代码库

Blazar

2017 年 10 月 11 日

目录

1 数论

1.1 快速求逆元

返回结果：

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

使用条件：$x \in [0, mod)$ 并且 x 与 mod 互质

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

1.2 扩展欧几里德算法

返回结果：

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

时间复杂度：$\mathcal{O}(nlogn)$

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

1.3 中国剩余定理

返回结果：

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

使用条件：p_i 需两两互质

```
LL china(int n,int *a,int *m){
	LL M=1,d,x=0,y;
	for(int i=0;i<n;i++)
		M*=m[i];
```

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

1.4 中国剩余定理 2

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

1.5 组合数取模

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

        ans=ans*i%P;
    }
    return make_pair(cnt,ans);
}
pair<LL,LL> calc(LL n,LL p,LL k){
    prod=1;P=pow(p,k,1e18);
    for(int i=1;i<P;i++)if(i%p)prod=prod*i%P;
```

```
    pair<LL,LL> res=comput(n,p,k);
//  res.second=res.second*pow(p,res.first%k,P)%P;
//  res.first-=res.first%k;
    return res;
}
LL calc(LL n,LL m,LL p,LL k){
    pair<LL,LL>A,B,C;
    LL P=pow(p,k,1e18);
    A=calc(n,p,k);
    B=calc(m,p,k);
    C=calc(n-m,p,k);
    LL ans=1;
    ans=pow(p,A.first-B.first-C.first,P);
    ans=ans*A.second%P*inv(B.second,P)%P*inv(C.second,P)%P;
    return ans;
}
```

1.6 扩展小步大步

```
LL solve2(LL a,LL b,LL p){
    //a^x=b (mod p)
    b%=p;
    LL e=1%p;
    for(int i=0;i<100;i++){
        if(e==b)return i;
        e=e*a%p;
    }
    int r=0;
    while(gcd(a,p)!=1){
        LL d=gcd(a,p);
        if(b%d)return -1;
        p/=d;b/=d;b=b*inv(a/d,p);
        r++;
    }LL res=BSGS(a,b,p);
    if(res==-1)return -1;
    return res+r;
}
```

1.7 卢卡斯定理

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

1.8 小步大步

返回结果：

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

使用条件：p 为质数

时间复杂度：$\mathcal{O}(\sqrt{n})$

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

1.9 Miller Rabin 素数测试

```
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
```

```
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

1.10 Pollard Rho 大数分解

时间复杂度：$\mathcal{O}(n^{1/4})$

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

1.11 快速数论变换（zky）

返回结果：

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

使用说明：$magic$ 是 mod 的原根

时间复杂度：$\mathcal{O}(nlogn)$

```
/*
{(mod,G)}={(81788929,7),(101711873,3),(167772161,3)
                        ,(377487361,7),(998244353,3),(1224736769,3)
                        ,(1300234241,3),(1484783617,5)}
*/
int mo=998244353,G=3;
void NTT(int a[],int n,int f){
        for(register int i=0;i<n;i++)
                if(i<rev[i])
                        swap(a[i],a[rev[i]]);
        for (register int i=2;i<=n;i<<=1){
                static int exp[maxn];
                exp[0]=1;exp[1]=pw(G,(mo-1)/i);
                if(f==-1)exp[1]=pw(exp[1],mo-2);
                for(register int k=2;k<(i>>1);k++)
                        exp[k]=1LL*exp[k-1]*exp[1]%mo;
                for(register int j=0;j<n;j+=i){
                        for(register int k=0;k<(i>>1);k++){
                                register int &pA=a[j+k],&pB=a[j+k+(i>>1)];
                                register int A=pA,B=1LL*pB*exp[k]%mo;
                                pA=(A+B)%mo;
                                pB=(A-B+mo)%mo;
                        }
                }
        }
        if(f==-1){
                int rv=pw(n,mo-2)%mo;
                for(int i=0;i<n;i++)
                        a[i]=1LL*a[i]*rv%mo;
        }
}
void mul(int m,int a[],int b[],int c[]){
        int n=1,len=0;
        while(n<m)n<<=1,len++;
        for (int i=1;i<n;i++)
```

```
        rev[i]=(rev[i>>1]>>1)|((i&1)<<(len-1));
    NTT(a,n,1);
    NTT(b,n,1);
    for(int i=0;i<n;i++)
        c[i]=1LL*a[i]*b[i]%mo;
    NTT(c,n,-1);
}
```

1.12 原根

```
vector<LL>fct;
bool check(LL x,LL g){
    for(int i=0;i<fct.size();i++)
        if(pw(g,(x-1)/fct[i],x)==1)
            return 0;
    return 1;
}
LL findrt(LL x){
    LL tmp=x-1;
    for(int i=2;i*i<=tmp;i++){
        if(tmp%i==0){
            fct.push_back(i);
            while(tmp%i==0)tmp/=i;
        }
    }if(tmp>1)fct.push_back(tmp);
    // x is 1,2,4,p^n,2p^n
    // x has phi(phi(x)) primitive roots
    for(int i=2;i<int(1e9);i++)if(check(x,i))
        return i;
    return -1;
}
const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool check(long long n,int base) {
    long long n2=n-1,res;
    int s=0;
    while(n2%2==0) n2>>=1,s++;
    res=pw(base,n2,n);
    if((res==1)||(res==n-1)) return 1;
    while(s--) {
        res=mul(res,res,n);
        if(res==n-1) return 1;
    }
```

```
    return 0; // n is not a strong pseudo prime
}
bool isprime(const long long &n) {
    if(n==2)
        return true;
    if(n<2 || n%2==0)
        return false;
    for(int i=0;i<12&&BASE[i]<n;i++){
        if(!check(n,BASE[i]))
            return false;
    }
    return true;
}
```

1.13 线性递推

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

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

```
            }
            copy(u, u + m, v);
        }
        //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
        for(int i(m); i < 2 * m; i++) {
            a[i] = 0;
            for(int j(0); j < m; j++) {
                a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
            }
        }
        for(int j(0); j < m; j++) {
            b[j] = 0;
            for(int i(0); i < m; i++) {
                b[j] = (b[j] + v[i] * a[i + j]) % p;
            }
        }
        for(int j(0); j < m; j++) {
            a[j] = b[j];
        }
}
```

1.14 线性筛

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

```
                        f[i*p[j]]=i*p[j]-i;//Special Case for f(p^k)
                    }else{
                        f[i*p[j]]=f[i/minpw[i]]*f[minpw[i]*p[j]];
                    }
                    break;
                }else{
                    //Case 3 coprime
                    minpw[i*p[j]]=p[j];
                    f[i*p[j]]=f[i]*f[p[j]];
                    phi[i*p[j]]=phi[i]*(p[j]-1);
                    mu[i*p[j]]=-mu[i];
                }
            }
        }
}
```

1.15 直线下整点个数

返回结果：

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

使用条件：$n,m>0$，$a,b\ge 0$

时间复杂度：$\mathcal{O}(nlogn)$

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

2 数值

2.1 高斯消元

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

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

```
                                A[i][k]=(A[i][k]-A[j][k]*t%m+m)%m;
                                swap(A[i],A[j]);
                                ans=-ans;
                        }
                }ans=ans*A[i][i]%m;
        }return (ans%m+m)%m;
}
int Gauss(){//求秩
        int r,now=-1;
        int ans=0;
        for(int i = 0; i <n; i++){
                r = now + 1;
                for(int j = now + 1; j < m; j++)
                        if(fabs(A[j][i]) > fabs(A[r][i]))
                                r = j;
                if (!sgn(A[r][i])) continue;
                ans++;
                now++;
                if(r != now)
                        for(int j = 0; j < n; j++)
                                swap(A[r][j], A[now][j]);

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

2.2 快速傅立叶变换

返回结果：

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

时间复杂度：$\mathcal{O}(nlogn)$

```
typedef complex<double> cp;
const double pi = acos(-1);
void FFT(vector<cp>&num,int len,int ty){
    for(int i=1,j=0;i<len-1;i++){
```

```cpp
        for(int k=len;j^=k>>=1,~j&k;);
        if(i<j)
            swap(num[i],num[j]);
    }
    for(int h=0;(1<<h)<len;h++){
        int step=1<<h,step2=step<<1;
        cp w0(cos(2.0*pi/step2),ty*sin(2.0*pi/step2));
        for(int i=0;i<len;i+=step2){
            cp w(1,0);
            for(int j=0;j<step;j++){
                cp &x=num[i+j+step];
                cp &y=num[i+j];
                cp d=w*x;
                x=y-d;
                y=y+d;
                w=w*w0;
            }
        }
    }
    if(ty==-1)
        for(int i=0;i<len;i++)
            num[i]=cp(num[i].real()/(double)len,num[i].imag());
}
vector<cp> mul(vector<cp>a,vector<cp>b){
    int len=a.size()+b.size();
    while((len&-len)!=len)len++;
    while(a.size()<len)a.push_back(cp(0,0));
    while(b.size()<len)b.push_back(cp(0,0));
    FFT(a,len,1);
    FFT(b,len,1);
    vector<cp>ans(len);
    for(int i=0;i<len;i++)
        ans[i]=a[i]*b[i];
    FFT(ans,len,-1);
    return ans;
}
```

2.3 单纯形法求解线性规划

返回结果：

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

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

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

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

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

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

2.4 自适应辛普森

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

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

```
        return area_total + (area_total - area_sum) / 15;
    }
    return simpson(left, mid, eps / 2, area_left)
         + simpson(mid, right, eps / 2, area_right);
}

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

2.5 多项式求根

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

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

3 数据结构

3.1 平衡的二叉查找树

3.1.1 Treap

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+5;
#define sz(x) (x?x->siz:0)
struct Treap{
```

```
	struct node{
		int key,val;
		int siz,s;
		node *c[2];
		node(int v=0){
			val=v;
			key=rand();
			siz=1,s=1;
			c[0]=c[1]=0;
		}
		void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
	}pool[maxn],*cur,*root;
	Treap(){cur=pool;}
	node* newnode(int val){return *cur=node(val),cur++;}
	void rot(node *&t,int d){
		if(!t->c[d])t=t->c[!d];
		else{
			node *p=t->c[d];t->c[d]=p->c[!d];
			p->c[!d]=t;t->rz();p->rz();t=p;
		}
	}
	void insert(node *&t,int x){
		if(!t){t=newnode(x);return;}
		if(t->val==x){t->s++;t->siz++;return;}
		insert(t->c[x>t->val],x);
		if(t->key<t->c[x>t->val]->key)
			rot(t,x>t->val);
		else t->rz();
	}
	void del(node *&t,int x){
		if(!t)return;
		if(t->val==x){
			if(t->s>1){t->s--;t->siz--;return;}
			if(!t->c[0]||!t->c[1]){
				if(!t->c[0])t=t->c[1];
				else t=t->c[0];
				return;
			}
			int d=t->c[0]->key<t->c[1]->key;
			rot(t,d);
			del(t,x);
			return;
```

```
			}
			del(t->c[x>t->val],x);
			t->rz();
		}
		int pre(node *t,int x){
			if(!t)return INT_MIN;
			int ans=pre(t->c[x>t->val],x);
			if(t->val<x)ans=max(ans,t->val);
			return ans;
		}
		int nxt(node *t,int x){
			if(!t)return INT_MAX;
			int ans=nxt(t->c[x>=t->val],x);
			if(t->val>x)ans=min(ans,t->val);
			return ans;
		}
		int rank(node *t,int x){
			if(!t)return 0;
			if(t->val==x)return sz(t->c[0]);
			if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
			if(t->val>x)return rank(t->c[0],x);
		}
		int kth(node *t,int x){
			if(sz(t->c[0])>=x)return kth(t->c[0],x);
			if(sz(t->c[0])+t->s>=x)return t->val;
			return kth(t->c[1],x-t->s-sz(t->c[0]));
		}
		void deb(node *t){
			if(!t)return;
			deb(t->c[0]);
			printf("%d ",t->val);
			deb(t->c[1]);
		}
		void insert(int x){insert(root,x);}
		void del(int x){del(root,x);}
		int pre(int x){return pre(root,x);}
		int nxt(int x){return nxt(root,x);}
		int rank(int x){return rank(root,x);}
		int kth(int x){return kth(root,x);}
		void deb(){deb(root);puts("");}
}T;
int main(){
```

```
    srand(12121);
    int m;
    scanf("%d",&m);
    while(m--){
        int opt,x;
        scanf("%d",&opt);
        switch(opt){
            case 1:
                scanf("%d",&x);
                T.insert(x);
                break;
            case 2:
                scanf("%d",&x);
                T.del(x);
                break;
            case 3:
                scanf("%d",&x);
                printf("%d\n",T.rank(x)+1);
                break;
            case 4:
                scanf("%d",&x);
                printf("%d\n",T.kth(x));
                break;
            case 5:
                scanf("%d",&x);
                printf("%d\n",T.pre(x));
                break;
            case 6:
                scanf("%d",&x);
                printf("%d\n",T.nxt(x));
                break;
        }
    }
        return 0;
}
```

3.1.2 Splay

```
void Rotate(int x, int c){
        int y = T[x].c[c];
        int z = T[y].c[1 - c];

```

```
        if (T[x].fa){
                if (T[T[x].fa].c[0] == x) T[T[x].fa].c[0] = y;
                else T[T[x].fa].c[1] = y;
        }

        T[z].fa = x; T[x].c[c] = z;
        T[y].fa = T[x].fa; T[x].fa = y; T[y].c[1 - c] = x;

