

51. $\int \frac{x}{\sqrt{a+bx}}\,dx = \frac{1}{\sqrt{2}}\ln\left|\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right|, \quad a>0,$

52. $\int \frac{\sqrt{a^2-x^2}}{x}\,dx = \sqrt{a^2-x^2} - a\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|,$

53. $\int x\sqrt{a^2-x^2}\,dx = -\frac{1}{3}(a^2-x^2)^{3/2},$

54. $\int x^2\sqrt{a^2-x^2}\,dx = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2} + \frac{a^4}{8}\arcsin\frac{x}{a}, \quad a>0,$

55. $\int \frac{dx}{\sqrt{a^2-x^2}} = -\frac{1}{a}\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|,$

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

57. $\int \frac{x^2\,dx}{\sqrt{a^2-x^2}} = -\frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a,} \quad a>0,$

58. $\int \frac{\sqrt{a^2+x^2}}{x}\,dx = \sqrt{a^2+x^2} - a\ln\left|\frac{a+\sqrt{a^2+x^2}}{x}\right|,$

59. $\int \frac{\sqrt{x^2-a^2}}{x}\,dx = \sqrt{x^2-a^2} - a\arccos\frac{a}{|x|}, \quad a>0,$

60. $\int x\sqrt{x^2\pm a^2}\,dx = \frac{1}{3}(x^2\pm a^2)^{3/2},$

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

62. $\displaystyle\int \frac{dx}{x\sqrt{x^2-a^2}} = \tfrac{1}{a}\arccos\tfrac{a}{|x|}, \quad a>0,$

63. $\displaystyle\int \frac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x},$

64. $\displaystyle\int \frac{x\,dx}{\sqrt{x^2\pm a^2}} = \sqrt{x^2\pm a^2},$

65. $\displaystyle\int \frac{\sqrt{x^2\pm a^2}}{x^4}\,dx = \mp\frac{(x^2+a^2)^{3/2}}{3a^2x^3},$

66.
$$\int \frac{dx}{ax^2+bx+c} = \begin{cases} \dfrac{1}{\sqrt{b^2-4ac}}\ln\left|\dfrac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, & \text{if } b^2>4ac,\\[2ex] \dfrac{2}{\sqrt{4ac-b^2}}\arctan\dfrac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2<4ac, \end{cases}$$

67.
$$\int \frac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, & \text{if } a>0,\\[2ex] \dfrac{1}{\sqrt{-a}}\arcsin\dfrac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a<0, \end{cases}$$

68. $\displaystyle\int \sqrt{ax^2+bx+c}\,dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ax-b^2}{8a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$

69. $\displaystyle\int \frac{x\,dx}{\sqrt{ax^2+bx+c}} = \frac{\sqrt{ax^2+bx+c}}{a} - \frac{b}{2a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$

70.
$$\int \frac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{-1}{\sqrt{c}}\ln\left|\dfrac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, & \text{if } c>0,\\[2ex] \dfrac{1}{\sqrt{-c}}\arcsin\dfrac{bx+2c}{|x|\sqrt{b^2-4ac}}, & \text{if } c<0, \end{cases}$$

71. $\displaystyle\int x^3\sqrt{x^2+a^2}\,dx = (\tfrac{1}{3}x^2-\tfrac{2}{15}a^2)(x^2+a^2)^{3/2},$

72. $\displaystyle\int x^n\sin(ax)\,dx = -\tfrac{1}{a}x^n\cos(ax) + \tfrac{n}{a}\int x^{n-1}\cos(ax)\,dx,$

73. $\displaystyle\int x^n\cos(ax)\,dx = \tfrac{1}{a}x^n\sin(ax) - \tfrac{n}{a}\int x^{n-1}\sin(ax)\,dx,$

74. $\displaystyle\int x^n e^{ax}\,dx = \frac{x^n e^{ax}}{a} - \tfrac{n}{a}\int x^{n-1}e^{ax}\,dx,$

75. $\displaystyle\int x^n\ln(ax)\,dx = x^{n+1}\left(\frac{\ln(ax)}{n+1} - \frac{1}{(n+1)^2}\right),$

