代码库

Blazar

2017 年 10 月 15 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

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

1.2 莫比乌斯反演

```cpp
#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];
```

```
34              }
35          }
36  }
37  int sum[100001];
38  //找 [1,n],[1,m] 内互质的数的对数
39  inline long long solve(int n,int m)
40  {
41      long long ans=0;
42      if(n>m)
43          swap(n,m);
44      int i,la=0;
45      for(i=1;i<=n;i=la+1)
46      {
47          la=min(n/(n/i),m/(m/i));
48          ans+=(long long)(sum[la]-sum[i-1])*(n/i)*(m/i);
49      }
50      return ans;
51  }
52  int main()
53  {
54      //freopen("b.in","r",stdin);
55      // freopen("b.out","w",stdout);
56      findmu();
57      sum[0]=0;
58      int i;
59      for(i=1;i<=100000;i++)
60          sum[i]=sum[i-1]+mu[i];
61      int a,b,c,d,k;
62      int T;
63      scanf("%d",&T);
64      while(T--)
65      {
66          scanf("%d%d%d%d%d",&a,&b,&c,&d,&k);
67          long long ans=0;
68          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);
69          printf("%lld\n",ans);
70      }
71      return 0;
72  }
```

1.3 扩展欧几里德算法

返回结果：

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

时间复杂度：$\mathcal{O}(nlogn)$

```
1  LL exgcd(LL a,LL b,LL &x,LL &y){
2    if(!b){
3      x=1;y=0;return a;
```

```
4    }else{
5      LL d=exgcd(b,a%b,x,y);
6      LL t=x;x=y;y=t-a/b*y;
7      return d;
8    }
9  }
```

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

```
1  LL china(int n,int *a,int *m){
2    LL M=1,d,x=0,y;
3    for(int i=0;i<n;i++)
4      M*=m[i];
5    for(int i=0;i<n;i++){
6      LL w=M/m[i];
7      d=exgcd(m[i],w,d,y);
8      y=(y%M+M)%M;
9      x=(x+y*w%M*a[i])%M;
10   }
11   while(x<0)x+=M;
12   return x;
13 }
```

1.5 中国剩余定理 2

```
1  //merge Ax=B and ax=b to A'x=B'
2  void merge(LL &A,LL &B,LL a,LL b){
3      LL x,y;
4      sol(A,-a,b-B,x,y);
5      A=lcm(A,a);
6      B=(a*y+b)%A;
7      B=(B+A)%A;
8  }
```

1.6 组合数取模

```
1  LL prod=1,P;
2  pair<LL,LL> comput(LL n,LL p,LL k){
3      if(n<=1)return make_pair(0,1);
4      LL ans=1,cnt=0;
5      ans=pow(prod,n/P,P);
6      cnt=n/p;
7      pair<LL,LL>res=comput(n/p,p,k);
```

```
8      cnt+=res.first;
9      ans=ans*res.second%P;
10     for(int i=n-n%P+1;i<=n;i++)if(i%p){
11
12         ans=ans*i%P;
13     }
14     return make_pair(cnt,ans);
15 }
16 pair<LL,LL> calc(LL n,LL p,LL k){
17     prod=1;P=pow(p,k,1e18);
18     for(int i=1;i<P;i++)if(i%p)prod=prod*i%P;
19     pair<LL,LL> res=comput(n,p,k);
20 //  res.second=res.second*pow(p,res.first%k,P)%P;
21 //  res.first-=res.first%k;
22     return res;
23 }
24 LL calc(LL n,LL m,LL p,LL k){
25     pair<LL,LL>A,B,C;
26     LL P=pow(p,k,1e18);
27     A=calc(n,p,k);
28     B=calc(m,p,k);
29     C=calc(n-m,p,k);
30     LL ans=1;
31     ans=pow(p,A.first-B.first-C.first,P);
32     ans=ans*A.second%P*inv(B.second,P)%P*inv(C.second,P)%P;
33     return ans;
34 }
```

1.7 扩展小步大步

```
1  LL solve2(LL a,LL b,LL p){
2      //a^x=b (mod p)
3      b%=p;
4      LL e=1%p;
5      for(int i=0;i<100;i++){
6          if(e==b)return i;
7          e=e*a%p;
8      }
9      int r=0;
10     while(gcd(a,p)!=1){
11         LL d=gcd(a,p);
12         if(b%d)return -1;
13         p/=d;b/=d;b=b*inv(a/d,p);
14         r++;
15     }LL res=BSGS(a,b,p);
16     if(res==-1)return -1;
17     return res+r;
18 }
```

1.8 卢卡斯定理

```
1  LL Lucas(LL n,LL m,LL p){
2    LL ans=1;
3    while(n&&m){
4      LL a=n%p,b=m%p;
5      if(a<b)return 0;
6      ans=(ans*C(a,b,p))%p;
7      n/=p;m/=p;
8    }return ans%p;
9  }
```

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: $\mathcal{O}(\sqrt{n})$

```
1   LL BSGS(LL a,LL b,LL p){
2     LL m=sqrt(p)+.5,v=inv(pw(a,m,p),p),e=1;
3     map<LL,LL>hash;hash[1]=0;
4     for(int i=1;i<m;i++)
5       e=e*a%p,hash[e]=i;
6     for(int i=0;i<=m;i++){
7       if(hash.count(b))return i*m+hash[b];
8       b=b*v%p;
9     }return -1;
10  }
```

1.10 Miller Rabin 素数测试

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

```
18        return false;
19    for(int i=0;i<12&&BASE[i]<n;i++){
20        if(!check(n,BASE[i]))
21            return false;
22    }
23    return true;
24 }
```

1.11 Pollard Rho 大数分解

时间复杂度：$\mathcal{O}(n^{1/4})$

```
1  LL prho(LL n,LL c){
2    LL i=1,k=2,x=rand()%(n-1)+1,y=x;
3    while(1){
4      i++;x=(x*x%n+c)%n;
5      LL d=__gcd((y-x+n)%n,n);
6      if(d>1&&d<n)return d;
7      if(y==x)return n;
8      if(i==k)y=x,k<<=1;
9    }
10 }
11 void factor(LL n,vector<LL>&fat){
12   if(n==1)return;
13   if(isprime(n)){
14     fat.push_back(n);
15     return;
16   }LL p=n;
17   while(p>=n)p=prho(p,rand()%(n-1)+1);
18   factor(p,fat);
19   factor(n/p,fat);
20 }
```

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：$\mathcal{O}(nlogn)$

```
1  /*
2  {(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
3        ,(377487361,7),(998244353,3),(1224736769,3)
4        ,(1300234241,3),(1484783617,5)}
5  */
6  int mo=998244353,G=3;
7  void NTT(int a[],int n,int f){
8    for(register int i=0;i<n;i++)
```

```
9          if(i<rev[i])
10            swap(a[i],a[rev[i]]);
11      for (register int i=2;i<=n;i<<=1){
12        static int exp[maxn];
13        exp[0]=1;exp[1]=pw(G,(mo-1)/i);
14        if(f==-1)exp[1]=pw(exp[1],mo-2);
15        for(register int k=2;k<(i>>1);k++)
16          exp[k]=1LL*exp[k-1]*exp[1]%mo;
17        for(register int j=0;j<n;j+=i){
18          for(register int k=0;k<(i>>1);k++){
19            register int &pA=a[j+k],&pB=a[j+k+(i>>1)];
20            register int A=pA,B=1LL*pB*exp[k]%mo;
21            pA=(A+B)%mo;
22            pB=(A-B+mo)%mo;
23          }
24        }
25      }
26      if(f==-1){
27        int rv=pw(n,mo-2)%mo;
28        for(int i=0;i<n;i++)
29          a[i]=1LL*a[i]*rv%mo;
30      }
31    }
32    void mul(int m,int a[],int b[],int c[]){
33      int n=1,len=0;
34      while(n<m)n<<=1,len++;
35      for (int i=1;i<n;i++)
36        rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
37      NTT(a,n,1);
38      NTT(b,n,1);
39      for(int i=0;i<n;i++)
40        c[i]=1LL*a[i]*b[i]%mo;
41      NTT(c,n,-1);
42    }
```

1.13　原根

```
1    vector<LL>fct;
2    bool check(LL x,LL g){
3      for(int i=0;i<fct.size();i++)
4        if(pw(g,(x-1)/fct[i],x)==1)
5          return 0;
6      return 1;
7    }
8    LL findrt(LL x){
9      LL tmp=x-1;
10     for(int i=2;i*i<=tmp;i++){
11       if(tmp%i==0){
12         fct.push_back(i);
```

```
13          while(tmp%i==0)tmp/=i;
14      }
15  }if(tmp>1)fct.push_back(tmp);
16  // x is 1,2,4,p^n,2p^n
17  // x has phi(phi(x)) primitive roots
18  for(int i=2;i<int(1e9);i++)if(check(x,i))
19      return i;
20      return -1;
21  }
22  const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
23  bool check(long long n,int base) {
24      long long n2=n-1,res;
25      int s=0;
26      while(n2%2==0) n2>>=1,s++;
27      res=pw(base,n2,n);
28      if((res==1)||(res==n-1)) return 1;
29      while(s--) {
30          res=mul(res,res,n);
31          if(res==n-1) return 1;
32      }
33      return 0; // n is not a strong pseudo prime
34  }
35  bool isprime(const long long &n) {
36      if(n==2)
37          return true;
38      if(n<2 || n%2==0)
39          return false;
40      for(int i=0;i<12&&BASE[i]<n;i++){
41          if(!check(n,BASE[i]))
42              return false;
43      }
44      return true;
45  }
```

1.14 线性递推

```
1  //已知 a_0, a_1, ..., a_{m-1}\\
2    a_n = c_0 * a_{n-m} + ... + c_{m-1} * a_{n-1}\\
3    求 a_n = v_0 * a_0 + v_1 * a_1 + ... + v_{m-1} * a_{m-1}\\
4
5  void linear_recurrence(long long n, int m, int a[], int c[], int p) {
6    long long v[M] = {1 % p}, u[M << 1], msk = !!n;
7    for(long long i(n); i > 1; i >>= 1) {
8      msk <<= 1;
9    }
10   for(long long x(0); msk; msk >>= 1, x <<= 1) {
11     fill_n(u, m << 1, 0);
12     int b(!!(n & msk));
13     x |= b;
```

```
14      if(x < m) {
15        u[x] = 1 % p;
16      }else {
17        for(int i(0); i < m; i++) {
18          for(int j(0), t(i + b); j < m; j++, t++) {
19            u[t] = (u[t] + v[i] * v[j]) % p;
20          }
21        }
22        for(int i((m << 1) - 1); i >= m; i--) {
23          for(int j(0), t(i - m); j < m; j++, t++) {
24            u[t] = (u[t] + c[j] * u[i]) % p;
25          }
26        }
27      }
28      copy(u, u + m, v);
29    }
30    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
31    for(int i(m); i < 2 * m; i++) {
32      a[i] = 0;
33      for(int j(0); j < m; j++) {
34        a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
35      }
36    }
37    for(int j(0); j < m; j++) {
38      b[j] = 0;
39      for(int i(0); i < m; i++) {
40        b[j] = (b[j] + v[i] * a[i + j]) % p;
41      }
42    }
43    for(int j(0); j < m; j++) {
44      a[j] = b[j];
45    }
46  }
```

1.15   线性筛

```
1   void sieve(){
2     f[1]=mu[1]=phi[1]=1;
3     for(int i=2;i<maxn;i++){
4       if(!minp[i]){
5         minp[i]=i;
6         minpw[i]=i;
7         mu[i]=-1;
8         phi[i]=i-1;
9         f[i]=i-1;
10        p[++p[0]]=i;//Case 1 prime
11      }
12      for(int j=1;j<=p[0]&&(LL)i*p[j]<maxn;j++){
13        minp[i*p[j]]=p[j];
```

```
14        if(i%p[j]==0){
15            //Case 2 not coprime
16            minpw[i*p[j]]=minpw[i]*p[j];
17            phi[i*p[j]]=phi[i]*p[j];
18            mu[i*p[j]]=0;
19            if(i==minpw[i]){
20                f[i*p[j]]=i*p[j]-i;//Special Case for f(p^k)
21            }else{
22                f[i*p[j]]=f[i/minpw[i]]*f[minpw[i]*p[j]];
23            }
24            break;
25        }else{
26            //Case 3 coprime
27            minpw[i*p[j]]=p[j];
28            f[i*p[j]]=f[i]*f[p[j]];
29            phi[i*p[j]]=phi[i]*(p[j]-1);
30            mu[i*p[j]]=-mu[i];
31        }
32    }
33  }
34 }
```

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：$\mathcal{O}(nlogn)$

```
1  //calc \sum_{i=0}^{n-1} [(a+bi)/m]
2  // n,a,b,m >0
3  LL solve(LL n,LL a,LL b,LL m){
4      if(b==0)
5          return n*(a/m);
6      if(a>=m || b>=m)
7          return n*(a/m)+(n-1)*n/2*(b/m)+solve(n,a%m,b%m,m);
8      return solve((a+b*n)/m,(a+b*n)%m,m,b);
9  }
```

2 数值

2.1 高斯消元

```
1  void Gauss(){
2    int r,k;
3    for(int i=0;i<n;i++){
4      r=i;
```

```cpp
    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
```

```
54     int r,now=-1;
55     int ans=0;
56     for(int i = 0; i <n; i++){
57       r = now + 1;
58       for(int j = now + 1; j < m; j++)
59         if(fabs(A[j][i]) > fabs(A[r][i]))
60           r = j;
61       if (!sgn(A[r][i])) continue;
62       ans++;
63       now++;
64       if(r != now)
65         for(int j = 0; j < n; j++)
66           swap(A[r][j], A[now][j]);
67
68       for(int k = now + 1; k < m; k++){
69         double t = A[k][i] / A[now][i];
70         for(int j = 0; j < n; j++){
71           A[k][j] -= t * A[now][j];
72         }
73       }
74     }
75     return ans;
76   }
```

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：$\mathcal{O}(nlogn)$

```
1   typedef complex<double> cp;
2   const double pi = acos(-1);
3   void FFT(vector<cp>&num,int len,int ty){
4       for(int i=1,j=0;i<len-1;i++){
5           for(int k=len;j^=k>>=1,~j&k;);
6           if(i<j)
7               swap(num[i],num[j]);
8       }
9       for(int h=0;(1<<h)<len;h++){
10          int step=1<<h,step2=step<<1;
11          cp w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
12          for(int i=0;i<len;i+=step2){
13              cp w(1,0);
14              for(int j=0;j<step;j++){
15                  cp &x=num[i+j+step];
16                  cp &y=num[i+j];
17                  cp d=w*x;
18                  x=y-d;
```

```
19              y=y+d;
20              w=w*w0;
21          }
22      }
23  }
24  if(ty==-1)
25      for(int i=0;i<len;i++)
26          num[i]=cp(num[i].real()/(double)len,num[i].imag());
27  }
28  vector<cp> mul(vector<cp>a,vector<cp>b){
29      int len=a.size()+b.size();
30      while((len&-len)!=len)len++;
31      while(a.size()<len)a.push_back(cp(0,0));
32      while(b.size()<len)b.push_back(cp(0,0));
33      FFT(a,len,1);
34      FFT(b,len,1);
35      vector<cp>ans(len);
36      for(int i=0;i<len;i++)
37          ans[i]=a[i]*b[i];
38      FFT(ans,len,-1);
39      return ans;
40  }
```

2.3   1e9+7 FFT

```
1  // double 精度对 10^9 + 7 取模最多可以做到 2^20
2  const int MOD = 1000003;
3  const double PI = acos(-1);
4  typedef complex<double> Complex;
5  const int N = 65536, L = 15, MASK = (1 << L) - 1;
6  Complex w[N];
7  void FFTInit() {
8    for (int i = 0; i < N; ++i)
9      w[i] = Complex(cos(2 * i * PI / N), sin(2 * i * PI / N));
10  }
11  void FFT(Complex p[], int n) {
12    for (int i = 1, j = 0; i < n - 1; ++i) {
13      for (int s = n; j ^= s >>= 1, ~j & s;);
14      if (i < j) swap(p[i], p[j]);
15    }
16    for (int d = 0; (1 << d) < n; ++d) {
17      int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
18      for (int i = 0; i < n; i += m2) {
19        for (int j = 0; j < m; ++j) {
20          Complex &p1 = p[i + j + m], &p2 = p[i + j];
21          Complex t = w[rm * j] * p1;
22          p1 = p2 - t, p2 = p2 + t;
23        } } }
24  }
```

```cpp
Complex A[N], B[N], C[N], D[N];
void mul(int a[N], int b[N]) {
  for (int i = 0; i < N; ++i) {
    A[i] = Complex(a[i] >> L, a[i] & MASK);
    B[i] = Complex(b[i] >> L, b[i] & MASK);
  }
  FFT(A, N), FFT(B, N);
  for (int i = 0; i < N; ++i) {
    int j = (N - i) % N;
    Complex da = (A[i] - conj(A[j])) * Complex(0, -0.5),
        db = (A[i] + conj(A[j])) * Complex(0.5, 0),
        dc = (B[i] - conj(B[j])) * Complex(0, -0.5),
        dd = (B[i] + conj(B[j])) * Complex(0.5, 0);
    C[j] = da * dd + da * dc * Complex(0, 1);
    D[j] = db * dd + db * dc * Complex(0, 1);
  }
  FFT(C, N), FFT(D, N);
  for (int i = 0; i < N; ++i) {
    long long da = (long long)(C[i].imag() / N + 0.5) % MOD,
        db = (long long)(C[i].real() / N + 0.5) % MOD,
        dc = (long long)(D[i].imag() / N + 0.5) % MOD,
        dd = (long long)(D[i].real() / N + 0.5) % MOD;
    a[i] = ((dd << (L * 2)) + ((db + dc) << L) + da) % MOD;
  }
}
```

2.4   单纯形法求解线性规划

返回结果：

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

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

```cpp
vector<double> solve(vector<vector<double> >A,vector<double>B,vector<double>C){
    n=C.size();
    m=B.size();
    for(int i=0;i<C.size();i++)
        a[0][i+1]=C[i];
    for(int i=0;i<B.size();i++)
        a[i+1][0]=B[i];

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

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

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

```
69    }
```

2.5 自适应辛普森

```
1  double area(const double &left, const double &right) {
2      double mid = (left + right) / 2;
3      return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
4  }
5
6  double simpson(const double &left, const double &right,
7                 const double &eps, const double &area_sum) {
8      double mid = (left + right) / 2;
9      double area_left = area(left, mid);
10     double area_right = area(mid, right);
11     double area_total = area_left + area_right;
12     if (std::abs(area_total - area_sum) < 15 * eps) {
13         return area_total + (area_total - area_sum) / 15;
14     }
15     return simpson(left, mid, eps / 2, area_left)
16          + simpson(mid, right, eps / 2, area_right);
17 }
18
19 double simpson(const double &left, const double &right, const double &eps) {
20     return simpson(left, right, eps, area(left, right));
21 }
```

2.6 多项式求根

```
1  const double eps=1e-12;
2  double a[10][10];
3  typedef vector<double> vd;
4  int sgn(double x) { return x < -eps ? -1 : x > eps; }
5  double mypow(double x,int num){
6    double ans=1.0;
7    for(int i=1;i<=num;++i)ans*=x;
8    return ans;
9  }
10 double f(int n,double x){
11   double ans=0;
12   for(int i=n;i>=0;--i)ans+=a[n][i]*mypow(x,i);
13   return ans;
14 }
15 double getRoot(int n,double l,double r){
16   if(sgn(f(n,l))==0)return l;
17   if(sgn(f(n,r))==0)return r;
18   double temp;
19   if(sgn(f(n,l))>0)temp=-1;else temp=1;
20   double m;
21   for(int i=1;i<=10000;++i){
```

```
22    m=(l+r)/2;
23    double mid=f(n,m);
24    if(sgn(mid)==0){
25      return m;
26    }
27    if(mid*temp<0)l=m;else r=m;
28    }
29    return (l+r)/2;
30  }
31  vd did(int n){
32    vd ret;
33    if(n==1){
34      ret.push_back(-1e10);
35      ret.push_back(-a[n][0]/a[n][1]);
36      ret.push_back(1e10);
37      return ret;
38    }
39    vd mid=did(n-1);
40    ret.push_back(-1e10);
41    for(int i=0;i+1<mid.size();++i){
42      int t1=sgn(f(n,mid[i])),t2=sgn(f(n,mid[i+1]));
43      if(t1*t2>0)continue;
44      ret.push_back(getRoot(n,mid[i],mid[i+1]));
45    }
46    ret.push_back(1e10);
47    return ret;
48  }
49  int main(){
50    int n; scanf("%d",&n);
51    for(int i=n;i>=0;--i){
52      scanf("%lf",&a[n][i]);
53    }
54    for(int i=n-1;i>=0;--i)
55      for(int j=0;j<=i;++j)a[i][j]=a[i+1][j+1]*(j+1);
56    vd ans=did(n);
57    sort(ans.begin(),ans.end());
58    for(int i=1;i+1<ans.size();++i)printf("%.10f\n",ans[i]);
59    return 0;
60  }
```

3 快速求逆

```
1  long long inverse(const long long &x, const long long &mod) {
2      if (x == 1) {
3          return 1;
4      } else {
5          return (mod - mod / x) * inverse(mod % x, mod) % mod;
6      }
7  }
```

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。

```
1  int prepare(int n) {
2    int len = 1;
3    for (; len <= 2 * n; len <<= 1);
4    for (int i = 0; i < len; i++) {
5      e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
6      e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
7    }
8    return len;
9  }
10 void DFT(Complex *a, int n, int f) {
11   for (int i = 0, j = 0; i < n; i++) {
12     if (i > j) std::swap(a[i], a[j]);
13     for (int t = n >> 1; (j ^= t) < t; t >>= 1);
14   }
15   for (int i = 2; i <= n; i <<= 1)
16     for (int j = 0; j < n; j += i)
17       for (int k = 0; k < (i >> 1); k++) {
18         Complex A = a[j + k];
19         Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
20         a[j + k] = A + B;
21         a[j + k + (i >> 1)] = A - B;
22       }
23   if (f == 1) {
24     for (int i = 0; i < n; i++)
25       a[i].a /= n;
26   }
27 }
```

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。