        Update(x);
        Update(y);
}

int stack[M], fx[M];

void Splay(int x, int fa){
        int top = 0;
        for (int  u = x; u != fa; u = T[u].fa)
                stack[++top] = u;
        for (int i = 2; i <= top; i++)
                if (T[stack[i]].c[0] == stack[i - 1]) fx[i] = 0;
                else fx[i] = 1;

        for (int i = 2; i <= top; i += 2){
                if (i == top) Rotate(stack[i], fx[i]);
                else {
                        if (fx[i] == fx[i + 1]){
                                Rotate(stack[i + 1], fx[i + 1]);
                                Rotate(stack[i], fx[i]);
                        } else {
                                Rotate(stack[i], fx[i]);
                                Rotate(stack[i + 1], fx[i + 1]);
                        }
                }
        }

        if (fa == 0) Root = x;
}
```

3.2 坚固的数据结构

3.2.1 坚固的平衡树

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

```
            c[0]->split(ned,p,q);c[0]=0;rz();
            q=merge(q,this);
        }else{
            c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
            p=merge(this,p);
        }
}
int main(){
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }
}
```

3.2.2 坚固的字符串

1. ext 库中的 rope

```
#include <ext/rope>

using __gnu_cxx::crope;
using __gnu_cxx::rope;

crope a, b;

int main(void) {
    a = b.substr(pos, len);     // [pos, pos + len)
    a = b.substr(pos);          // [pos, pos]
    b.c_str();                  // might lead to memory leaks
    b.delete_c_str();           // delete the c_str that created before
    a.insert(pos, text);        // insert text before position pos
    a.erase(pos, len);          // erase [pos, pos + len)
}
```

2. 可持久化平衡树实现的 rope

```
class Rope {
private:
    class Node {
    public:
        Node *left, *right;
```

```
        int size;
        char key;

        Node(char key = 0, Node *left = NULL, Node *right = NULL)
                : key(key), left(left), right(right) {
            update();
        }

        void update() {
            size = (left ? left->size : 0) + 1 + (right ? right->size : 0);
        }

        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;
        }
        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;
        }
        std::pair<Node*, Node*> part = split(root, left);
        root = merge(part.first, split(part.second, right - left).second);
    }

    std::string substr(int left, int right) {
        if (left < 0 || left >= size() ||
            right < 1 || right > size()) {
            throw "Out of range";
        }
        if (left >= right) {
            return "";
        }
        return split(split(root, left).second, right - left).first->to_string();
    }

```

```cpp
    void copy(int left, int right, int pos) {
        if (left < 0 || left >= size() ||
            right < 1 || right > size() ||
            pos < 0 || pos > size()) {
            throw "Out of range";
        }
        if (left >= right) {
            return;
        }
        std::pair<Node*, Node*> part = split(root, pos);
        root = merge(merge(part.first,
                           split(split(root, left).second, right - left).first),
                     part.second);
    }
};
```

3.2.3 坚固的左偏树

```cpp
int Merge(int x, int y){
  if (x == 0 || y == 0) return x + y;
  if (Heap[x].Key < Heap[y].Key) swap(x, y);
  Heap[x].Ri = Merge(Heap[x].Ri, y);
  if (Heap[Heap[x].Le].Dis < Heap[Heap[x].Ri].Dis) swap(Heap[x].Le, Heap[x].Ri);
  if (Heap[x].Ri == 0) Heap[x].Dis = 0;
  else Heap[x].Dis = Heap[Heap[x].Ri].Dis + 1;
  return x;
}

for (int i = 0; i <= n; i++){
        Heap[i].Le = Heap[i].Ri = 0;
        Heap[i].Dis = 0;
        Heap[i].Key = Cost[i];
}
Heap[0].Dis = -1;
```

3.2.4 不坚固的斜堆

```cpp
struct node;
node *Null,*root[maxn];
struct node{
        node* c[2];
```

```
        int val,ind;
        node(int _val=0,int _ind=0){
                val=_val;c[0]=c[1]=Null;ind=_ind;
        }
};
node* merge(node *p,node *q){
        if(p==Null)return q;
        if(q==Null)return p;
        if(p->val>q->val)swap(p,q);
        p->c[1]=merge(p->c[1],q);
        swap(p->c[0],p->c[1]);
        return p;
}

Null=new node(0);
Null->c[0]=Null->c[1]=Null;
```

3.3 树上的魔术师

3.3.1 轻重树链剖分 (zky)

```
vector<int>G[maxn];
int fa[maxn],top[maxn],siz[maxn],son[maxn],mp[maxn],z,dep[maxn];
void dfs(int u){
        siz[u]=1;
        for(int i=0;i<G[u].size();i++){
                int v=G[u][i];
                if(v!=fa[u]){
                        fa[v]=u;dep[v]=dep[u]+1;
                        dfs(v);
                        siz[u]+=siz[v];
                        if(siz[son[u]]<siz[v])son[u]=v;
                }
        }
}
void build(int u,int tp){
        top[u]=tp;mp[u]=++z;
        if(son[u])build(son[u],tp);
        for(int v,i=0;i<G[u].size();i++)if((v=G[u][i])!=son[u]&&v!=fa[u])build(v,v);
}
```

3.3.2 Link Cut Tree(zky)

```
struct LCT{
    struct node{
        bool rev;
        int mx,val;
        node *f,*c[2];
        bool d(){return this==f->c[1];}
        bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
        void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
        void makerv(){rev^=1;swap(c[0],c[1]);}
        void pd(){
            if(rev){
                if(c[0])c[0]->makerv();
                if(c[1])c[1]->makerv();
                rev=0;
            }
        }
        void rz(){
            mx=val;
            if(c[0])mx=max(mx,c[0]->mx);
            if(c[1])mx=max(mx,c[1]->mx);
        }
    }nd[int(1e4)+1];
    void rot(node *x){
        node *y=x->f;if(!y->rt())y->f->pd();
        y->pd();x->pd();bool d=x->d();
        y->sets(x->c[!d],d);
        if(y->rt())x->f=y->f;
        else y->f->sets(x,y->d());
        x->sets(y,!d);
    }
    void splay(node *x){
        while(!x->rt())
            if(x->f->rt())rot(x);
            else if(x->d()==x->f->d())rot(x->f),rot(x);
            else rot(x),rot(x);
    }
    node* access(node *x){
        node *y=0;
        for(;x;x=x->f){
            splay(x);
```

```
            x->sets(y,1);y=x;
        }return y;
    }
    void makert(node *x){
        access(x)->makerv();
        splay(x);
    }
    void link(node *x,node *y){
        makert(x);
        x->f=y;
        access(x);
    }
    void cut(node *x,node *y){
        makert(x);access(y);splay(y);
        y->c[0]=x->f=0;
        y->rz();
    }
    void link(int x,int y){link(nd+x,nd+y);}
    void cut(int x,int y){cut(nd+x,nd+y);}
}T;
```

3.3.3 AAA Tree

```
#define rep(i,a,n) for(int i=a;i<n;i++)
int n,m;
struct info{
    int mx,mn,sum,sz;
    info(){}
    info(int mx,int mn,int sum,int sz):
        mx(mx),mn(mn),sum(sum),sz(sz){}
    void deb(){printf("sum:%d size:%d",(int)sum,sz);}
};
struct flag{
    int mul,add;
    flag(){mul=1;}
    flag(int mul,int add):
        mul(mul),add(add){}
    bool empty(){return mul==1&&add==0;}
};
info operator+(const info &a,const flag &b) {
    return a.sz?info(a.mx*b.mul+b.add,a.mn*b.mul+b.add,a.sum*b.mul+b.add*a.sz,a.sz):a;
}
```

```
info operator+(const info &a,const info &b) {
    return info(max(a.mx,b.mx),min(a.mn,b.mn),a.sum+b.sum,a.sz+b.sz);
}
flag operator+(const flag &a,const flag &b) {
    return flag(a.mul*b.mul,a.add*b.mul+b.add);
}
struct node{
    node *c[4],*f;
    flag Cha,All;
    info cha,sub,all;
    bool rev,inr;
    int val;
    void makerev(){rev^=1;swap(c[0],c[1]);}
    void makec(const flag &a){
        Cha=Cha+a;cha=cha+a;val=val*a.mul+a.add;
        all=cha+sub;
    }
    void makes(const flag &a,bool _=1){
        All=All+a;all=all+a;sub=sub+a;
        if(_)makec(a);
    }
    void rz(){
        cha=all=sub=info(-(1<<30),1<<30,0,0);
        if(!inr)all=cha=info(val,val,val,1);
        rep(i,0,2)if(c[i])cha=cha+c[i]->cha,sub=sub+c[i]->sub;
        rep(i,0,4)if(c[i])all=all+c[i]->all;
        rep(i,2,4)if(c[i])sub=sub+c[i]->all;
    }
    void pd(){
        if(rev){
            if(c[0])c[0]->makerev();
            if(c[1])c[1]->makerev();
            rev=0;
        }
        if(!All.empty()){
            rep(i,0,4)if(c[i])c[i]->makes(All,i>=2);
            All=flag(1,0);
        }
        if(!Cha.empty()){
            rep(i,0,2)if(c[i])c[i]->makec(Cha);
            Cha=flag(1,0);
        }
```

```
    }
    node *C(int i){if(c[i])c[i]->pd();return c[i];}
    bool d(int ty){return f->c[ty+1]==this;}
    int D(){rep(i,0,4)if(f->c[i]==this)return i;}
    void sets(node *x,int d){if(x)x->f=this;c[d]=x;}
    bool rt(int ty){
        if(ty==0)return !f||(f->c[0]!=this&&f->c[1]!=this);
        else return !f||!f->inr||!inr;
    }
}nd[maxn*2],*cur=nd+maxn,*pool[maxn],**Cur=pool;
int _cnt;
node *newnode(){
    _cnt++;
    node *x=(Cur==pool)?cur++:*(--Cur);
    rep(i,0,4)x->c[i]=0;x->f=0;
    x->All=x->Cha=flag(1,0);
    x->all=x->cha=info(-(1<<30),(1<<30),0,0);
    x->inr=1;x->rev=0;x->val=0;
    return x;
}
void dele(node *x){*(Cur++)=x;}
void rot(node *x,int ty){
    node *p=x->f;int d=x->d(ty);
    if(!p->f)x->f=0;else p->f->sets(x,p->D());
    p->sets(x->c[!d+ty],d+ty);x->sets(p,!d+ty);p->rz();
}
void splay(node *x,int ty=0){
    while(!x->rt(ty)){
        if(x->f->rt(ty))rot(x,ty);
        else if(x->d(ty)==x->f->d(ty))rot(x->f,ty),rot(x,ty);
        else rot(x,ty),rot(x,ty);
    }x->rz();
}
void add(node *u,node *w){
    w->pd();
    rep(i,2,4)if(!w->c[i]){w->sets(u,i);return;}
    node *x=newnode(),*v;
    for(v=w;v->c[2]->inr;v=v->C(2));
    x->sets(v->c[2],2);x->sets(u,3);
    v->sets(x,2);splay(x,2);
}
```

```
void del(node *w){
    if(w->f->inr){
        w->f->f->sets(w->f->c[5-w->D()],w->f->D());
        dele(w->f);splay(w->f->f,2);
    }else w->f->sets(0,w->D());
    w->f=0;
}
void access(node *w){
    static node *sta[maxn];
    static int top=0;
    node *v=w,*u;
    for(u=w;u;u=u->f)sta[top++]=u;
    while(top)sta[--top]->pd();
    splay(w);
    if(w->c[1])u=w->c[1],w->c[1]=0,add(u,w),w->rz();
    while(w->f){
        for(u=w->f;u->inr;u=u->f);
        splay(u);
        if(u->c[1])w->f->sets(u->c[1],w->D()),splay(w->f,2);
        else del(w);
        u->sets(w,1);
        (w=u)->rz();
    }splay(v);
}
void makert(node *x){
    access(x);x->makerev();
}
node *findp(node *u){
    access(u);u=u->C(0);
    while(u&&u->c[1])u=u->C(1);
    return u;
}
node *findr(node *u){for(;u->f;u=u->f);return u;}
node* cut(node *u){
    node *v=findp(u);
    if(v)access(v),del(u),v->rz();
    return v;
}
void link(node *u,node *v) {
    node* p=cut(u);
    if(findr(u)!=findr(v))p=v;
    if(p)access(p),add(u,p),p->rz();
```

```
}
int main(){
//  freopen("bzoj3153.in","r",stdin);
    n=getint();m=getint();
    static int _u[maxn],_v[maxn];
    rep(i,1,n)_u[i]=getint(),_v[i]=getint();
    rep(i,1,n+1){
        nd[i].val=getint();
        nd[i].rz();
    }
    rep(i,1,n)makert(nd+_u[i]),link(nd+_u[i],nd+_v[i]);
    int root=getint();
    makert(nd+root);
//  deb();
    int x,y,z;
    node *u,*v;
    while(m--){
        int k=getint();x=getint();
        u=nd+x;
        if(k==0||k==3||k==4||k==5||k==11){
            access(u);
            if(k==3||k==4||k==11){
                int ans=u->val;
                rep(i,2,4)if(u->c[i]){
                    info res=u->c[i]->all;
                    if(k==3) ans=min(ans,res.mn);
                    else if(k==4) ans=max(ans,res.mx);
                    else if(k==11) ans+=res.sum;
                }printf("%d\n",ans);
            }else{
                y=getint();
                flag fg(k==5,y);
                u->val=u->val*fg.mul+fg.add;
                rep(i,2,4)if(u->c[i])u->c[i]->makes(fg);
                u->rz();
            }
        }else if(k==2||k==6||k==7||k==8||k==10){
            y=getint();
            makert(u),access(nd+y),splay(u);
            if (k==7||k==8||k==10) {
                info ans=u->cha;
                if (k==7) printf("%d\n",ans.mn);
```