76. $\displaystyle\int x^n(\ln ax)^m\,dx = \frac{x^{n+1}}{n+1}(\ln ax)^m - \frac{m}{n+1}\int x^n(\ln ax)^{m-1}\,dx.$

$$\begin{aligned}
x^1 &= x^{\underline{1}} &&= x^{\overline{1}}\\
x^2 &= x^{\underline{2}} + x^{\underline{1}} &&= x^{\overline{2}} - x^{\overline{1}}\\
x^3 &= x^{\underline{3}} + 3x^{\underline{2}} + x^{\underline{1}} &&= x^{\overline{3}} - 3x^{\overline{2}} + x^{\overline{1}}\\
x^4 &= x^{\underline{4}} + 6x^{\underline{3}} + 7x^{\underline{2}} + x^{\underline{1}} &&= x^{\overline{4}} - 6x^{\overline{3}} + 7x^{\overline{2}} - x^{\overline{1}}\\
x^5 &= x^{\underline{5}} + 15x^{\underline{4}} + 25x^{\underline{3}} + 10x^{\underline{2}} + x^{\underline{1}} &&= x^{\overline{5}} - 15x^{\overline{4}} + 25x^{\overline{3}} - 10x^{\overline{2}} + x^{\overline{1}}
\end{aligned}$$

$$\begin{aligned}
x^{\overline{1}} &= x^1 & x^{\underline{1}} &= x^1\\
x^{\overline{2}} &= x^2 + x^1 & x^{\underline{2}} &= x^2 - x^1\\
x^{\overline{3}} &= x^3 + 3x^2 + 2x^1 & x^{\underline{3}} &= x^3 - 3x^2 + 2x^1\\
x^{\overline{4}} &= x^4 + 6x^3 + 11x^2 + 6x^1 & x^{\underline{4}} &= x^4 - 6x^3 + 11x^2 - 6x^1\\
x^{\overline{5}} &= x^5 + 10x^4 + 35x^3 + 50x^2 + 24x^1 & x^{\underline{5}} &= x^5 - 10x^4 + 35x^3 - 50x^2 + 24x^1
\end{aligned}$$

Finite Calculus

Difference, shift operators:

$$\Delta f(x) = f(x+1) - f(x),$$
$$\mathrm{E}\,f(x) = f(x+1).$$

Fundamental Theorem:

$$f(x) = \Delta F(x) \Leftrightarrow \sum f(x)\delta x = F(x) + C.$$
$$\sum_a^b f(x)\delta x = \sum_{i=a}^{b-1} f(i).$$

Differences:

$\Delta(cu) = c\Delta u, \qquad \Delta(u+v) = \Delta u + \Delta v,$

$\Delta(uv) = u\Delta v + \mathrm{E}\,v\Delta u,$

$\Delta(x^{\underline{n}}) = nx^{\underline{n-1}},$

$\Delta(H_x) = x^{\underline{-1}}, \qquad \Delta(2^x) = 2^x,$

$\Delta(c^x) = (c-1)c^x, \qquad \Delta\binom{x}{m} = \binom{x}{m-1}.$

Sums:

$\sum cu\,\delta x = c\sum u\,\delta x,$

$\sum(u+v)\,\delta x = \sum u\,\delta x + \sum v\,\delta x,$

$\sum u\Delta v\,\delta x = uv - \sum \mathrm{E}\,v\Delta u\,\delta x,$

$\sum x^{\underline{n}}\,\delta x = \frac{x^{\underline{n+1}}}{m+1}, \qquad \sum x^{\underline{-1}}\delta x = H_x,$

$\sum c^x\,\delta x = \frac{c^x}{c-1}, \qquad \sum\binom{x}{m}\,\delta x = \binom{x}{m+1}.$

Falling Factorial Powers:

$$x^{\underline{n}} = x(x-1)\cdots(x-n+1), \quad n>0,$$
$$x^{\underline{0}} = 1,$$
$$x^{\underline{n}} = \frac{1}{(x+1)\cdots(x+|n|)}, \quad n<0,$$
$$x^{\underline{n+m}} = x^{\underline{m}}(x-m)^{\underline{n}}.$$

