STRING

KMP:

```
int k,l,c;
char txt[1000001],pat[10000001];
int fnd[1000001],fn[1000001];
void comp(){
    fn[1]=0;
    int m=0;
    for(int x=2;x<=l;x++){
 while(m>0 && pat[m+1]!=pat[x])m=fn[m];
        if(pat[m+1]==pat[x])m++;
        fn[x]=m;
    }
}
void job(){
    int b,x;
    b=0;
    for(x=1;x<=k;x++){
 while(b>0 && pat[b+1]!=txt[x])b=fn[b];
        if(pat[b+1]==txt[x])b++;
        if(b==l){
            c++;
            b=fn[b];
        }
    }
}
```

SUFFIX AUTOMATA:

```
int TF[1001][27];
char pat[1001];
void computeFA(int M){
   int lps = 0;
for(int i = 0; i < 26; i++) TF[0][i] = 0;
   TF[0][pat[0] - 'a'] = 1;
   for(int i = 1; i < M; i++){
        for(int j = 0; j < 26; j++) TF[i][j] = TF[lps]
[j];
        TF[i][pat[i] - 'a'] = i + 1;
  if(i < M) lps = TF[lps][pat[i] - 'a'];
    }
}
```

Z-ALGO:

```
char s[1000003];
int zz[1000003];
void Z_algo(int n) {
    int L = 0, R = 0;
    for (int i = 1; i < n; i++) {
        if (i > R) {
            L = R = i;
   while (R < n && s[R-L] == s[R]) R++;
            zz[i] = R-L;
            R--;
        } else {
            int k = i-L;
      if (zz[k] < R-i+1) zz[i] = zz[k];
            else {
                L = i;
   while (R < n && s[R-L] == s[R]) R++;
                zz[i] = R-L;
                R--;
            }
        }
    }
}
```

MANACHER:

```
string tmp, s;
void pre_process(){
    s = "%";
    REP(i,tmp.size())s += '#' + tmp[i];
    s += "#&";
}
LL P[300001];
void manacher(){
    LL c = 0, r = 0;
    FOR(i, 1, s.size()-1){
        LL i_mirror = 2 * c - i;
       P[i]=(r>i)?min(r-i,P[i_mirror]):0;
  while(s[i+1+P[i]]==s[i-1- P[i]])P[i]++;
        if (i + P[i] > r) {
            c = i;
            r = i + P[i];
        }
    }
}
```

SUFFIX ARRAY:

```
const int N = 3e5+10; // 3* MX size
int A[N] , rank[N] , sa[N] , height[N];
char str[N];
void ini(){
    memset(A,0,sizeof A);
    memset(rank,0,sizeof rank);
    memset(sa,0,sizeof sa);
    memset(height,0,sizeof height);
}
namespace Suffix_array{
  # define F(x) ((x)/3+((x)%3==1?0:tb))
# define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
 int wa[N*3],wb[N*3],wv[N*3],ws[N * 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++) ws[i] = 0;
    for(i=0;i<n;i++) ws[wv[i]]++;
    for(i=1;i<m;i++) ws[i] += ws[i-1];
  for(i=n-1;i>=0;i--)b[--ws[wv[i]]]=a[i];
        return;
    }
 void dc3(int *r, int *sa, int n, int m){
    int i, j, *rn = r + n;
       int *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++];
  }
    void da(int str[],int sa[],int rank[],int height[],int
n,int m){
     for (int i = n; i < n * 3; i++)
            str[i] = 0;
        dc3 (str , sa , n + 1 , m);
        int i, j, k;
        for (i = 0; i < n; i++){
            sa[i] = sa[i + 1];
            rank[sa[i]] = i;
```

```
        }
for (i = 0, j = 0, k = 0; i < n; height[rank[i ++]] = k)
       if (rank[i] > 0)
    for (k ? k--: 0 , j = sa[rank[i]-1];
    i + k < n && j + k < n && str[i + k]
     == str[j + k]; k++);
  }
} using namespace Suffix_array;
```

AHO CORASIK:

```
struct trie{
    int next[26],fail,score;
    void clear(){ REP(i,0,26) next[i]=0;
   fail=score=0; }
  };
  trie node[MX];
 int A[110][110],len[110],sc[110],root=1;
void init(){
    n=root+1;
    REP(i,2) node[i].clear();
    REP(i,26) node[0].next[i]=root;
    return;
}
void insert(int x){
    int pos=root;
    for(int i=0;i<len[x];i++){
        if(!node[pos].next[A[x][i]]){
            node[n].clear();
           node[pos].next[A[x][i]]=n++;
        }
        pos=node[pos].next[A[x][i]];
    }
    node[pos].score+=sc[x];
    return;
 }
void get_failure(){
    queue<int>q;
    q.push(root);
    while(!q.empty()){
        int u,v;u=q.front();q.pop();
        REP(i,26){
            if(node[u].next[i]){
node[node[u].next[i]].fail=node[node[u].fail].next[i];
node[node[u].next[i]].score+=node[node[node[u].next[i]].
fail].score;
        q.push(node[u].next[i]);
      } else {
node[u].next[i]=node[node[u].fail].next[i];
            }
        }
    }
    return;
}
```