```
                else if (k==8) printf("%d\n",ans.mx);
                else printf("%d\n",ans.sum);
            }else u->makec(flag(k==6,getint()));
            makert(nd+root);
        }else if(k==9)link(u,nd+getint());
        else if(k==1)makert(u),root=x;
    }
    return 0;
}
```

3.4 ST

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

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

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

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

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

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

3.5 可持久化线段树

```
struct node1 {
        int L, R, Lson, Rson, Sum;
```

```
} tree[N * 40];
int root[N], a[N], b[N];
int tot, n, m;
int Real[N];
int Same(int x) {
        ++tot;
        tree[tot] = tree[x];
        return tot;
}
int build(int L, int R) {
        ++tot;
        tree[tot].L = L;
        tree[tot].R = R;
        tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
        if (L == R) return tot;
        int s = tot;
        int mid = (L + R) >> 1;
        tree[s].Lson = build(L, mid);
        tree[s].Rson = build(mid + 1, R);
        return s;
}
int Ask(int Lst, int Cur, int L, int R, int k) {
        if (L == R) return L;
        int Mid = (L + R) >> 1;
        int Left = tree[tree[Cur].Lson].Sum - tree[tree[Lst].Lson].Sum;
        if (Left >= k) return Ask(tree[Lst].Lson, tree[Cur].Lson, L, Mid, k);
        k -= Left;
        return Ask(tree[Lst].Rson, tree[Cur].Rson, Mid + 1, R, k);
}
int Add(int Lst, int pos) {
        int root = Same(Lst);
        tree[root].Sum++;
        if (tree[root].L == tree[root].R) return root;
        int mid = (tree[root].L + tree[root].R) >> 1;
        if (pos <= mid) tree[root].Lson = Add(tree[root].Lson, pos);
        else tree[root].Rson = Add(tree[root].Rson, pos);
        return root;
}
int main() {
        scanf("%d%d", &n, &m);
        int up = 0;
        for (int i = 1; i <= n; i++){
```

```
            scanf("%d", &a[i]);
            b[i] = a[i];
        }
        sort(b + 1, b + n + 1);
        up = unique(b + 1, b + n + 1) - b - 1;
        for (int i = 1; i <= n; i++){
            int tmp = lower_bound(b + 1, b + up + 1, a[i]) - b;
            Real[tmp] = a[i];
            a[i] = tmp;
        }
        tot = 0;
        root[0] = build(1, up);
        for (int i = 1; i <= n; i++){
            root[i] = Add(root[i - 1], a[i]);
        }
        for (int i = 1; i <= m; i++){
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            printf("%d\n", Real[Ask(root[u - 1], root[v], 1, up, w)]);
        }
        return 0;
}
```

3.6 可持久化 Trie

```
int Pre[N];
int n, q, Len, cnt, Lstans;
char s[N];
int First[N], Last[N];
int Root[N];
int Trie_tot;
struct node{
    int To[30];
    int Lst;
}Trie[N];
int tot;
struct node1{
    int L, R, Lson, Rson, Sum;
}tree[N * 25];
int Build(int L, int R){
    ++tot;
    tree[tot].L = L;
```

```cpp
    tree[tot].R = R;
    tree[tot].Lson = tree[tot].Rson = tree[tot].Sum = 0;
    if (L == R) return tot;
    int s = tot;
    int mid = (L + R) >> 1;
    tree[s].Lson = Build(L, mid);
    tree[s].Rson = Build(mid + 1, R);
    return s;
}
int Same(int x){
    ++tot;
    tree[tot] = tree[x];
    return tot;
}
int Add(int Lst, int pos){
    int s = Same(Lst);
    tree[s].Sum++;
    if (tree[s].L == tree[s].R) return s;
    int Mid = (tree[s].L + tree[s].R) >> 1;
    if (pos <= Mid) tree[s].Lson = Add(tree[Lst].Lson, pos);
    else tree[s].Rson = Add(tree[Lst].Rson, pos);
    return s;
}

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

int main(){
    while (scanf("%d", &n) == 1){
        for (int i = 1; i <= Trie_tot; i++){
            for (int j = 1; j <= 26; j++)
                Trie[i].To[j] = 0;
            Trie[i].Lst = 0;
        }
        Trie_tot = 1;
        cnt = 0;
```

```
        for (int ii = 1; ii <= n; ii++){
            scanf("%s", s + 1);
            Len = strlen(s + 1);
            int Cur = 1;
            First[ii] = cnt + 1;
            for (int i = 1; i <= Len; i++){
                int ch = s[i] - 'a' + 1;
                if (Trie[Cur].To[ch] == 0){
                    ++Trie_tot;
                    Trie[Cur].To[ch] = Trie_tot;
                }
                Cur = Trie[Cur].To[ch];
                Pre[++cnt] = Trie[Cur].Lst;
                Trie[Cur].Lst = ii;
            }
            Last[ii] = cnt;
        }
        tot = 0;
        Root[0] = Build(0, n);
        for (int i = 1; i <= cnt; i++){
            Root[i] = Add(Root[i - 1], Pre[i]);
        }
        Lstans = 0;
        scanf("%d", &q);
        for (int ii = 1; ii <= q; ii++){
            int L, R;
            scanf("%d%d", &L, &R);
            L = (L + Lstans) % n + 1;
            R = (R + Lstans) % n + 1;
            if (L > R) swap(L, R);
            int Ret = Ask(Root[First[L] - 1], Root[Last[R]], 0, n, L);
            printf("%d\n", Ret);
            Lstans = Ret;
        }
    }
    return 0;
}
```

3.7 k-d 树

```
long long norm(const long long &x) {
    //    For manhattan distance
```

```
    return std::abs(x);
    //    For euclid distance
    return x * x;
}

struct Point {
    int x, y, id;

    const int& operator [] (int index) const {
        if (index == 0) {
            return x;
        } else {
            return y;
        }
    }

    friend long long dist(const Point &a, const Point &b) {
        long long result = 0;
        for (int i = 0; i < 2; ++i) {
            result += norm(a[i] - b[i]);
        }
        return result;
    }
} point[N];

struct Rectangle {
    int min[2], max[2];

    Rectangle() {
        min[0] = min[1] = INT_MAX;
        max[0] = max[1] = INT_MIN;
    }

    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);
    tree[x].reset(point[mid]);
    for (int i = l; i < r; ++i) {
```

```cpp
            tree[x].rectangle.add(point[i]);
        }
        tree[x].child[0] = build(l, mid, type ^ 1);
        tree[x].child[1] = build(mid + 1, r, type ^ 1);
        return x;
}

int insert(int x, const Point &p, int type = 1) {
        pivot = type;
        if (x == 0) {
            tree[++size].reset(p);
            return size;
        }
        tree[x].rectangle.add(p);
        if (compare(p, tree[x].seperator)) {
            tree[x].child[0] = insert(tree[x].child[0], p, type ^ 1);
        } else {
            tree[x].child[1] = insert(tree[x].child[1], p, type ^ 1);
        }
        return x;
}

//    For minimum distance
void query(int x, const Point &p, std::pair<long long, int> &answer, int type = 1) {
        pivot = type;
        if (x == 0 || tree[x].rectangle.dist(p) > answer.first) {
            return;
        }
        answer = std::min(answer,
                 std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
        if (compare(p, tree[x].seperator)) {
            query(tree[x].child[0], p, answer, type ^ 1);
            query(tree[x].child[1], p, answer, type ^ 1);
        } else {
            query(tree[x].child[1], p, answer, type ^ 1);
            query(tree[x].child[0], p, answer, type ^ 1);
        }
}

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

void query(int x, const Point &p, int k, int type = 1) {
```

```cpp
    pivot = type;
    if (x == 0 ||
        (int)answer.size() == k && tree[x].rectangle.dist(p) > answer.top().first) {
        return;
    }
    answer.push(std::make_pair(dist(tree[x].seperator, p), tree[x].seperator.id));
    if ((int)answer.size() > k) {
        answer.pop();
    }
    if (compare(p, tree[x].seperator)) {
        query(tree[x].child[0], p, k, type ^ 1);
        query(tree[x].child[1], p, k, type ^ 1);
    } else {
        query(tree[x].child[1], p, k, type ^ 1);
        query(tree[x].child[0], p, k, type ^ 1);
    }
}
```

3.8 莫队算法

```cpp
struct node{
        int l, r, id;
        friend bool operator < (const node &a, const node &b){
                if (a.l / Block == b.l / Block) return a.r / Block < b.r / Block;
                return a.l / Block < b.l / Block;
        }
}q[N];
Block = int(sqrt(n));
for (int i = 1; i <= m; i++){
        scanf("%d%d", &q[i].l, &q[i].r);
        q[i].id = i;
}
sort(q + 1, q + 1 + m);
Cur = a[1]; /// Hints: adjust by yourself
Le = Ri = 1;
for (int i = 1; i <= m; i++){
        while (q[i].r > Ri) Ri++, ChangeRi(1, Le, Ri);
        while (q[i].l > Le) ChangeLe(-1, Le, Ri), Le++;
        while (q[i].l < Le) Le--, ChangeLe(1, Le, Ri);
        while (q[i].r < Ri) ChangeRi(-1, Le, Ri), Ri--;
        Ans[q[i].id] = Cur;
}
```

3.9 树上在线莫队

```
bool operator<(qes a,qes b){
    if(dfn[a.x]/B!=dfn[b.x]/B)return dfn[a.x]/B<dfn[b.x]/B;
    if(dfn[a.y]/B!=dfn[b.y]/B)return dfn[a.y]/B<dfn[b.y]/B;
    if(a.tm/B!=b.tm/B)return a.tm/B<b.tm/B;
    return a.tm<b.tm;
}
void vxor(int x){
    if(vis[x])ans-=(LL)W[cnt[col[x]]]*V[col[x]],cnt[col[x]]--;
    else cnt[col[x]]++,ans+=(LL)W[cnt[col[x]]]*V[col[x]];
    vis[x]^=1;
}
void change(int x,int y){
    if(vis[x]){
        vxor(x);col[x]=y;vxor(x);
    }else col[x]=y;
}
void TimeMachine(int tar){//XD
    for(int i=now+1;i<=tar;i++)change(C[i].x,C[i].y);
    for(int i=now;i>tar;i--)change(C[i].x,C[i].pre);
    now=tar;
}
void vxor(int x,int y){
    while(x!=y)if(dep[x]>dep[y])vxor(x),x=fa[x];
    else vxor(y),y=fa[y];
}
    for(int i=1;i<=q;i++){
        int ty=getint(),x=getint(),y=getint();
        if(ty&&dfn[x]>dfn[y])swap(x,y);
        if(ty==0) C[++Csize]=(oper){x,y,pre[x],i},pre[x]=y;
        else Q[Qsize+1]=(qes){x,y,Qsize+1,Csize},Qsize++;
    }sort(Q+1,Q+1+Qsize);
    int u=Q[1].x,v=Q[1].y;
    TimeMachine(Q[1].tm);
    vxor(Q[1].x,Q[1].y);
    int LCA=lca(Q[1].x,Q[1].y);
    vxor(LCA);anss[Q[1].id]=ans;vxor(LCA);
    for(int i=2;i<=Qsize;i++){
        TimeMachine(Q[i].tm);
```

```
        vxor(Q[i-1].x,Q[i].x);
        vxor(Q[i-1].y,Q[i].y);
        int LCA=lca(Q[i].x,Q[i].y);
        vxor(LCA);
        anss[Q[i].id]=ans;
        vxor(LCA);
    }
```

3.10 整体二分

```
struct BIT{
        LL d[maxn];
        inline int lowbit(int x){return x&-x;}
        LL get(int x){
                LL ans=0;
                while(x)ans+=d[x],x-=lowbit(x);
                return ans;
        }
        void updata(int x,LL f){
                while(x<=m)d[x]+=f,x+=lowbit(x);
        }
        void add(int l,int r,LL f){
                updata(l,f);
                updata(r+1,-f);
        }
}T,T2;
int anss[maxn],wana[maxn];
struct qes{
        LL x,y,z;
        qes(LL _x=0,LL _y=0,LL _z=0):
                x(_x),y(_y),z(_z){}
}q[maxn],p[maxn];
bool part(qes &q){
        if(q.y+q.z>=wana[q.x])return 1;
        q.z+=q.y;q.y=0;return 0;
}
void solve(int lef,int rig,int l,int r){
        if(l==r){
                for(int i=lef;i<=rig;i++)if(anss[p[i].x]!=-1)
                anss[p[i].x]=l;return;
        }int mid=(l+r)>>1;
        for(int i=l;i<=mid;i++){
```

```
                if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,q[i].z);
                else T.add(1,q[i].y,q[i].z),T.add(q[i].x,m,q[i].z);
        }for(int i=lef;i<=rig;i++){
                p[i].y=0;
                for(int j=0;j<O[p[i].x].size()&&p[i].y<=int(1e9)+1;j++)
                p[i].y+=T.get(O[p[i].x][j]);
        }for(int i=l;i<=mid;i++){
                if(q[i].x<=q[i].y)T.add(q[i].x,q[i].y,-q[i].z);
                else T.add(1,q[i].y,-q[i].z),T.add(q[i].x,m,-q[i].z);
        }int dv=stable_partition(p+lef,p+rig+1,part)-p-1;
        if(lef<=dv)
        solve(lef,dv,l,mid);
        if(dv+1<=rig)
        solve(dv+1,rig,mid+1,r);
}
```

3.11 树状数组 kth

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

3.12 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;
}
```

3.13 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){
        d[u]=0;
    for(int i=0;i<G[u].size();i++){
        edge e=G[u][i];
        if(cant[e.v])continue;
        d[e.v]=e.w;
```