Rising Factorial Powers:

$$x^{\overline{n}} = x(x+1)\cdots(x+n-1), \quad n>0,$$
$$x^{\overline{0}} = 1,$$
$$x^{\overline{n}} = \frac{1}{(x-1)\cdots(x-|n|)}, \quad n<0,$$
$$x^{\overline{n+m}} = x^{\overline{m}}(x+m)^{\overline{n}}.$$

Conversion:

$$x^{\underline{n}} = (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}} = 1/(x+1)^{\overline{-n}},$$
$$x^{\overline{n}} = (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}} = 1/(x-1)^{\underline{-n}},$$
$$x^n = \sum_{k=1}^n \left\{ n \atop k \right\} x^{\underline{k}} = \sum_{k=1}^n \left\{ n \atop k \right\}(-1)^{n-k}x^{\overline{k}},$$
$$x^{\underline{n}} = \sum_{k=1}^n \left[n \atop k \right](-1)^{n-k}x^k,$$
$$x^{\overline{n}} = \sum_{k=1}^n \left[n \atop k \right]x^k.$$

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \cdots = \sum_{i=0}^{\infty} \frac{(x-a)^i}{i!} f^{(i)}(a).$$

Expansions:

$$\begin{aligned}
\frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \cdots &&= \sum_{i=0}^{\infty} x^i, \\
\frac{1}{1-cx} &= 1 + cx + c^2x^2 + c^3x^3 + \cdots &&= \sum_{i=0}^{\infty} c^i x^i, \\
\frac{1}{1-x^n} &= 1 + x^n + x^{2n} + x^{3n} + \cdots &&= \sum_{i=0}^{\infty} x^{ni}, \\
\frac{x}{(1-x)^2} &= x + 2x^2 + 3x^3 + 4x^4 + \cdots &&= \sum_{i=0}^{\infty} ix^i, \\
x^k \frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) &= x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \cdots &&= \sum_{i=0}^{\infty} i^n x^i, \\
e^x &= 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots &&= \sum_{i=0}^{\infty} \frac{x^i}{i!}, \\
\ln(1+x) &= x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots &&= \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}, \\
\ln\frac{1}{1-x} &= x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots &&= \sum_{i=1}^{\infty} \frac{x^i}{i}, \\
\sin x &= x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!}, \\
\cos x &= 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}, \\
\tan^{-1} x &= x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots &&= \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)}, \\
(1+x)^n &= 1 + nx + \tfrac{n(n-1)}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{n}{i} x^i, \\
\frac{1}{(1-x)^{n+1}} &= 1 + (n+1)x + \tbinom{n+2}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{i+n}{i} x^i, \\
\frac{x}{e^x - 1} &= 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots &&= \sum_{i=0}^{\infty} \frac{B_i x^i}{i!}, \\
\frac{1}{2x}(1 - \sqrt{1-4x}) &= 1 + x + 2x^2 + 5x^3 + \cdots &&= \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i, \\
\frac{1}{\sqrt{1-4x}} &= 1 + x + 2x^2 + 6x^3 + \cdots &&= \sum_{i=0}^{\infty} \binom{2i}{i} x^i, \\
\frac{1}{\sqrt{1-4x}}\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n &= 1 + (2+n)x + \tbinom{4+n}{2}x^2 + \cdots &&= \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i, \\
\frac{1}{1-x}\ln\frac{1}{1-x} &= x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots &&= \sum_{i=1}^{\infty} H_i x^i, \\
\frac{1}{2}\left(\ln\frac{1}{1-x}\right)^2 &= \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots &&= \sum_{i=2}^{\infty} \frac{H_{i-1}x^i}{i}, \\
\frac{x}{1-x-x^2} &= x + x^2 + 2x^3 + 3x^4 + \cdots &&= \sum_{i=0}^{\infty} F_i x^i, \\
\frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2} &= F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots &&= \sum_{i=0}^{\infty} F_{ni} x^i.
\end{aligned}$$