```
1  // meminit(A, l, r) 是将数组 A 的 [l, r) 清 0。
2  // memcopy(target, source, l, r) 是将 source 的 [l, r) 复制到 target 的 [l, r)
3  #define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
4  #define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
5  void DFT(int *a, int n, int f) { // 封闭形式，常数小 (10^7 跑 2.23 秒)
6    for (register int i = 0, j = 0; i < n; i++) {
7      if (i > j) std::swap(a[i], a[j]);
8      for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
9    }
10   for (register int i = 2; i <= n; i <<= 1) {
11     static int exp[MAXN];
12     exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i);
```

```
13        if (f == 1) exp[1] = fpm(exp[1], MOD - 2);
14        for (register int k = 2; k < (i >> 1); k++) {
15          exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
16        }
17        for (register int j = 0; j < n; j += i) {
18          for (register int k = 0; k < (i >> 1); k++) {
19            register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
20            register int A = pA, B = 1ll * pB * exp[k] % MOD;
21            pA = (A + B) % MOD;
22            pB = (A - B + MOD) % MOD;
23          }
24        }
25      }
26      if (f == 1) {
27        register int rev = fpm(n, MOD - 2, MOD);
28        for (register int i = 0; i < n; i++) {
29          a[i] = 1ll * a[i] * rev % MOD;
30        }
31      }
32    }
33    // 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
34    // 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
35    int CRT(int *a) {
36      static int x[3];
37      for (int i = 0; i < 3; i++) {
38        x[i] = a[i];
39        for (int j = 0; j < i; j++) {
40          int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
41          if (t < 0) t += FFT[i] -> MOD;
42          x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
43        }
44      }
45      int sum = 1, ret = x[0] % MOD;
46      for (int i = 1; i < 3; i ++) {
47        sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
48        ret += 1LL * x[i] * sum % MOD;
49        if(ret >= MOD) ret -= MOD;
50      }
51      return ret;
52    }
53    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
54      for (int j = 0; j < 3; j++)
55        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：

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

注意事项：

1. $F(x)$ 的常数项系数必然不为 `0`，否则没有逆元；

2. 复杂度是 $O(n \log n)$ 但是常数比较大（10^5 大概需要 `0.3` 秒左右）；

3. 传入的两个数组必须不同，但传入的次数界没有必要是 `2` 的次幂；

```
1  void getInv(int *a, int *b, int n) {
2    static int tmp[MAXN];
3    b[0] = fpm(a[0], MOD - 2, MOD);
4    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
5      for (; M <= 3 * (c - 1); M <<= 1);
6      meminit(b, c, M);
7      meminit(tmp, c, M);
8      memcopy(tmp, a, 0, c);
9      DFT(tmp, M, 0);
10     DFT(b, M, 0);
11     for (int i = 0; i < M; i++) {
12       b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] * b[i] % MOD + MOD) % MOD;
13     }
14     DFT(b, M, 1);
15     meminit(b, c, M);
16   }
17 }
```

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。

```cpp
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);
```

```
32    }
33    void getExp(int *a, int *b, int n) {
34      static int ln[MAXN], tmp[MAXN];
35      b[0] = 1;
36      for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
37        for (; M <= 2 * (c - 1); M <<= 1);
38        int bound = std::min(c, n);
39        memcopy(tmp, a, 0, bound);
40        meminit(tmp, bound, M);
41        meminit(b, c, M);
42        getLn(b, ln, c);
43        meminit(ln, c, M);
44        DFT(b, M, 0);
45        DFT(tmp, M, 0);
46        DFT(ln, M, 0);
47        for (int i = 0; i < M; i++) {
48          b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + MOD) % MOD;
49        }
50        DFT(b, M, 1);
51        meminit(b, c, M);
52      }
53    }
```

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。

```
1    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
2      static int M, tA[MAXN], tB[MAXN], inv[MAXN], tD[MAXN];
3      for (; n > 0 && a[n - 1] == 0; n--);
4      for (; m > 0 && b[m - 1] == 0; m--);
5      for (int i = 0; i < n; i++) tA[i] = a[n - i - 1];
6      for (int i = 0; i < m; i++) tB[i] = b[m - i - 1];
7      for (M = 1; M <= n - m + 1; M <<= 1);
8      meminit(tB, m, M);
9      getInv(tB, inv, M);
10     for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
11     meminit(inv, n - m + 1, M);
12     meminit(tA, n - m + 1, M);
13     DFT(inv, M, 0);
14     DFT(tA, M, 0);
```

```
15    for (int i = 0; i < M; i++) {
16      d[i] = 1ll * inv[i] * tA[i] % MOD;
17    }
18    DFT(d, M, 1);
19    std::reverse(d, d + n - m + 1);
20    for (M = 1; M <= n; M <<= 1);
21    memcopy(tB, b, 0, m); meminit(tB, m, M);
22    memcopy(tD, d, 0, n - m + 1); meminit(tD, n - m + 1, M);
23    DFT(tD, M, 0);
24    DFT(tB, M, 0);
25    for (int i = 0; i < M; i++) {
26      r[i] = 1ll * tD[i] * tB[i] % MOD;
27    }
28    DFT(r, M, 1);
29    meminit(r, n, M);
30    for (int i = 0; i < n; i++) {
31      r[i] = (a[i] - r[i] + MOD) % MOD;
32    }
33  }
```

5 数据结构

5.1 平衡的二叉查找树

5.1.1 Treap

```cpp
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int maxn=1e5+5;
4   #define sz(x) (x?x->siz:0)
5   struct Treap{
6     struct node{
7       int key,val;
8       int siz,s;
9       node *c[2];
10      node(int v=0){
11        val=v;
12        key=rand();
13        siz=1,s=1;
14        c[0]=c[1]=0;
15      }
16      void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
17    }pool[maxn],*cur,*root;
18    Treap(){cur=pool;}
19    node* newnode(int val){return *cur=node(val),cur++;}
20    void rot(node *&t,int d){
21      if(!t->c[d])t=t->c[!d];
22      else{
23        node *p=t->c[d];t->c[d]=p->c[!d];
```

```
24        p->c[!d]=t;t->rz();p->rz();t=p;
25      }
26    }
27    void insert(node *&t,int x){
28      if(!t){t=newnode(x);return;}
29      if(t->val==x){t->s++;t->siz++;return;}
30      insert(t->c[x>t->val],x);
31      if(t->key<t->c[x>t->val]->key)
32        rot(t,x>t->val);
33      else t->rz();
34    }
35    void del(node *&t,int x){
36      if(!t)return;
37      if(t->val==x){
38        if(t->s>1){t->s--;t->siz--;return;}
39        if(!t->c[0]||!t->c[1]){
40          if(!t->c[0])t=t->c[1];
41          else t=t->c[0];
42          return;
43        }
44        int d=t->c[0]->key<t->c[1]->key;
45        rot(t,d);
46        del(t,x);
47        return;
48      }
49      del(t->c[x>t->val],x);
50      t->rz();
51    }
52    int pre(node *t,int x){
53      if(!t)return INT_MIN;
54      int ans=pre(t->c[x>t->val],x);
55      if(t->val<x)ans=max(ans,t->val);
56      return ans;
57    }
58    int nxt(node *t,int x){
59      if(!t)return INT_MAX;
60      int ans=nxt(t->c[x>=t->val],x);
61      if(t->val>x)ans=min(ans,t->val);
62      return ans;
63    }
64    int rank(node *t,int x){
65      if(!t)return 0;
66      if(t->val==x)return sz(t->c[0]);
67      if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
68      if(t->val>x)return rank(t->c[0],x);
69    }
70    int kth(node *t,int x){
71      if(sz(t->c[0])>=x)return kth(t->c[0],x);
72      if(sz(t->c[0])+t->s>=x)return t->val;
```

```
73          return kth(t->c[1],x-t->s-sz(t->c[0]));
74      }
75      void deb(node *t){
76          if(!t)return;
77          deb(t->c[0]);
78          printf("%d ",t->val);
79          deb(t->c[1]);
80      }
81      void insert(int x){insert(root,x);}
82      void del(int x){del(root,x);}
83      int pre(int x){return pre(root,x);}
84      int nxt(int x){return nxt(root,x);}
85      int rank(int x){return rank(root,x);}
86      int kth(int x){return kth(root,x);}
87      void deb(){deb(root);puts("");}
88  }T;
89  int main(){
90      srand(12121);
91      int m;
92      scanf("%d",&m);
93      while(m--){
94          int opt,x;
95          scanf("%d",&opt);
96          switch(opt){
97              case 1:
98                  scanf("%d",&x);
99                  T.insert(x);
100                 break;
101             case 2:
102                 scanf("%d",&x);
103                 T.del(x);
104                 break;
105             case 3:
106                 scanf("%d",&x);
107                 printf("%d\n",T.rank(x)+1);
108                 break;
109             case 4:
110                 scanf("%d",&x);
111                 printf("%d\n",T.kth(x));
112                 break;
113             case 5:
114                 scanf("%d",&x);
115                 printf("%d\n",T.pre(x));
116                 break;
117             case 6:
118                 scanf("%d",&x);
119                 printf("%d\n",T.nxt(x));
120                 break;
121         }
```

```
122        }
123    return 0;
124  }
```

5.1.2 Splay

```
1   void Rotate(int x, int c){
2     int y = T[x].c[c];
3     int z = T[y].c[1 - c];
4
5     if (T[x].fa){
6       if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
7       else T[T[x].fa].c[1] = y;
8     }
9
10    T[z].fa = x; T[x].c[c] = z;
11    T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;
12
13    Update(x);
14    Update(y);
15  }
16
17  int stack[M], fx[M];
18
19  void Splay(int x, int fa){
20    int top = 0;
21    for (int  u = x; u != fa; u = T[u].fa)
22      stack[++top] = u;
23    for (int i = 2; i <= top; i++)
24      if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
25      else fx[i] = 1;
26
27    for (int i = 2; i <= top; i += 2){
28      if (i == top) Rotate(stack[i], fx[i]);
29      else {
30        if (fx[i] == fx[i + 1]){
31          Rotate(stack[i + 1], fx[i + 1]);
32          Rotate(stack[i], fx[i]);
33        } else {
34          Rotate(stack[i], fx[i]);
35          Rotate(stack[i + 1], fx[i + 1]);
36        }
37      }
38    }
39
40    if (fa == 0) Root = x;
41  }
```

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();
```

```
47          p=merge(this,p);
48      }
49  }
50  int main(){
51      for(int i=1;i<=n;i++){
52          nd[i].val=in();
53          nd[i].key=rand();
54          nd[i].rz();
55          root=merge(root,nd+i);
56      }
57  }
```

5.2.2　坚固的字符串

1. ext 库中的 rope

```
1   #include <ext/rope>
2
3   using __gnu_cxx::crope;
4   using __gnu_cxx::rope;
5
6   crope a, b;
7
8   int main(void) {
9       a = b.substr(pos, len);      // [pos, pos + len)
10      a = b.substr(pos);           // [pos, pos]
11      b.c_str();                   // might lead to memory leaks
12      b.delete_c_str();            // delete the c_str that created before
13      a.insert(pos, text);         // insert text before position pos
14      a.erase(pos, len);           // erase [pos, pos + len)
15  }
```

2. 可持久化平衡树实现的 rope

```
1   class Rope {
2   private:
3       class Node {
4       public:
5           Node *left, *right;
6           int size;
7           char key;
8
9           Node(char key = 0, Node *left = NULL, Node *right = NULL)
10              : key(key), left(left), right(right) {
11              update();
12          }
13
14          void update() {
15              size = (left ? left->size : 0) + 1 + (right ? right->size : 0);
16          }
17
```

33

```cpp
    std::string to_string() {
        return (left ? left->to_string() : "") + key
            + (right ? right->to_string() : "");
    }
};

bool random(int a, int b) {
    return rand() % (a + b) < a;
}

Node* merge(Node *x, Node *y) {
    if (!x) {
        return y;
    }
    if (!y) {
        return x;
    }
    if (random(x->size, y->size)) {
        return new Node(x->key, x->left, merge(x->right, y));
    } else {
        return new Node(y->key, merge(x, y->left), y->right);
    }
}

std::pair<Node*, Node*> split(Node *x, int size) {
    if (!x) {
        return std::make_pair<Node*, Node*>(NULL, NULL);
    }
    if (size == 0) {
        return std::make_pair<Node*, Node*>(NULL, x);
    }
    if (size > x->size) {
        return std::make_pair<Node*, Node*>(x, NULL);
    }
    if (x->left && size <= x->left->size) {
        std::pair<Node*, Node*> part =
            split(x->left, size);
        return std::make_pair(part.first, new Node(x->key, part.second, x->right));
    } else {
        std::pair<Node*, Node*> part =
            split(x->right, size - (x->left ? x->left->size : 0) - 1);
        return std::make_pair(new Node(x->key, x->left, part.first), part.second);
    }
}

Node* build(const std::string &text, int left, int right) {
    if (left > right) {
        return NULL;
    }
```

```cpp
            int mid = left + right >> 1;
            return new Node(text[mid],
                            build(text, left, mid - 1),
                            build(text, mid + 1, right));
        }

public:
    Node *root;

    Rope() {
        root = NULL;
    }

    Rope(const std::string &text) {
        root = build(text, 0, (int)text.length() - 1);
    }

    Rope(const Rope &other) {
        root = other.root;
    }

    Rope& operator = (const Rope &other) {
        if (this == &other) {
            return *this;
        }
        root = other.root;
        return *this;
    }

    int size() {
        return root ? root->size : 0;
    }

    void insert(int pos, const std::string &text) {
        if (pos < 0 || pos > size()) {
            throw "Out of range";
        }
        std::pair<Node*, Node*> part = split(root, pos);
        root = merge(merge(part.first, build(text, 0, (int)text.length() - 1)),
                     part.second);
    }

    void erase(int left, int right) {
        if (left < 0 || left >= size() ||
            right < 1 || right > size()) {
            throw "Out of range";
        }
        if (left >= right) {
            return;
```

```
116            }
117            std::pair<Node*, Node*> part = split(root, left);
118            root = merge(part.first, split(part.second, right - left).second);
119        }
120
121        std::string substr(int left, int right) {
122            if (left < 0 || left >= size() ||
123                right < 1 || right > size()) {
124                throw "Out of range";
125            }
126            if (left >= right) {
127                return "";
128            }
129            return split(split(root, left).second, right - left).first->to_string();
130        }
131
132        void copy(int left, int right, int pos) {
133            if (left < 0 || left >= size() ||
134                right < 1 || right > size() ||
135                pos < 0 || pos > size()) {
136                throw "Out of range";
137            }
138            if (left >= right) {
139                return;
140            }
141            std::pair<Node*, Node*> part = split(root, pos);
142            root = merge(merge(part.first,
143                             split(split(root, left).second, right - left).first),
144                       part.second);
145        }
146    };
```

5.2.3 坚固的左偏树

```
1   int Merge(int x, int y){
2     if (x == 0 || y == 0) return x + y;
3     if (Heap[x].Key < Heap[y].Key) swap(x, y);
4     Heap[x].Ri = Merge(Heap[x].Ri, y);
5     if (Heap[Heap[x].Le].Dis < Heap[Heap[x].Ri].Dis) swap(Heap[x].Le, Heap[x].Ri);
6     if (Heap[x].Ri == 0) Heap[x].Dis = 0;
7     else Heap[x].Dis = Heap[Heap[x].Ri].Dis + 1;
8     return x;
9   }
10
11  for (int i = 0; i <= n; i++){
12    Heap[i].Le = Heap[i].Ri = 0;
13    Heap[i].Dis = 0;
14    Heap[i].Key = Cost[i];
15  }
16  Heap[0].Dis = -1;
```

5.2.4 不坚固的斜堆

```
1   struct node;
2   node *Null,*root[maxn];
3   struct node{
4     node* c[2];
5     int val,ind;
6     node(int _val=0,int _ind=0){
7       val=_val;c[0]=c[1]=Null;ind=_ind;
8     }
9   };
10  node* merge(node *p,node *q){
11    if(p==Null)return q;
12    if(q==Null)return p;
13    if(p->val>q->val)swap(p,q);
14    p->c[1]=merge(p->c[1],q);
15    swap(p->c[0],p->c[1]);
16    return p;
17  }
18
19  Null=new node(0);
20  Null->c[0]=Null->c[1]=Null;
```

5.3 树上的魔术师

5.3.1 轻重树链剖分（zky）

```
1   vector<int>G[maxn];
2   int fa[maxn],top[maxn],siz[maxn],son[maxn],mp[maxn],z,dep[maxn];
3   void dfs(int u){
4     siz[u]=1;
5     for(int i=0;i<G[u].size();i++){
6       int v=G[u][i];
7       if(v!=fa[u]){
8         fa[v]=u;dep[v]=dep[u]+1;
9         dfs(v);
10        siz[u]+=siz[v];
11        if(siz[son[u]]<siz[v])son[u]=v;
12      }
13    }
14  }
15  void build(int u,int tp){
16    top[u]=tp;mp[u]=++z;
17    if(son[u])build(son[u],tp);
18    for(int v,i=0;i<G[u].size();i++)if((v=G[u][i])!=son[u]&&v!=fa[u])build(v,v);
19  }
```

5.3.2 Link Cut Tree(zky)

```
1   struct LCT{
2       struct node{
3           bool rev;
4           int mx,val;
5           node *f,*c[2];
6           bool d(){return this==f->c[1];}
7           bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
8           void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
9           void makerv(){rev^=1;swap(c[0],c[1]);}
10          void pd(){
11              if(rev){
12                  if(c[0])c[0]->makerv();
13                  if(c[1])c[1]->makerv();
14                  rev=0;
15              }
16          }
17          void rz(){
18              mx=val;
19              if(c[0])mx=max(mx,c[0]->mx);
20              if(c[1])mx=max(mx,c[1]->mx);
21          }
22      }nd[int(1e4)+1];
23      void rot(node *x){
24          node *y=x->f;if(!y->rt())y->f->pd();
25          y->pd();x->pd();bool d=x->d();
26          y->sets(x->c[!d],d);
27          if(y->rt())x->f=y->f;
28          else y->f->sets(x,y->d());
29          x->sets(y,!d);
30      }
31      void splay(node *x){
32          while(!x->rt())
33              if(x->f->rt())rot(x);
34              else if(x->d()==x->f->d())rot(x->f),rot(x);
35              else rot(x),rot(x);
36      }
37      node* access(node *x){
38          node *y=0;
39          for(;x;x=x->f){
40              splay(x);
41              x->sets(y,1);y=x;
42          }return y;
43      }
44      void makert(node *x){
45          access(x)->makerv();
46          splay(x);
47      }
48      void link(node *x,node *y){
```

```
49          makert(x);
50          x->f=y;
51          access(x);
52      }
53      void cut(node *x,node *y){
54          makert(x);access(y);splay(y);
55          y->c[0]=x->f=0;
56          y->rz();
57      }
58      void link(int x,int y){link(nd+x,nd+y);}
59      void cut(int x,int y){cut(nd+x,nd+y);}
60  }T;
```

5.4 RMQ

```
1   for (int i = 1; i <= n; i++)
2     Log[i] = int(log2(i));
3
4   for (int i = 1; i <= n; i++)
5     Rmq[i][0] = i;
6
7   for (int k = 1; (1 << k) <= n; k++)
8     for (int i = 1; i + (1 << k) - 1 <= n; i++){
9       int x = Rmq[i][k - 1], y = Rmq[i + (1 << (k - 1))][k - 1];
10      if (a[x] < a[y])
11        Rmq[i][k] = x;
12      else
13        Rmq[i][k] = y;
14    }
15
16  int Smallest(int l, int r){
17    int k = Log[r - l + 1];
18
19    int x = Rmq[l][k];
20    int y = Rmq[r - (1 << k) + 1][k];
21
22    if (a[x] < a[y]) return x;
23    else return y;
24  }
```

5.5 可持久化线段树

```
1   struct node1 {
2     int L, R, Lson, Rson, Sum;
3   } tree[N * 40];
4   int root[N], a[N], b[N];
5   int tot, n, m;
6   int Real[N];
7   int Same(int x) {
```

```
8      ++tot;
9      tree[tot] = tree[x];
10     return tot;
11   }
12   int build(int L, int R) {
13     ++tot;
14     tree[tot].L = L;
15     tree[tot].R = R;
16     tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
17     if (L == R) return tot;
18     int s = tot;
19     int mid = (L + R) >> 1;
20     tree[s].Lson = build(L, mid);
21     tree[s].Rson = build(mid + 1, R);
22     return s;
23   }
24   int Ask(int Lst, int Cur, int L, int R, int k) {
25     if (L == R) return L;
26     int Mid = (L + R) >> 1;
27     int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
28     if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
29     k -= Left;
30     return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
31   }
32   int Add(int Lst, int pos) {
33     int root = Same(Lst);
34     tree[root].Sum++;
35     if (tree[root].L == tree[root].R) return root;
36     int mid = (tree[root].L + tree[root].R) >> 1;
37     if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
38     else tree[root].Rson = Add(tree[root].Rson, pos);
39     return root;
40   }
41   int main() {
42     scanf("%d%d", &n, &m);
43     int up = 0;
44     for (int i = 1; i <= n; i++){
45       scanf("%d", &a[i]);
46       b[i] = a[i];
47     }
48     sort(b + 1, b + n + 1);
49     up = unique(b + 1, b + n + 1) - b - 1;
50     for (int i = 1; i <= n; i++){
51       int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
52       Real[tmp] = a[i];
53       a[i] = tmp;
54     }
55     tot = 0;
56     root[0] = build(1, up);
```

```
57      for (int i = 1; i <= n; i++){
58        root[i] = Add(root[i - 1], a[i]);
59      }
60      for (int i = 1; i <= m; i++){
61        int u, v, w;
62        scanf("%d%d%d", &u, &v, &w);
63        printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
64      }
65      return 0;
66    }
```

5.6 可持久化 Trie