```
        dfs(e.v,u);

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

3.14 元芳树

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e4+1e4+233;
const int BIT=18;
int n,m,q;
struct edge{
        int u,v,w;
        bool operator==(edge oth)const{
                return u==oth.u && v==oth.v && w==oth.w;
        }
        bool operator!=(edge oth)const{
                return !(*this==oth);
        }
};
vector<edge>G[maxn],T[maxn];

int dfn[maxn],low[maxn],tot,rlen[maxn];
bool ins[maxn];
stack<edge>S;
int Rcnt=0;
```

```
vector<edge>ring[maxn];
vector<int>bel[maxn],sum[maxn],dis[maxn];
int fa[maxn][BIT];
int dep[maxn],dep2[maxn],fw[maxn];
vector<pair<int,int> >ind[maxn];
map<pair<int,int>,int>Mw;
pair<int,int>pack(int a,int b){
        if(a>b)swap(a,b);
        return make_pair(a,b);
}
void tarjan(int u){
        dfn[u]=low[u]=++tot;
        for(int i=0;i<G[u].size();i++){
                edge e=G[u][i];
                if(dfn[e.v])
                        low[u]=min(low[u],dfn[e.v]);
                else{
                        S.push(e);
                        tarjan(e.v);
                        if(low[e.v]==dfn[u]){

                                if(S.top()==e){
                                        fa[e.v][0]=u;
                                        fw[e.v]=e.w;
                                        S.pop();
                                        continue;
                                }

                                Rcnt++;
                                edge ed;
                                do{
                    ed=S.top();S.pop();
                    ring[Rcnt].push_back(ed);
                }while(ed!=e);
                    reverse(ring[Rcnt].begin(),ring[Rcnt].end());
                int last=ring[Rcnt].back().v;
                    ring[Rcnt].push_back((edge){last,u,Mw[pack(last,u)]});
                        }
                        low[u]=min(low[u],low[e.v]);
                }
        }
}
```

```cpp
void up(int u){
	if(dep[u]||u==1)return ;
	if(fa[u][0])up(fa[u][0]);
	dep[u]=dep[fa[u][0]]+1;
	fw[u]+=fw[fa[u][0]];
}
void build(){
	S.push((edge){0,1,0});
	tarjan(1);

	for(int i=1;i<=Rcnt;i++){
		rlen[i]=0;
		sum[i].resize(ring[i].size());
		dis[i].resize(ring[i].size());
		for(int j=0;j<ring[i].size();j++){
			rlen[i]+=ring[i][j].w;
			ind[i].push_back(make_pair(ring[i][j].u,j));
		}
		sum[i][0]=0;
		fw[i+n]=0;
		fa[i+n][0]=ring[i][0].u;
		for(int j=1;j<ring[i].size();j++){
			sum[i][j]=sum[i][j-1]+ring[i][j-1].w;
			dis[i][j]=min(sum[i][j],rlen[i]-sum[i][j]);
			fw[ring[i][j].u]=dis[i][j];
			fa[ring[i][j].u][0]=i+n;
		}
		sort(ind[i].begin(),ind[i].end());
	}

	for(int i=1;i<=n+Rcnt;i++)
		up(i);

	for(int j=1;j<BIT;j++)
	for(int i=1;i<=n+Rcnt;i++)if(fa[i][j-1])
		fa[i][j]=fa[fa[i][j-1]][j-1];

}
pair<int,int>second_lca;
int lca(int u,int v){
	if(dep[u]<dep[v])swap(u,v);
	int d=dep[u]-dep[v];
```

```
	for(int i=0;i<BIT;i++)if(d>>i&1)
		u=fa[u][i];
	if(u==v)return u;
	for(int i=BIT-1;i>=0;i--)if(fa[u][i]!=fa[v][i]){
		u=fa[u][i];
		v=fa[v][i];
	}
	second_lca=make_pair(u,v);
	return fa[u][0];
}
int main(){

	freopen("bzoj2125.in","r",stdin);

	scanf("%d%d%d",&n,&m,&q);
	for(int i=1;i<=m;i++){
		int u,v,w;scanf("%d%d%d",&u,&v,&w);
		G[u].push_back((edge){u,v,w});
		G[v].push_back((edge){v,u,w});
		Mw[pack(u,v)]=w;
	}

	build();
	while(q--){
		int u,v;
		scanf("%d%d",&u,&v);
		int LCA=lca(u,v);
		if(LCA<=n)printf("%d\n",fw[u]+fw[v]-2*fw[LCA]);
		else{
			if(dep[u]<dep[v])swap(u,v);
			int R=LCA-n;
			int uu=second_lca.first;
			int vv=second_lca.second;
			int ans=fw[u]-fw[uu]+fw[v]-fw[vv];
			int uid,vid;
			uid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(uu,-1))->second;
			vid=lower_bound(ind[R].begin(),ind[R].end(),make_pair(vv,-1))->second;
			ans+=min(abs(sum[R][uid]-sum[R][vid]),rlen[R]-abs(sum[R][uid]-sum[R][vid]));
			printf("%d\n",ans);
		}
	}
	return 0;
```

```
}
```

4 图论

4.1 强连通分量

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

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

void solve() {
    stamp = comps = top = 0;
    std::fill(dfn, dfn + n, 0);
    std::fill(comp, comp + n, 0);
    for (int i = 0; i < n; ++i) {
        if (!dfn[i]) {
            tarjan(i);
        }
    }
}
```

4.2 2-SAT 问题

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

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

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

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

```
            return false;
        }
        answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
    }
    return true;
}
```

4.3 二分图最大匹配

4.3.1 Hungary 算法

时间复杂度：$\mathcal{O}(V \cdot E)$

```
vector<int>G[maxn];
int Link[maxn],vis[maxn],T;
bool find(int x){
        for(int i=0;i<G[x].size();i++){
                int v=G[x][i];
                if(vis[v]==T)continue;
                vis[v]=T;
                if(!Link[v]||find(Link[v])){
                        Link[v]=x;
                        return 1;
                }
        }return 0;
}
int Hungarian(int n){
        int ans=0;
        memset(Link,0,sizeof Link);
        for(int i=1;i<=n;i++){
                T++;
                ans+=find(i);
        }return ans;
}
```

4.3.2 Hopcroft Karp 算法

时间复杂度：$\mathcal{O}(\sqrt{V} \cdot E)$

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

bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
```

```
        int y = edge[x][i];
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}

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

```
        if (delta == 0) {
            return answer;
        } else {
            answer += delta;
        }
    }
}
```

4.4 二分图最大权匹配

时间复杂度：$\mathcal{O}(V^4)$

```
int labelx[N], labely[N], match[N], slack[N];
bool visitx[N], visity[N];

bool dfs(int x) {
    visitx[x] = true;
    for (int y = 0; y < n; ++y) {
        if (visity[y]) {
            continue;
        }
        int delta = labelx[x] + labely[y] - graph[x][y];
        if (delta == 0) {
            visity[y] = true;
            if (match[y] == -1 || dfs(match[y])) {
                match[y] = x;
                return true;
            }
        } else {
            slack[y] = std::min(slack[y], delta);
        }
    }
    return false;
}

int solve() {
    for (int i = 0; i < n; ++i) {
        match[i] = -1;
        labelx[i] = INT_MIN;
        labely[i] = 0;
        for (int j = 0; j < n; ++j) {
            labelx[i] = std::max(labelx[i], graph[i][j]);
```

```
        }
    }
    for (int i = 0; i < n; ++i) {
        while (true) {
            std::fill(visitx, visitx + n, 0);
            std::fill(visity, visity + n, 0);
            for (int j = 0; j < n; ++j) {
                slack[j] = INT_MAX;
            }
            if (dfs(i)) {
                break;
            }
            int delta = INT_MAX;
            for (int j = 0; j < n; ++j) {
                if (!visity[j]) {
                    delta = std::min(delta, slack[j]);
                }
            }
            for (int j = 0; j < n; ++j) {
                if (visitx[j]) {
                    labelx[j] -= delta;
                }
                if (visity[j]) {
                    labely[j] += delta;
                } else {
                    slack[j] -= delta;
                }
            }
        }
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += graph[match[i]][i];
    }
    return answer;
}
```

4.5 最大流 (dinic)

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
struct edge{int u,v,cap,flow;};
vector<edge>edges;
```

```
vector<int>G[maxn];
int s,t;
int cur[maxn],d[maxn];
void add(int u,int v,int cap){
        edges.push_back((edge){u,v,cap,0});
        G[u].push_back(edges.size()-1);
        edges.push_back((edge){v,u,0,0});
        G[v].push_back(edges.size()-1);
}
bool bfs(){
        static int vis[maxn];
        memset(vis,0,sizeof vis);vis[s]=1;
        queue<int>q;q.push(s);d[s]=0;
        while(!q.empty()){
                int u=q.front();q.pop();
                for(int i=0;i<G[u].size();i++){
                        edge e=edges[G[u][i]];if(vis[e.v]||e.cap==e.flow)continue;
                        d[e.v]=d[u]+1;vis[e.v]=1;q.push(e.v);
                }
        }return vis[t];
}
int dfs(int u,int a){
        if(u==t||!a)return a;
        int flow=0,f;
        for(int &i=cur[u];i<G[u].size();i++){
                edge e=edges[G[u][i]];
                if(d[e.v]==d[u]+1&&(f=dfs(e.v,min(a,e.cap-e.flow)))>0){
                        edges[G[u][i]].flow+=f;
                        edges[G[u][i]^1].flow-=f;
                        flow+=f;a-=f;if(!a)break;
                }
        }return flow;
}
int dinic(){
        int flow=0,x;
        while(bfs()){
                memset(cur,0,sizeof cur);
                while(x=dfs(s,INT_MAX)){
                        flow+=x;
                        memset(cur,0,sizeof cur);
                }
        }return flow;
```

```
}
```

4.6 最大流（sap）

时间复杂度：$\mathcal{O}(V^2 \cdot E)$

```
int g[T], adj[M], nxt[M], f[M];
int cnt[T], dist[T], cur[T], fa[T], dat[T];
void Ins(int x, int y, int ff, int rf){
        adj[++tot] = y; nxt[tot] = g[x]; g[x] = tot; f[tot] = ff;
        adj[++tot] = x; nxt[tot] = g[y]; g[y] = tot; f[tot] = rf;
}
int sap(int s, int t){
        int x, sum;
        for (int i = 1; i <= t; i++){
                dist[i] = 1;
                cur[i] = g[i];
                fa[i] = 0;
                dat[i] = 0;
                cnt[i] = 0;
        }
        cnt[0] = 1; cnt[1] = t - 1;
        dist[t] = 0;
        dat[s] = INF;
        x = s;
        sum = 0;
        while (1){
                int p;
                for (p = cur[x]; p; p = nxt[p]){
                        if (f[p] > 0 && dist[adj[p]] == dist[x] - 1) break;
                }
                if (p > 0){
                        cur[x] = p;
                        fa[adj[p]] = p;
                        dat[adj[p]] = min(dat[x], f[p]);
                        x = adj[p];
                        if (x == t){
                                sum += dat[x];
                                while (x != s){
                                        f[fa[x]] -= dat[t];
                                        f[fa[x] ^ 1] += dat[t];
                                        x = adj[fa[x] ^ 1];
                                }
```

```
            }
        } else {
            cnt[dist[x]] --;
            if (cnt[dist[x]] == 0) return sum;
            dist[x] = t + 1;
            for (int p = g[x]; p; p = nxt[p]){
                if (f[p] > 0 && dist[adj[p]] + 1 < dist[x]){
                    dist[x] = dist[adj[p]] + 1;
                    cur[x] = p;
                }
            }
            cnt[dist[x]]++;
            if (dist[s] > t) return sum;
            if (x != s) x = adj[fa[x] ^ 1];
        }
    }
}
/*
tot = 1
edges' id start from 2
remember to clean g
t is the number of points
*/
```

4.7 最小费用最大流

4.7.1 稀疏图

时间复杂度：$\mathcal{O}(V \cdot E^2)$

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

```
            cost[size++] = w;
        }
} e;

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

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

bool augment() {
    static int dist[N], occur[N];
    std::vector<int> queue;
    std::fill(dist, dist + n, INT_MAX);
    std::fill(occur, occur + n, 0);
    dist[source] = 0;
    occur[source] = true;
    queue.push_back(source);
    for (int head = 0; head < (int)queue.size(); ++head) {
        int x = queue[head];
        for (int i = e.last[x]; ~i; i = e.succ[i]) {
            int y = e.other[i];
            if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
                dist[y] = dist[x] + e.cost[i];
                prev[y] = i;
                if (!occur[y]) {
                    occur[y] = true;
                    queue.push_back(y);
                }
            }
        }
        occur[x] = false;
    }
    return dist[target] < INT_MAX;
}

std::pair<int, int> solve() {
    std::pair<int, int> answer = std::make_pair(0, 0);
    while (augment()) {
        int number = INT_MAX;
```

```
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            number = std::min(number, e.flow[prev[i]]);
        }
        answer.first += number;
        for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
            e.flow[prev[i]] -= number;
            e.flow[prev[i] ^ 1] += number;
            answer.second += number * e.cost[prev[i]];
        }
    }
    return answer;
}
```

4.7.2 稠密图

使用条件：费用非负

时间复杂度：$\mathcal{O}(V \cdot E^2)$

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

int n, source, target, flow, cost;
int slack[N], dist[N];
bool visit[N];

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