MATRIX

Gaussian Elimination:

```
int gauss() {
    int x,y;
    for(x=0,y=0;x<row && y<col;y++) {
        int p=x;
        rep(i,x,row-1) if(a[i][y]) {p=i;break;} ///for
GF(2)
        rep(i,x+1,row-1) if(abs(a[i][y])>abs(a[p][y]))
p=i; ///for mod k
        rep(i,x+1,row-1) if(fabs(a[i][y])>fabs(a[p][y])
+eps) p=i;   ///for real values
     if(!a[p][y]) continue;  ///for GF(2) and mod k
 if(fabs(a[p][y])<eps) continue; ///for real values
      if(p!=x) rep(j,y,col) swap(a[p][j],a[x][j]);
        loc[y]=x; ///only for real value
        rep(i,0,row-1){
            if(i==x || !a[i][y]) continue; ///for GF(2)
and mod k
            if(i==x || fabs(a[i][y])<eps) continue;
///for real values
      rep(j,y,col) a[i][j]^=a[x][j];  ///for GF(2)
            ll b=a[i][y],c=a[x][y]; ///for mod k
            rep(j,y,col) a[i][j] = ((c*a[i][j] - b*a[x][j])
%k+k)%k ; ///for mod k
     double tmp=a[i][y]/a[x][y]; ///for real values
            rep(j,y,col) a[i][j] -= a[x][j]*tmp; ///for
real values
        }
        x++;
    }
    /// for GF(2) and mod k
    int ans=1;
    rep(i,1,row-x) ans=(2LL*ans)%MOD;
    ans--;
    return (ans%MOD+MOD)%MOD;
    ///for real values only
    rep(j,0,col-1) {
        if(loc[j]==-1) continue;
        a[loc[j]][col]/=a[loc[j]][j];
        a[loc[j]][j]/=a[loc[j]][j];
    }
}
void init() {
    num=row=col=100;
    CLR(a);  SET(s);  SET(loc);
}
printf("%.10lf\n",a[loc[pos]][col]);
```

MATRIX EXPO:

```
#define MX 2
LL MODSQ = mod*mod,mul_res[MX][MX];
void multiply(LL a[][MX], LL b[][MX]){
    set0(mul_res);
    for(int i = 0; i < MX; i++){
        for(int j = 0; j < MX; j++){
            for(int k = 0; k < MX; k++){
      mul_res[i][j] = (mul_res[i][j] + a[i][k] * b[k][j]);
      if(mul_res[i][j] >= MODSQ) mul_res[i][j] -= MODSQ;
            }
            mul_res[i][j] %= mod;
        }
    }   }
LL big_res[MX][MX],init[MX][MX];
void matrix_expo(LL n){
    if(n == 0){
        set0(big_res);
        REP(i, MX)big_res[i][i] = 1;
        return;
    }
    if(n % 2 == 0){
        matrix_expo(n / 2);
        multiply(big_res, big_res);
        REP(i, MX){
           REP(j, MX)big_res[i][j] = mul_res[i][j];
        }
    }
    else {
        matrix_expo(n - 1);
        multiply(big_res, init);
        REP(i, MX){
           REP(j, MX)big_res[i][j] = mul_res[i][j];
        }
    }
}
```

MATRIX DETERMINANT:

```
ld data[64][64];
ld det(int n) {
    ld ans = 1;
    for (int j = 0; j < n; j++) {
        for (int i = j; i < n; i++) if (data[i][j]) { //
find non-zero
  for (int k = j; k < n; k++) swap(data[i][k], data[j][k]);
      if (i != j) ans *= -1;
```

```
        break ;
            }
  if (data[j][j]) for (int i = j + 1; i < n; i++) {
      ld mult = data[i][j]/data[j][j];
                data[i][j] = 0;
     for (int k = j + 1; k < n; k++) data[i][k] -=
mult*data[j][k];
            }
    }
    for (int i = 0; i < n; i++) ans *= data[i][i];
    return ans;
}
```