```
1   int Pre[N];
2   int n, q, Len, cnt, Lstans;
3   char s[N];
4   int First[N], Last[N];
5   int Root[N];
6   int Trie_tot;
7   struct node{
8       int To[30];
9       int Lst;
10  }Trie[N];
11  int tot;
12  struct node1{
13      int L, R, Lson, Rson, Sum;
14  }tree[N * 25];
15  int Build(int L, int R){
16      ++tot;
17      tree[tot].L = L;
18      tree[tot].R = R;
19      tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
20      if (L == R) return tot;
21      int s = tot;
22      int mid = (L + R) >> 1;
23      tree[s].Lson = Build(L, mid);
24      tree[s].Rson = Build(mid + 1, R);
25      return s;
26  }
27  int Same(int x){
28      ++tot;
29      tree[tot] = tree[x];
30      return tot;
31  }
32  int Add(int Lst, int pos){
33      int s = Same(Lst);
34      tree[s].Sum++;
35      if (tree[s].L == tree[s].R) return s;
36      int Mid = (tree[s].L + tree[s].R) >> 1;
```

```
37      if (pos <= Mid) tree[s].Lson = Add(tree[Lst].Lson, pos);
38      else tree[s].Rson = Add(tree[Lst].Rson, pos);
39      return s;
40  }

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  }

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);
90          int Ret = Ask(Root[First[L] - 1], Root[Last[R]], 0, n, L);
91          printf("%d\n", Ret);
92          Lstans = Ret;
93      }
94    }
95    return 0;
96 }
```

5.7 k-d 树

```
1  long long norm(const long long &x) {
2      //    For manhattan distance
3      return std::abs(x);
4      //    For euclid distance
5      return x * x;
6  }
7
8  struct Point {
9      int x, y, id;
10
11     const int& operator [] (int index) const {
12         if (index == 0) {
13             return x;
14         } else {
15             return y;
16         }
17     }
18
19     friend long long dist(const Point &a, const Point &b) {
20         long long result = 0;
21         for (int i = 0; i < 2; ++i) {
22             result += norm(a[i] - b[i]);
23         }
24         return result;
25     }
26 } point[N];
27
28 struct Rectangle {
29     int min[2], max[2];
30
31     Rectangle() {
32         min[0] = min[1] = INT_MAX;
33         max[0] = max[1] = INT_MIN;
34     }
35
```

```cpp
    void add(const Point &p) {
        for (int i = 0; i < 2; ++i) {
            min[i] = std::min(min[i], p[i]);
            max[i] = std::max(max[i], p[i]);
        }
    }

    long long dist(const Point &p) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            //    For minimum distance
            result += norm(std::min(std::max(p[i], min[i]), max[i]) - p[i]);
            //    For maximum distance
            result += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
        }
        return result;
    }
};

struct Node {
    Point seperator;
    Rectangle rectangle;
    int child[2];

    void reset(const Point &p) {
        seperator = p;
        rectangle = Rectangle();
        rectangle.add(p);
        child[0] = child[1] = 0;
    }
} tree[N << 1];

int size, pivot;

bool compare(const Point &a, const Point &b) {
    if (a[pivot] != b[pivot]) {
        return a[pivot] < b[pivot];
    }
    return a.id < b.id;
}

int build(int l, int r, int type = 1) {
    pivot = type;
    if (l >= r) {
        return 0;
    }
    int x = ++size;
    int mid = l + r >> 1;
    std::nth_element(point + l, point + mid, point + r, compare);
```

```
85      tree[x].reset(point[mid]);
86      for (int i = l; i < r; ++i) {
87          tree[x].rectangle.add(point[i]);
88      }
89      tree[x].child[0] = build(l, mid, type ^ 1);
90      tree[x].child[1] = build(mid + 1, r, type ^ 1);
91      return x;
92  }

94  int insert(int x, const Point &p, int type = 1) {
95      pivot = type;
96      if (x == 0) {
97          tree[++size].reset(p);
98          return size;
99      }
100     tree[x].rectangle.add(p);
101     if (compare(p, tree[x].seperator)) {
102         tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
103     } else {
104         tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
105     }
106     return x;
107 }

109 //     For minimum distance
110 void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
111     pivot = type;
112     if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
113         return;
114     }
115     answer = std::min(answer,
116             std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
117     if (compare(p, tree[x].seperator)) {
118         query(tree[x].child[0], p, answer, type ^ 1);
119         query(tree[x].child[1], p, answer, type ^ 1);
120     } else {
121         query(tree[x].child[1], p, answer, type ^ 1);
122         query(tree[x].child[0], p, answer, type ^ 1);
123     }
124 }

126 std::priority_queue<std::pair<long long, int> > answer;

128 void query(int x, const Point &p, int k, int type = 1) {
129     pivot = type;
130     if (x == 0 ||
131         (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().first) {
132         return;
133     }
```

```
134      answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
135      if ((int)answer.size() > k) {
136          answer.pop();
137      }
138      if (compare(p, tree[x].seperator)) {
139          query(tree[x].child[0], p, k, type ^ 1);
140          query(tree[x].child[1], p, k, type ^ 1);
141      } else {
142          query(tree[x].child[1], p, k, type ^ 1);
143          query(tree[x].child[0], p, k, type ^ 1);
144      }
145  }
```

5.8 莫队算法

```
1   struct node{
2     int l, r, id;
3     friend bool operator < (const node &a, const node &b){
4       if (a.l / Block == b.l / Block) return a.r / Block < b.r / Block;
5       return a.l / Block < b.l / Block;
6     }
7   }q[N];
8   Block = int(sqrt(n));
9   for (int i = 1; i <= m; i++){
10    scanf("%d%d", &q[i].l, &q[i].r);
11    q[i].id = i;
12  }
13  sort(q + 1, q + 1 + m);
14  Cur = a[1]; /// Hints: adjust by yourself
15  Le = Ri = 1;
16  for (int i = 1; i <= m; i++){
17    while (q[i].r > Ri) Ri++, ChangeRi(1, Le, Ri);
18    while (q[i].l > Le) ChangeLe(-1, Le, Ri), Le++;
19    while (q[i].l < Le) Le--, ChangeLe(1, Le, Ri);
20    while (q[i].r < Ri) ChangeRi(-1, Le, Ri), Ri--;
21    Ans[q[i].id] = Cur;
22  }
```

5.9 树上在线莫队

```
1   bool operator<(qes a,qes b){
2       if(dfn[a.x]/B!=dfn[b.x]/B)return dfn[a.x]/B<dfn[b.x]/B;
3       if(dfn[a.y]/B!=dfn[b.y]/B)return dfn[a.y]/B<dfn[b.y]/B;
4       if(a.tm/B!=b.tm/B)return a.tm/B<b.tm/B;
5       return a.tm<b.tm;
6   }
7   void vxor(int x){
8       if(vis[x])ans-=(LL)W[cnt[col[x]]]*V[col[x]],cnt[col[x]]--;
9       else cnt[col[x]]++,ans+=(LL)W[cnt[col[x]]]*V[col[x]];
```

```
10        vis[x]^=1;
11    }
12    void change(int x,int y){
13        if(vis[x]){
14            vxor(x);col[x]=y;vxor(x);
15        }else col[x]=y;
16    }
17    void TimeMachine(int tar){//XD
18        for(int i=now+1;i<=tar;i++)change(C[i].x,C[i].y);
19        for(int i=now;i>tar;i--)change(C[i].x,C[i].pre);
20        now=tar;
21    }
22    void vxor(int x,int y){
23        while(x!=y)if(dep[x]>dep[y])vxor(x),x=fa[x];
24        else vxor(y),y=fa[y];
25    }
26        for(int i=1;i<=q;i++){
27            int ty=getint(),x=getint(),y=getint();
28            if(ty&&dfn[x]>dfn[y])swap(x,y);
29            if(ty==0) C[++Csize]=(oper){x,y,pre[x],i},pre[x]=y;
30            else Q[Qsize+1]=(qes){x,y,Qsize+1,Csize},Qsize++;
31        }sort(Q+1,Q+1+Qsize);
32        int u=Q[1].x,v=Q[1].y;
33        TimeMachine(Q[1].tm);
34        vxor(Q[1].x,Q[1].y);
35        int LCA=lca(Q[1].x,Q[1].y);
36        vxor(LCA);anss[Q[1].id]=ans;vxor(LCA);
37        for(int i=2;i<=Qsize;i++){
38            TimeMachine(Q[i].tm);
39            vxor(Q[i-1].x,Q[i].x);
40            vxor(Q[i-1].y,Q[i].y);
41            int LCA=lca(Q[i].x,Q[i].y);
42            vxor(LCA);
43            anss[Q[i].id]=ans;
44            vxor(LCA);
45        }
```

5.10 整体二分

```
1    struct BIT{
2      LL d[maxn];
3      inline int lowbit(int x){return x&-x;}
4      LL get(int x){
5        LL ans=0;
6        while(x)ans+=d[x],x-=lowbit(x);
7        return ans;
8      }
9      void updata(int x,LL f){
10        while(x<=m)d[x]+=f,x+=lowbit(x);
```

```
11          }
12      void add(int l,int r,LL f){
13          updata(l,f);
14          updata(r+1,-f);
15      }
16  }T,T2;
17  int anss[maxn],wana[maxn];
18  struct qes{
19      LL x,y,z;
20      qes(LL _x=0,LL _y=0,LL _z=0):
21          x(_x),y(_y),z(_z){}
22  }q[maxn],p[maxn];
23  bool part(qes &q){
24      if(q.y+q.z>=wana[q.x])return 1;
25      q.z+=q.y;q.y=0;return 0;
26  }
27  void solve(int lef,int rig,int l,int r){
28      if(l==r){
29          for(int i=lef;i<=rig;i++)if(anss[p[i].x]!=-1)
30          anss[p[i].x]=l;return;
31      }int mid=(l+r)>>1;
32      for(int i=l;i<=mid;i++){
33          if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,q[i].z);
34          else T.add(1,q[i].y,q[i].z),T.add(q[i].x,m,q[i].z);
35      }for(int i=lef;i<=rig;i++){
36          p[i].y=0;
37          for(int j=0;j<O[p[i].x].size()&&p[i].y<=int(1e9)+1;j++)
38          p[i].y+=T.get(O[p[i].x][j]);
39      }for(int i=l;i<=mid;i++){
40          if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,-q[i].z);
41          else T.add(1,q[i].y,-q[i].z),T.add(q[i].x,m,-q[i].z);
42      }int dv=stable_partition(p+lef,p+rig+1,part)-p-1;
43      if(lef<=dv)
44      solve(lef,dv,l,mid);
45      if(dv+1<=rig)
46      solve(dv+1,rig,mid+1,r);
47  }
```

5.11 树状数组 kth

```
1  int find(int k){
2      int cnt=0,ans=0;
3      for(int i=22;i>=0;i--){
4          ans+=(1<<i);
5          if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
6          else cnt+=d[ans];
7      }
8      return ans+1;
9  }
```

5.12 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;
}
```

5.13 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){
```

```
25        d[u]=0;
26        for(int i=0;i<G[u].size();i++){
27            edge e=G[u][i];
28            if(cant[e.v])continue;
29            d[e.v]=e.w;
30            dfs(e.v,u);
31
32        }
33    }
34    void solve(int u){
35        calc(u);cant[u]=1;
36        for(int i=0;i<G[u].size();i++){
37            edge e=G[u][i];
38            if(cant[e.v])continue;
39            All=siz[e.v];
40            f[root=0]=n+1;
41            makert(e.v,0);
42            solve(root);
43        }
44    }
45    All=n
46    f[root=0]=n+1;
47    makert(1,1);
48    solve(root);
```

5.14   元芳树

```
1    #include<bits/stdc++.h>
2    using namespace std;
3    const int maxn=1e4+1e4+233;
4    const int BIT=18;
5    int n,m,q;
6    struct edge{
7      int u,v,w;
8      bool operator==(edge oth)const{
9        return u==oth.u && v==oth.v && w==oth.w;
10      }
11      bool operator!=(edge oth)const{
12        return !(*this==oth);
13      }
14    };
15    vector<edge>G[maxn],T[maxn];
16
17    int dfn[maxn],low[maxn],tot,rlen[maxn];
18    bool ins[maxn];
19    stack<edge>S;
20    int Rcnt=0;
21    vector<edge>ring[maxn];
22    vector<int>bel[maxn],sum[maxn],dis[maxn];
```

```
23  int fa[maxn][BIT];
24  int dep[maxn],dep2[maxn],fw[maxn];
25  vector<pair<int,int> >ind[maxn];
26  map<pair<int,int>,int>Mw;
27  pair<int,int>pack(int a,int b){
28      if(a>b)swap(a,b);
29      return make_pair(a,b);
30  }
31  void tarjan(int u){
32      dfn[u]=low[u]=++tot;
33      for(int i=0;i<G[u].size();i++){
34          edge e=G[u][i];
35          if(dfn[e.v])
36              low[u]=min(low[u],dfn[e.v]);
37          else{
38              S.push(e);
39              tarjan(e.v);
40              if(low[e.v]==dfn[u]){
41
42                  if(S.top()==e){
43                      fa[e.v][0]=u;
44                      fw[e.v]=e.w;
45                      S.pop();
46                      continue;
47                  }
48
49                  Rcnt++;
50                  edge ed;
51                  do{
52                          ed=S.top();S.pop();
53                          ring[Rcnt].push_back(ed);
54                      }while(ed!=e);
55                  reverse(ring[Rcnt].begin(),ring[Rcnt].end());
56                      int last=ring[Rcnt].back().v;
57                  ring[Rcnt].push_back((edge){last,u,Mw[pack(last,u)]});
58              }
59          low[u]=min(low[u],low[e.v]);
60          }
61      }
62  }
63  void up(int u){
64      if(dep[u]||u==1)return ;
65      if(fa[u][0])up(fa[u][0]);
66      dep[u]=dep[fa[u][0]]+1;
67      fw[u]+=fw[fa[u][0]];
68  }
69  void build(){
70      S.push((edge){0,1,0});
71      tarjan(1);
```

51

```
72
73    for(int i=1;i<=Rcnt;i++){
74      rlen[i]=0;
75      sum[i].resize(ring[i].size());
76      dis[i].resize(ring[i].size());
77      for(int j=0;j<ring[i].size();j++){
78        rlen[i]+=ring[i][j].w;
79        ind[i].push_back(make_pair(ring[i][j].u,j));
80      }
81      sum[i][0]=0;
82      fw[i+n]=0;
83      fa[i+n][0]=ring[i][0].u;
84      for(int j=1;j<ring[i].size();j++){
85        sum[i][j]=sum[i][j-1]+ring[i][j-1].w;
86        dis[i][j]=min(sum[i][j],rlen[i]-sum[i][j]);
87        fw[ring[i][j].u]=dis[i][j];
88        fa[ring[i][j].u][0]=i+n;
89      }
90      sort(ind[i].begin(),ind[i].end());
91    }
92
93    for(int i=1;i<=n+Rcnt;i++)
94      up(i);
95
96    for(int j=1;j<BIT;j++)
97    for(int i=1;i<=n+Rcnt;i++)if(fa[i][j-1])
98      fa[i][j]=fa[fa[i][j-1]][j-1];
99
100 }
101 pair<int,int>second_lca;
102 int lca(int u,int v){
103    if(dep[u]<dep[v])swap(u,v);
104    int d=dep[u]-dep[v];
105    for(int i=0;i<BIT;i++)if(d>>i&1)
106      u=fa[u][i];
107    if(u==v)return u;
108    for(int i=BIT-1;i>=0;i--)if(fa[u][i]!=fa[v][i]){
109      u=fa[u][i];
110      v=fa[v][i];
111    }
112    second_lca=make_pair(u,v);
113    return fa[u][0];
114 }
115 int main(){
116
117    freopen("bzoj2125.in","r",stdin);
118
119    scanf("%d%d%d",&n,&m,&q);
120    for(int i=1;i<=m;i++){
```

```
121    int u,v,w;scanf("%d%d%d",&u,&v,&w);
122    G[u].push_back((edge){u,v,w});
123    G[v].push_back((edge){v,u,w});
124    Mw[pack(u,v)]=w;
125  }
126
127  build();
128  while(q--){
129    int u,v;
130    scanf("%d%d",&u,&v);
131    int LCA=lca(u,v);
132    if(LCA<=n)printf("%d\n",fw[u]+fw[v]-2*fw[LCA]);
133    else{
134      if(dep[u]<dep[v])swap(u,v);
135      int R=LCA-n;
136      int uu=second_lca.first;
137      int vv=second_lca.second;
138      int ans=fw[u]-fw[uu]+fw[v]-fw[vv];
139      int uid,vid;
140      uid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(uu,-1))->second;
141      vid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(vv,-1))->second;
142      ans+=min(abs(sum[R][uid]-sum[R][vid]),rlen[R]-abs(sum[R][uid]-sum[R][vid]));
143      printf("%d\n",ans);
144    }
145  }
146  return 0;
147 }
```

6 图论

6.1 点双连通分量

```
1  // 坚固的点双连通分量
2  int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N];
3  vector<int> a[N];
4  pair<int, int> stack[M];
5  void tarjan(int x, int p) {
6    dfn[x] = low[x] = ++ind2;
7    for (int i = 0; i < a[x].size(); ++i)
8      if (dfn[x] > dfn[a[x][i]] && a[x][i] != p){
9        stack[++size] = make_pair(x, a[x][i]);
10       if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
11         if (!dfn[a[x][i]]){
12           tarjan(a[x][i], x);
13           low[x] = min(low[x], low[a[x][i]]);
14           if (low[a[x][i]] >= dfn[x]){
15             tot1 = tot2 = 0;
16             ++flag;
17             for (; ; ){
```

```
18          if (block[stack[size].first] != flag) {
19             ++tot1;
20             block[stack[size].first] = flag;
21          }
22          if (block[stack[size].second] != flag) {
23             ++tot1;
24             block[stack[size].second] = flag;
25          }
26          if (stack[size].first == x && stack[size].second == a[x][i])
27             break;
28          ++tot2;
29          --size;
30       }
31       for (; stack[size].first == x && stack[size].second == a[x][i]; --size)
32          ++tot2;
33       if (tot2 < tot1)
34          ans1 += tot2;
35       if (tot2 > tot1)
36          ans2 += tot2;
37       }
38    }
39    else
40       low[x] = min(low[x], dfn[a[x][i]]);
41    }
42 }
43 int main(){
44    for (; ; ){
45       scanf("%d%d", &n, &m);
46       if (n == 0 && m == 0) return 0;
47       for (int i = 1; i <= n; ++i) {
48          a[i].clear();
49          dfn[i] = 0;
50       }
51       for (int i = 1; i <= m; ++i){
52          scanf("%d%d",&x, &y);
53          ++x, ++y;
54          a[x].push_back(y);
55          a[y].push_back(x);
56       }
57       for (int i = 1; i <= n; ++i)
58          sort(a[i].begin(), a[i].end());
59       ans1 = ans2 = ind2 = 0;
60       for (int i = 1; i <= n; ++i)
61          if (!dfn[i]) {
62             size = 0;
63             tarjan(i, 0);
64          }
65       printf("%d %d\n", ans1, ans2);
66    }
```

```
67      return 0;
68    }
69    // 朴素的点双连通分量
70    void tarjan(int x){
71      dfn[x] = low[x] = ++ind2;
72      v[x] = 1;
73      for (int i = nt[x]; pt[i]; i = nt[i])
74        if (!dfn[pt[i]]){
75          tarjan(pt[i]);
76          low[x] = min(low[x], low[pt[i]]);
77          if (dfn[x] <= low[pt[i]])
78            ++v[x];
79        }
80        else
81          low[x] = min(low[x], dfn[pt[i]]);
82    }
83    int main(){
84      for (; ; ){
85        scanf("%d%d", &n, &m);
86        if (n == 0 && m == 0)
87          return 0;
88        for (int i = 1; i <= ind; ++i)
89          nt[i] = pt[i] = 0;
90        ind = n;
91        for (int i = 1; i <= ind; ++i)
92          last[i] = i;
93        for (int i = 1; i <= m; ++i){
94          scanf("%d%d", &x, &y);
95          ++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 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;
}
```

6.3 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);
```

```
20      for (int answer = 0; ; ) {
21          std::vector<int> queue;
22          for (int i = 0; i < n; ++i) {
23              if (matchx[i] == -1) {
24                  level[i] = 0;
25                  queue.push_back(i);
26              } else {
27                  level[i] = -1;
28              }
29          }
30          for (int head = 0; head < (int)queue.size(); ++head) {
31              int x = queue[head];
32              for (int i = 0; i < (int)edge[x].size(); ++i) {
33                  int y = edge[x][i];
34                  int w = matchy[y];
35                  if (w != -1 && level[w] < 0) {
36                      level[w] = level[x] + 1;
37                      queue.push_back(w);
38                  }
39              }
40          }
41          int delta = 0;
42          for (int i = 0; i < n; ++i) {
43              if (matchx[i] == -1 && dfs(i)) {
44                  delta++;
45              }
46          }
47          if (delta == 0) {
48              return answer;
49          } else {
50              answer += delta;
51          }
52      }
53  }
```

6.4 KM 带权匹配

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

```
1   int DFS(int x){
2       visx[x] = 1;
3       for (int y = 1;y <= ny;y ++){
4           if (visy[y]) continue;
5           int t = lx[x] + ly[y] - w[x][y];
6           if (t == 0) {
7               visy[y] = 1;
8               if (link[y] == -1||DFS(link[y])){
9                   link[y] = x;
10                  return 1;
11              }
```

```
12              }
13              else slack[y] = min(slack[y],t);
14          }
15      return 0;
16  }
17  int KM(){
18      int i,j;
19      memset(link,-1,sizeof(link));
20      memset(ly,0,sizeof(ly));
21      for (i = 1; i <= nx; i++)
22          for (j = 1, lx[i] = -inf; j <= ny; j++)
23              lx[i] = max(lx[i],w[i][j]);
24      for (int x = 1; x <= nx; x++){
25          for (i = 1; i <= ny; i++) slack[i] = inf;
26          while (true) {
27              memset(visx, 0, sizeof(visx));
28              memset(visy, 0, sizeof(visy));
29              if (DFS(x)) break;
30              int d = inf;
31              for (i = 1; i <= ny;i++)
32                  if (!visy[i] && d > slack[i]) d = slack[i];
33              for (i = 1; i <= nx; i++)
34                  if (visx[i]) lx[i] -= d;
35              for (i = 1; i <= ny; i++)
36                  if (visy[i]) ly[i] += d;
37                  else slack[i] -= d;
38          }
39      }
40      int res = 0;
41      for (i = 1;i <= ny;i ++)
42          if (link[i] > -1) res += w[link[i]][i];
43      return res;
44  }
```