```

bool relabel() {
    int delta = INT_MAX;
    for (int i = 0; i < n; ++i) {
        if (!visit[i]) {
            delta = std::min(delta, slack[i]);
        }
        slack[i] = INT_MAX;
    }
    if (delta == INT_MAX) {
        return true;
    }
    for (int i = 0; i < n; ++i) {
        if (visit[i]) {
            dist[i] += delta;
        }
    }
    return false;
}

int dfs(int x, int answer) {
    if (x == target) {
        flow += answer;
        cost += answer * (dist[source] - dist[target]);
        return answer;
    }
    visit[x] = true;
    int delta = answer;
    for (int i = e.last[x]; ~i; i = e.succ[i]) {
        int y = e.other[i];
        if (e.flow[i] > 0 && !visit[y]) {
            if (dist[y] + e.cost[i] == dist[x]) {
                int number = dfs(y, std::min(e.flow[i], delta));
                e.flow[i] -= number;
                e.flow[i ^ 1] += number;
                delta -= number;
                if (delta == 0) {
                    dist[x] = INT_MIN;
                    return answer;
                }
            } else {
                slack[y] = std::min(slack[y], dist[y] + e.cost[i] - dist[x]);
```

```
            }
        }
    }
    return answer - delta;
}

std::pair<int, int> solve() {
    flow = cost = 0;
    std::fill(dist, dist + n, 0);
    do {
        do {
            fill(visit, visit + n, 0);
        } while (dfs(source, INT_MAX));
    } while (!relabel());
    return std::make_pair(flow, cost);
}
```

4.8 一般图最大匹配

时间复杂度：$\mathcal{O}(V^3)$

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

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

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

int lca(int x, int y) {
    static int stamp = 0;
    stamp++;
```

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

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

void augment(int source) {
    queue.clear();
    for (int i = 0; i < n; ++i) {
        next[i] = visit[i] = -1;
        belong[i] = i;
```

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

int solve() {
```

```cpp
    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;
}
```

4.9 无向图全局最小割

时间复杂度：$\mathcal{O}(V^3)$

注意事项：处理重边时，应该对边权累加

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

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

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

4.10 有根树的同构

时间复杂度：$\mathcal{O}(VlogV)$

```
const unsigned long long MAGIC = 4423;

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

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

```
    }
    for (int index = n - 1; index >= 0; --index) {
        int x = queue[index];
        hash[x] = std::make_pair(0, 0);

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

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

4.11 哈密尔顿回路（ORE 性质的图）

ORE 性质：

$$\forall x, y \in V \wedge (x, y) \notin E \quad s.t. \quad deg_x + deg_y \geq n$$

返回结果：从顶点 1 出发的一个哈密尔顿回路

使用条件：$n \geq 3$

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

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

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

```
    }
    return 0;
}

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

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

4.12 必经点树

```
vector<int>G[maxn],rG[maxn],dom[maxn];
int n,m;
int dfn[maxn],rdfn[maxn],dfs_c,semi[maxn],idom[maxn],fa[maxn];
```

```cpp
struct ufsets{
    int fa[maxn],best[maxn];
    int find(int x){
        if(fa[x]==x)
            return x;
        int f=find(fa[x]);
        if(dfn[semi[best[x]]]>dfn[semi[best[fa[x]]]])
            best[x]=best[fa[x]];
        fa[x]=f;
        return f;
    }
    int getbest(int x){
        find(x);
        return best[x];
    }
    void init(){
        for(int i=1;i<=n;i++)
            fa[i]=best[i]=i;
    }
}uf;
void init(){
    uf.init();
    for(int i=1;i<=n;i++){
        semi[i]=i;
        idom[i]=0;
        fa[i]=0;
        dfn[i]=rdfn[i]=0;
    }
    dfs_c=0;
}
void dfs(int u){
    dfn[u]=++dfs_c;
    rdfn[dfn[u]]=u;
    for(int i=0;i<G[u].size();i++){
        int v=G[u][i];
        if(!dfn[v]){
            fa[v]=u;
            dfs(v);
        }
    }
}
```

```
void tarjan(){
    for(int i=n;i>1;i--){
        int tmp=1e9;
        int y=rdfn[i];
        for(int i=0;i<rG[y].size();i++){
            int x=rG[y][i];
            tmp=min(tmp,dfn[semi[uf.getbest(x)]]);
        }
        semi[y]=rdfn[tmp];
        int x=fa[y];
        dom[semi[y]].push_back(y);
        uf.fa[y]=x;
        for(int i=0;i<dom[x].size();i++){
            int z=dom[x][i];
            if(dfn[semi[uf.getbest(z)]]<dfn[x])
                idom[z]=uf.getbest(z);
            else
                idom[z]=semi[z];
        }
        dom[x].clear();
    }
    semi[rdfn[1]]=1;
    for(int i=2;i<=n;i++){
        int x=rdfn[i];
        if(idom[x]!=semi[x])
            idom[x]=idom[idom[x]];

    }
    idom[rdfn[1]]=0;
}
init();
dfs(1);
tarjan();
```

5 字符串

5.1 模式匹配

5.1.1 KMP 算法

```
void build(char *pattern) {
    int length = (int)strlen(pattern + 1);
```

```cpp
    fail[0] = -1;
    for (int i = 1, j; i <= length; ++i) {
        for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j = fail[j]);
        fail[i] = j + 1;
    }
}

void solve(char *text, char *pattern) {
    int length = (int)strlen(text + 1);
    for (int i = 1, j; i <= length; ++i) {
        for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);
        match[i] = j + 1;
    }
}
///Hint: 1 - Base
```

5.1.2 扩展 KMP 算法

返回结果：

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

```cpp
void solve(char *text, int length, int *next) {
    int j = 0, k = 1;
    for (; j + 1 < length && text[j] == text[j + 1]; j++);
    next[0] = length - 1;
    next[1] = j;
    for (int i = 2; i < length; ++i) {
        int far = k + next[k] - 1;
        if (next[i - k] < far - i + 1) {
            next[i] = next[i - k];
        } else {
            j = std::max(far - i + 1, 0);
            for (; i + j < length && text[j] == text[i + j]; j++);
            next[i] = j;
            k = i;
        }
    }
}
/// 0 - Base
```

5.1.3 AC 自动机

```
struct Node{
	int Next[30], fail, mark;
}Tree[N];

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

	for (int i = 1; i <= n; i++){
		char c;
		int now = 1;
		scanf("%s", s + 1);
		int Length = strlen(s + 1);
		for (int j = 1; j <= Length; j++){
			c = s[j];
			if (Tree[now].Next[c - 'a']) now = Tree[now].Next[c - 'a']; else
				Tree[now].Next[c - 'a'] = ++ cnt, now = cnt;
		}
	}
}

void Build_Ac(){
	int en = 0;
	Q[0] = 1;
	for (int fi = 0; fi <= en; fi++){
		int now = Q[fi];
		for (int next = 0; next < 26; next++)
			if (Tree[now].Next[next])
			{
				int k = Tree[now].Next[next];
				if (now == 1) Tree[k].fail = 1; else
				{
					int h = Tree[now].fail;
					while (h && !Tree[h].Next[next]) h = Tree[h].fail;
					if (!h) Tree[k].fail = 1;
					else Tree[k].fail = Tree[h].Next[next];
				}
				Q[++ en] = k;
			}
	}
}

```

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

5.2 后缀三姐妹

5.2.1 后缀数组

```
struct Sa{
	int heap[N],s[N],sa[N],r[N],tr[N],sec[N],m,cnt;
	int h[19][N];

	void Prep(){
		for (int i=1; i<=m; i++) heap[i]=0;
		for (int i=1; i<=n; i++) heap[s[i]]++;
		for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
		for (int i=n; i>=1; i--) sa[heap[s[i]]--]=i;
		r[sa[1]]=1; cnt=1;
		for (int i=2; i<=n; i++){
			if (s[sa[i]]!=s[sa[i-1]]) cnt++;
			r[sa[i]]=cnt;
		}
		m=cnt;
	}

	void Suffix(){
		int j=1;
		while (cnt<n){
			cnt=0;
			for (int i=n-j+1; i<=n; i++) sec[++cnt]=i;
			for (int i=1; i<=n; i++) if (sa[i]>j)
				sec[++cnt]=sa[i]-j;
			for (int i=1; i<=n; i++) tr[i]=r[sec[i]];
			for (int i=1; i<=m; i++) heap[i]=0;
			for (int i=1; i<=n; i++) heap[tr[i]]++;
			for (int i=2; i<=m; i++) heap[i]+=heap[i-1];
			for (int i=n; i>=1; i--)
					sa[heap[tr[i]]--]=sec[i];
			tr[sa[1]]=1; cnt=1;
			for (int i=2; i<=n; i++){
				if ((r[sa[i]]!=r[sa[i-1]]) || (r[sa[i]+j]!=r[sa[i-1]+j]))
					cnt++;
				tr[sa[i]]=cnt;
			}
```

```
                for (int i=1; i<=n; i++) r[i]=tr[i];
                m=cnt; j=j+j;
            }
        }

        void Calc(){
            int k=0;
            for (int i=1; i<=n; i++){
                if (r[i]==1) continue;
                int j=sa[r[i]-1];
                while ((i+k<=n) && (j+k<=n) && (s[i+k]==s[j+k])) k++;
                h[0][r[i]]=k;
                if (k) k--;
            }
            for (int i=1; i<19; i++)
                for (int j=1; j+(1 << i)-1<=n; j++)
                    h[i][j]=min(h[i-1][j],h[i-1][j + (1 << (i - 1)) + 1]);
        }

        int Query(int L,int R){
            L=r[L], R=r[R];
            if (L>R) swap(L,R);
            L++;
            int l0 = Lg[R-L+1];
            return min(h[l0][L],h[l0][R-(1 << l0)+1]);
        }

        void Work(){
            Prep(); Suffix(); Calc();
        }
}P,S;

/// Hints : 1 - Base
```

5.2.2 后缀数组 (dc3)

```
//`DC3 待排序的字符串放在 r 数组中，从 r[0] 到 r[n-1]，长度为 n，且最大值小于 m.`
//`约定除 r[n-1] 外所有的 r[i] 都大于 0，r[n-1]=0。`
//`函数结束后，结果放在 sa 数组中，从 sa[0] 到 sa[n-1]。`
//`r 必须开长度乘 3`
#define maxn 10000
#define F(x) ((x)/3+((x)%3==1?0:tb))
```

```
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)

int wa[maxn],wb[maxn],wv[maxn],wss[maxn];
int s[maxn*3],sa[maxn*3];
int c0(int *r,int a,int b)
{
        return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];
}
int c12(int k,int *r,int a,int b)
{
        if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);
        else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];
}
void sort(int *r,int *a,int *b,int n,int m)
{
        int i;
        for(i=0;i<n;i++) wv[i]=r[a[i]];
        for(i=0;i<m;i++) wss[i]=0;
        for(i=0;i<n;i++) wss[wv[i]]++;
        for(i=1;i<m;i++) wss[i]+=wss[i-1];
        for(i=n-1;i>=0;i--) b[--wss[wv[i]]]=a[i];
}
void dc3(int *r,int *sa,int n,int m)
{
        int i,j,*rn=r+n,*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;
        r[n]=r[n+1]=0;
        for(i=0;i<n;i++)
                if(i%3!=0) wa[tbc++]=i;
        sort(r+2,wa,wb,tbc,m);
        sort(r+1,wb,wa,tbc,m);
        sort(r,wa,wb,tbc,m);
        for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)
                rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;
        if (p<tbc) dc3(rn,san,tbc,p);
        else for (i=0;i<tbc;i++) san[rn[i]]=i;
        for (i=0;i<tbc;i++)
                if(san[i]<tb) wb[ta++]=san[i]*3;
        if(n%3==1) wb[ta++]=n-1;
        sort(r,wb,wa,ta,m);
        for(i=0;i<tbc;i++)
                wv[wb[i]=G(san[i])]=i;
        for(i=0,j=0,p=0;i<ta && j<tbc;p++)
```

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

int main(){
        int n,m=0;
        scanf("%d",&n);
        for (int i=0;i<n;i++) scanf("%d",&s[i]),s[i]++,m=max(s[i]+1,m);
        printf("%d\n",m);
        s[n++]=0;
        dc3(s,sa,n,m);
        for (int i=0;i<n;i++) printf("%d ",sa[i]);printf("\n");
}
```

5.2.3 后缀自动机-多串 LCS

对一个串建后缀自动机，其他串在上面匹配，因为是求所有串的公共子串，所以每个点记录每个串最长匹配长度的最小值，最后找到所有点中最长的一个即可。一个注意事项就是，当走到一个点时，还要更新它的 parent 树上的祖先的匹配长度，数组开两倍啦啦啦!

```
struct Node{
        int len, fail;
        int To[30];
}T[N];
int Lst, Root, tot, ans;
char s[N];
int Len[N], Ans[N], Ord[N];
void Add(int x, int l){
        int Nt = ++tot, p = Lst;
        T[Nt].len = l;
        for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
        if (!p) T[Nt].fail = Root; else
        if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
        else{
                int q = ++tot, qt = T[p].To[x];
                T[q] = T[qt];
                T[q].len = T[p].len + 1;
                T[qt].fail = T[Nt].fail = q;
                for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
```

```cpp
bool cmp(int a, int b){
	return T[a].len < T[b].len;
}
int main(){
	scanf("%s", s + 1);
	int n = strlen(s + 1);
	ans = n;
	Root = tot = Lst = 1;
	for (int i = 1; i <= n; i++)
		Add(s[i] - 'a' + 1, i);
	for (int i = 1; i <= tot; i++)
		Ord[i] = i;
	sort(Ord + 1, Ord + tot + 1, cmp);
	for (int i = 1; i <= tot; i++)
		Ans[i] = T[i].len;
	bool flag = 0;
	while (scanf("%s", s + 1) != EOF){
		flag = 1;
		int n = strlen(s + 1);
		int p = Root, len = 0;
		for (int i = 1; i <= tot; i++) Len[i] = 0;
		for (int i = 1; i <= n; i++){
			int x = s[i] - 'a' + 1;
			if (T[p].To[x]) len++, p = T[p].To[x];
			else {
				while (p && !T[p].To[x]) p = T[p].fail;
				if (!p) p = Root, len = 0;
				else len = T[p].len + 1, p = T[p].To[x];
			}
			Len[p] = max(Len[p], len);
		}
		for (int i = tot; i >= 1; i--){
			int Cur = Ord[i];
			Ans[Cur] = min(Ans[Cur], Len[Cur]);
			if (Len[Cur] && T[Cur].fail)
				Len[T[Cur].fail] = T[T[Cur].fail].len;
		}
	}
	if (flag){
		ans = 0;
		for (int i = 1; i <= tot; i++){
			ans = max(ans, Ans[i]);
```

```
                }
        }
        printf("%d\n", ans);
        return 0;
}
```