FLOW

DINIC:

```
struct dinic_maxflow {
    struct edge {
        int a, b, cap, flow ;
     edge(int _a,int _b,int _c,int _d) {
            a=_a,b=_b,cap=_c,flow=_d;
        }
    } ;
    int INF = 1500000001 ;
int n, s, t, d [ 30001 ],ptr[30001],q[30001* 10 ] ;
    vector < edge > e ;
    vector < int > g [ 30001 ] ;
void add_edge ( int a, int b, int cap ) {
        edge e1 =edge( a, b, cap, 0 ) ;
        edge e2 =edge( b, a, 0 , 0 ) ;
        g[a].push_back((int)e.size());
        e. push_back ( e1 ) ;
        g[b].push_back((int)e.size());
        e. push_back ( e2 ) ;
    }
    bool bfs ( ) {
        int qh = 0 , qt = 0 ;
        q [ qt ++ ] = s ;
        memset ( d, -1 , sizeof d ) ;
        d [ s ] = 0 ;
        while(qh<qt&&d[t] == - 1 ) {
            int v = q [ qh ++ ] ;
 for (size_t i=0;i<g[v].size();++i){
       int id=g[v][i], to=e[id].b ;
  if(d[to]==-1 && e[id ].flow  < e [ id ] . cap ) {
            q [ qt ++ ] = to ;
            d [ to ] = d [ v ] + 1 ;
                }
            }
        }
        return d [ t ] != - 1 ;
 }
int dfs ( int v, int flow ) {
     if ( ! flow )  return 0 ;
     if ( v == t )  return flow ;
for(;ptr[v]<(int)g[v].size();++ptr [ v ] ) {
      int id = g [v][ptr[v]], to = e [ id ] . b ;
      if(d[to]!=d[v] + 1 )  continue ;
      int pushed = dfs ( to, min ( flow, e [ id ] . cap
- e [ id ] . flow ) ) ;
      if ( pushed ) {
           e [ id ] . flow += pushed ;
          e [ id ^ 1 ] . flow -= pushed ;
                return pushed ;
            }
        }
        return 0 ;
    }
    int dinic ( ) {
        int flow = 0 ;
        for ( ;; ) {
            if ( !bfs () )  break ;
        memset ( ptr, 0 , sizeof ptr ) ;
  while ( int pushed = dfs ( s, INF ) ) {
                flow += pushed ;
                if(pushed == 0)break;
            }
        }
        return flow ;
    }
 dinic_maxflow( int _n, int _s, int _t ){
        n = _n; s = _s; t = _t;
    }
};
```

MIN COST FLOW:

```
const int MAXN = 7210;
const int MAXM = 100010;
const int INF = 1000000000;
struct Edge{
    int u, v, cap, cost;
    int next;
    Edge(){}
    Edge(int u,int v,int cap,int cost,int
next):u(u),v(v),cap(cap),cost(cost),next(next){}
}edge[MAXM*3];
int NE;
int head[MAXN], dist[MAXN], pp[MAXN];
bool vis[MAXN];
void init(int n){
    NE = 0;
    fill(head,head+n+1,-1);
}
void addedge(int u, int v, int cap, int cost){
    edge[NE] = Edge(u,v,cap,cost,head[u]);
    head[u] = NE++;
    edge[NE] = Edge(v,u,0,-cost,head[v]);
    head[v] = NE++;
}
int qu[300010];
inline bool Min(int &x,int y){
    if(y < x) {x = y;return true;}
    return false;
}
bool SPFA(int s, int t, int n){
    int i, u, v , tmp;
    int he = 0 , ta = 0;
    fill(vis,vis+n+1,false);
    fill(pp,pp+n+1,-1);
    fill(dist,dist+n+1,INF);
    vis[s] = true; dist[s] = 0;
    qu[++ta] = s;
    while(he!=ta){
        u = qu[++he];  vis[u] = false;
        if(he >= 250000) he = 0;
        for(i = head[u];~ i; i = edge[i].next){
            v = edge[i].v;
           tmp = dist[u] + edge[i].cost;
      if(edge[i].cap &&  dist[v] > tmp){
                dist[v] = tmp;
                pp[v] = i;
                if(!vis[v]){
                    qu[++ta] = v;
                if(ta >= 250000) ta = 0;
                    vis[v] = true;
                }
            }
        }
    }
    if(dist[t] == INF) return false;
    return true;
}
pair<int,int> MCMF(int s, int t, int n){
    int flow = 0;
    int i, minflow, mincost;
    mincost = 0;
   for(;SPFA(s, t, n);){
        minflow = INF + 1;
```

```
   for(i = pp[t]; ~i ; i = pp[edge[i].u])
            Min( minflow , edge[i].cap);
        flow += minflow;
        for(i = pp[t]; ~i; i = pp[edge[i].u]){
            edge[i].cap -= minflow;
            edge[i^1].cap += minflow;
        }
        mincost += dist[t] * minflow;
    }
    return make_pair(flow,mincost); }
```