6.5 稀疏图最大流

注意事项： 适用于比较稀疏的一般图。

```
1   int Maxflow_Isap(int s,int t,int n) {
2     std::fill(pre + 1, pre + n + 1, 0);
3     std::fill(d + 1, d + n + 1, 0);
4     std::fill(gap + 1, gap + n + 1, 0);
5     for (int i = 1; i <= n; i++) cur[i] = h[i];
6     gap[0] = n;
7     int u = pre[s] = s, v, maxflow = 0;
8     while (d[s] < n) {
9       v = n + 1;
10      for (int i = cur[u]; i; i = e[i].next)
11      if (e[i].flow && d[u] == d[e[i].node] + 1) {
12        v = e[i].node; cur[u]=i; break;
```

```
13          }
14          if (v <= n) {
15            pre[v] = u; u = v;
16            if (v == t) {
17              int dflow = INF, p = t; u = s;
18              while (p != s) {
19                p = pre[p];
20                dflow = std::min(dflow, e[cur[p]].flow);
21              }
22              maxflow += dflow; p = t;
23              while (p != s) {
24                p = pre[p];
25                e[cur[p]].flow -= dflow;
26                e[e[cur[p]].opp].flow += dflow;
27              }
28            }
29          }
30          else{
31            int mindist = n + 1;
32            for (int i = h[u]; i; i = e[i].next)
33              if (e[i].flow && mindist > d[e[i].node]) {
34                mindist = d[e[i].node]; cur[u] = i;
35              }
36            if (!--gap[d[u]]) return maxflow;
37            gap[d[u] = mindist + 1]++; u = pre[u];
38          }
39        }
40        return maxflow;
41  }
```

6.6 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。

```
1   int bfs(){
2     for (int i = 1;i <= t;i ++) d[i] = -1;
3     int l,r;
4     q[l = r = 0] = s, d[s] = 0;
5     for (;l <= r;l ++)
6       for (int k = h[q[l]]; k > -1; k = nxt[k])
7         if (d[p[k]] == -1 && c[k] > 0) d[p[k]] = d[q[l]] + 1, q[++ r] = p[k];
8     return d[t] > -1 ? 1 : 0;
9   }
10  int dfs(int u,int ext){
11    if (u == t) return ext;
12    int k = w[u],ret = 0;
13    for (; k > -1; k = nxt[k], w[u] = k){     //w 数组为当前弧
14      if (ext == 0) break;
15      if (d[p[k]] == d[u] + 1 && c[k] > 0){
16        int flow = dfs(p[k], min(c[k], ext));
```

```
17        if (flow > 0){
18            c[k] -= flow, c[k ^ 1] += flow;
19            ret += flow, ext -= flow;      //ret 累计增广量, ext 记录还可增广的量
20        }
21    }
22  }
23  if (k == -1) d[u] = -1;
24  return ret;
25 }
26 void dinic() {
27   while (bfs()) {
28     for (int i = 1; i <= t;i ++) w[i] = h[i];
29     dfs(s, inf);
30   }
31 }
```

6.7 稀疏图费用流

```
1  struct EdgeList {
2      int size;
3      int last[N];
4      int succ[M], other[M], flow[M], cost[M];
5      void clear(int n) {
6          size = 0;
7          std::fill(last, last + n, -1);
8      }
9      void add(int x, int y, int c, int w) {
10         succ[size] = last[x];
11         last[x] = size;
12         other[size] = y;
13         flow[size] = c;
14         cost[size++] = w;
15     }
16 } e;
17
18 int n, source, target;
19 int prev[N];
20
21 void add(int x, int y, int c, int w) {
22     e.add(x, y, c, w);
23     e.add(y, x, 0, -w);
24 }
25
26 bool augment() {
27     static int dist[N], occur[N];
28     std::vector<int> queue;
29     std::fill(dist, dist + n, INT_MAX);
30     std::fill(occur, occur + n, 0);
31     dist[source] = 0;
```

```
32      occur[source] = true;
33      queue.push_back(source);
34      for (int head = 0; head < (int)queue.size(); ++head) {
35          int x = queue[head];
36          for (int i = e.last[x]; ~i; i = e.succ[i]) {
37              int y = e.other[i];
38              if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
39                  dist[y] = dist[x] + e.cost[i];
40                  prev[y] = i;
41                  if (!occur[y]) {
42                      occur[y] = true;
43                      queue.push_back(y);
44                  }
45              }
46          }
47          occur[x] = false;
48      }
49      return dist[target] < INT_MAX;
50  }
51
52  std::pair<int, int> solve() {
53      std::pair<int, int> answer = std::make_pair(0, 0);
54      while (augment()) {
55          int number = INT_MAX;
56          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
57              number = std::min(number, e.flow[prev[i]]);
58          }
59          answer.first += number;
60          for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
61              e.flow[prev[i]] -= number;
62              e.flow[prev[i] ^ 1] += number;
63              answer.second += number * e.cost[prev[i]];
64          }
65      }
66      return answer;
67  }
```

6.8 稠密图费用流

```
1   int aug(int no,int res) {
2       if(no == t) return cost += pi1 * res,res;
3       v[no] = true;
4       int flow = 0;
5       for(int i = h[no]; ~ i ;i = nxt[i])
6       if(cap[i] && !expense[i] && !v[p[i]]) {
7           int d = aug(p[i],min(res,cap[i]));
8           cap[i] -= d,cap[i ^ 1] += d,flow += d,res -= d;
9           if( !res ) return flow;
10      }
```

```
11        return flow;
12    }
13    bool modlabel() {
14        int d = maxint;
15        for(int i = 1;i <= t;++ i)
16        if(v[i]) {
17          for(int j = h[i]; ~ j ;j = nxt[j])
18            if(cap[j] && !v[p[j]] && expense[j] < d) d = expense[j];
19        }
20        if(d == maxint)return false;
21        for(int i = 1;i <= t;++ i)
22        if(v[i]) {
23          for(int j = h[i];~ j;j = nxt[j])
24            expense[j] -= d,expense[j ^ 1] += d;
25        }
26        pi1 += d;
27        return true;
28    }
29    void minimum_cost_flow_zkw() {
30      cost = 0;
31      do{
32        do{
33          memset(v,false,sizeof v);
34        }while (aug(s,maxint));
35      }while (modlabel());
36    }
```

6.9 2-SAT 问题

```
1    int stamp, comps, top;
2    int dfn[N], low[N], comp[N], stack[N];
3
4    void add(int x, int a, int y, int b) {
5        edge[x << 1 | a].push_back(y << 1 | b);
6    }
7
8    void tarjan(int x) {
9        dfn[x] = low[x] = ++stamp;
10       stack[top++] = x;
11       for (int i = 0; i < (int)edge[x].size(); ++i) {
12           int y = edge[x][i];
13           if (!dfn[y]) {
14               tarjan(y);
15               low[x] = std::min(low[x], low[y]);
16           } else if (!comp[y]) {
17               low[x] = std::min(low[x], dfn[y]);
18           }
19       }
20       if (low[x] == dfn[x]) {
```

```
21          comps++;
22          do {
23              int y = stack[--top];
24              comp[y] = comps;
25          } while (stack[top] != x);
26      }
27  }
28
29  bool solve() {
30      int counter = n + n + 1;
31      stamp = top = comps = 0;
32      std::fill(dfn, dfn + counter, 0);
33      std::fill(comp, comp + counter, 0);
34      for (int i = 0; i < counter; ++i) {
35          if (!dfn[i]) {
36              tarjan(i);
37          }
38      }
39      for (int i = 0; i < n; ++i) {
40          if (comp[i << 1] == comp[i << 1 | 1]) {
41              return false;
42          }
43          answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
44      }
45      return true;
46  }
```

6.10 有根树的同构

```
1   const unsigned long long MAGIC = 4423;
2
3   unsigned long long magic[N];
4   std::pair<unsigned long long, int> hash[N];
5
6   void solve(int root) {
7       magic[0] = 1;
8       for (int i = 1; i <= n; ++i) {
9           magic[i] = magic[i - 1] * MAGIC;
10      }
11      std::vector<int> queue;
12      queue.push_back(root);
13      for (int head = 0; head < (int)queue.size(); ++head) {
14          int x = queue[head];
15          for (int i = 0; i < (int)son[x].size(); ++i) {
16              int y = son[x][i];
17              queue.push_back(y);
18          }
19      }
20      for (int index = n - 1; index >= 0; --index) {
```

```
21          int x = queue[index];
22          hash[x] = std::make_pair(0, 0);
23
24          std::vector<std::pair<unsigned long long, int> > value;
25          for (int i = 0; i < (int)son[x].size(); ++i) {
26              int y = son[x][i];
27              value.push_back(hash[y]);
28          }
29          std::sort(value.begin(), value.end());
30
31          hash[x].first = hash[x].first * magic[1] + 37;
32          hash[x].second++;
33          for (int i = 0; i < (int)value.size(); ++i) {
34              hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
35              hash[x].second += value[i].second;
36          }
37          hash[x].first = hash[x].first * magic[1] + 41;
38          hash[x].second++;
39      }
40  }
```

6.11 Dominator Tree

```
1   #define foreach(A, x, it) for (int it = A.h[x]; it; it = A.e[it].next)
2
3   const int MAXN = 200001;
4   const int MAXM = 400001;
5
6   template<class T, int MAXN, int MAXM>
7   struct AdjList{
8     struct Edge{
9       T data;
10      int next;
11    }e[MAXM];
12    int h[MAXN], t;
13    void add(int x, const T &data) {
14      t++;
15      e[t].data = data;
16      e[t].next = h[x];
17      h[x] = t;
18    }
19    void drop(int x) {
20      h[x] = e[h[x]].next;
21    }
22    T & operator [](const int &index) {
23      return e[index].data;
24    }
25    void clear(int n) {
26      std::fill(h + 1, h + n + 1, t = 0);
```

```
27         }
28     };
29     // 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
30     // pred 是前驱链表，succ 是后继链表
31     int dfn[MAXN], sdom[MAXN], idom[MAXN], id[MAXN], f[MAXN], fa[MAXN], smin[MAXN];
32     AdjList<int, MAXN, MAXM> pred, succ;
33     long long answer[MAXN];
34
35     void predfs(int x) {
36         id[dfn[x] = ++stamp] = x;
37         foreach(succ, x, i) {
38             int y = succ[i];
39             if (!dfn[y]) {
40                 f[y] = x;
41                 predfs(y);
42             }
43         }
44     }
45
46     int getfa(int x) {
47         if (fa[x] == x) return x;
48         int ret = getfa(fa[x]);
49         if (dfn[sdom[smin[fa[x]]]] < dfn[sdom[smin[x]]]) {
50             smin[x] = smin[fa[x]];
51         }
52         return fa[x] = ret;
53     }
54
55     void tarjan(int s) {
56         static AdjList<int, MAXN, MAXN> tmp;
57         stamp = tmp.t = 0;
58         predfs(s);
59         for (int i = 1; i <= stamp; i++) {
60             fa[id[i]] = smin[id[i]] = id[i];
61             tmp.h[id[i]] = idom[id[i]] = 0;
62         }
63         for (int o = stamp; o >= 1; o--) {
64             int x = id[o];
65             if (o != 1) {
66                 sdom[x] = f[x];
67                 foreach(pred, x, i) {
68                     int p = pred[i];
69                     if (!dfn[p]) continue;
70                     if (dfn[p] > dfn[x]) {
71                         getfa(p);
72                         p = sdom[smin[p]];
73                     }
74                     if (dfn[sdom[x]] > dfn[p]) {
75                         sdom[x] = p;
```

```
76              }
77          }
78          tmp.add(sdom[x], x);
79      }
80      while (tmp.h[x] != 0) {
81          int y = tmp[tmp.h[x]];
82          tmp.drop(x);
83          getfa(y);
84          if (x != sdom[smin[y]]) {
85              idom[y] = smin[y];
86          } else {
87              idom[y] = x;
88          }
89      }
90      foreach(succ, x, i) {
91          if (f[succ[i]] == x) {
92              fa[succ[i]] = x;
93          }
94      }
95  }
96  idom[s] = s;
97  for (int i = 2; i <= stamp; i++) {
98      int x = id[i];
99      if (idom[x] != sdom[x]) {
100         idom[x] = idom[idom[x]];
101     }
102 }
103 }
```

6.12 哈密尔顿回路（ORE 性质的图）

```
1  int left[N], right[N], next[N], last[N];
2
3  void cover(int x) {
4      left[right[x]] = left[x];
5      right[left[x]] = right[x];
6  }
7
8  int adjacent(int x) {
9      for (int i = right[0]; i <= n; i = right[i]) {
10         if (graph[x][i]) {
11             return i;
12         }
13     }
14     return 0;
15 }
16
17 std::vector<int> solve() {
18     for (int i = 1; i <= n; ++i) {
```

```
19          left[i] = i - 1;
20          right[i] = i + 1;
21      }
22      int head, tail;
23      for (int i = 2; i <= n; ++i) {
24          if (graph[1][i]) {
25              head = 1;
26              tail = i;
27              cover(head);
28              cover(tail);
29              next[head] = tail;
30              break;
31          }
32      }
33      while (true) {
34          int x;
35          while (x = adjacent(head)) {
36              next[x] = head;
37              head = x;
38              cover(head);
39          }
40          while (x = adjacent(tail)) {
41              next[tail] = x;
42              tail = x;
43              cover(tail);
44          }
45          if (!graph[head][tail]) {
46              for (int i = head, j; i != tail; i = next[i]) {
47                  if (graph[head][next[i]] && graph[tail][i]) {
48                      for (j = head; j != i; j = next[j]) {
49                          last[next[j]] = j;
50                      }
51                      j = next[head];
52                      next[head] = next[i];
53                      next[tail] = i;
54                      tail = j;
55                      for (j = i; j != head; j = last[j]) {
56                          next[j] = last[j];
57                      }
58                      break;
59                  }
60              }
61          }
62          next[tail] = head;
63          if (right[0] > n) {
64              break;
65          }
66          for (int i = head; i != tail; i = next[i]) {
67              if (adjacent(i)) {
```

```
68          head = next[i];
69          tail = i;
70          next[tail] = 0;
71          break;
72        }
73      }
74    }
75    std::vector<int> answer;
76    for (int i = head; ; i = next[i]) {
77      if (i == 1) {
78        answer.push_back(i);
79        for (int j = next[i]; j != i; j = next[j]) {
80          answer.push_back(j);
81        }
82        answer.push_back(i);
83        break;
84      }
85      if (i == tail) {
86        break;
87      }
88    }
89    return answer;
90  }
```

6.13 无向图最小割

```
1   int node[N], dist[N];
2   bool visit[N];
3
4   int solve(int n) {
5     int answer = INT_MAX;
6     for (int i = 0; i < n; ++i) {
7       node[i] = i;
8     }
9     while (n > 1) {
10      int max = 1;
11      for (int i = 0; i < n; ++i) {
12        dist[node[i]] = graph[node[0]][node[i]];
13        if (dist[node[i]] > dist[node[max]]) {
14          max = i;
15        }
16      }
17      int prev = 0;
18      memset(visit, 0, sizeof(visit));
19      visit[node[0]] = true;
20      for (int i = 1; i < n; ++i) {
21        if (i == n - 1) {
22          answer = std::min(answer, dist[node[max]]);
23          for (int k = 0; k < n; ++k) {
```

```
24              graph[node[k]][node[prev]] =
25                    (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
26              }
27              node[max] = node[--n];
28          }
29          visit[node[max]] = true;
30          prev = max;
31          max = -1;
32          for (int j = 1; j < n; ++j) {
33              if (!visit[node[j]]) {
34                  dist[node[j]] += graph[node[prev]][node[j]];
35                  if (max == -1 || dist[node[max]] < dist[node[j]]) {
36                      max = j;
37                  }
38              }
39          }
40      }
41  }
42  return answer;
43 }
```

6.14 弦图判定

```
1  int n, m, first[1001], l, next[2000001], where[2000001],f[1001], a[1001], c[1001], L[1001], R[1001],
2  v[1001], idx[1001], pos[1001];
3  bool b[1001][1001];
4
5  inline void makelist(int x, int y){
6      where[++l] = y;
7      next[l] = first[x];
8      first[x] = l;
9  }
10
11 bool cmp(const int &x, const int &y){
12     return(idx[x] < idx[y]);
13 }
14
15 int main(){
16     for (;;)
17     {
18         n = read(); m = read();
19         if (!n && !m) return 0;
20         memset(first, 0, sizeof(first)); l = 0;
21         memset(b, false, sizeof(b));
22         for (int i = 1; i <= m; i++)
23         {
24             int x = read(), y = read();
25             if (x != y && !b[x][y])
26             {
```

```
27              b[x][y] = true; b[y][x] = true;
28              makelist(x, y); makelist(y, x);
29          }
30      }
31      memset(f, 0, sizeof(f));
32      memset(L, 0, sizeof(L));
33      memset(R, 255, sizeof(R));
34      L[0] = 1; R[0] = n;
35      for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
36      memset(idx, 0, sizeof(idx));
37      memset(v, 0, sizeof(v));
38      for (int i = n; i; --i)
39      {
40          int now = c[i];
41          R[f[now]]--;
42          if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
43          idx[now] = i; v[i] = now;
44          for (int x = first[now]; x; x = next[x])
45              if (!idx[where[x]])
46              {
47                  swap(c[pos[where[x]]], c[R[f[where[x]]]]);
48                  pos[c[pos[where[x]]]] = pos[where[x]];
49                  pos[where[x]] = R[f[where[x]]];
50                  L[f[where[x]] + 1] = R[f[where[x]]]--;
51                  if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
52                  if (R[f[where[x]] + 1] == -1)
53                      R[f[where[x]] + 1] = L[f[where[x]] + 1];
54                  ++f[where[x]];
55              }
56      }
57      bool ok = true;
58      //v 是完美消除序列.
59      for (int i = 1; i <= n && ok; i++)
60      {
61          int cnt = 0;
62          for (int x = first[v[i]]; x; x = next[x])
63              if (idx[where[x]] > i) c[++cnt] = where[x];
64          sort(c + 1, c + cnt + 1, cmp);
65          bool can = true;
66          for (int j = 2; j <= cnt; j++)
67              if (!b[c[1]][c[j]])
68              {
69                  ok = false;
70                  break;
71              }
72      }
73      if (ok) printf("Perfect\n");
74      else printf("Imperfect\n");
75      printf("\n");
```

```
76            }
77    }
```

6.15 弦图求最大团

```
1   int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c[100001],
  ↪   f[100001],
2   pos[100001], idx[100001], v[100001], ans;
3
4   inline void makelist(int x, int y){
5       where[++l] = y;
6       next[l] = first[x];
7       first[x] = l;
8   }
9
10  int read(){
11      char ch;
12      for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar());
13      int cnt = 0;
14      for (; ch >= '0' && ch <= '9'; ch = getchar()) cnt = cnt * 10 + ch - '0';
15      return(cnt);
16  }
17
18  int main(){
19      //freopen("1006.in", "r", stdin);
20      //freopen("1006.out", "w", stdout);
21      memset(first, 0, sizeof(first)); l = 0;
22      n = read(); m = read();
23      for (int i = 1; i <= m; i++)
24      {
25          int x, y;
26          x = read(); y = read();
27          makelist(x, y); makelist(y, x);
28      }
29      memset(L, 0, sizeof(L));
30      memset(R, 255, sizeof(R));
31      memset(f, 0, sizeof(f));
32      memset(idx, 0, sizeof(idx));
33      for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
34      L[0] = 1; R[0] = n; ans = 0;
35      for (int i = n; i; --i)
36      {
37          int now = c[i], cnt = 1;
38          idx[now] = i; v[i] = now;
39          if (--R[f[now]] < L[f[now]]) R[f[now]] = -1;
40          for (int x = first[now]; x; x = next[x])
41              if (!idx[where[x]])
42              {
43                  swap(c[pos[where[x]]], c[R[f[where[x]]]]);
```

```
44                pos[c[pos[where[x]]]] = pos[where[x]];
45                pos[where[x]] = R[f[where[x]]];
46                L[f[where[x]] + 1] = R[f[where[x]]]--;
47                if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
48                if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
49                ++f[where[x]];
50            }
51            else ++cnt;
52        ans = max(ans, cnt);
53    }
54    printf("%d\n", ans);
55 }
```

6.16　最大团搜索

```
1  // mc[i] 代表只用 i-n 号点的答案
2  // g 代表连通性
3  void dfs(int size) {
4    int i, j, k;
5    if (len[size] == 0) {
6      if (size > ans) {
7        ans = size;
8        found = true;
9      }
10     return;
11   }
12   for (k = 0; k < len[size] && !found; k ++) {
13     if (size + len[size] - k <= ans) break;
14     i = list[size][k];
15     if (size + mc[i] <= ans) break;
16     for (j = k + 1, len[size + 1] = 0; j < len[size]; j ++)
17       if (g[i][list[size][j]]) list[size + 1][len[size + 1] ++] = list[size][j];
18     dfs(size + 1);
19     if (found) {
20       prin[size + 1] = i;
21     }
22   }
23 }
24 void work() {
25   int i, j;
26   mc[n] = ans = 1;
27   ansi = 1;
28   for (i = n - 1; i; i --) {
29     found = false;
30     len[1] = 0;
31     for (j = i + 1; j <= n; j ++) if (g[i][j]) list[1][len[1]++] = j;
32     dfs(1);
33     mc[i] = ans;
34     if (found) prin[1] = i;
```

```
35      }
36   }
37   void print() {
38      printf("%d\n", ans);
39      for (int i = 1; i < ans; i ++) printf("%d ", prin[i]);
40      printf("%d\n", prin[ans]);
41   }
```