5.2.4 后缀自动机-各长度字串出现次数最大值

给一个字符串 S，令 F(x) 表示 S 的所有长度为 x 的子串中，出现次数的最大值。 构建字符串的自动机，对于每个节点，right 集合大小就是出现次数，maxs 就是它代表的最长长度，那么我们用 |right(x)| 去更新 f[maxs[x]] 的值，最后从大到小用 f[i] 去更新 f[i-1] 的值即可

```
struct Node{
        int len, fail;
        int To[30];
}T[N];
int Lst, Root, tot, n;
char s[N];
int Ord[N], Ans[N], Ways[N], heap[N];
void Add(int x, int l){
        int Nt = ++tot, p = Lst;
        T[Nt].len = l;
        for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
        if (!p) T[Nt].fail = Root; else
        if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
        else{
                int q = ++tot, qt = T[p].To[x];
                T[q] = T[qt];
                T[q].len = T[p].len + 1;
                T[qt].fail = T[Nt].fail = q;
                for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
bool cmp(int a, int b){
        return T[a].len < T[b].len;
}
void sort(){
        for (int i = 1; i <= tot; i++) heap[T[i].len]++;
        for (int i = 1; i <= n; i++) heap[i] += heap[i-1];
        for (int i = 1; i <= tot; i++) Ord[heap[T[i].len]--]=i;
}
int main(){
```

```
	scanf("%s", s + 1);
	n = strlen(s + 1);
	Root = tot = Lst = 1;
	for (int i = 1; i <= n; i++)
		Add(s[i] - 'a' + 1, i);
	sort();
	memset(Ways , 0, sizeof(Ways));
	for (int i = 1, p = Root; i <= n; i++)
		p = T[p].To[s[i] - 'a' + 1], Ways[p] = 1;
	for (int i = tot; i >= 1; i--){
		int Cur = Ord[i];
		if (T[Cur].fail == 0) continue;
		Ways[T[Cur].fail] += Ways[Cur];
	}
	for (int i = 1; i <= tot; i++)
		Ans[T[i].len] = max(Ans[T[i].len], Ways[i]);
	for (int i = n; i >= 1; i--)
		Ans[i] = max(Ans[i + 1], Ans[i]);
	for (int i = 1; i <= n; i++)
		printf("%d\n", Ans[i]);
	return 0;
}
```

5.2.5 后缀自动机-两串 LCS

```
struct node{
	int len, fail;
	int To[27];
}T[N];
char a[N], b[N];
int Lst, Root, tot;
void add(int x, int l){
	int Nt = ++tot, p = Lst;
	T[Nt].len = l;
	for (;p && !T[p].To[x]; p = T[p].fail) T[p].To[x] = Nt;
	if (!p) T[Nt].fail = Root;
	else
	if (T[T[p].To[x]].len == T[p].len + 1) T[Nt].fail = T[p].To[x];
	else{
		int q = ++tot, qt = T[p].To[x];
		T[q] = T[qt];
		T[q].len  = T[p].len + 1;
```

```
            T[qt].fail = T[Nt].fail = q;
            for (;p && T[p].To[x] == qt; p = T[p].fail) T[p].To[x] = q;
        }
        Lst = Nt;
}
int main(){
        while (scanf("%s%s", a + 1, b + 1) == 2){
                int n = strlen(a + 1);
                Lst = Root = tot = 1;
                for (int i = 1; i <= n; i++)
                        add(a[i] - 'a' + 1, i);
                int m = strlen(b + 1);
                int p = Root, len = 0;
                int Ans = 0;
                for (int i = 1; i <= m; i++){
                        int x = b[i] - 'a' + 1;
                        if (T[p].To[x]) len++, p = T[p].To[x];
                        else {
                                while (p && !T[p].To[x]) p = T[p].fail;
                                if (!p) p = Root, len = 0;
                                else len = T[p].len + 1, p = T[p].To[x];
                        }
                        if (len > Ans) Ans = len;
                }
                printf("%d\n", Ans);
                for (int i = 1; i <= tot; i++){
                        T[i].len = T[i].fail = 0;
                        for (int j = 1; j <= 26; j++)
                                T[i].To[j] = 0;
                }
        }
        return 0;
}
//Hints£ºSAM + Longest common subsequence
```

5.3 回文三兄弟

5.3.1 马拉车

```
void Manacher(){
        R[1] = 1;
        for (int i = 2, j = 1; i <= length; i++){
```

```
            if (j + R[j] <= i){
                R[i] = 0;
            } else {
                R[i] = min(R[j * 2 - i], j + R[j] - i);
            }
            while (i - R[i] >= 1 && i + R[i] <= length
                && text[i - R[i]] == text[i + R[i]]){
                R[i]++;
            }
            if (i + R[i] > j + R[j]){
                j = i;
            }
        }
}
        length = 0;
        int n = strlen(s + 1);
        for (int i = 1; i <= n; i++){
            text[++length] = '*';
            text[++length] = s[i];
        }
        text[++length] = '*';
/// Hints: 1 - Base
```

5.3.2 回文自动机 (zky)

```
struct PAM{
        int tot,last,str[maxn],nxt[maxn][26],n;
        int len[maxn],suf[maxn],cnt[maxn];
        int newnode(int l){
                len[tot]=l;
                return tot++;
        }
        void init(){
                tot=0;
                newnode(0);// tree0 is node 0
                newnode(-1);// tree-1 is node 1
                str[0]=-1;
                suf[0]=1;
        }
        int find(int x){
                while(str[n-len[x]-1]!=str[n])x=suf[x];
                return x;
```

```
    }
    void add(int c){
        str[++n]=c;
        int u=find(last);
        if(!nxt[u][c]){
            int v=newnode(len[u]+2);
            suf[v]=nxt[find(suf[u])][c];
            nxt[u][c]=v;
        }last=nxt[u][c];
        cnt[last]++;
    }
    void count(){
        for(int i=tot-1;i>=0;i--)cnt[suf[i]]+=cnt[i];
    }
}P;
int main(){
    P.init();
    for(int i=0;i<n;i++)
        P.add(s[i]-'a');
    P.count();
```

5.4 循环串最小表示

```
string sol(char *s){
    int n=strlen(s);
    int i=0,j=1,k=0,p;
    while(i<n&&j<n&&k<n){
        int t=s[(i+k)%n]-s[(j+k)%n];
        if(t==0)k++;
        else if(t<0)j+=k+1,k=0;
        else i+=k+1,k=0;
        if(i==j)j++;
    }p=min(i,j);
    string S;
    for(int i=p;i<p+n;i++)S.push_back(s[i%n]);
    return S;
}
```

6 计算几何

6.1 二维基础

6.1.1 点类

```
int sgn(double x){return (x>eps)-(x<-eps);}
int sgn(double a,double b){return sgn(a-b);}
double sqr(double x){return x*x;}
struct P{
	double x,y;
	P(){}
	P(double x,double y):x(x),y(y){}
	double len2(){
		return sqr(x)+sqr(y);
	}
	double len(){
		return sqrt(len2());
	}
	void print(){
		printf("(%.3f,%.3f)\n",x,y);
	}
	P turn90(){return P(-y,x);}
	P norm(){return P(x/len(),y/len());}
};
bool operator==(P a,P b){
	return !sgn(a.x-b.x) and !sgn(a.y-b.y);
}
P operator+(P a,P b){
	return P(a.x+b.x,a.y+b.y);
}
P operator-(P a,P b){
	return P(a.x-b.x,a.y-b.y);
}
P operator*(P a,double b){
	return P(a.x*b,a.y*b);
}
P operator/(P a,double b){
	return P(a.x/b,a.y/b);
}
double operator^(P a,P b){
	return a.x*b.x + a.y*b.y;
}
```

```
double operator*(P a,P b){
        return a.x*b.y - a.y*b.x;
}
double det(P a,P b,P c){
        return (b-a)*(c-a);
}
double dis(P a,P b){
        return (b-a).len();
}
double Area(vector<P>poly){
        double ans=0;
        for(int i=1;i<poly.size();i++)
                ans+=(poly[i]-poly[0])*(poly[(i+1)%poly.size()]-poly[0]);
        return fabs(ans)/2;
}
struct L{
        P a,b;
        L(){}
        L(P a,P b):a(a),b(b){}
        P v(){return b-a;}
};
bool onLine(P p,L l){
        return sgn((l.a-p)*(l.b-p))==0;
}
bool onSeg(P p,L s){
        return onLine(p,s) and sgn((s.b-s.a)^(p-s.a))>=0 and sgn((s.a-s.b)^(p-s.b))>=0;
}
bool parallel(L l1,L l2){
        return sgn(l1.v()*l2.v())==0;
}
P intersect(L l1,L l2){
        double s1=det(l1.a,l1.b,l2.a);
        double s2=det(l1.a,l1.b,l2.b);
        return (l2.a*s2-l2.b*s1)/(s2-s1);
}
P project(P p,L l){
        return l.a+l.v()*((p-l.a)^l.v())/l.v().len2();
}
double dis(P p,L l){
        return fabs((p-l.a)*l.v())/l.v().len();
}
```

6.1.2 凸包

```
vector<P> convex(vector<P>p){
        sort(p.begin(),p.end());
        vector<P>ans,S;
    for(int i=0;i<p.size();i++){
            while(S.size()>=2
                            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                                    S.pop_back();
            S.push_back(p[i]);
    }//dw
    ans=S;
    S.clear();
        for(int i=(int)p.size()-1;i>=0;i--){
            while(S.size()>=2
                            && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                                    S.pop_back();
            S.push_back(p[i]);
        }//up
        for(int i=1;i+1<S.size();i++)
                ans.push_back(S[i]);
        return ans;
}
```

6.1.3 半平面交

```
struct P{
        int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
};
struct L{
        bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
        L push() const{ // push out eps
                const double eps = 1e-10;
                P delta = (b - a).turn90().norm() * eps;
                return L(a - delta, b - delta);
        }
};
bool sameDir(const L &l0, const L &l1) {
        return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
}
bool operator < (const P &a, const P &b) {
        if (a.quad() != b.quad())
```

```cpp
                return a.quad() < b.quad();
        else
                return sgn((a*b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
        if (sameDir(l0, l1))
                return l1.onLeft(l0.a);
        else
                return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(const L &u, const L &v, const L &w) {
        return w.onLeft(intersect(u, v));
}
vector<P> intersection(vector<L> &l) {
        sort(l.begin(), l.end());
        deque<L> q;
        for (int i = 0; i < (int)l.size(); ++i) {
                if (i && sameDir(l[i], l[i - 1])) {
                        continue;
                }
                while (q.size() > 1
                        && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
                                q.pop_back();
                while (q.size() > 1
                        && !check(q[1], q[0], l[i]))
                                q.pop_front();
                q.push_back(l[i]);
        }
        while (q.size() > 2
                && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
                        q.pop_back();
        while (q.size() > 2
                && !check(q[1], q[0], q[q.size() - 1]))
                        q.pop_front();
        vector<P> ret;
        for (int i = 0; i < (int)q.size(); ++i)
        ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
        return ret;
}
```

6.1.4 最近点对

```cpp
bool byY(P a,P b){return a.y<b.y;}
LL solve(P *p,int l,int r){
	LL d=1LL<<62;
	if(l==r)
		return d;
	if(l+1==r)
		return dis2(p[l],p[r]);
	int mid=(l+r)>>1;
	d=min(solve(l,mid),d);
	d=min(solve(mid+1,r),d);
	vector<P>tmp;
	for(int i=l;i<=r;i++)
		if(sqr(p[mid].x-p[i].x)<=d)
			tmp.push_back(p[i]);
	sort(tmp.begin(),tmp.end(),byY);
	for(int i=0;i<tmp.size();i++)
		for(int j=i+1;j<tmp.size()&&j-i<10;j++)
			d=min(d,dis2(tmp[i],tmp[j]));
	return d;
}
```

6.1.5 最小圆覆盖

```cpp
struct line{
	point p,v;
};
point Rev(point v){return point(-v.y,v.x);}
point operator*(line A,line B){
	point u=B.p-A.p;
	double t=(B.v*u)/(B.v*A.v);
	return A.p+A.v*t;
}
point get(point a,point b){
	return (a+b)/2;
}
point get(point a,point b,point c){
	if(a==b)return get(a,c);
	if(a==c)return get(a,b);
	if(b==c)return get(a,b);
	line ABO=(line){(a+b)/2,Rev(a-b)};
	line BCO=(line){(c+b)/2,Rev(b-c)};
	return ABO*BCO;
```

```
}
int main(){
	scanf("%d",&n);
	for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
	random_shuffle(p+1,p+1+n);
	O=p[1];r=0;
	for(int i=2;i<=n;i++){
		if(dis(p[i],O)<r+1e-6)continue;
		O=get(p[1],p[i]);r=dis(O,p[i]);
		for(int j=1;j<i;j++){
			if(dis(p[j],O)<r+1e-6)continue;
			O=get(p[i],p[j]);r=dis(O,p[i]);
			for(int k=1;k<j;k++){
				if(dis(p[k],O)<r+1e-6)continue;
				O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
			}
		}
	}printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
	return 0;
}s
```