WEIGHTED MATCHING:

```
const int N = 1024*5; // Yury's
const int inf = 1<<30;
int w[N][N], ml[N], mr[N], lbl[N], lbr[N], s[N],
slack[N], par[N];
int wmatch(int n, int m) { //assume n<=m;
    int i, j, k, done, al, next, out = 0;
    for(i=0; i<n; i++) for(j=0,lbl[i]=0,ml[i]=-1; j<m;
j++) lbl[i] >?= w[i][j];
    for(i = 0; i < m; i++) lbr[i] = 0, mr[i] = -1;
   for(i=0; i<n; i++) for(j = 0; j < m; j++)if(w[i][j]
== lbl[i] && mr[j] == -1){
      ml[i] = j, mr[j] = i; break ;
       }
   for(i = 0; i < n; i++)if(ml[i] == -1){
            for(j=0; j<m; j++) s[j]=(j==i),
slack[j]=(lbl[i]+lbr[j]-w[i][j]), par[j] = -1;
            for(done = 0; !done;) {
                for(j = 0, al = inf; j < m; j++)
if(par[j] == -1) al <?= slack[j];
               for(j = 0, lbl[j] -= s[j]*al; j < m; ++j,
lbl[j] -= s[j]*al)
          if(par[j] != -1) lbr[j] += al;
          else slack[j] -= al;
                for(j = 0; j < m; j++) if(!slack[j] &&
par[j] == -1) {
                      for(k=0; k<n; k++) if(s[k]&&(lbl[k]
+lbr[j]==w[k][j])) {
                          par[j] = k;
                          break ;
                          }
                          if(mr[j] == -1) {
                              done = 1;
                          do {
        next=ml[par[j]], mr[j]=par[j],
        ml[par[j]]=j, j=next;
                          } while(j!=-1);
                      }
          else for(k = 0, s[mr[j]] = 1; k < m; k++)
  slack[k] <?= (lbl[mr[j]] + lbr[k] - w[mr[j]][k]);
                          break ;
                      }
            }
        }
for(i=0; i < n; i++) out += w[i][ml[i]];
    return out;
}
// Nhan's, O(m^2*n), m<=n
#define _clr(x) memset(x,-1,sizeof(int)*n)
int hungary(int m,int n,int a[][maxn],int* x,int* y) {
    int s[maxn+1],t[maxn],fx[maxn], fy[maxn],
p,q,ret=0,i,j,k;
   for(i=0; i<m; i++) for(fx[i]=-inf,j=0; j<n; j++)
    fx[i]=a[i][j]>fx[i]?a[i][j]:fx[i];
    for(i=0; i<n; fy[i++]=0);
    for(_clr(x),_clr(y),i=0; i<m; i++) {
        for(_clr(t),s[p=q=0]=i; p<=q&&x[i]<0; p++)
       for(k=s[p],j=0; j<n&&x[i]<0; j++)
                if(fx[k]+fy[j]==a[k][j]&&t[j]<0) {
                    s[++q]=y[j],t[j]=k;
                    if(s[q]<0) for(p=j; p>=0; j=p)
y[j]=k=t[j],p=x[k],x[k]=j;
                }
        if(x[i]<0) {
            for(i--,p=inf,k=0; k<=q; k++)
                for(j=0; j<n; j++) if(t[j]<0&&fx[s[k]]
+fy[j]-a[s[k]][j]<p)
            p=fx[s[k]]+fy[j]-a[s[k]][j];
   for(j=0; j<n; fy[j]+=t[j]<0?0:p,j++);
          for(k=0; k<=q; fx[s[k++]]-=p);
        }
    }
    for(i=0; i<m; i++) ret+=a[i][x[i]];
    return ret;
}
```