6.17   极大团计数

```
1    bool g[N][N];
2    int ne[N], ce[N], list[N][N], ans;
3    void dfs(int size) {
4       if (ans > 1000) return;
5       int i, j, k, t, cnt, best = 0;
6       bool bb;
7       if (ne[size] == ce[size]) {
8          if (ce[size] == 0) ++ans;
9          return;
10      }
11      for (t = 0, i = 1; i <= ne[size]; ++i) {
12         for (cnt = 0, j = ne[size] + 1; j <= ce[size]; ++j)
13            if (!g[list[size][i]][list[size][j]]) ++cnt;
14         if (t == 0 || cnt < best) t = i, best = cnt;
15      }
16      if (t && best <= 0) return;
17      for (k = ne[size] + 1; k <= ce[size]; ++k) {
18         if (t > 0) {
19            for (i = k; i <= ce[size]; ++i)
20               if (!g[list[size][t]][list[size][i]]) break;
21            swap(list[size][k], list[size][i]);
22         }
23         i = list[size][k];
24         ne[size + 1] = ce[size + 1] = 0;
25         for (j = 1; j < k; ++j)
26            if (g[i][list[size][j]])
27               list[size + 1][++ne[size + 1]] = list[size][j];
28         for (ce[size + 1] = ne[size + 1], j = k + 1; j <= ce[size]; ++j)
29            if (g[i][list[size][j]]) list[size + 1][++ce[size + 1]] = list[size][j];
30         dfs(size + 1);
31         ++ne[size];
32         --best;
33         for (j = k + 1, cnt = 0; j <= ce[size]; ++j)
34            if (!g[i][list[size][j]]) ++cnt;
35         if (t == 0 || cnt < best) t = k, best = cnt;
36         if (t && best <= 0) break;
37      }
38   }
39   int main(){
```

```
40    int n, m;
41    while (scanf("%d%d", &n, &m) == 2) {
42      for (int i = 1; i <= n; ++i)
43        for (int j = 1; j <= n; ++j)
44          g[i][j] = false;
45      while (m--) {
46        int x, y;
47        scanf("%d%d", &x, &y);
48        g[x][y] = g[y][x] = true;
49      }
50      ne[0] = 0;
51      ce[0] = 0;
52      for (int i = 1; i <= n; ++i)
53        list[0][++ce[0]] = i;
54      ans = 0;
55      dfs(0);
56      if (ans > 1000) puts("Too many maximal sets of friends.");
57      else printf("%d\n", ans);
58    }
59    return 0;
60  }
```

6.18　最小树形图

```
1   int n, m, used[N], pass[N], eg[N], more, queue[N];
2   double g[N][N];
3
4   void combine(int id, double &sum) {
5     int tot = 0, from, i, j, k;
6     for (; id != 0 && !pass[id]; id = eg[id]) {
7       queue[tot++] = id;
8       pass[id] = 1;
9     }
10
11    for (from = 0; from < tot && queue[from] != id; from++);
12    if (from == tot) return;
13    more = 1;
14    for (i = from; i < tot; i++) {
15      sum += g[eg[queue[i]]][queue[i]];
16      if (i != from) {
17        used[queue[i]] = 1;
18        for (j = 1; j <= n; j++) if (!used[j]) {
19          if (g[queue[i]][j] < g[id][j]) g[id][j] = g[queue[i]][j];
20        }
21      }
22    }
23
24    for (i = 1; i <= n; i++) if (!used[i] && i != id) {
25      for (j = from; j < tot; j++) {
```

```
26        k = queue[j];
27        if (g[i][id] > g[i][k] - g[eg[k]][k]) g[i][id] = g[i][k] - g[eg[k]][k];
28      }
29    }
30  }
31
32  double mdst(int root) {
33    int i, j, k;
34    double sum = 0;
35    memset(used, 0, sizeof(used));
36    for (more = 1; more; ) {
37      more = 0;
38      memset(eg, 0, sizeof(eg));
39      for (i = 1; i <= n; i++) if (!used[i] && i != root) {
40        for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
41          if (k == 0 || g[j][i] < g[k][i]) k = j;
42        eg[i] = k;
43      }
44
45      memset(pass, 0, sizeof(pass));
46      for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root) combine(i, sum);
47    }
48
49    for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
50    return sum;
51  }
```

6.19   带花树

```
1   int match[N], belong[N], next[N], mark[N], visit[N];
2   std::vector<int> queue;
3
4   int find(int x) {
5       if (belong[x] != x) {
6           belong[x] = find(belong[x]);
7       }
8       return belong[x];
9   }
10
11  void merge(int x, int y) {
12      x = find(x);
13      y = find(y);
14      if (x != y) {
15          belong[x] = y;
16      }
17  }
18
19  int lca(int x, int y) {
20      static int stamp = 0;
```

75

```
21        stamp++;
22        while (true) {
23            if (x != -1) {
24                x = find(x);
25                if (visit[x] == stamp) {
26                    return x;
27                }
28                visit[x] = stamp;
29                if (match[x] != -1) {
30                    x = next[match[x]];
31                } else {
32                    x = -1;
33                }
34            }
35            std::swap(x, y);
36        }
37    }
38
39    void group(int a, int p) {
40        while (a != p) {
41            int b = match[a], c = next[b];
42            if (find(c) != p) {
43                next[c] = b;
44            }
45            if (mark[b] == 2) {
46                mark[b] = 1;
47                queue.push_back(b);
48            }
49            if (mark[c] == 2) {
50                mark[c] = 1;
51                queue.push_back(c);
52            }
53            merge(a, b);
54            merge(b, c);
55            a = c;
56        }
57    }
58
59    void augment(int source) {
60        queue.clear();
61        for (int i = 0; i < n; ++i) {
62            next[i] = visit[i] = -1;
63            belong[i] = i;
64            mark[i] = 0;
65        }
66        mark[source] = 1;
67        queue.push_back(source);
68        for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
69            int x = queue[head];
```

```cpp
        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;
}
```

6.20 度限制生成树

```
1   int n, m, S, K, ans , cnt , Best[N], fa[N], FE[N];
2   int f[N], p[M], t[M], c[M], o, Cost[N];
3   bool u[M], d[M];
4   pair<int, int> MinCost[N];
5   struct Edge {
6     int a, b, c;
7     bool operator < (const Edge & E) const { return c < E.c; }
8   }E[M];
9   vector<int> SE;
10  inline int F(int x) {
11    return fa[x] == x ? x : fa[x] = F(fa[x]);
12  }
13  inline void AddEdge(int a, int b, int C) {
14    p[++o] = b; c[o] = C;
15    t[o] = f[a]; f[a] = o;
16  }
17  void dfs(int i, int father) {
18    fa[i] = father;
19    if (father == S) Best[i] = -1;
20    else {
21      Best[i] = i;
22      if (~Best[father] && Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
23    }
24    for (int j = f[i]; j; j = t[j])
25    if (!d[j] && p[j] != father) {
26      Cost[p[j]] = c[j];
27      FE[p[j]] = j;
28      dfs(p[j], i);
29    }
30  }
31  inline bool Kruskal() {
32    cnt = n - 1,ans = 0; o = 1;
33    for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
34    sort(E + 1, E + m + 1);
35    for (int i = 1; i <= m; i++) {
36      if (E[i].b == S) swap(E[i].a, E[i].b);
37      if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
38        fa[F(E[i].a)] = F(E[i].b);
39        ans += E[i].c;
40        cnt --;
41        u[i] = true;
42        AddEdge(E[i].a, E[i].b, E[i].c);
43        AddEdge(E[i].b, E[i].a, E[i].c);
44      }
45    }
46    for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
47    for (int i = 1; i <= m; i++)
48    if (E[i].a == S) {
```

```
49        SE.push_back(i);
50        MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
51      }
52      int dif = 0;
53      for (int i = 1; i <= n; i++)
54      if (i != S && fa[i] == i) {
55        if (MinCost[i].second == INF) return false;
56        if (++ dif > K) return false;
57        dfs(E[MinCost[i].second].b, S);
58        u[MinCost[i].second] = true;
59        ans += MinCost[i].first;
60      }
61      return true;
62    }
63  bool Solve() {
64      memset(d,false,sizeof d);
65      memset(u,false,sizeof u);
66      if (!Kruskal()) return false;
67      for (int i = cnt + 1; i <= K && i <= n; i++) {
68        int MinD = INF, MinID = -1;
69        for (int j = (int) SE.size() - 1; j >= 0; j--)
70        if (u[SE[j]])
71          SE.erase(SE.begin() + j);
72        for (int j = 0; j < (int) SE.size(); j++) {
73          int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
74          if (tmp < MinD) {
75            MinD = tmp;
76            MinID= SE[j];
77          }
78        }
79        if (MinID == -1) return true;
80        if (MinD >= 0) break;
81        ans += MinD;
82        u[MinID] = true;
83        d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
84        dfs(E[MinID].b, S);
85      }
86      return true;
87    }
88  int main(){
89      Solve();
90      return 0;
91    }
```

7 字符串

7.1 KMP 算法

```
1  void build(char *pattern) {
2      int length = (int)strlen(pattern + 1);
3      fail[0] = -1;
4      for (int i = 1, j; i <= length; ++i) {
5          for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j = fail[j]);
6          fail[i] = j + 1;
7      }
8  }
9
10 void solve(char *text, char *pattern) {
11     int length = (int)strlen(text + 1);
12     for (int i = 1, j; i <= length; ++i) {
13         for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
14         match[i] = j + 1;
15     }
16 }
17 ///Hint: 1 - Base
```

7.2 扩展 KMP 算法

返回结果:

$$next_i = lcp(text, text_{i \ldots n-1})$$

```
1  void solve(char *text, int length, int *next) {
2      int j = 0, k = 1;
3      for (; j + 1 < length && text[j] == text[j + 1]; j++);
4      next[0] = length - 1;
5      next[1] = j;
6      for (int i = 2; i < length; ++i) {
7          int far = k + next[k] - 1;
8          if (next[i - k] < far - i + 1) {
9              next[i] = next[i - k];
10         } else {
11             j = std::max(far - i + 1, 0);
12             for (; i + j < length && text[j] == text[i + j]; j++);
13             next[i] = j;
14             k = i;
15         }
16     }
17 }
18 /// 0 - Base
```

7.3 AC 自动机

```
1  struct Node{
2    int Next[30], fail, mark;
```

```
3    }Tree[N];

4

5    void Init(){
6        memset(Tree, 0, sizeof Tree);
7        cnt = 1;

8

9        for (int i = 1; i <= n; i++){
10           char c;
11           int now = 1;
12           scanf("%s", s + 1);
13           int Length = strlen(s + 1);
14           for (int j = 1; j <= Length; j++){
15               c = s[j];
16               if (Tree[now].Next[c - 'a']) now = Tree[now].Next[c - 'a']; else
17                   Tree[now].Next[c - 'a'] = ++ cnt, now = cnt;
18           }
19       }
20   }

21

22   void Build_Ac(){
23       int en = 0;
24       Q[0] = 1;
25       for (int fi = 0; fi <= en; fi++){
26           int now = Q[fi];
27           for (int next = 0; next < 26; next++)
28               if (Tree[now].Next[next])
29               {
30                   int k = Tree[now].Next[next];
31                   if (now == 1) Tree[k].fail = 1; else
32                   {
33                       int h = Tree[now].fail;
34                       while (h && !Tree[h].Next[next]) h = Tree[h].fail;
35                       if (!h) Tree[k].fail = 1;
36                       else Tree[k].fail = Tree[h].Next[next];
37                   }
38                   Q[++ en] = k;
39               }
40       }
41   }

42

43   /// Hints : when not match , fail = 1
```

7.4　广义后缀自动机

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。

```
1    void add(int x, int &last) {
2        int lastnode = last;
3        if (c[lastnode][x]) {
4            int nownode = c[lastnode][x];
```

```
5        if (l[nownode] == l[lastnode] + 1) last = nownode;
6        else{
7          int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
8          for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
9          f[auxnode] = f[nownode]; f[nownode] = auxnode;
10         for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
11           c[lastnode][x] = auxnode;
12         }
13         last = auxnode;
14       }
15     }
16     else{ // Naive Suffix Automaton
17       int newnode = ++size; l[newnode] = l[lastnode] + 1;
18       for (; lastnode && !c[lastnode][x]; lastnode = f[lastnode]) c[lastnode][x] = newnode;
19       if (!lastnode) f[newnode] = 1;
20       else{
21         int nownode = c[lastnode][x];
22         if (l[lastnode] + 1 == l[nownode]) f[newnode] = nownode;
23         else{
24           int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
25           for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
26           f[auxnode] = f[nownode]; f[nownode] = f[newnode] = auxnode;
27           for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
28             c[lastnode][x] = auxnode;
29           }
30         }
31       }
32       last = newnode;
33     }
34   }
```

7.5　后缀数组

注意事项： $\mathcal{O}(n \log n)$ 倍增构造。

```
1  namespace suffix_array{
2    int wa[MAXN], wb[MAXN], ws[MAXN], wv[MAXN];
3    bool cmp(int *r, int a, int b, int l) {
4      return r[a] == r[b] && r[a + l] == r[b + l];
5    }
6    void DA(int *r, int *sa, int n, int m) {
7      int *x = wa, *y = wb, *t;
8      for (int i = 0; i < m; i++) ws[i] = 0;
9      for (int i = 0; i < n; i++) ws[x[i] = r[i]]++;
10     for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
11     for (int i = n - 1; i >= 0; i--) sa[--ws[x[i]]] = i;
12     for (int i, j = 1, p = 1; p < n; j <<= 1, m = p) {
13       for (p = 0, i = n - j; i < n; i++) y[p++] = i;
14       for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
15       for (i = 0; i < n; i++) wv[i] = x[y[i]];
```

```
16        for (i = 0; i < m; i++) ws[i] = 0;
17        for (i = 0; i < n; i++) ws[wv[i]]++;
18        for (i = 1; i < m; i++) ws[i] += ws[i - 1];
19        for (i = n - 1; i >= 0; i--) sa[--ws[wv[i]]] = y[i];
20        for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
21          x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
22      }
23    }
24    void getheight(int *r, int *sa, int *rk, int *h, int n) {
25      for (int i = 1; i <= n; i++) rk[sa[i]] = i;
26      for (int i = 0, j, k = 0; i < n; h[rk[i++]] = k)
27        for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);
28    }
29  };
```

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];
3     int allocate(int len) {
4       l[nTree] = len;
5       r[nTree] = 0;
6       fail[nTree] = 0;
7       memset(c[nTree], 0, sizeof(c[nTree]));
8       return nTree++;
9     }
10    void init() {
11      nTree = nStr = 0;
12      int newEven = allocate(0);
13      int newOdd = allocate(-1);
14      last = newEven;
15      fail[newEven] = newOdd;
16      fail[newOdd] = newEven;
17      s[0] = -1;
18    }
19    void add(int x) {
20      s[++nStr] = x;
21      int nownode = last;
22      while (s[nStr - l[nownode] - 1] != s[nStr]) nownode = fail[nownode];
23      if (!c[nownode][x]) {
24        int newnode = allocate(l[nownode] + 2), &newfail = fail[newnode];
25        newfail = fail[nownode];
26        while (s[nStr - l[newfail] - 1] != s[nStr]) newfail = fail[newfail];
27        newfail = c[newfail][x];
28        c[nownode][x] = newnode;
29      }
30      last = c[nownode][x];
31      r[last]++;
```

```
32      }
33    void count() {
34      for (int i = nTree - 1; i >= 0; i--) {
35        r[fail[i]] += r[i];
36      }
37    }
38  }
```

7.7   Manacher

注意事项： 1-based 算法，请注意下表。

```
1  int manacher(char *text, int length, int *palindrome) {
2    static char buffer[MAXN];
3    for (int i = 1; i <= length; i++) {
4      buffer[2 * i - 1] = text[i];
5      if (i != 0) buffer[2 * i] = '#';
6    }
7
8    palindrome[1] = 1;
9    for (int i = 2, j = 0; i <= 2 * length - 1; ++i) {
10     if (j + palindrome[j] <= i) {
11       palindrome[i] = 0;
12     } else {
13       palindrome[i] = std::min(palindrome[(j << 1) - i], j + palindrome[j] - i);
14     }
15     while (i - palindrome[i] >= 1 && i + palindrome[i] <= 2 * length - 1
16         && buffer[i - palindrome[i]] == buffer[i + palindrome[i]]) {
17       palindrome[i]++;
18     }
19     if (i + palindrome[i] > j + palindrome[j]) {
20       j = i;
21     }
22   }
23
24   int answer = 0;
25   for (int i = 1; i < 2 * length; i++) {
26     if (i & 1) answer = std::max(answer, 2 * (palindrome[i] - 1 >> 1) + 1);
27     else answer = std::max(answer, 2 * (palindrome[i] >> 1));
28   }
29   return answer;
30 }
```

7.8   循环串的最小表示

注意事项： 0-Based 算法，请注意下标。

```
1  int minrep(const char *s) {
2    int length = strlen(s);
3    int i = 0, j = 1, k = 0;
```

```
4    while (i < length && j < length && k < length) {
5      if (s[(i + k) % length] == s[(j + k) % length]) {
6        k++;
7      } else {
8        if (s[(i + k) % length] > s[(j + k) % length]) {
9          i += k + 1;
10       } else {
11         j += k + 1;
12       }
13       if (i == j) j++;
14       k = 0;
15     }
16   }
17   return std::min(i, j);
18 }
```

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;

```
1  const int MAXL = 100001; // The length of the string being inserted into the ST.
2  const int MAXD = 27;     // The size of the alphabet.
3
4  struct SuffixTree{
5    int size, length, pCur, dCur, lCur, lBuf, text[MAXL];
6    std::pair<int, int> suffix[MAXL];
7
8    struct Node{
9      int left, right, sLink, next[MAXD];
10   }tree[MAXL * 2];
11
12   int getLength(const int &rhs) {
13     return tree[rhs].right ? tree[rhs].right - tree[rhs].left : length + 1 - tree[rhs].left;
14   }
15   void addLink(int &last, int node) {
16     if (last != 0) tree[last].sLink = node;
17     last = node;
18   }
19   int alloc(int left, int right = 0) {
20     size++;
21     memset(&tree[size], 0, sizeof(tree[size]));
22     tree[size].left = left;
23     tree[size].right = right;
24     tree[size].sLink = 1;
25     return size;
```

```
26        }
27      bool move(int node) {
28        int length = getLength(node);
29        if (lCur >= length) {
30          lCur -= length;
31          dCur += length;
32          pCur = node;
33          return true;
34        }
35        return false;
36      }
37      void init() {
38        size = length = 0;
39        lCur = dCur = lBuf = 0;
40        pCur = alloc(0);
41      }
42      void extend(int x) {
43        text[++length] = x;
44        lBuf++;
45        for (int last = 0; lBuf > 0; ) {
46          if (lCur == 0) dCur = length;
47          if (!tree[pCur].next[text[dCur]]) {
48            int newleaf = alloc(length);
49            tree[pCur].next[text[dCur]] = newleaf;
50            suffix[length + 1 - lBuf] = std::make_pair(pCur, newleaf);
51            addLink(last, pCur);
52          } else {
53            int nownode = tree[pCur].next[text[dCur]];
54            if (move(nownode)) continue;
55            if (text[tree[nownode].left + lCur] == x) {
56              lCur++;
57              addLink(last, pCur);
58              break;
59            }
60            int newleaf = alloc(length), newnode = alloc(tree[nownode].left, tree[nownode].left + lCur);
61            tree[nownode].left += lCur;
62            tree[pCur].next[text[dCur]] = newnode;
63            tree[newnode].next[x] = newleaf;
64            tree[newnode].next[text[tree[nownode].left]] = nownode;
65            suffix[length + 1 - lBuf] = std::make_pair(newnode, newleaf);
66            addLink(last, newnode);
67          }
68          lBuf--;
69          if (pCur == 1 && lCur > 0) lCur--, dCur++;
70          else pCur = tree[pCur].sLink;
71        }
72      }
73    };
```

8 计算几何

8.1 二维几何基础

```cpp
inline int sign(double x) { return x < -EPS ? -1 : x > EPS; }
inline double sqr(double x) { return x * x; }

struct point {
  double x, y;
  point(double x = 0, double y = 0) : x(x), y(y) {}
  inline double length() const { return sqrt(x * x + y * y); }
  inline double norm() const { return length(); }
  inline double norm2() const { return x * x + y * y; }
  inline point unit() const {
    double len = length();
    return point(x / len, y / len);
  }
  inline point negate() const { return point(-x, -y); }
  inline point rot90() const {  // counter - clockwise
    return point(-y, x);
  }
  inline point _rot90() const {  // clockwise
    return point(y, -x);
  }
  inline point rotate(double theta) const {  // counter - clockwise
    double c = cos(theta), s = sin(theta);
    return point(x * c - y * s, x * s + y * c);
  }
  int get() { return scanf("%lf %lf", &x, &y); }
  void out() { printf("(%.5f, %.5f)\n", x, y); }
};

inline bool operator==(const point &a, const point &b) {
  return fabs(a.x - b.x) < EPS && fabs(a.y - b.y) < EPS;
}
inline bool operator!=(const point &a, const point &b) {
  return fabs(a.x - b.x) > EPS || fabs(a.y - b.y) > EPS;
}
inline bool operator<(const point &a, const point &b) {
  if (fabs(a.x - b.x) > EPS) return a.x < b.x;
  return a.y + EPS < b.y;
}
inline point operator+(const point &a, const point &b) {
  return point(a.x + b.x, a.y + b.y);
}
inline point operator-(const point &a, const point &b) {
  return point(a.x - b.x, a.y - b.y);
}
inline point operator*(const point &a, const double &b) {
  return point(a.x * b, a.y * b);
```