6.1.6 凸包快速询问

```
/*
   给定凸包，log n 内完成各种询问，具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   INF 为坐标范围，需要定义点类大于号
   改成实数只需修改 sign 函数，以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点，面积非空，以及 pair(x,y) 的最小点放在第一个
*/
const int INF = 1000000000;
struct Convex
{
	int n;
	vector<Point> a, upper, lower;
	Convex(vector<Point> _a) : a(_a) {
		n = a.size();
		int ptr = 0;
		for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
```

```cpp
        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
        upper.push_back(a[0]);
    }
    int sign(long long x) { return x < 0 ? -1 : x > 0; }
    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
        int l = 0, r = (int)convex.size() - 2;
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
            else l = mid;
        }
        return max(make_pair(vec.det(convex[r]), r)
            , make_pair(vec.det(convex[0]), 0));
    }
    void update_tangent(const Point &p, int id, int &i0, int &i1) {
        if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
        if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
    }
    void binary_search(int l, int r, Point p, int &i0, int &i1) {
        if (l == r) return;
        update_tangent(p, l % n, i0, i1);
        int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        update_tangent(p, r % n, i0, i1);
    }
    int binary_search(Point u, Point v, int l, int r) {
        int sl = sign((v - u).det(a[l % n] - u));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((v - u).det(a[mid % n] - u));
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 判定点是否在凸包内，在边界返回 true
```

```cpp
	bool contain(Point p) {
		if (p.x < lower[0].x || p.x > lower.back().x) return false;
		int id = lower_bound(lower.begin(), lower.end()
			, Point(p.x, -INF)) - lower.begin();
		if (lower[id].x == p.x) {
			if (lower[id].y > p.y) return false;
		} else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
		id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
			, greater<Point>()) - upper.begin();
		if (upper[id].x == p.x) {
			if (upper[id].y < p.y) return false;
		} else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
		return true;
	}
	// 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
	// 共线的多个切点返回任意一个，否则返回 false
	bool get_tangent(Point p, int &i0, int &i1) {
		if (contain(p)) return false;
		i0 = i1 = 0;
		int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
		binary_search(0, id, p, i0, i1);
		binary_search(id, (int)lower.size(), p, i0, i1);
		id = lower_bound(upper.begin(), upper.end(), p
			, greater<Point>()) - upper.begin();
		binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
		binary_search((int)lower.size() - 1 + id
			, (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
		return true;
	}
	// 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
	int get_tangent(Point vec) {
		pair<long long, int> ret = get_tangent(upper, vec);
		ret.second = (ret.second + (int)lower.size() - 1) % n;
		ret = max(ret, get_tangent(lower, vec));
		return ret.second;
	}
	// 求凸包和直线 u,v 的交点，如果无严格相交返回 false.
	//如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
	bool get_intersection(Point u, Point v, int &i0, int &i1) {
		int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
		if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
			if (p0 > p1) swap(p0, p1);
```

```
                        i0 = binary_search(u, v, p0, p1);
                        i1 = binary_search(u, v, p1, p0 + n);
                        return true;
                } else {
                        return false;
                }
        }
};
```

6.2 多边形

6.2.1 判断点在多边形内部

```
bool InPoly(P p,vector<P>poly){
        int cnt=0;
        for(int i=0;i<poly.size();i++){
                P a=poly[i],b=poly[(i+1)%poly.size()];
                if(OnLine(p,L(a,b)))
                        return false;
                int x=sgn(det(a,p,b));
                int y=sgn(a.y-p.y);
                int z=sgn(b.y-p.y);
                cnt+=(x>0&&y<=0&&z>0);
                cnt-=(x<0&&z<=0&&y>0);
        }
        return cnt;
}
```

7 其他

7.1 斯坦纳树

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

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

for (int s = 0; s < (1 << m); s++){
        for (int i = 1; i <= n; i++){
                for (int s0 = (s&(s-1)); s0 ; s0=(s&(s0-1))){
                                f[s][i] = min(f[s][i], f[s0][i] + f[s - s0][i]);
```

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

7.2 无敌的读入优化

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

```
                for (; getchar(ch), ch >= '0' && ch <= '9'; )
                        x = x * 10 + ch - '0';
                return true;
        }
}
Reader::getint(x);
Reader::getint(y);
```

7.3 最小树形图

```
const int maxn=1100;

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

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

int mdst( int root ) { // return the total length of MDST
        int i , j , k , sum = 0 ;
        memset ( used , 0 , sizeof ( used ) ) ;
        for ( more =1; more ; ) {
```

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

7.4 DLX

```
int n,m,K;
struct DLX{
        int L[maxn],R[maxn],U[maxn],D[maxn];
        int sz,col[maxn],row[maxn],s[maxn],H[maxn];
        bool vis[233];
        int ans[maxn],cnt;
        void init(int m){
                for(int i=0;i<=m;i++){
                        L[i]=i-1;R[i]=i+1;
                        U[i]=D[i]=i;s[i]=0;
                }
                memset(H,-1,sizeof H);
                L[0]=m;R[m]=0;sz=m+1;
        }
        void Link(int r,int c){
                U[sz]=c;D[sz]=D[c];U[D[c]]=sz;D[c]=sz;
                if(H[r]<0)H[r]=L[sz]=R[sz]=sz;
                else{
                        L[sz]=H[r];R[sz]=R[H[r]];
                        L[R[H[r]]]=sz;R[H[r]]=sz;
                }
                s[c]++;col[sz]=c;row[sz]=r;sz++;
        }
        void remove(int c){
                for(int i=D[c];i!=c;i=D[i])
                        L[R[i]]=L[i],R[L[i]]=R[i];
```

```
    }
    void resume(int c){
        for(int i=U[c];i!=c;i=U[i])
            L[R[i]]=R[L[i]]=i;
    }
    int A(){
        int res=0;
        memset(vis,0,sizeof vis);
        for(int i=R[0];i;i=R[i])if(!vis[i]){
            vis[i]=1;res++;
            for(int j=D[i];j!=i;j=D[j])
                for(int k=R[j];k!=j;k=R[k])
                    vis[col[k]]=1;
        }
        return res;
    }
    void dfs(int d,int &ans){
        if(R[0]==0){ans=min(ans,d);return;}
        if(d+A()>=ans)return;
        int tmp=23333,c;
        for(int i=R[0];i;i=R[i])
            if(tmp>s[i])tmp=s[i],c=i;
        for(int i=D[c];i!=c;i=D[i]){
            remove(i);
            for(int j=R[i];j!=i;j=R[j])remove(j);
            dfs(d+1,ans);
            for(int j=L[i];j!=i;j=L[j])resume(j);
            resume(i);
        }
    }
    void del(int c){//exactly cover
    L[R[c]]=L[c];R[L[c]]=R[c];
        for(int i=D[c];i!=c;i=D[i])
            for(int j=R[i];j!=i;j=R[j])
                U[D[j]]=U[j],D[U[j]]=D[j],--s[col[j]];
}
void add(int c){  //exactly cover
    R[L[c]]=L[R[c]]=c;
        for(int i=U[c];i!=c;i=U[i])
            for(int j=L[i];j!=i;j=L[j])
                ++s[col[U[D[j]]=D[U[j]]=j]];
}
```

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

7.5 插头 DP

```
int n,m,l;
struct L{
    int d[11];
    int& operator[](int x){return d[x];}
    int mc(int x){
        int an=1;
        if(d[x]==1){
            for(x++;x<l;x++)if(d[x]){
                an=an+(d[x]==1?1:-1);
                if(!an)return x;
            }
        }else{
```

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

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

```
                }else
                if(d1==2&&d2==1){
                    f[cur][S.s(j-1,0).s(j,0)]+=w;
                }else
                if((d1==1&&d2==1)||(d1==2&&d2==2)){
                    int m1=S.mc(j),m2=S.mc(j-1);
                    f[cur][S.s(j-1,0).s(j,0).s(m1,1).s(m2,2)]+=w;
                }
            }
        }
        cur^=1;f[cur].cl();
        for(int k=0;k<f[!cur].sz;k++){
            L S=f[!cur].s[k];Int w=f[!cur][S];
            f[cur][S>>1]=w;
        }
    }
    return 0;
}
```

7.6 某年某月某日是星期几

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

7.7 枚举大小为 k 的子集

使用条件：$k > 0$

```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;
    }
}
```

7.8 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];
            if (!has(i, j)) {
                upleft = INT_MIN;
            }
            int max = std::max(left, std::max(upleft, up));
            if (left == max) {
                from[i][j] = 0;
            } else if (upleft == max) {
                from[i][j] = 1;
            } else {
                from[i][j] = 2;
            }
            left = max;
        }
        if (i >= n) {
```

```
            int count = 0;
            for (int x = i, y = n; y; ) {
                int t = from[x][y];
                count += t == 1;
                x += DELTA[t][0];
                y += DELTA[t][1];
            }
            ret = std::max(ret, count);
            int x = i - n + 1;
            from[x][0] = 0;
            int y = 0;
            while (y <= n && from[x][y] == 0) {
                y++;
            }
            for (; x <= i; ++x) {
                from[x][y] = 0;
                if (x == i) {
                    break;
                }
                for (; y <= n; ++y) {
                    if (from[x + 1][y] == 2) {
                        break;
                    }
                    if (y + 1 <= n && from[x + 1][y + 1] == 1) {
                        y++;
                        break;
                    }
                }
            }
        }
    }
    return ret;
}
```

7.9 LLMOD

```
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;
}
```

8 vimrc

```
autocmd BufNewFile *.cpp exec ":call Setfilehead()"
func! Setfilehead()
    call append(0, '// Create: '.strftime("%Y-%m-%d %H:%M:%S"))
endfunc

colo morning
set fdm=syntax
set foldlevel=100

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
set autochdir
set makeprg=g++\ %:r.cpp\ -o\ %:r\ -g\ -std=c++11\ -Wall\ -Wextra\ -Wconversion

syntax on

nmap <C-A> ggVG
vmap <C-C> "+y
noremap <C-V> "+P

filetype plugin indent on

autocmd FileType cpp set cindent
autocmd FileType cpp map <F9> :make<CR>
autocmd FileType cpp map <C-F9> :!g++ %:r.cpp -o %:r -g -O2 -std=c++11 -Wall -Wextra<CR>
autocmd FileType cpp map <F8> :!time ./%:r < %:r.in <CR>
autocmd FileType cpp map <F5> :!time ./%:r <CR>
autocmd FileType cpp map <F10> :!gdb ./%:r <CR>

autocmd FileType python set smartindent autoindent
autocmd FileType python map <F5> :!python ./%<CR>

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

9 常用结论

9.1 上下界网络流

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

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

无源汇的上下界可行流

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

有源汇的上下界可行流

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

有源汇的上下界最大流

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

有源汇的上下界最小流

1. 在**有源汇的上下界可行流**中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在**无源汇的上下界可行流**即为原图的最小流。
2. 按照**无源汇的上下界可行流**的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意这一次不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^*，T^* 无影响，再直接求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

9.2 上下界费用流

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

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

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

9.3 弦图相关

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

9.4 Bernoulli 数

1. 初始化：$B_0(n) = 1$
2. 递推公式：
$$B_m(n) = n^m - \sum_{k=0}^{m-1} \binom{m}{k} \frac{B_k(n)}{m-k+1}$$
3. 应用：
$$\sum_{k=1}^{n} k^m = \frac{1}{m+1} \sum_{k=0}^{m} \binom{m+1}{k} n^{m+1-k}$$

10 常见错误

1. 数组或者变量类型开错，例如将 double 开成 int；
2. 函数忘记返回返回值；
3. 初始化数组没有初始化完全；
4. 对空间限制判断不足导致 MLE；

11 测试列表

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

12 Java

12.1 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;
        }
}
```

```java
class InputReader {
    public BufferedReader reader;
    public StringTokenizer tokenizer;

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

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

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

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

12.2 样历代码

```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));
```

```java

                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);
                //Scanner cin = new Scanner(System.in);
                int N = 70;

                long k[] = new long[N];
                int n;
```

```

			while(true){
				n=cin.nextInt();
				if (n == 0) break;

				for (int i = 1; i <= n; i ++){
					k[i]=cin.nextLong();
					//System.out.println(k[i]);

				}

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

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

					if(Len<=0)continue;

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

				/*
								Sum = 0;
				for (int j = 1; j <= n; j++)
					Sum += (1LL << (j - 1)) * k[j];



				LL x = Len;



				if ((2 * Sum) % Len > 0) continue;

				LL y = (2 * Sum) / Len;
				*/
					long x = Len;
```

```
//System.out.println("len="+Len);

if ((Sum.multiply(BigInteger.valueOf(2))) .mod (BigInteger.valueOf(L

long y = Sum.multiply(BigInteger.valueOf(2)).divide(BigInteger.value

if ((y - x + 1) % 2 > 0) continue;

A = (y - x + 1) / 2;

if ((x + y - 1) % 2 > 0) continue;

B = (x + y - 1) / 2;

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

int flag = 1;

long Cnt;
long Cnt_B;
long Cnt_A;

//printf("%lld %lld\n", A, B);

for (int j = 1; j <= n; j++){

        Cnt_B = Calc(B, j);
        Cnt_A = Calc(A - 1, j);



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

if (flag==1){
        Ans++;
```

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

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

        }
}
```

compact1, compact2, compact3

java.math

Class BigInteger

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

All Implemented Interfaces:

`Serializable, Comparable<BigInteger>`

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

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

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

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

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

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

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

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

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

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

Implementation Note:

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

Since:

JDK1.1

See Also:

BigDecimal, Serialized Form

Field Summary

Fields

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

Constructor Summary

Constructors

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

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

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

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

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

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

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

	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.

compact1, compact2, compact3

java.util

Class TreeMap<K,V>

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

Type Parameters:

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

`V` - the type of mapped values

All Implemented Interfaces:

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

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

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

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

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

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

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

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

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

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

This class is a member of the Java Collections Framework.