GRAPH MATCHING:

```
#define _ufind_run(x) for(;p[t=x];x=p[x],p[t]=(p[x]?
p[x]:x))
#define _run_both _ufind_run(i);_ufind_run(j)
struct ufind {
    int p[maxn+1],t;
    void init() { memset(p,0,sizeof(p)); }
    void set_friend(int i,int j) {
        _run_both;
        p[i]=(i==j?0:j);
    }
    int is_friend(int i,int j) {
        _run_both;
        return i==j&&i;
    }
    int find(int i) {
        _ufind_run(i);
        return i;
    }
} uf;
int base(int i) {
    return uf.find(i+1)-1;
}
void bunion(int i,int j) { uf.set_friend(i+1,j+1); }
void clear(int i) { uf.p[i+1]=0; }
int pre[maxn],visit[maxn],path1[maxn],path2[maxn],strue;
int source[maxn],target[maxn],match[maxn];
void shrink(int b,int v,int w,queue<int>& Q) {
    int x = base(v);
    while (x!=b) {
    bunion(x,b), x=match[x], bunion(x,b);
        Q.push(x);
        source[x]=v, target[x]=w;
        x = base(pre[x]);
    }
  }
void path(list<int>& P,int x,int y) {
    if(x==y) {
        P.push_back(x);
        return;
    }
    if(visit[x]==0) {
        P.push_back(x),
P.push_back(match[x]),path(P,pre[match[x]],y);
        return ;
    }
    P.push_back(x);
    list<int> P2;
    path(P2,source[x],match[x]),
reverse(P2.begin(),P2.end());
    P.splice(P.end(),P2), path(P,target[x],y);
}
int graph_match(int n,int mat[][maxn],int* match) {
    uf.init(), strue=0;
    REP(i,n)
    match[i]=-1,pre[i]=-1,visit[i]=0,
path1[i]=path2[i]=0,source[i]=target[i]=-1;
    REP(i,n) {
        if(match[i]>=0) continue ;
        queue<int> Q;
        vector<int> T;
```

```
        Q.push(i),T.push_back(i);
        bool flag = false ;
        while (!flag && !Q.empty()) {
            int v = Q.front();
            Q.pop();
            REP(w,n) if(mat[v][w]) {
                if(base(v)==base(w) ||
visit[base(w)]==1) continue ;
                if(visit[w]==-1) {
                    visit[w]=1, T.push_back(w),
pre[w]=v;
                    visit[match[w]]=0,
T.push_back(match[w]);
                    Q.push(match[w]);
                } else {
            int hv=base(v), hw=base(w);
           path1[hv]=path2[hw]=++strue;
                    while ((path1[hw]!=strue &&
path2[hv]!=strue)&&                 (match[hv]>=0||
match[hw]>=0)) {
                        if(match[hv]>=0)
hv=base(pre[match[hv]]), path1[hv]=strue;
                        if(match[hw]>=0)
hw=base(pre[match[hw]]), path2[hw]=strue;
                    }
                    if(path1[hw]==strue ||
path2[hv]==strue) {
       int b = (path1[hw]==strue)?hw:hv;
        shrink(b,v,w,Q), shrink(b,w,v,Q);
                    } else {
                        list<int> P;
                        path(P,v,hv);
                        P.push_front(w);
                  while (!P.empty()) {
                   int a=P.front();
                   P.pop_front();
                   int b=P.front();
                   P.pop_front();
                 match[a]=b, match[b]=a;
                   }
                   T.push_back(w);
              REP(j,T.size()) visit[T[j]]=1,clear(T[j]);
               flag = true ; break ;
                   }
                }
            }
        }
    }
    int ret=0;
    REP(i,n) ret+=match[i]>=0;
    return ret/2;
}
```

CIRCULATION: for each edge (a,b, lo,hi), add one edge of capacity lo from source to b, one edge of capacity hi from a to sink, one edge from a to b of capacity hi-lo. Finally add one edge from n to 1 of capacity (sum of lo). Flow should be equal to sum of lo. If it doesn't then there is no solution.

MAXIMUM INDEPENDENT SET: create a graph with weight of the left group vertices as the capacity from source to left group vertices, do the same for right and sink. add edge between left and right with cap inf. sum of all vertices - flow is ans.

GRAPH

ARTICULATION POINT:

```
vector<int>vc[100001];
int vis[100001];
int cnt = 0,in[100001],val[100001];
set< int > apNodes;
void dfs(int v,int p){
    vis[v] = 1;
    in[v] = val[v] = cnt++;
    int c=0;
    REP(i, vc[v].size()){
        if(vc[v][i] == p)continue;
        if(vis[vc[v][i]])val[v] = min(val[v], in[vc[v]
[i]]);
        else{
            dfs(vc[v][i],v);
      val[v]=min(val[v], val[vc[v][i]]);
     if (val[vc[v][i]] >= in[v] && p!=-1)
                apNodes.insert(v);
            c++;
        }
    }
    if(p==-1 && c>1)
        apNodes.insert(v);
}
```