```
47    }
48    inline point operator/(const point &a, const double &b) {
49      return point(a.x / b, a.y / b);
50    }
51    inline double det(const point &a, const point &b) {
52      return a.x * b.y - b.x * a.y;
53    }
54    inline double dot(const point &a, const point &b) {
55      return a.x * b.x + a.y * b.y;
56    }
57    inline double dis(const point &a, const point &b) {
58      return sqrt(sqr(a.x - b.x) + sqr(a.y - b.y));
59    }
60    struct line {
61      point s, t;
62      line(point s = point(), point t = point()) : s(s), t(t) {}
63      inline double length() const { return dis(s, t); }
64    };
65    //线段交点
66    //注意如果两条线段是共线的且有交点，那么 intersect_judgement 确实会返回 true,
67    //但是 line_intersect 会求错，所以这种情况需要特判.
68    inline bool point_on_line(const point &a, const line &b) {
69      return sign(det(a - b.s, b.t - b.s)) == 0 && dot(b.s - a, b.t - a) < EPS;
70    }
71    inline bool two_side(const point &a, const point &b, const line &c) {
72      return sign(det(a - c.s, c.t - c.s)) * sign(det(b - c.s, c.t - c.s)) < 0;
73    }
74    inline bool intersect_judgement(const line &a, const line &b) {
75      if (point_on_line(b.s, a) || point_on_line(b.t, a)) return true;
76      if (point_on_line(a.s, b) || point_on_line(a.t, b)) return true;
77      return two_side(a.s, a.t, b) && two_side(b.s, b.t, a);
78    }
79    inline point line_intersect(const line &a, const line &b) {
80      double s1 = det(a.t - a.s, b.s - a.s);
81      double s2 = det(a.t - a.s, b.t - a.s);
82      return (b.s * s2 - b.t * s1) / (s2 - s1);
83    }
84    //点到直线的距离
85    double point_to_line(const point &p, const line &l) {
86      return fabs(det(l.t - l.s, p - l.s)) / dis(l.s, l.t);
87    }
88    inline double min_point_to_line(const point &a, const line &b) {
89      if (dot(b.s - a, b.t - a) < EPS)
90        return fabs(det(b.s - a, b.t - a) / b.length());
91      return min(dis(a, b.s), dis(a, b.t));
92    }
93    //点在多边形内
94    bool in_polygon(const point &p, const vector<point> &poly) {
95      int n = (int)poly.size();
```

```
 96      int counter = 0;
 97      for (int i = 0; i < n; ++i) {
 98        point a = poly[i], b = poly[(i + 1) % n];
 99        if (point_on_line(p, line(a, b))) return false;  // bounded excluded
100        int x = sign(det(p - a, b - a));
101        int y = sign(a.y - p.y);
102        int z = sign(b.y - p.y);
103        if (x > 0 && y <= 0 && z > 0) counter++;
104        if (x < 0 && z <= 0 && y > 0) counter--;
105      }
106      return counter != 0;
107    }
```
//点到直线的投影
```
109    point project_to_line(const point &p, const line &l) {
110      return l.s + (l.t - l.s) * (dot(p - l.s, l.t - l.s) / (l.t - l.s).norm2());
111    }
```
//圆类
```
113    struct circle {
114      point center;
115      double radius;
116      circle(point center = point(), double radius = 0)
117          : center(center), radius(radius) {}
118    };
119    inline bool operator==(const circle &a, const circle &b) {
120      return a.center == b.center && fabs(a.radius - b.radius) < EPS;
121    }
122    inline bool operator!=(const circle &a, const circle &b) {
123      return a.center != b.center || fabs(a.radius - b.radius) > EPS;
124    }
125    inline bool in_circle(const point &p, const circle &c) {
126      return dis(p, c.center) < c.radius + EPS;
127    }
```
//圆的生成函数
```
129    circle make_circle(const point &a, const point &b) {
130      return circle((a + b) / 2, dis(a, b) / 2);
131    }
132    circle make_circle(const point &a, const point &b, const point &c) {
133      point center = circumcenter(a, b, c);
134      return circle(center, dis(center, a));
135    }
```
//点到圆的切线
```
137    pair<line, line> tangent(const point &p, const circle &c) {
138      circle a = make_circle(p, c.center);
139      return make_pair(circle_intersect(a, c), circle_intersect(c, a));
140    }
```
//直线与圆的交点
//返回 AB 方向的第一个交点
```
143    point line_circle_intersect(const line &l, const circle &c) {
144      double x = sqrt(sqr(c.radius) - sqr(point_to_line(c.center, l)));
```

```
145    return project_to_line(c.center, l) + (l.s - l.t).unit() * x;
146  }
147  //圆与圆的交点
148  point circle_intersect(const circle &a, const circle &b) {  // get another point
149    using circle_intersect(b, a) point r = (b.center - a.center).unit();
150    double d = dis(a.center, b.center);
151    double x = .5 * ((sqr(a.radius) - sqr(b.radius)) / d + d);
152    double h = sqrt(sqr(a.radius) - sqr(x));
153    return a.center + r * x + r.rot90() * h;
154  }
```

8.2 快速凸包

```
1   //水平序凸包
2   inline bool turn_left(const point &a, const point &b, const point &c) {
3     return det(b - a, c - a) > EPS;
4   }
5   inline bool turn_right(const point &a, const point &b, const point &c) {
6     return det(b - a, c - a) < -EPS;
7   }
8   inline vector<point> convex_hull(vector<point> a) {
9     int n = (int)a.size(), cnt = 0;
10    sort(a.begin(), a.end());
11    vector<point> ret;
12    for (int i = 0; i < n; ++i) {
13      while (cnt > 1 && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
14        --cnt;
15        ret.pop_back();
16      }
17      ret.push_back(a[i]);
18      ++cnt;
19    }
20    int fixed = cnt;
21    for (int i = n - 1; i >= 0; --i) {
22      while (cnt > fixed && turn_left(ret[cnt - 2], a[i], ret[cnt - 1])) {
23        --cnt;
24        ret.pop_back();
25      }
26      ret.push_back(a[i]);
27      ++cnt;
28    }
29    // this algorithm will preserve the points which are collineation
30    // the lowest point will occur twice , i.e. ret.front () == ret.back ()
31    return ret;
32  }
```

8.3 半平面交

```cpp
//半平面交
inline bool two_side(const point &a, const point &b, const line &c) {
  return sign(det(a - c.s, c.t - c.s)) * sign(det(b - c.s, c.t - c.s)) < 0;
}
vector<point> cut(const vector<point> &c, line p) {
  vector<point> ret;
  if (c.empty()) return ret;
  for (int i = 0; i < (int)c.size(); ++i) {
    int j = (i + 1) % (int)c.size();
    if (!turn_right(p.s, p.t, c[i])) ret.push_back(c[i]);
    if (two_side(c[i], c[j], p))
      ret.push_back(line_intersubsection(p, line(c[i], c[j])));
  }
  return ret;
}
static const double BOUND = 1e5;
/*
convex .clear ();
convex . push_back ( point (-BOUND , -BOUND ));
convex . push_back ( point (BOUND , -BOUND ));
convex . push_back ( point (BOUND , -BOUND ));
convex . push_back ( point (BOUND , -BOUND ));
convex = cut(convex , line(point , point));
Judgement : convex . empty ();
*/
//高效半平面交
// plane[] 按照法向量 (逆时针 90 度) 极角排序，去除平行半平面
inline bool turn_left(const line &l, const point &p) {
  return turn_left(l.s, l.t, p);
}
vector<line> half_plane_intersect(const vector<line> &h) {
  int fore = 0, rear = -1;
  vector<line> ret;
  for (int i = 0; i < (int)h.size(); ++i) {
    while (fore < rear &&
           !turn_left(h[i], line_intersect(ret[rear - 1], ret[rear])))
      --rear;
    while (fore < rear &&
           !turn_left(h[i], line_intersect(ret[fore], ret[fore + 1])))
      ++fore;
    ++rear;
    ret.push_back(h[i]);
  }
  while (rear - fore > 1 &&
         !turn_left(ret[fore], line_intersect(ret[rear - 1], ret[rear])))
    --rear;
  while (rear - fore > 1 &&
         !turn_left(ret[rear], line_intersect(ret[fore], ret[fore + 1])))
```

```
49        ++fore;
50    if (rear - fore < 2) return vector<line>();
51    return ret;
52    }
```

8.4 三角形的心

```
1    //三角形的内心
2    point incenter(const point &a, const point &b, const point &c) {
3        double p = (a - b).length() + (b - c).length() + (c - a).length();
4        return (a * (b - c).length() + b * (c - a).length() + c * (a - b).length()) /
5                p;
6    }
7    //三角形的外心
8    point circumcenter(const point &a, const point &b, const point &c) {
9        point p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
10        double d = det(p, q);
11        return a + point(det(s, point(p.y, q.y)), det(point(p.x, q.x), s)) / d;
12    }
13    //三角形的垂心
14    point orthocenter(const point &a, const point &b, const point &c) {
15        return a + b + c - circumcenter(a, b, c) * 2.0;
16    }
```

8.5 圆与多边形面积交

```
1    // 求扇形面积
2    double getSectorArea(const Point &a, const Point &b, const double &r) {
3        double c = (2.0 * r * r - sqrdist(a, b)) / (2.0 * r * r);
4        double alpha = acos(c);
5        return r * r * alpha / 2.0;
6    }
7    // 求二次方程 ax^2 + bx + c = 0 的解
8    std::pair<double, double> getSolution(const double &a, const double &b, const double &c) {
9        double delta = b * b - 4.0 * a * c;
10        if (dcmp(delta) < 0) return std::make_pair(0, 0);
11        else return std::make_pair((-b - sqrt(delta)) / (2.0 * a), (-b + sqrt(delta)) / (2.0 * a));
12    }
13    // 直线与圆的交点
14    std::pair<Point, Point> getIntersection(const Point &a, const Point &b, const double &r) {
15        Point d = b - a;
16        double A = dot(d, d);
17        double B = 2.0 * dot(d, a);
18        double C = dot(a, a) - r * r;
19        std::pair<double, double> s = getSolution(A, B, C);
20        return std::make_pair(a + d * s.first, a + d * s.second);
21    }
22    // 原点到线段 AB 的距离
23    double getPointDist(const Point &a, const Point &b) {
```

```
24    Point d = b - a;
25    int sA = dcmp(dot(a, d)), sB = dcmp(dot(b, d));
26    if (sA * sB <= 0) return det(a, b) / dist(a, b);
27    else return std::min(dist(a), dist(b));
28  }
29  // a 和 b 和原点组成的三角形与半径为 r 的圆的交的面积
30  double getArea(const Point &a, const Point &b, const double &r) {
31    double dA = dot(a, a), dB = dot(b, b), dC = getPointDist(a, b), ans = 0.0;
32    if (dcmp(dA - r * r) <= 0 && dcmp(dB - r * r) <= 0) return det(a, b) / 2.0;
33    Point tA = a / dist(a) * r;
34    Point tB = b / dist(b) * r;
35    if (dcmp(dC - r) > 0) return getSectorArea(tA, tB, r);
36    std::pair<Point, Point> ret = getIntersection(a, b, r);
37    if (dcmp(dA - r * r) > 0 && dcmp(dB - r * r) > 0) {
38      ans += getSectorArea(tA, ret.first, r);
39      ans += det(ret.first, ret.second) / 2.0;
40      ans += getSectorArea(ret.second, tB, r);
41      return ans;
42    }
43    if (dcmp(dA - r * r) > 0) return det(ret.first, b) / 2.0 + getSectorArea(tA, ret.first, r);
44    else return det(a, ret.second) / 2.0 + getSectorArea(ret.second, tB, r);
45  }
46  // 求圆与多边形的交的主过程
47  double getArea(int n, Point *p, const Point &c, const double r)  {
48    double ret = 0.0;
49    for (int i = 0; i < n; i++) {
50      int sgn = dcmp(det(p[i] - c, p[(i + 1) % n] - c));
51      if (sgn > 0) ret += getArea(p[i] - c, p[(i + 1) % n] - c, r);
52      else ret -= getArea(p[(i + 1) % n] - c, p[i] - c, r);
53    }
54    return fabs(ret);
55  }
```

8.6　圆并求面积

注意事项： 复杂度 $\mathcal{O}(n^2 \log n)$

```
1   struct arc {
2     double theta;
3     int delta;
4     point p;
5     arc(){};
6     arc(const double &theta, const point &p, int d)
7         : theta(theta), p(p), delta(d) {}
8   };
9
10  vector<arc> vec;
11  vector<double> ans;
12  vector<point> center;
13  int cnt = 0;
```

```
14
15  inline bool operator<(const arc &a, const arc &b) {
16    return a.theta + EPS < b.theta;
17  }
18
19  inline void psh(const double t1, const point p1, const double t2,
20                  const point p2) {
21    if (t2 + EPS < t1) cnt++;
22    vec.push_back(arc(t1, p1, 1));
23    vec.push_back(arc(t2, p2, -1));
24  }
25
26  inline double cub(const double &x) { return x * x * x; }
27  inline void combine(int d, const double &area, const point &o) {
28    if (sign(area) == 0) return;
29    center[d] = (center[d] * ans[d] + o * area) * (1 / (ans[d] + area));
30    ans[d] += area;
31  }
32
33  void area(vector<circle> &cir) {
34    int n = cir.size();
35    vector<bool> f;
36    f.resize(n);
37    vec.clear();
38    cnt = 0;
39    for (int i = 0; i < n; i++) {
40      f[i] = true;
41      for (int j = 0; j < n; j++)
42        if (i != j) {
43          if ((cir[i] == cir[j] && i < j) ||
44              (cir[i] != cir[j] && cir[i].radius < cir[j].radius + EPS &&
45               (cir[i].center - cir[j].center).length() <
46                  fabs(cir[i].radius - cir[j].radius) + EPS)) {
47            f[i] = false;
48            break;
49          }
50        }
51    }
52    int n1 = 0;
53    for (int i = 0; i < n; i++)
54      if (f[i]) cir[n1++] = cir[i];
55    n = n1;
56    ans.clear();
57    center.clear();
58    ans.resize(n + 1);
59    center.resize(n + 1);
60    point dvd;
61    for (int i = 0; i < n; i++) {
62      dvd = cir[i].center - point(cir[i].radius, 0);
```

```
63      vec.clear();
64      vec.push_back(arc(-PI, dvd, 1));
65      cnt = 0;
66      for (int j = 0; j < n; j++)
67        if (j != i) {
68          double d = (cir[j].center - cir[i].center).norm2();
69          if (d < sqr(cir[j].radius - cir[i].radius) + EPS) {
70            if (cir[i].radius + i * EPS < cir[j].radius + j * EPS)
71              psh(-PI, dvd, PI, dvd);
72          } else if (d + EPS < sqr(cir[j].radius + cir[i].radius)) {
73            double lambda =
74                0.5 * (1 + (sqr(cir[i].radius) - sqr(cir[j].radius)) / d);
75            point cp(cir[i].center + (cir[j].center - cir[i].center) * lambda);
76            point nor((cir[j].center - cir[i].center)._rot90().unit() *
77                (sqrt(sqr(cir[i].radius) - (cp - cir[i].center).norm2())));
78            point frm(cp + nor);
79            point to(cp - nor);
80            psh(atan2((frm - cir[i].center).y, (frm - cir[i].center).x), frm,
81                atan2((to - cir[i].center).y, (to - cir[i].center).x), to);
82          }
83        }
84      sort(vec.begin() + 1, vec.end());
85      vec.push_back(arc(PI, dvd, -1));
86      for (int j = 0; j + 1 < vec.size(); j++) {
87        cnt += vec[j].delta;
88        double theta(vec[j + 1].theta - vec[j].theta);
89        double area(sqr(cir[i].radius) * theta * 0.5);
90        combine(cnt, area, cir[i].center +
91                            point(sin(vec[j + 1].theta) - sin(vec[j].theta),
92                                cos(vec[j].theta) - cos(vec[j + 1].theta)) *
93                              (1. / area / 3 * cub(cir[i].radius)));
94        combine(cnt, -sqr(cir[i].radius) * sin(theta) * 0.5,
95              (cir[i].center + vec[j].p + vec[j + 1].p) / 3.);
96        combine(cnt, det(vec[j].p, vec[j + 1].p) * 0.5,
97              (vec[j].p + vec[j + 1].p) / 3.);
98      }
99    }
100 }
```

8.7 最小覆盖圆

```
1   circle minimum_circle(vector<point> p) {
2     circle ret;
3     random_shuffle(p.begin(), p.end());
4     for (int i = 0; i < (int)p.size(); ++i)
5       if (!in_circle(p[i], ret)) {
6         ret = circle(p[i], 0);
7         for (int j = 0; j < i; ++j)
8           if (!in_circle(p[j], ret)) {
```

```
 9          ret = make_circle(p[j], p[i]);
10          for (int k = 0; k < j; ++k)
11            if (!in_circle(p[k], ret)) ret = make_circle(p[i], p[j], p[k]);
12        }
13      }
14    return ret;
15  }
```

8.8   最小覆盖球

```
 1  double eps(1e-8);
 2  int sign(const double & x) {
 3    return (x > eps) - (x + eps < 0);
 4  }
 5  bool equal(const double & x, const double & y) {
 6    return x + eps > y and y + eps > x;
 7  }
 8  struct Point {
 9    double x, y, z;
10    Point() {
11    }
12    Point(const double & x, const double & y, const double & z) : x(x), y(y), z(z){
13    }
14    void scan() {
15      scanf("%lf%lf%lf", &x, &y, &z);
16    }
17    double sqrlen() const {
18      return x * x + y * y + z * z;
19    }
20    double len() const {
21      return sqrt(sqrlen());
22    }
23    void print() const {
24      printf("(%lf %lf %lf)\n", x, y, z);
25    }
26  } a[33];
27  Point operator + (const Point & a, const Point & b) {
28    return Point(a.x + b.x, a.y + b.y, a.z + b.z);
29  }
30  Point operator - (const Point & a, const Point & b) {
31    return Point(a.x - b.x, a.y - b.y, a.z - b.z);
32  }
33  Point operator * (const double & x, const Point & a) {
34    return Point(x * a.x, x * a.y, x * a.z);
35  }
36  double operator % (const Point & a, const Point & b) {
37    return a.x * b.x + a.y * b.y + a.z * b.z;
38  }
39  Point operator * (const Point & a, const Point & b) {
```

```cpp
40      return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
41    }
42    struct Circle {
43      double r;
44      Point o;
45      Circle() {
46        o.x = o.y = o.z = r = 0;
47      }
48      Circle(const Point & o, const double & r) : o(o), r(r) {
49      }
50      void scan() {
51        o.scan();
52        scanf("%lf", &r);
53      }
54      void print() const {
55        o.print();
56        printf("%lf\n", r);
57      }
58    };
59    struct Plane {
60      Point nor;
61      double m;
62      Plane(const Point & nor, const Point & a) : nor(nor){
63        m = nor % a;
64      }
65    };
66    Point intersect(const Plane & a, const Plane & b, const Plane & c) {
67      Point 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);
68      return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
69    }
70    bool in(const Point & a, const Circle & b) {
71      return sign((a - b.o).len() - b.r) <= 0;
72    }
73    bool operator < (const Point & a, const Point & b) {
74      if(!equal(a.x, b.x)) {
75        return a.x < b.x;
76      }
77      if(!equal(a.y, b.y)) {
78        return a.y < b.y;
79      }
80      if(!equal(a.z, b.z)) {
81        return a.z < b.z;
82      }
83      return false;
84    }
85    bool operator == (const Point & a, const Point & b) {
86      return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
87    }
```

```cpp
vector<Point> 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 {
    Point 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() {
  int n;
  for(;;) {
    scanf("%d", &n);
    if(!n) {
      break;
    }
    for(int i(0); i < n; i++) {
      a[i].scan();
    }
    sort(a, a + n);
    n = unique(a, a + n) - a;
```

```
136        vec.clear();
137        printf("%.10f\n", miniBall(n).r);
138    }
139 }
```

8.9 三维几何基础

```
1  int dcmp(const double &x) {
2      return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1);
3  }
4
5  struct TPoint{
6      double x, y, z;
7      TPoint() {}
8      TPoint(double x, double y, double z) : x(x), y(y), z(z) {}
9      TPoint operator +(const TPoint &p)const {
10         return TPoint(x + p.x, y + p.y, z + p.z);
11     }
12     TPoint operator -(const TPoint &p)const {
13         return TPoint(x - p.x, y - p.y, z - p.z);
14     }
15     TPoint operator *(const double &p)const {
16         return TPoint(x * p, y * p, z * p);
17     }
18     TPoint operator /(const double &p)const {
19         return TPoint(x / p, y / p, z / p);
20     }
21     bool operator <(const TPoint &p)const {
22         int dX = dcmp(x - p.x), dY = dcmp(y - p.y), dZ = dcmp(z - p.z);
23         return dX < 0 || (dX == 0 && (dY < 0 || (dY == 0 && dZ < 0)));
24     }
25     bool read() {
26         return scanf("%lf%lf%lf", &x, &y, &z) == 3;
27     }
28 };
29
30 double sqrdist(const TPoint &a) {
31     double ret = 0;
32     ret += a.x * a.x;
33     ret += a.y * a.y;
34     ret += a.z * a.z;
35     return ret;
36 }
37 double sqrdist(const TPoint &a, const TPoint &b) {
38     double ret = 0;
39     ret += (a.x - b.x) * (a.x - b.x);
40     ret += (a.y - b.y) * (a.y - b.y);
41     ret += (a.z - b.z) * (a.z - b.z);
42     return ret;
```

```
43    }
44    double dist(const TPoint &a) {
45      return sqrt(sqrdist(a));
46    }
47    double dist(const TPoint &a, const TPoint &b) {
48      return sqrt(sqrdist(a, b));
49    }
50    TPoint det(const TPoint &a, const TPoint &b) {
51      TPoint ret;
52      ret.x = a.y * b.z - b.y * a.z;
53      ret.y = a.z * b.x - b.z * a.x;
54      ret.z = a.x * b.y - b.x * a.y;
55      return ret;
56    }
57    double dot(const TPoint &a, const TPoint &b) {
58      double ret = 0;
59      ret += a.x * b.x;
60      ret += a.y * b.y;
61      ret += a.z * b.z;
62      return ret;
63    }
64    double detdot(const TPoint &a, const TPoint &b, const TPoint &c, const TPoint &d) {
65      return dot(det(b - a, c - a), d - a);
66    }
```

8.10 三维凸包