Since:

1.2

See Also:

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

Nested Class Summary

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

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

Constructor Summary

Constructors

Constructor and Description
`TreeMap()` Constructs a new, empty tree map, using the natural ordering of its keys.
`TreeMap(Comparator<? super K> comparator)` Constructs a new, empty tree map, ordered according to the given comparator.
`TreeMap(Map<? extends K,? extends V> m)` Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.
`TreeMap(SortedMap<K,? extends V> m)` Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

Method Summary

All Methods Instance Methods Concrete Methods

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

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

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

Methods inherited from class java.util.AbstractMap

`equals`, `hashCode`, `isEmpty`, `toString`

13 gedit

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

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

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

14 数学

14.1 常用数学公式

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

14.1.2 斐波那契数列

1. $fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$
2. $fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$
3. $fib_{-n} = (-1)^{n-1} fib_n$

4. $fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$

5. $gcd(fib_m, fib_n) = fib_{gcd(m,n)}$

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

14.1.3 错排公式

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

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

14.1.4 莫比乌斯函数

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

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

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

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

14.1.5 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：

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

14.1.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有

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

14.1.7 树的计数

1. 有根树计数：$n+1$ 个结点的有根树的个数为

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

其中，

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

2. 无根树计数：当 n 为奇数时，n 个结点的无根树的个数为

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

当 n 为偶数时，n 个结点的无根树的个数为

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

3. n 个结点的完全图的生成树个数为

$$n^{n-2}$$

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

14.1.8 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：

$$V-E+F=2$$

14.1.9 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：

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

14.1.10 牛顿恒等式

设

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

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

则

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

特别地，对于

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

有

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

14.2 平面几何公式

14.2.1 三角形

1. 半周长

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

2. 面积

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

3. 中线

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

4. 角平分线

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

5. 高线

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

6. 内切圆半径

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

7. 外接圆半径

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

14.2.2 四边形

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

1. $a^2+b^2+c^2+d^2 = D_1^2+D_2^2+4M^2$

2. $S = \frac{1}{2}D_1D_2sinA$

3. 对于圆内接四边形

$$ac + bd = D_1 D_2$$

4. 对于圆内接四边形

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

14.2.3 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

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

14.2.4 圆

1. 弧长

$$l = rA$$

2. 弦长

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

3. 弓形高

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

4. 扇形面积

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

5. 弓形面积

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

14.2.5 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

$$T = S + 2A$$

14.2.6 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

$$T = S + 2A$$

14.2.7 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

$$T = S + A_1 + A_2$$

14.2.8 圆柱

1. 侧面积

$$S = 2\pi rh$$

2. 全面积

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

3. 体积

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

14.2.9 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

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

14.2.10 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

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

14.2.11 球

1. 全面积

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

2. 体积

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

14.2.12 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

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

14.2.13 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

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

14.3 积分表

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

14.4 立体几何公式

14.4.1 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 $A + B + C - \pi$

14.4.2 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

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

15 附录

15.1 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

Theoretical Computer Science Cheat Sheet

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Theoretical Computer Science Cheat Sheet

Identities Cont.

38. $\left[{n+1 \atop m+1}\right] = \sum_k \left[{n \atop k}\right]\binom{k}{m} = \sum_{k=0}^{n} \left[{k \atop m}\right] n^{\underline{n-k}} = n!\sum_{k=0}^{n} \frac{1}{k!}\left[{k \atop m}\right]$,

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

40. $\left\{{n \atop m}\right\} = \sum_k \binom{n}{k}\left\{{k+1 \atop m+1}\right\}(-1)^{n-k}$,

41. $\left[{n \atop m}\right] = \sum_k \left[{n+1 \atop k+1}\right]\binom{k}{m}(-1)^{m-k}$,

42. $\left\{{m+n+1 \atop m}\right\} = \sum_{k=0}^{m} k\left\{{n+k \atop k}\right\}$,

43. $\left[{m+n+1 \atop m}\right] = \sum_{k=0}^{m} k(n+k)\left[{n+k \atop k}\right]$,

44. $\binom{n}{m} = \sum_k \left\{{n+1 \atop k+1}\right\}\left[{k \atop m}\right](-1)^{m-k}$,

45. $(n-m)!\binom{n}{m} = \sum_k \left[{n+1 \atop k+1}\right]\left\{{k \atop m}\right\}(-1)^{m-k}$, for $n \ge m$,

46. $\left\{{n \atop n-m}\right\} = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left[{m+k \atop k}\right]$,

47. $\left[{n \atop n-m}\right] = \sum_k \binom{m-n}{m+k}\binom{m+n}{n+k}\left\{{m+k \atop k}\right\}$,

48. $\left\{{n \atop \ell+m}\right\}\binom{\ell+m}{\ell} = \sum_k \left\{{k \atop \ell}\right\}\left\{{n-k \atop m}\right\}\binom{n}{k}$,

49. $\left[{n \atop \ell+m}\right]\binom{\ell+m}{\ell} = \sum_k \left[{k \atop \ell}\right]\left[{n-k \atop m}\right]\binom{n}{k}$.

Trees

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

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

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

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

Recurrences

Master method:

$$T(n) = aT(n/b) + f(n), \quad a \ge 1, b > 1$$

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 $af(n/b) \le cf(n)$ for large n, then

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

Substitution (example): Consider the following recurrence

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

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

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

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

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

Substituting we find

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

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

Summing factors (example): Consider the following recurrence

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

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

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

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

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

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

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

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

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

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

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

Note that

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

Subtracting we find

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

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

Generating functions:

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

Example:

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

Multiply and sum:

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

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

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

Simplify:

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

Solve for $G(x)$:

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

Expand this using partial fractions:

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

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

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

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

Pascal's Triangle

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

General

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

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

Change of base, quadratic formula:

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

Euler's number e:

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

Harmonic numbers:

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

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

Factorial, Stirling's approximation:

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

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

Ackermann's function and inverse:

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

Binomial distribution:

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

Poisson distribution:

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

Normal (Gaussian) distribution:

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

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

$$nH_n.$$

Probability

Continuous distributions: If

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

then p is the probability density function of X. If

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

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

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

Expectation: If X is discrete

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

If X continuous then

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

Variance, standard deviation:

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

For events A and B:

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

iff A and B are independent.

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

For random variables X and Y:

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

if X and Y are independent.

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

Bayes' theorem:

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

Inclusion-exclusion:

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

Moment inequalities:

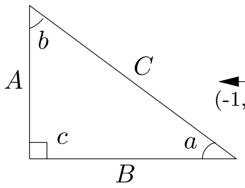
$$\Pr\left[|X| \geq \lambda\,\mathrm{E}[X]\right] \leq \frac{1}{\lambda},$$
$$\Pr\left[\left|X - \mathrm{E}[X]\right| \geq \lambda \cdot \sigma\right] \leq \frac{1}{\lambda^2}.$$

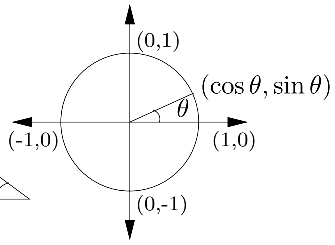
Geometric distribution:

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

Theoretical Computer Science Cheat Sheet

Trigonometry





Pythagorean theorem:
$$C^2 = A^2 + B^2.$$

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

Area, radius of inscribed circle:
$$\tfrac{1}{2}AB, \qquad \frac{AB}{A+B+C}.$$

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

sseiden@acm.org
http://www.csc.lsu.edu/~seiden

Matrices

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

Determinants: $\det A \neq 0$ iff A is non-singular.
$$\det A \cdot B = \det A \cdot \det B,$$
$$\det A = \sum_{\pi} \prod_{i=1}^{n} \operatorname{sign}(\pi) a_{i,\pi(i)}.$$

2×2 and 3×3 determinant:
$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc,$$
$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = g \begin{vmatrix} b & c \\ e & f \end{vmatrix} - h \begin{vmatrix} a & c \\ d & f \end{vmatrix} + i \begin{vmatrix} a & b \\ d & e \end{vmatrix}$$
$$= \begin{array}{c} aei + bfg + cdh \\ -ceg - fha - ibd. \end{array}$$

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

Hyperbolic Functions

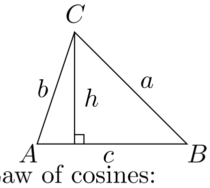
Definitions:
$$\sinh x = \frac{e^x - e^{-x}}{2}, \qquad \cosh x = \frac{e^x + e^{-x}}{2},$$
$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \operatorname{csch} x = \frac{1}{\sinh x},$$
$$\operatorname{sech} x = \frac{1}{\cosh x}, \qquad \coth x = \frac{1}{\tanh x}.$$

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

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

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

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

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

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

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

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

Fermat's theorem:

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

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

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

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

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

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

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

Möbius inversion:

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

If

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

then

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

Prime numbers:

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

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

Graph Theory

Definitions:

Loop	An edge connecting a vertex to itself.
Directed	Each edge has a direction.
Simple	Graph with no loops or multi-edges.
Walk	A sequence $v_0 e_1 v_1 \ldots e_\ell v_\ell$.
Trail	A walk with distinct edges.
Path	A trail with distinct vertices.
Connected	A graph where there exists a path between any two vertices.
Component	A maximal connected subgraph.
Tree	A connected acyclic graph.
Free tree	A tree with no root.
DAG	Directed acyclic graph.
Eulerian	Graph with a trail visiting each edge exactly once.
Hamiltonian	Graph with a cycle visiting each vertex exactly once.
Cut	A set of edges whose removal increases the number of components.
Cut-set	A minimal cut.
Cut edge	A size 1 cut.
k-Connected	A graph connected with the removal of any $k - 1$ vertices.
k-Tough	$\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G - S) \leq \|S\|$.
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
$\mathrm{r}(k, \ell)$	Ramsey number

Geometry

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

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

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

Distance formula, L_p and L_∞ metric:

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

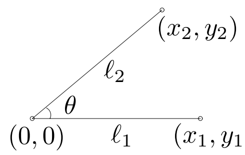
$$\big[|x_1 - x_0|^p + |y_1 - y_0|^p\big]^{1/p},$$

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

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

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

Angle formed by three points:



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

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

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

Area of circle, volume of sphere:

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

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

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:

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

Brouncker's continued fraction expansion:

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

Gregrory's series:

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

Newton's series:

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

Sharp's series:

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

Euler's series:

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

$$\tfrac{\pi^2}{8} = \tfrac{1}{1^2} + \tfrac{1}{3^2} + \tfrac{1}{5^2} + \tfrac{1}{7^2} + \tfrac{1}{9^2} + \cdots$$

$$\tfrac{\pi^2}{12} = \tfrac{1}{1^2} - \tfrac{1}{2^2} + \tfrac{1}{3^2} - \tfrac{1}{4^2} + \tfrac{1}{5^2} - \cdots$$

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

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

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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:

$x^{\underline{n}} = x(x-1)\cdots(x-n+1), \quad n>0,$

$x^{\underline{0}} = 1,$

$x^{\underline{n}} = \dfrac{1}{(x+1)\cdots(x+|n|)}, \quad n<0,$

$x^{\underline{n+m}} = x^{\underline{m}}(x-m)^{\underline{n}}.$

Rising Factorial Powers:

$x^{\overline{n}} = x(x+1)\cdots(x+n-1), \quad n>0,$

$x^{\overline{0}} = 1,$

$x^{\overline{n}} = \dfrac{1}{(x-1)\cdots(x-|n|)}, \quad n<0,$

$x^{\overline{n+m}} = x^{\overline{m}}(x+m)^{\overline{n}}.$

Conversion:

$x^{\underline{n}} = (-1)^n(-x)^{\overline{n}} = (x-n+1)^{\overline{n}}$
$\quad = 1/(x+1)^{\overline{-n}},$

$x^{\overline{n}} = (-1)^n(-x)^{\underline{n}} = (x+n-1)^{\underline{n}}$
$\quad = 1/(x-1)^{\underline{-n}},$

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

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

$$x^{\overline{n}} = \sum_{k=1}^n \left[n \atop k \right]x^k.$$

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

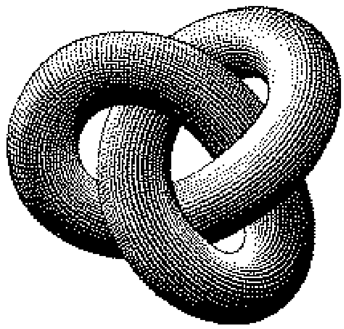
$$\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = \sum_{i=0}^{\infty} \frac{n(2i+n-1)!}{i!(n+i)!} x^i,$$

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

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

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

Escher's Knot



Stieltjes Integration

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

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

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

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

If the integrals involved exist

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

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

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

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

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

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

Cramer's Rule

If we have equations:

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

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

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

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

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:

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

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 $k_m \ge 2$.

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