BRIDGE:

```
vector<int>vc[100001];
int vis[100001];
int cnt,in[100001],val[100001];
void dfs(int v,int p){
    vis[v] = 1;
    in[v] = val[v] = cnt++;
    REP(i, vc[v].size()){
        if(vc[v][i] == p)continue;
        if(vis[vc[v][i]])val[v] = min(val[v], in[vc[v]
[i]]);
        else{
            dfs(vc[v][i], v);
     val[v] = min(val[v], val[vc[v][i]]);
            if (val[vc[v][i]] > in[v]){
                isBridge;;;;
            }
        }
    }
}
```

SCC: first run a dfs from each non-visited node. After processing the node put it in the stack. Then take elements one by one from stack and run a dfs again on the inverted graph. The visited components in the 2nd dfs are in scc.

TWO-SAT: for each (a || b) relation add an edge from -a to b and another from -b to a. then find scc. Check if any x and -x belong in the same scc, then no valid solution. Other run topo sort to greedily assign values.

EULER TOUR:

doit(u):

for each edge e = (u, v) in E, do: erase e, doit(v).

prepend u to the list of vertices in the tour.

HAVEL HAKIMI: given degree find if ok.

```
f = 0;
REP(j, b) {
    sort(tmp + j, tmp + b);
    reverse(tmp + j, tmp + b);
    if(tmp[j] < 0) {
        f = 1;
        break;
    }
    for(int k = 1; k <= tmp[j]; k++) {
        if(j + k >= b) {
            f = 1;
            break;
        }
        tmp[j + k]--;
    }
}
// havel-hakimi ends here. f = 1 shows invalid graph
```

Euler's theorem: For any planar graph, V − E + F = 1 + C, where V is the number of graph's vertices, E is the number of edges, F is the number of faces in graph's planar drawing, and C is the number of connected components. Corollary: V − E + F = 2 for a 3D polyhedron.

Vertex covers and independent sets: Let M , C, I be a max matching, a min vertex cover, and

a max independent set. Then |M | <= |C| = N − |I|, with equality for bipartite graphs. Complement of an MVC is always a MIS, and vice versa. Given a bipartite graph with partitions (A, B), build a network: connect source to A, and B to sink with edges of capacities, equal to the corresponding nodes' weights, or 1 in the unweighted case. Set capacities of the original graph's edges to the infinity. Let (S, T) be a minimum s-t cut. Then a maximum(-weighted) independent set is I = (A ∩ S)∪(B∩T), and a minimum(-weighted) vertex cover is C = (A ∩ T) ∪ (B ∩ S).

Matrix-tree theorem: Let matrix T = [t_{ij}], where t_{ij} is the number of multiedges between i and j, for i = j, and t_{ii} = −deg i . Number of spanning trees of a graph is equal to the determinant of a matrix obtained by deleting any k-th row and k-th column from T .

Prufer code of a tree: Label vertices with integers 1 to n. Repeatedly remove the leaf with the smallest label, and output its only neighbor's label, until only one edge remains. The sequence has length n − 2. Two isomorphic trees have the same sequence, and every sequence of integers from 1 and n corresponds to a tree. Corollary: the number of labeled trees with n vertices is n n−2 .

STABLE MARRIAGE:

```
scanf("%d", &n);
for( int i = 0; i < n; i++ ) {
    for( int j = 0; j < n; j++ ) {
        scanf("%d", &mPref[i][j]);
        mPref[i][j] -= (n+1);
    }
}
for( int i = 0; i < n; i++ ) {
    for( int j = 0; j < n; j++ ) {
        int k;
        scanf("%d", &k);
        wPref[i][k-1] = n - j;
    }
}
memset( R, -1, sizeof(R) );
memset( P, 0, sizeof(P) );
for( int i = 0; i < n; i++ ) {
    int man = i;
    while( man >= 0 ) {
     int woman = mPref[man][ P[man]++ ];
        if( R[woman] == -1 || wPref[woman][ R[woman] ]<
wPref[woman][ man ] ) {
            L[man] = woman;
            swap( man, R[woman] );
        }
    }
}
printf("Case %d:", ++caseno);
for( int i = 0; i < n; i++ ) printf(" (%d %d)", i+1,
L[i]+n+1);
```