```
1     struct Triangle{
2       TPoint a, b, c;
3       Triangle() {}
4       Triangle(TPoint a, TPoint b, TPoint c) : a(a), b(b), c(c) {}
5       double getArea() {
6         TPoint ret = det(b - a, c - a);
7         return dist(ret) / 2.0;
8       }
9     };
10    namespace Convex_Hull {
11      struct Face{
12        int a, b, c;
13        bool isOnConvex;
14        Face() {}
15        Face(int a, int b, int c) : a(a), b(b), c(c) {}
16      };
17
18      int nFace, left, right, whe[MAXN][MAXN];
19      Face queue[MAXF], tmp[MAXF];
20
21      bool isVisible(const std::vector<TPoint> &p, const Face &f, const TPoint &a) {
22        return dcmp(detdot(p[f.a], p[f.b], p[f.c], a)) > 0;
```

```cpp
    }

    bool init(std::vector<TPoint> &p) {
      bool check = false;
      for (int i = 1; i < (int)p.size(); i++) {
        if (dcmp(sqrdist(p[0], p[i]))) {
          std::swap(p[1], p[i]);
          check = true;
          break;
        }
      }
      if (!check) return false;
      check = false;
      for (int i = 2; i < (int)p.size(); i++) {
        if (dcmp(sqrdist(det(p[i] - p[0], p[1] - p[0])))) {
          std::swap(p[2], p[i]);
          check = true;
          break;
        }
      }
      if (!check) return false;
      check = false;
      for (int i = 3; i < (int)p.size(); i++) {
        if (dcmp(detdot(p[0], p[1], p[2], p[i]))) {
          std::swap(p[3], p[i]);
          check = true;
          break;
        }
      }
      if (!check) return false;
      for (int i = 0; i < (int)p.size(); i++)
        for (int j = 0; j < (int)p.size(); j++) {
          whe[i][j] = -1;
        }
      return true;
    }

    void pushface(const int &a, const int &b, const int &c) {
      nFace++;
      tmp[nFace] = Face(a, b, c);
      tmp[nFace].isOnConvex = true;
      whe[a][b] = nFace;
      whe[b][c] = nFace;
      whe[c][a] = nFace;
    }

    bool deal(const std::vector<TPoint> &p, const std::pair<int, int> &now, const TPoint &base) {
      int id = whe[now.second][now.first];
      if (!tmp[id].isOnConvex) return true;
```

```
72      if (isVisible(p, tmp[id], base)) {
73        queue[++right] = tmp[id];
74        tmp[id].isOnConvex = false;
75        return true;
76      }
77      return false;
78    }
79
80    std::vector<Triangle> getConvex(std::vector<TPoint> &p) {
81      static std::vector<Triangle> ret;
82      ret.clear();
83      if (!init(p)) return ret;
84      if (!isVisible(p, Face(0, 1, 2), p[3])) pushface(0, 1, 2); else pushface(0, 2, 1);
85      if (!isVisible(p, Face(0, 1, 3), p[2])) pushface(0, 1, 3); else pushface(0, 3, 1);
86      if (!isVisible(p, Face(0, 2, 3), p[1])) pushface(0, 2, 3); else pushface(0, 3, 2);
87      if (!isVisible(p, Face(1, 2, 3), p[0])) pushface(1, 2, 3); else pushface(1, 3, 2);
88      for (int a = 4; a < (int)p.size(); a++) {
89        TPoint base = p[a];
90        for (int i = 1; i <= nFace; i++) {
91          if (tmp[i].isOnConvex && isVisible(p, tmp[i], base)) {
92            left = 0, right = 0;
93            queue[++right] = tmp[i];
94            tmp[i].isOnConvex = false;
95            while (left < right) {
96              Face now = queue[++left];
97              if (!deal(p, std::make_pair(now.a, now.b), base)) pushface(now.a, now.b, a);
98              if (!deal(p, std::make_pair(now.b, now.c), base)) pushface(now.b, now.c, a);
99              if (!deal(p, std::make_pair(now.c, now.a), base)) pushface(now.c, now.a, a);
100           }
101           break;
102         }
103       }
104     }
105     for (int i = 1; i <= nFace; i++) {
106       Face now = tmp[i];
107       if (now.isOnConvex) {
108         ret.push_back(Triangle(p[now.a], p[now.b], p[now.c]));
109       }
110     }
111     return ret;
112   }
113 };
114
115 int n;
116 std::vector<TPoint> p;
117 std::vector<Triangle> answer;
118
119 int main() {
120   scanf("%d", &n);
```

```
121    for (int i = 1; i <= n; i++) {
122      TPoint a;
123      a.read();
124      p.push_back(a);
125    }
126    answer = Convex_Hull::getConvex(p);
127    double areaCounter = 0.0;
128    for (int i = 0; i < (int)answer.size(); i++) {
129      areaCounter += answer[i].getArea();
130    }
131    printf("%.3f\n", areaCounter);
132    return 0;
133  }
```

8.11 三维绕轴旋转

注意事项： 逆时针绕轴 AB 旋转 θ 角。

```
1   Matrix getTrans(const double &a, const double &b, const double &c) {
2       Matrix ret;
3       ret.a[0][0] = 1; ret.a[0][1] = 0; ret.a[0][2] = 0; ret.a[0][3] = 0;
4       ret.a[1][0] = 0; ret.a[1][1] = 1; ret.a[1][2] = 0; ret.a[1][3] = 0;
5       ret.a[2][0] = 0; ret.a[2][1] = 0; ret.a[2][2] = 1; ret.a[2][3] = 0;
6       ret.a[3][0] = a; ret.a[3][1] = b; ret.a[3][2] = c; ret.a[3][3] = 1;
7       return ret;
8   }
9   Matrix getRotate(const double &a, const double &b, const double &c, const double &theta) {
10      Matrix ret;
11      ret.a[0][0] = a * a * (1 - cos(theta)) + cos(theta);
12      ret.a[0][1] = a * b * (1 - cos(theta)) + c * sin(theta);
13      ret.a[0][2] = a * c * (1 - cos(theta)) - b * sin(theta);
14      ret.a[0][3] = 0;
15
16      ret.a[1][0] = b * a * (1 - cos(theta)) - c * sin(theta);
17      ret.a[1][1] = b * b * (1 - cos(theta)) + cos(theta);
18      ret.a[1][2] = b * c * (1 - cos(theta)) + a * sin(theta);
19      ret.a[1][3] = 0;
20
21      ret.a[2][0] = c * a * (1 - cos(theta)) + b * sin(theta);
22      ret.a[2][1] = c * b * (1 - cos(theta)) - a * sin(theta);
23      ret.a[2][2] = c * c * (1 - cos(theta)) + cos(theta);
24      ret.a[2][3] = 0;
25
26      ret.a[3][0] = 0;
27      ret.a[3][1] = 0;
28      ret.a[3][2] = 0;
29      ret.a[3][3] = 1;
30      return ret;
31  }
```

```
32  Matrix getRotate(const double &ax, const double &ay, const double &az, const double &bx, const
    ↪    double &by, const double &bz, const double &theta) {
33      double l = dist(Point(0, 0, 0), Point(bx, by, bz));
34      Matrix ret = getTrans(-ax, -ay, -az);
35      ret = ret * getRotate(bx / l, by / l, bz / l, theta);
36      ret = ret * getTrans(ax, ay, az);
37      return ret;
38  }
```

8.12 Delaunay 三角剖分

```
1   /*
2   Delaunay Triangulation 随机增量算法 :
3   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
4   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)
5   Triangulation::find 返回包含某点的三角形
6   Triangulation::add_point 将某点加入三角剖分
7   某个 Triangle 在三角剖分中当且仅当它的 has_children 为 0
8   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]
9   */
10  const int N = 100000 + 5, MAX_TRIS = N * 6;
11  const double EPSILON = 1e-6, PI = acos(-1.0);
12  struct Point {
13    double x,y; Point():x(0),y(0){} Point(double x, double y):x(x),y(y){}
14    bool operator ==(Point const& that)const {return x==that.x&&y==that.y;}
15  };
16  inline double sqr(double x) { return x*x; }
17  double dist_sqr(Point const& a, Point const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
18  bool in_circumcircle(Point const& p1, Point const& p2, Point const& p3, Point const& p4) {
19    double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;
20    double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
21    double u13 = sqr(p1.x) - sqr(p4.x) + sqr(p1.y) - sqr(p4.y);
22    double u23 = sqr(p2.x) - sqr(p4.x) + sqr(p2.y) - sqr(p4.y);
23    double u33 = sqr(p3.x) - sqr(p4.x) + sqr(p3.y) - sqr(p4.y);
24    double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 + u11*u22*u33;
25    return det > EPSILON;
26  }
27  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);}
28  typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
29  struct Edge {
30    TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
31    Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
32  };
33  struct Triangle {
34    Point p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
35    Triangle(Point const& p0, Point const& p1, Point const& p2) {
36      p[0]=p0;p[1]=p1;p[2]=p2;children[0]=children[1]=children[2]=0;
37    }
```

```
38      bool has_children() const { return children[0] != 0; }
39      int num_children() const {
40        return children[0] == 0 ? 0
41          : children[1] == 0 ? 1
42          : children[2] == 0 ? 2 : 3;
43      }
44      bool contains(Point const& q) const {
45        double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
46        return a >= -EPSILON && b >= -EPSILON && c >= -EPSILON;
47      }
48    } triange_pool[MAX_TRIS], *tot_triangles;
49    void set_edge(Edge a, Edge b) {
50      if (a.tri) a.tri->edge[a.side] = b;
51      if (b.tri) b.tri->edge[b.side] = a;
52    }
53    class Triangulation {
54      public:
55        Triangulation() {
56          const double LOTS = 1e6;
57          the_root = new(tot_triangles++)
    ↪   Triangle(Point(-LOTS,-LOTS),Point(+LOTS,-LOTS),Point(0,+LOTS));
58        }
59        TriangleRef find(Point p) const { return find(the_root,p); }
60        void add_point(Point const& p) { add_point(find(the_root,p),p); }
61      private:
62        TriangleRef the_root;
63        static TriangleRef find(TriangleRef root, Point const& p) {
64          for( ; ; ) {
65            if (!root->has_children()) return root;
66            else for (int i = 0; i < 3 && root->children[i] ; ++i)
67                if (root->children[i]->contains(p))
68                  {root = root->children[i]; break;}
69          }
70        }
71        void add_point(TriangleRef root, Point const& p) {
72          TriangleRef tab,tbc,tca;
73          tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
74          tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
75          tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
76          set_edge(Edge(tab,0),Edge(tbc,1));set_edge(Edge(tbc,0),Edge(tca,1));
77          set_edge(Edge(tca,0),Edge(tab,1));set_edge(Edge(tab,2),root->edge[2]);
78          set_edge(Edge(tbc,2),root->edge[0]);set_edge(Edge(tca,2),root->edge[1]);
79          root->children[0]=tab;root->children[1]=tbc;root->children[2]=tca;
80          flip(tab,2); flip(tbc,2); flip(tca,2);
81        }
82        void flip(TriangleRef tri, SideRef pi) {
83          TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
84          if(!trj||!in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
85          TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj], tri->p[pi]);
```

```
86        TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi], trj->p[pj]);
87        set_edge(Edge(trk,0), Edge(trl,0));
88        set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
89        set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
90        tri->children[0]=trk;tri->children[1]=trl;tri->children[2]=0;
91        trj->children[0]=trk;trj->children[1]=trl;trj->children[2]=0;
92        flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
93      }
94    };
95    int n; Point ps[N];
96    void build(){
97      tot_triangles = triange_pool; cin >> n;
98      for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
99      random_shuffle(ps, ps + n); Triangulation tri;
100     for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
101   }
```

9 其他

9.1 斯坦纳树

```
1   priority_queue<pair<int, int> > Q;
2
3   // m is key point
4   // n is all point
5
6   for (int s = 0; s < (1 << m); s++){
7     for (int i = 1; i <= n; i++){
8       for (int s0 = (s&(s-1)); s0 ; s0=(s&(s0-1))){
9           f[s][i] = min(f[s][i], f[s0][i] + f[s - s0][i]);
10      }
11    }
12    for (int i = 1; i <= n; i++) vis[i] = 0;
13      while (!Q.empty()) Q.pop();
14    for (int i = 1; i <= n; i++){
15      Q.push(mp(-f[s][i], i));
16    }
17    while (!Q.empty()){
18      while (!Q.empty() && Q.top().first != -f[s][Q.top().second]) Q.pop();
19        if (Q.empty()) break;
20      int Cur = Q.top().second; Q.pop();
21      for (int p = g[Cur]; p; p = nxt[p]){
22        int y = adj[p];
23        if ( f[s][y] > f[s][Cur] + 1){
24            f[s][y] = f[s][Cur] + 1;
25            Q.push(mp(-f[s][y], y));
26        }
27      }
28    }
```

```
29    }
```

9.2 无敌的读入优化

```
1   namespace Reader {
2     const int L = (1 << 20) + 5;
3     char buffer[L], *S, *T;
4     __inline bool getchar(char &ch) {
5       if (S == T) {
6         T = (S = buffer) + fread(buffer, 1, L, stdin);
7         if (S == T) {
8           ch = EOF;
9           return false;
10        }
11      }
12      ch = *S ++;
13      return true;
14    }
15    __inline bool getint(int &x) {
16      char ch;
17      for (; getchar(ch) && (ch < '0' || ch > '9'); );
18      if (ch == EOF) return false;
19      x = ch - '0';
20      for (; getchar(ch), ch >= '0' && ch <= '9'; )
21        x = x * 10 + ch - '0';
22      return true;
23    }
24  }
25  Reader::getint(x);
26  Reader::getint(y);
```

9.3 最小树形图

```
1   const int maxn=1100;
2
3   int n,m , g[maxn][maxn] , used[maxn] , pass[maxn] , eg[maxn] , more , queue[maxn];
4
5   void combine (int id , int &sum ) {
6     int tot = 0 , from , i , j , k ;
7     for ( ; id!=0 && !pass[ id ] ; id=eg[id] ) {
8       queue[tot++]=id ; pass[id]=1;
9     }
10    for ( from=0; from<tot && queue[from]!=id ; from++);
11    if ( from==tot ) return ;
12    more = 1 ;
13    for ( i=from ; i<tot ; i++) {
14      sum+=g[eg[queue[i]]][queue[i]] ;
15      if ( i!=from ) {
16        used[queue[i]]=1;
```

```
17        for ( j = 1 ; j <= n ; j++) if ( !used[j] )
18          if ( g[queue[i]][j]<g[id][j] ) g[id][j]=g[queue[i]][j] ;
19      }
20    }
21    for ( i=1; i<=n ; i++) if ( !used[i] && i!=id ) {
22      for ( j=from ; j<tot ; j++){
23        k=queue[j];
24        if ( g[i][id]>g[i][k]-g[eg[k]][k] ) g[i][id]=g[i][k]-g[eg[k]][k];
25      }
26    }
27  }
28
29  int mdst( int root ) { // return the total length of MDST
30    int i , j , k , sum = 0 ;
31    memset ( used , 0 , sizeof ( used ) ) ;
32    for ( more =1; more ; ) {
33      more = 0 ;
34      memset (eg,0,sizeof(eg)) ;
35      for ( i=1 ; i <= n ; i ++) if ( !used[i] && i!=root ) {
36        for ( j=1 , k=0 ; j <= n ; j ++) if ( !used[j] && i!=j )
37          if ( k==0 || g[j][i] < g[k][i] ) k=j ;
38        eg[i] = k ;
39      }
40      memset(pass,0,sizeof(pass));
41      for ( i=1; i<=n ; i++) if ( !used[i] && !pass[i] && i!= root ) combine ( i , sum ) ;
42    }
43    for ( i =1; i<=n ; i ++) if ( !used[i] && i!= root ) sum+=g[eg[i]][i];
44    return sum ;
45  }
```

9.4 DLX

```
1   int n,m,K;
2   struct DLX{
3     int L[maxn],R[maxn],U[maxn],D[maxn];
4     int sz,col[maxn],row[maxn],s[maxn],H[maxn];
5     bool vis[233];
6     int ans[maxn],cnt;
7     void init(int m){
8       for(int i=0;i<=m;i++){
9         L[i]=i-1;R[i]=i+1;
10        U[i]=D[i]=i;s[i]=0;
11      }
12      memset(H,-1,sizeof H);
13      L[0]=m;R[m]=0;sz=m+1;
14    }
15    void Link(int r,int c){
16      U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
17      if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
```

```
18        else{
19          L[sz]=H[r];R[sz]=R[H[r]];
20          L[R[H[r]]]=sz;R[H[r]]=sz;
21        }
22        s[c]++;col[sz]=c;row[sz]=r;sz++;
23      }
24      void remove(int c){
25        for(int i=D[c];i!=c;i=D[i])
26          L[R[i]]=L[i],R[L[i]]=R[i];
27      }
28      void resume(int c){
29        for(int i=U[c];i!=c;i=U[i])
30          L[R[i]]=R[L[i]]=i;
31      }
32      int A(){
33        int res=0;
34        memset(vis,0,sizeof vis);
35        for(int i=R[0];i;i=R[i])if(!vis[i]){
36          vis[i]=1;res++;
37          for(int j=D[i];j!=i;j=D[j])
38            for(int k=R[j];k!=j;k=R[k])
39              vis[col[k]]=1;
40        }
41        return res;
42      }
43      void dfs(int d,int &ans){
44        if(R[0]==0){ans=min(ans,d);return;}
45        if(d+A()>=ans)return;
46        int tmp=23333,c;
47        for(int i=R[0];i;i=R[i])
48          if(tmp>s[i])tmp=s[i],c=i;
49        for(int i=D[c];i!=c;i=D[i]){
50          remove(i);
51          for(int j=R[i];j!=i;j=R[j])remove(j);
52          dfs(d+1,ans);
53          for(int j=L[i];j!=i;j=L[j])resume(j);
54          resume(i);
55        }
56      }
57      void del(int c){//exactly cover
58          L[R[c]]=L[c];R[L[c]]=R[c];
59        for(int i=D[c];i!=c;i=D[i])
60          for(int j=R[i];j!=i;j=R[j])
61            U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
62        }
63        void add(int c){  //exactly cover
64          R[L[c]]=L[R[c]]=c;
65        for(int i=U[c];i!=c;i=U[i])
66          for(int j=L[i];j!=i;j=L[j])
```

```
67              ++s[col[U[D[j]]=D[U[j]]=j]];
68          }
69      bool dfs2(int k){//exactly cover
70          if(!R[0]){
71              cnt=k;return 1;
72          }
73          int c=R[0];
74      for(int i=R[0];i;i=R[i])
75          if(s[c]>s[i])c=i;
76          del(c);
77      for(int i=D[c];i!=c;i=D[i]){
78          for(int j=R[i];j!=i;j=R[j])
79          del(col[j]);
80              ans[k]=row[i];if(dfs2(k+1))return true;
81          for(int j=L[i];j!=i;j=L[j])
82          add(col[j]);
83          }
84          add(c);
85      return 0;
86      }
87  }dlx;
88  int main(){
89      dlx.init(n);
90      for(int i=1;i<=m;i++)
91          for(int j=1;j<=n;j++)
92              if(dis(station[i],city[j])<mid-eps)
93                  dlx.Link(i,j);
94              dlx.dfs(0,ans);
95  }
```

9.5 插头 DP

```
1   int n,m,l;
2   struct L{
3       int d[11];
4       int& operator[](int x){return d[x];}
5       int mc(int x){
6           int an=1;
7           if(d[x]==1){
8               for(x++;x<l;x++)if(d[x]){
9                   an=an+(d[x]==1?1:-1);
10                  if(!an)return x;
11              }
12          }else{
13              for(x--;x>=0;x--)if(d[x]){
14                  an=an+(d[x]==2?1:-1);
15                  if(!an)return x;
16              }
17          }
```

```
18          }
19          int h(){int an=0;for(int i=l-1;i>=0;i--)an=an*3+d[i];return an;}
20          L s(int x,int y){
21              L S=*this;
22              S[x]=y;return S;
23          }
24          L operator>>(int _){
25              L S=*this;
26              for(int i=l-1;i>=1;i--)S[i]=S[i-1];
27              S[0]=0;return S;
28          }
29      };
30      struct Int{
31          int len;
32          int a[40];
33          Int(){len=1;memset(a,0,sizeof a);}
34          Int operator+=(const Int &o){
35              int l=max(len,o.len);
36              for(int i=0;i<l;i++)
37                  a[i]=a[i]+o.a[i];
38              for(int i=0;i<l;i++)
39                  a[i+1]+=a[i]/10,a[i]%=10;
40              if(a[l])l++;len=l;
41              return *this;
42          }
43          void print(){
44              for(int i=len-1;i>=0;i--)
45                  printf("%d",a[i]);
46              puts("");
47          }
48      };
49      struct hashtab{
50          int sz;
51          int tab[177147];
52          Int w[177147];
53          L s[177147];
54          hashtab(){memset(tab,-1,sizeof tab);}
55          void cl(){
56              for(int i=0;i<sz;i++)tab[s[i].h()]=-1;
57              sz=0;
58          }
59          Int& operator[](L S){
60              int h=S.h();
61              if(tab[h]==-1)tab[h]=sz,s[sz]=S,w[sz]=Int(),sz++;
62              return w[tab[h]];
63          }
64      }f[2];
65      bool check(L S){
66          int cn1=0,cn2=0;
```