Ordinary power series:

$$A(x) = \sum_{i=0}^{\infty} a_i x^i.$$

Exponential power series:

$$A(x) = \sum_{i=0}^{\infty} a_i \frac{x^i}{i!}.$$

Dirichlet power series:

$$A(x) = \sum_{i=1}^{\infty} \frac{a_i}{i^x}.$$

Binomial theorem:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k.$$

Difference of like powers:

$$x^n - y^n = (x-y)\sum_{k=0}^{n-1} x^{n-1-k} y^k.$$

For ordinary power series:

$$\begin{aligned}
\alpha A(x) + \beta B(x) &= \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i, \\
x^k A(x) &= \sum_{i=k}^{\infty} a_{i-k} x^i, \\
\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} &= \sum_{i=0}^{\infty} a_{i+k} x^i, \\
A(cx) &= \sum_{i=0}^{\infty} c^i a_i x^i, \\
A'(x) &= \sum_{i=0}^{\infty} (i+1) a_{i+1} x^i, \\
xA'(x) &= \sum_{i=1}^{\infty} i a_i x^i, \\
\int A(x)\,dx &= \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i, \\
\frac{A(x) + A(-x)}{2} &= \sum_{i=0}^{\infty} a_{2i} x^{2i}, \\
\frac{A(x) - A(-x)}{2} &= \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.
\end{aligned}$$

Summation: If $b_i = \sum_{j=0}^{i} a_i$ then

$$B(x) = \frac{1}{1-x}A(x).$$

Convolution:

$$A(x)B(x) = \sum_{i=0}^{\infty} \left(\sum_{j=0}^{i} a_j b_{i-j}\right) x^i.$$

God made the natural numbers;
all the rest is the work of man.
– Leopold Kronecker

Theoretical Computer Science Cheat Sheet

Series

Expansions:

$$\frac{1}{(1-x)^{n+1}}\ln\frac{1}{1-x} = \sum_{i=0}^{\infty}(H_{n+i}-H_n)\binom{n+i}{i}x^i, \qquad \left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty}\left\{ {i \atop n} \right\}x^i,$$

$$x^{\overline{n}} = \sum_{i=0}^{\infty}\left[{n \atop i} \right]x^i, \qquad (e^x-1)^n = \sum_{i=0}^{\infty}\left\{ {i \atop n} \right\}\frac{n!x^i}{i!},$$

$$\left(\ln\frac{1}{1-x}\right)^n = \sum_{i=0}^{\infty}\left[{i \atop n} \right]\frac{n!x^i}{i!}, \qquad x\cot x = \sum_{i=0}^{\infty}\frac{(-4)^iB_{2i}x^{2i}}{(2i)!},$$

$$\tan x = \sum_{i=1}^{\infty}(-1)^{i-1}\frac{2^{2i}(2^{2i}-1)B_{2i}x^{2i-1}}{(2i)!}, \qquad \zeta(x) = \sum_{i=1}^{\infty}\frac{1}{i^x},$$

$$\frac{1}{\zeta(x)} = \sum_{i=1}^{\infty}\frac{\mu(i)}{i^x}, \qquad \frac{\zeta(x-1)}{\zeta(x)} = \sum_{i=1}^{\infty}\frac{\phi(i)}{i^x},$$

$$\zeta(x) = \prod_p \frac{1}{1-p^{-x}},$$

$$\zeta^2(x) = \sum_{i=1}^{\infty}\frac{d(i)}{x^i} \quad \text{where } d(n)=\textstyle\sum_{d|n}1,$$

$$\zeta(x)\zeta(x-1) = \sum_{i=1}^{\infty}\frac{S(i)}{x^i} \quad \text{where } S(n)=\textstyle\sum_{d|n}d,$$

$$\zeta(2n) = \frac{2^{2n-1}|B_{2n}|}{(2n)!}\pi^{2n}, \quad n\in\mathbb{N},$$

$$\frac{x}{\sin x} = \sum_{i=0}^{\infty}(-1)^{i-1}\frac{(4^i-2)B_{2i}x^{2i}}{(2i)!},$$