```
67        for(int i=0;i<l;i++){
68            cn1+=S[i]==1;
69            cn2+=S[i]==2;
70        }return cn1==1&&cn2==1;
71    }
72    int main(){
73        Int One;One.a[0]=1;
74        scanf("%d%d",&n,&m);if(n<m)swap(n,m);l=m+1;
75        if(n==1||m==1){puts("1");return 0;}
76        int cur=0;f[cur].cl();
77        for(int i=1;i<=n;i++){
78            for(int j=1;j<=m;j++){
79                if(i==1&&j==1){
80                    f[cur][L().s(0,1).s(1,2)]+=One;
81                    continue;
82                }
83                cur^=1;f[cur].cl();
84                for(int k=0;k<f[!cur].sz;k++){
85                    L S=f[!cur].s[k];Int w=f[!cur][S];
86                    int d1=S[j-1],d2=S[j];
87                    if(d1==0&&d2==0){
88                        if(i!=n&&j!=m)f[cur][S.s(j-1,1).s(j,2)]+=w;
89                    }else
90                    if(d1==0||d2==0){
91                        if(i!=n)f[cur][S.s(j-1,d1|d2).s(j,0)]+=w;
92                        if(j!=m)f[cur][S.s(j-1,0).s(j,d1|d2)]+=w;
93                    }else
94                    if(d1==1&&d2==2){
95                        if(i==n&&j==m&&check(S))
96                            (w+=w).print();
97                    }else
98                    if(d1==2&&d2==1){
99                        f[cur][S.s(j-1,0).s(j,0)]+=w;
100                   }else
101                   if((d1==1&&d2==1)||(d1==2&&d2==2)){
102                       int m1=S.mc(j),m2=S.mc(j-1);
103                       f[cur][S.s(j-1,0).s(j,0).s(m1,1).s(m2,2)]+=w;
104                   }
105               }
106           }
107           cur^=1;f[cur].cl();
108           for(int k=0;k<f[!cur].sz;k++){
109               L S=f[!cur].s[k];Int w=f[!cur][S];
110               f[cur][S>>1]=w;
111           }
112       }
113       return 0;
114   }
```

9.6 某年某月某日是星期几

```cpp
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：$k > 0$

```cpp
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

```cpp
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];
```

```cpp
            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

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

10 vimrc

```vim
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

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

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 ≤ 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 $|N(v)| + 1 \le |N(w)|$ 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

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 基础模板

```java
import java.io.*;
import java.util.*;
import java.math.*;
public class Main {
    public static void main(String[] args) {
        InputStream inputStream = System.in;
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        PrintWriter out = new PrintWriter(outputStream);
    }
}
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;
  }
}
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

    public InputReader(InputStream stream) {
```

```java
30          reader = new BufferedReader(new InputStreamReader(stream), 32768);
31          tokenizer = null;
32      }
33
34      public String next() {
35          while (tokenizer == null || !tokenizer.hasMoreTokens()) {
36              try {
37                  tokenizer = new StringTokenizer(reader.readLine());
38              } catch (IOException e) {
39                  throw new RuntimeException(e);
40              }
41          }
42          return tokenizer.nextToken();
43      }
44
45      public int nextInt() {
46          return Integer.parseInt(next());
47      }
48
49      public long nextLong() {
50          return Long.parseLong(next());
51      }
52  }
```

14.2 Java Hints

```java
1  import java.util.*;
2  import java.math.*;
3  import java.io.*;
4
5  public class Main{
6    static class Task{
7      void solve(int testId, InputReader cin, PrintWriter cout) {
8        // Write down the code you want
9      }
10   };
11
12   public static void main(String args[]) {
13     InputStream inputStream = System.in;
14     OutputStream outputStream = System.out;
15     InputReader in = new InputReader(inputStream);
16     PrintWriter out = new PrintWriter(outputStream);
17     TaskA solver = new TaskA();
18     solver.solve(1, in, out);
19     out.close();
20   }
21
22   static class InputReader {
23     public BufferedReader reader;
```

```java
    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 ()
// 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
```

14.3　样例代码

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

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);

    return Ret;
  }
  private static class InputReader {

      public BufferedReader rea;
      public StringTokenizer tok;

      public InputReader(InputStream stream) {
          rea = new BufferedReader(new InputStreamReader(stream), 32768);
          tok = null;
      }

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

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

      public long nextLong() {
          return Long.parseLong(next());
      }
  }

  public static void main(String arg[]){
    InputReader cin = new InputReader(System.in);
```

```java
49        //Scanner cin = new Scanner(System.in);
50        int N = 70;
51
52        long k[] = new long[N];
53        int n;
54
55        while(true){
56          n=cin.nextInt();
57          if (n == 0) break;
58
59          for (int i = 1; i <= n; i ++){
60            k[i]=cin.nextLong();
61            //System.out.println(k[i]);
62          }
63
64          long Len;
65          BigInteger Sum;
66          long A, B;
67          long AnsA = -1, AnsB = -1;
68          int Ans = 0;
69
70          for (int i = -1; i <= 1; i++){
71            Len = k[1] * 2 + i;
72
73            if(Len<=0)continue;
74
75            Sum = BigInteger.ZERO;
76            for (int j = 1; j <= n; j++){
77              Sum = Sum.add(BigInteger.valueOf(1L << (j - 1)) .multiply(BigInteger.valueOf(k[j])));
78            }
79            //System.out.println(Sum);
80
81            long x = Len;
82
83            if ((Sum.multiply(BigInteger.valueOf(2))) .mod
     ↪    (BigInteger.valueOf(Len)).compareTo(BigInteger.ZERO) > 0) continue;
84
85            long y = Sum.multiply(BigInteger.valueOf(2)).divide(BigInteger.valueOf(Len)).longValue();
86            if ((y - x + 1) % 2 > 0) continue;
87            A = (y - x + 1) / 2;
88            if ((x + y - 1) % 2 > 0) continue;
89            B = (x + y - 1) / 2;
90
91            if (A < 1 || A > (long)1e18) continue;
92            if (B < 1 || B > (long)1e18) continue;
93
94            int flag = 1;
95
96            long Cnt;
```

```java
97          long Cnt_B;
98          long Cnt_A;
99
100         for (int j = 1; j <= n; j++){
101           Cnt_B = Calc(B, j);
102           Cnt_A = Calc(A - 1, j);
103
104           if (Cnt_B - Cnt_A != k[j]){
105             flag = 0;
106             break;
107           }
108         }
109
110         if (flag==1){
111           Ans++;
112           //printf("%lld %lld\n", A, B);
113           //System.out.println(A+" "+B);
114           AnsA = A;
115           AnsB = B;
116         }
117       }
118
119     if (Ans == 0) System.out.println("None");//puts("None");
120     else
121     if (Ans == 1)
122       System.out.println(AnsA+" "+AnsB);//cout << AnsA << ' ' << AnsB << endl;
123     else
124       System.out.println("Many");//puts("Many");
125   }
126
127   }
128 }
```

Java™ Platform
Standard Ed. 8

OVERVIEW   PACKAGE   CLASS   USE   TREE   DEPRECATED   INDEX   HELP

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   ALL CLASSES
SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

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 $-2^{Integer.MAX\_VALUE}$ (exclusive) to $+2^{Integer.MAX\_VALUE}$ (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 $-2^{Integer.MAX\_VALUE}$ (exclusive) to $+2^{Integer.MAX\_VALUE}$ (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 ($this^{-1}$ mod m).	
BigInteger	**modPow**(**BigInteger** exponent, **BigInteger** m)	
	Returns a BigInteger whose value is ($this^{exponent}$ mod 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 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.

Java™ Platform
Standard Ed. 8

OVERVIEW   PACKAGE   CLASS   USE   TREE   DEPRECATED   INDEX   HELP

PREV CLASS   NEXT CLASS   FRAMES   NO FRAMES   ALL CLASSES
SUMMARY: NESTED | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

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

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`

```
1  Compile:
2  #!/bin/sh
3  full=$GEDIT_CURRENT_DOCUMENT_NAME
4  name=`echo $full | cut -d. -f1`
5  g++ $full -o $name -g -Wall
6
7  Debug:
8  #!/bin/bash
9  name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
10 gnome-terminal -x bash -c "gdb ./$name"
11
12 Run:
13 #!/bin/bash
14 name=`echo $GEDIT_CURRENT_DOCUMENT_NAME | cut -d. -f1`
15 gnome-terminal -x bash -c "time ./$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. $fib_m|fib_n^2 \Leftrightarrow n fib_n|m$

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

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

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

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

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

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

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

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

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

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

7．外接圆半径

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

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．$S = \frac{1}{2}D_1D_2 sinA$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 巴什博弈

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，$s \leq m$），那么先取者要拿走 s 个物品，如果后取者拿走 $k(k \leq m)$ 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 阶梯博弈

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

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

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

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

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

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

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

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

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．$S = \frac{1}{2}D_1 D_2 sinA$

3．对于圆内接四边形

$$ac + bd = D_1 D_2$$

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

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

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

$$S = 2\pi rh$$

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

Definitions		Series

$f(n) = O(g(n))$	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 $|a_n - a| < \epsilon$, $\forall n \ge n_0$.	
$\sup S$	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.	

Series column:

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

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:

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

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

$\left[\begin{smallmatrix} n \\ k \end{smallmatrix}\right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{\begin{smallmatrix} n \\ k \end{smallmatrix}\right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle\begin{smallmatrix} n \\ k \end{smallmatrix}\right\rangle$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \ldots \pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\left\langle\begin{smallmatrix} n \\ k \end{smallmatrix}\right\rangle\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

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

4. $\binom{n}{k} = \dfrac{n}{k}\binom{n-1}{k-1}$, **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.** $\left\{\begin{smallmatrix} n \\ 1 \end{smallmatrix}\right\} = \left\{\begin{smallmatrix} n \\ n \end{smallmatrix}\right\} = 1$,

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

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

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

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

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

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

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

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

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

Identities Cont.		Trees

38. $\left[\begin{matrix} n+1 \\ m+1 \end{matrix}\right] = \sum_k \left[\begin{matrix} n \\ k \end{matrix}\right]\binom{k}{m} = \sum_{k=0}^{n}\left[\begin{matrix} k \\ m \end{matrix}\right]n^{\underline{n-k}} = n!\sum_{k=0}^{n}\frac{1}{k!}\left[\begin{matrix} k \\ m \end{matrix}\right],$ **39.** $\left[\begin{matrix} x \\ x-n \end{matrix}\right] = \sum_{k=0}^{n}\left\langle\!\!\left\langle\begin{matrix} n \\ k \end{matrix}\right\rangle\!\!\right\rangle\binom{x+k}{2n},$

40. $\left\{\begin{matrix} n \\ m \end{matrix}\right\} = \sum_k \binom{n}{k}\left\{\begin{matrix} k+1 \\ m+1 \end{matrix}\right\}(-1)^{n-k},$ **41.** $\left[\begin{matrix} n \\ m \end{matrix}\right] = \sum_k \left[\begin{matrix} n+1 \\ k+1 \end{matrix}\right]\binom{k}{m}(-1)^{m-k},$

42. $\left\{\begin{matrix} m+n+1 \\ m \end{matrix}\right\} = \sum_{k=0}^{m} k\left\{\begin{matrix} n+k \\ k \end{matrix}\right\},$ **43.** $\left[\begin{matrix} m+n+1 \\ m \end{matrix}\right] = \sum_{k=0}^{m} k(n+k)\left[\begin{matrix} n+k \\ k \end{matrix}\right],$

44. $\binom{n}{m} = \sum_k \left\{\begin{matrix} n+1 \\ k+1 \end{matrix}\right\}\left[\begin{matrix} k \\ m \end{matrix}\right](-1)^{m-k},$ **45.** $(n-m)!\binom{n}{m} = \sum_k \left[\begin{matrix} n+1 \\ k+1 \end{matrix}\right]\left\{\begin{matrix} k \\ m \end{matrix}\right\}(-1)^{m-k},$ for $n \geq m,$

46. $\left\{\begin{matrix} n \\ n-m \end{matrix}\right\} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left[\begin{matrix} m+k \\ k \end{matrix}\right],$ **47.** $\left[\begin{matrix} n \\ n-m \end{matrix}\right] = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left\{\begin{matrix} m+k \\ k \end{matrix}\right\},$

48. $\left\{\begin{matrix} n \\ \ell+m \end{matrix}\right\}\binom{\ell+m}{\ell} = \sum_k \left\{\begin{matrix} k \\ \ell \end{matrix}\right\}\left\{\begin{matrix} n-k \\ m \end{matrix}\right\}\binom{n}{k},$ **49.** $\left[\begin{matrix} n \\ \ell+m \end{matrix}\right]\binom{\ell+m}{\ell} = \sum_k \left[\begin{matrix} k \\ \ell \end{matrix}\right]\left[\begin{matrix} n-k \\ m \end{matrix}\right]\binom{n}{k}.$

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"

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

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

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

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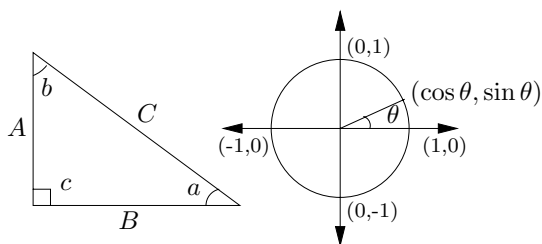$$G(x) = \frac{x}{(1-x)(1-2x)}.$$

Expand this using partial fractions:
$$G(x) = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)$$
$$= x\left(2\sum_{i\geq 0} 2^i x^i - \sum_{i\geq 0} x^i\right)$$
$$= \sum_{i\geq 0}(2^{i+1} - 1)x^{i+1}.$$

So $g_i = 2^i - 1.$

<div align="center">

Theoretical Computer Science Cheat Sheet

</div>

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

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:
$$e = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} + \cdots$$
$$\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n = e^x.$$
$$\left(1 + \tfrac{1}{n}\right)^n < e < \left(1 + \tfrac{1}{n}\right)^{n+1}.$$
$$\left(1 + \tfrac{1}{n}\right)^n = e - \frac{e}{2n} + \frac{11e}{24n^2} - O\left(\frac{1}{n^3}\right).$$

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

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

Factorial, Stirling's approximation:
$$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, \ldots$$
$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right).$$

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:
$$\Pr[A \vee B] = \Pr[A] + \Pr[B] - \Pr[A \wedge B]$$
$$\Pr[A \wedge B] = \Pr[A] \cdot \Pr[B],$$
$$\text{iff } A \text{ and } B \text{ are independent.}$$
$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]}$$

For random variables X and Y:
$$\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y],$$
$$\text{if } X \text{ and } Y \text{ are independent.}$$
$$\mathrm{E}[X + Y] = \mathrm{E}[X] + \mathrm{E}[Y],$$
$$\mathrm{E}[cX] = c\,\mathrm{E}[X].$$

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

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

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

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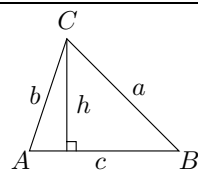$$(\cosh x + \sinh x)^n = \cosh nx + \sinh nx, \quad n \in \mathbb{Z},$$
$$2\sinh^2 \tfrac{x}{2} = \cosh x - 1, \qquad 2\cosh^2 \tfrac{x}{2} = \cosh x + 1.$$

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	Graph Theory

Number Theory

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

$$C \equiv r_1 \bmod m_1$$
$$\vdots \quad \vdots \quad \vdots$$
$$C \equiv r_n \bmod m_n$$

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

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

Prime numbers:

$$p_n = n \ln n + n \ln \ln n - n + n \frac{\ln \ln n}{\ln n}$$
$$+ O\left(\frac{n}{\ln n}\right),$$
$$\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3}$$
$$+ O\left(\frac{n}{(\ln n)^4}\right).$$

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 $k \cdot c(G - S) \leq |S|$.
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
$r(k,\ell)$	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:
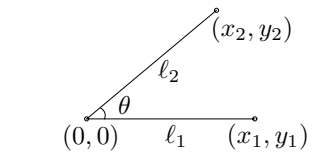
$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p},$$
$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p\right]^{1/p}.$$

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Integrals:

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

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

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

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

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

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

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

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

Calculus Cont.	Finite Calculus

Calculus Cont.

62. $\int \dfrac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a > 0,$ **63.** $\int \dfrac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\dfrac{\sqrt{x^2\pm a^2}}{a^2x},$

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

66. $\int \dfrac{dx}{ax^2+bx+c} = \begin{cases} \dfrac{1}{\sqrt{b^2-4ac}}\ln\left|\dfrac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, & \text{if } b^2 > 4ac, \\[3mm] \dfrac{2}{\sqrt{4ac-b^2}}\arctan\dfrac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2 < 4ac, \end{cases}$

67. $\int \dfrac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, & \text{if } a > 0, \\[3mm] \dfrac{1}{\sqrt{-a}}\arcsin\dfrac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a < 0, \end{cases}$

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

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

70. $\int \dfrac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases} \dfrac{-1}{\sqrt{c}}\ln\left|\dfrac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, & \text{if } c > 0, \\[3mm] \dfrac{1}{\sqrt{-c}}\arcsin\dfrac{bx+2c}{|x|\sqrt{b^2-4ac}}, & \text{if } c < 0, \end{cases}$

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

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

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

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

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

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

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

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

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:
$$\Delta(cu) = c\Delta u, \qquad \Delta(u+v) = \Delta u + \Delta v,$$
$$\Delta(uv) = u\Delta v + \mathrm{E}\,v\Delta u,$$
$$\Delta(x^{\underline{n}}) = nx^{\underline{n-1}},$$
$$\Delta(H_x) = x^{\underline{-1}}, \qquad\qquad \Delta(2^x) = 2^x,$$
$$\Delta(c^x) = (c-1)c^x, \qquad \Delta\binom{x}{m} = \binom{x}{m-1}.$$

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

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:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \cdots = \sum_{i=0}^{\infty} x^i,$$

$$\frac{1}{1-cx} = 1 + cx + c^2x^2 + c^3x^3 + \cdots = \sum_{i=0}^{\infty} c^i x^i,$$

$$\frac{1}{1-x^n} = 1 + x^n + x^{2n} + x^{3n} + \cdots = \sum_{i=0}^{\infty} x^{ni},$$

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + 4x^4 + \cdots = \sum_{i=0}^{\infty} i x^i,$$

$$x^k \frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) = x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \cdots = \sum_{i=0}^{\infty} i^n x^i,$$

$$e^x = 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!},$$

$$\ln(1+x) = x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots = \sum_{i=1}^{\infty} (-1)^{i+1}\frac{x^i}{i},$$

$$\ln\frac{1}{1-x} = x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots = \sum_{i=1}^{\infty} \frac{x^i}{i},$$

$$\sin x = x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},$$

$$\cos x = 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},$$

$$\tan^{-1} x = x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},$$

$$(1+x)^n = 1 + nx + \frac{n(n-1)}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{n}{i} x^i,$$

$$\frac{1}{(1-x)^{n+1}} = 1 + (n+1)x + \binom{n+2}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,$$

$$\frac{x}{e^x - 1} = 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},$$

$$\frac{1}{2x}(1 - \sqrt{1-4x}) = 1 + x + 2x^2 + 5x^3 + \cdots = \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}} = 1 + x + 2x^2 + 6x^3 + \cdots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$$

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

$$\frac{1}{1-x}\ln\frac{1}{1-x} = x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots = \sum_{i=1}^{\infty} H_i x^i,$$

$$\frac{1}{2}\left(\ln\frac{1}{1-x}\right)^2 = \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots = \sum_{i=2}^{\infty} \frac{H_{i-1} x^i}{i},$$

$$\frac{x}{1-x-x^2} = x + x^2 + 2x^3 + 3x^4 + \cdots = \sum_{i=0}^{\infty} F_i x^i,$$

$$\frac{F_n x}{1-(F_{n-1}+F_{n+1})x - (-1)^n x^2} = F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots = \sum_{i=0}^{\infty} F_{ni} x^i.$$

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:
$$\alpha A(x) + \beta B(x) = \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i,$$

$$x^k A(x) = \sum_{i=k}^{\infty} a_{i-k} x^i,$$

$$\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} = \sum_{i=0}^{\infty} a_{i+k} x^i,$$

$$A(cx) = \sum_{i=0}^{\infty} c^i a_i x^i,$$

$$A'(x) = \sum_{i=0}^{\infty} (i+1)a_{i+1} x^i,$$

$$xA'(x) = \sum_{i=1}^{\infty} i a_i x^i,$$

$$\int A(x)\, dx = \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i,$$

$$\frac{A(x) + A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i} x^{2i},$$

$$\frac{A(x) - A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.$$

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:

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

$$x^{\overline{n}} = \sum_{i=0}^{\infty}\begin{bmatrix} n \\ i \end{bmatrix}x^i,$$

$$\left(\ln\frac{1}{1-x}\right)^n = \sum_{i=0}^{\infty}\begin{bmatrix} i \\ n \end{bmatrix}\frac{n!x^i}{i!},$$

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

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

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

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

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

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

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

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

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

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

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

$$\left(\frac{1}{x}\right)^{\overline{-n}} = \sum_{i=0}^{\infty}\left\{\begin{matrix} i \\ n \end{matrix}\right\}x^i,$$
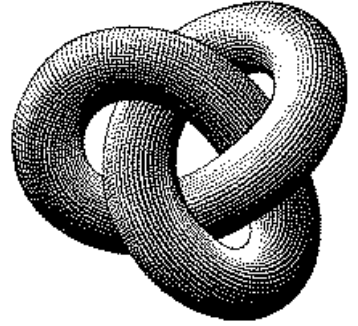
$$(e^x-1)^n = \sum_{i=0}^{\infty}\left\{\begin{matrix} i \\ n \end{matrix}\right\}\frac{n!x^i}{i!},$$

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

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

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

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:

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

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

$$F_i = F_{i-1}+F_{i-2},\quad F_0 = F_1 = 1,$$
$$F_{-i} = (-1)^{i-1}F_i,$$
$$F_i = \frac{1}{\sqrt{5}}\left(\phi^i - \hat{\phi}^i\right),$$

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

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

Additive rule:

$$F_{n+k} = F_k F_{n+1} + F_{k-1}F_n,$$
$$F_{2n} = F_n F_{n+1} + F_{n-1}F_n.$$

Calculation by matrices:

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