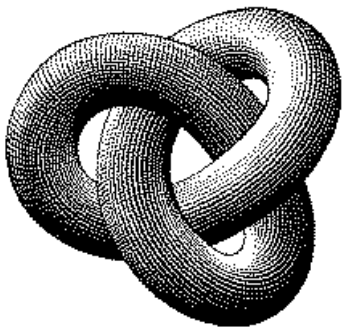
$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty}\frac{n(2i+n-1)!}{i!(n+i)!}x^i,$$

$$e^x\sin x = \sum_{i=1}^{\infty}\frac{2^{i/2}\sin\frac{i\pi}{4}}{i!}x^i,$$

$$\sqrt{\frac{1-\sqrt{1-x}}{x}} = \sum_{i=0}^{\infty}\frac{(4i)!}{16^i\sqrt{2}(2i)!(2i+1)!}x^i,$$

$$\left(\frac{\arcsin x}{x}\right)^2 = \sum_{i=0}^{\infty}\frac{4^ii!^2}{(i+1)(2i+1)!}x^{2i}.$$

Escher's Knot



Stieltjes Integration

If G is continuous in the interval $[a,b]$ and F is nondecreasing then

$$\int_a^b G(x)\,dF(x)$$

exists. If $a\le b\le c$ then

$$\int_a^c G(x)\,dF(x) = \int_a^b G(x)\,dF(x) + \int_b^c G(x)\,dF(x).$$

If the integrals involved exist

$$\int_a^b \big(G(x)+H(x)\big)\,dF(x) = \int_a^b G(x)\,dF(x) + \int_a^b H(x)\,dF(x),$$

$$\int_a^b G(x)\,d\big(F(x)+H(x)\big) = \int_a^b G(x)\,dF(x) + \int_a^b G(x)\,dH(x),$$

$$\int_a^b c\cdot G(x)\,dF(x) = \int_a^b G(x)\,d\big(c\cdot F(x)\big) = c\int_a^b G(x)\,dF(x),$$

$$\int_a^b G(x)\,dF(x) = G(b)F(b) - G(a)F(a) - \int_a^b F(x)\,dG(x).$$

If the integrals involved exist, and F possesses a derivative F' at every point in $[a,b]$ then

$$\int_a^b G(x)\,dF(x) = \int_a^b G(x)F'(x)\,dx.$$

Cramer's Rule

If we have equations:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ \vdots \qquad & \quad \vdots \qquad \quad \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n \end{aligned}$$

Let $A=(a_{i,j})$ and B be the column matrix (b_i). Then there is a unique solution iff $\det A\neq 0$. Let A_i be A with column i replaced by B. Then

$$x_i = \frac{\det A_i}{\det A}.$$

Improvement makes strait roads, but the crooked roads without Improvement, are roads of Genius.
– William Blake (The Marriage of Heaven and Hell)

00	47	18	76	29	93	85	34	61	52
86	11	57	28	70	39	94	45	02	63
95	80	22	67	38	71	49	56	13	04
59	96	81	33	07	48	72	60	24	15
73	69	90	82	44	17	58	01	35	26
68	74	09	91	83	55	27	12	46	30
37	08	75	19	92	84	66	23	50	41
14	25	36	40	51	62	03	77	88	99
21	32	43	54	65	06	10	89	97	78
42	53	64	05	16	20	31	98	79	87

The Fibonacci number system: Every integer n has a unique representation

$$n = F_{k_1} + F_{k_2} + \cdots + F_{k_m},$$

where $k_i \ge k_{i+1}+2$ for all i, $1\le i<m$ and $k_m\ge 2$.

Fibonacci Numbers

$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$

Definitions:

$$F_i = F_{i-1}+F_{i-2}, \quad F_0 = F_1 = 1,$$

$$F_{-i} = (-1)^{i-1}F_i,$$

$$F_i = \tfrac{1}{\sqrt{5}}\left(\phi^i - \hat{\phi}^i\right),$$

Cassini's identity: for $i>0$:

$$F_{i+1}F_{i-1} - F_i^2 = (-1)^i.$$

Additive rule:

$$F_{n+k} = F_kF_{n+1} + F_{k-1}F_n,$$

$$F_{2n} = F_nF_{n+1} + F_{n-1}F_n.$$

Calculation by matrices:

$$\begin{pmatrix} F_{n-2} & F_{n-1} \\ F_{n-1} & F_n \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n.$$