DATA STRUCTURE

LCA:

```
int pre[100001],lvl[100001];
int P[100001][22];
void pre_process(int N){
    memset(P,-1,sizeof P);
    for(int i = 1; i <= N; i++)P[i][0] = pre[i];
    for(int j = 1; (1<<j) <= N; j++){
        for(int i = 1; i <= N; i++){
if(P[i][j - 1] != -1) P[i][j] = P[ P[i][j - 1] ][j - 1];
        }
    }
}
int LCA(int p, int q){
    if(lvl[p] < lvl[q])swap(p,q);
    int log;
 for(log = 1; (1<<log) <= lvl[p]; log++);
    log--;
    for(int i = log; i >= 0; i--){
        if(lvl[p] - (1<<i) >= lvl[q])p = P[p][i];
    }
    if(p == q)return p;
    for(int i = log; i >= 0; i--){
        if(P[p][i] != -1 && P[p][i] != P[q][i]){
            p = P[p][i];
            q = P[q][i];
        }
    }
    return pre[p];
}
```

SPLAY TREE:

```
struct node{
    int v, l, r, p, sz;
} T[200001];
struct SplayTree{
    int n, root;
    SplayTree(){
        n = root = 0;
        T[0].sz = T[0].l = T[0].r = 0;
    }
    int newNode(int v, int p){
        n++;
        T[n].v = v;
        T[n].l = T[n].r = 0;
        T[n].sz = 1;
        T[n].p = p;
        return n;
    }
    void update(int c){
   T[c].sz = 1 + T[ T[c].l ].sz + T[ T[c].r ].sz;
    }
    void Zig(int c){
        int p = T[c].p, q = T[p].p;
        T[p].l = T[c].r;
        if(T[p].l != 0) T[T[p].l].p = p;
      T[c].r = p; T[c].p = q; T[p].p = c;
        if(q != 0){
            if(T[q].l == p) T[q].l = c;
            else T[q].r = c;
        }
        update(p);
    }
    void Zag(int c){
        int p = T[c].p, q = T[p].p;
        T[p].r = T[c].l;
        if(T[p].r != 0) T[T[p].r].p = p;
      T[c].l = p; T[c].p = q; T[p].p = c;
        if(q != 0){
            if(T[q].l == p) T[q].l = c;
            else T[q].r = c;
        }
        update(p);
    }
    void splay(int c){
        while(T[c].p != 0){
            int p = T[c].p, q = T[p].p;
            if(q != 0 && T[q].l == p){
                if(T[p].l == c) Zig(p),Zig(c);
                else Zag(c),Zig(c);
        } else if(q != 0 && T[q].r == p){
                if(T[p].r == c) Zag(p),Zag(c);
                else Zig(c),Zag(c);
            } else {
                if(T[p].l == c) Zig(c);
                else Zag(c);
            }
        }
        root = c;
        update(c);
    }
```

```
    int search(int p){
        if(T[root].sz == 0) return 0;
        int c = root;
        while(p > 0){
      if(T[ T[c].l ].sz >= p) c=T[c].l;
            else {
                p -= T[ T[c].l ].sz + 1;
                if(p == 0) {
                    splay(c);
                    return T[c].v;
                }
                c = T[c].r;
            }
        }
    }
    int build(int l, int r, int p){
        if(l > r) return 0;
        int mid = (l + r) / 2;
        int c = newNode(ar[mid], p);
        if(p == 0) root = c;
if(l < mid)T[c].l = build(l, mid - 1, c);
if(r > mid)T[c].r = build(mid + 1, r, c);
        update(c);
        return c;
    }
};
```

DIVIDE AND CONQUER

SHANKS:

```
LL solve ( LL a, LL b, LL m ) {
    LL n = ( LL ) sqrt ( m + .0 ) + 1 ;
    LL an = 1 ;
    for ( int i = 0 ; i < n ; ++ i )
        an = ( an * a ) % m ;
    map < LL , LL > vals ;
for(int i=1,cur = an ; i <= n ; ++ i ){
 if(!vals.count(cur)) vals[cur]=i;
        cur = ( cur * an ) % m ;
    }
    LL ret = m;
    for(int i=0,cur=b ; i <= n ; ++ i){
        if ( vals. count ( cur ) ) {
            int ans=vals[ cur ] * n - i;
            if ( ans < ret )ret = ans;
        }
        cur = ( cur * a ) % m ;
    }
    if(ret < m)return ret;
    return - 1 ;
}
```

DP OPTIMIZATIONS:

CONVEX HULL TRICK:

```
int M[20001];
int B[20001];
int bad(int l1,int l2,int l3){
    return (B[l3]-B[l1])*(M[l1]-M[l2])<(B[l2]-
B[l1])*(M[l1]-M[l3]);
}
int cnt;
void insert_line(int m,int b){
    M[cnt]=m;
    B[cnt++]=b;
 while(cnt>=3&&bad(cnt-3,cnt-2,cnt-1)>0){
        swap(M[cnt-1],M[cnt-2]);
        swap(B[cnt-1],B[cnt-2]);
        cnt--;
    }
}
int p;
int query(int x) {
    if(p>=cnt)p=cnt-1;
 while(p<cnt-1 && M[p+1]*x+B[p+1]<M[p]*x+B[p])p++;
    return M[p]*x+B[p];
 }
```

```
 dp[i] = min{j<i}(dp[j] + b[j]*a[i]),
 b[j] >= b[j+1]
DIVIDE AND CONQUER OPTIMIZATION:
dp[i][j]=min{k<j}(dp[i-1][k] + C[k][j]),
A[i][j]<=A[i][j+1].
Find mid, then call(i,mid, st, ed), take the pos which
gives the min result, call(mid+1,e,pos,ed).
KNUTH OPTIMIZATION:
dp[i][j] = min{i<k<j}(dp[i][k]+dp[k][j])+C[i][j],
A[i][j-1] <= A[i][j] <= A[i+1][j].
run a loop from A[i][j-1] to A[i+1][j]. A[i][j] is the
position which gives the min.
```

MATH

EXTENDED EUCLID:number of solutions to ax+by+c==0, given range.

```
struct Euclid {
    i64 x, y, d;
    Euclid() {}
    Euclid( i64 xx, i64 yy,i64 dd){x= xx, y = yy, d =
dd; } };
Euclid egcd( i64 a, i64 b ) {
    if( !b ) return Euclid(1, 0, a);
    Euclid r = egcd ( b, a % b );
    return Euclid( r.y, r.x - a / b * r.y, r.d );
}
i64 abs64( i64 a ) {
    return a > 0 ? a : -a;
}
i64 myfloor( i64 a, i64 b ) {
    i64 c = a / b;
    if( (a % b) && a < 0 ) c--;
    return c;
}
i64 myceil( i64 a, i64 b ) {
    i64 c = a / b;
    if( (a % b) && a > 0 ) c++;
    return c;
}
i64 solve() {
 i64 a, b, c, x1, x2, y1, y2, X, Y, n1, n2, m1, m2;
    scanf("%lld %lld %lld %lld %lld %lld %lld", &a, &b, &c,
&x1, &x2, &y1, &y2);
    c *= -1;
if(a < 0 ) a *= -1, x1 *= -1, x2*=-1, swap(x1, x2);
if(b < 0 ) b *= -1, y1 *= -1,y2 *=-1, swap(y1, y2);
if(!a&&!b)return !c?(x2-x1+1)  * (y2 - y1 + 1) : 0;
    if( b == 0 ) {
        if( c % a ) return 0;
        i64 x = c / a;
return (x>=x1 && x<=x2)*(y2 - y1 + 1);
    }
    if( a == 0 ) {
        if( c % b ) return 0;
        i64 y = c / b;
return (y>=y1 && y<=y2) * (x2 - x1 + 1);
    }
    Euclid s = egcd( a, b );
    if( c % s.d ) return 0;
    a /= s.d; b /= s.d;
    c /= s.d; s.d = 1;
    X = s.x * c; Y = s.y * c;
    n2 = min(myfloor(X-x1,b),myfloor(y2 - Y, a ) );
    n1 = -min(myfloor(Y-y1,a),myfloor(x2- X, b ) );
    return (n2 < n1) ? 0 : n2 - n1 + 1;
}
```

POLYNOMIAL ROOTS:

```
typedef complex<double> cdouble;
typedef vector<cdouble> poly;
pair<poly,cdouble>horner(const poly &a,cdouble x0){
    int n = a.size();
    poly b = poly(max(1, n - 1));
    for(int i = n - 1; i > 0; i--)
     b[i - 1] = a[i] + (i < n - 1 ? b[i] * x0 : 0);
    return make_pair(b, a[0] + b[0] * x0);
```

```
}
cdouble eval(const poly &p, cdouble x) {
    return horner(p, x).second;
}
poly derivative(const poly &p) {
    int n = p.size();
    poly r = poly(max(1, n - 1));
    for(int i = 1; i < n; i++)
        r[i - 1] = p[i] * cdouble(i);
    return r;
}
const double EPS = 1e-9;
int cmp(cdouble x, cdouble y) {
    double diff = abs(x) - abs(y);
    return diff < -EPS ? -1 : (diff > EPS ? 1 : 0);
}
cdouble find_one_root(const poly &p0, cdouble x) {
    int n = p0.size() - 1;
    poly p1 = derivative(p0);
    poly p2 = derivative(p1);
for(int step = 0; step < 10000; step++) {
        cdouble y0 = eval(p0, x);
        if (cmp(y0, 0) == 0) break;
        cdouble G = eval(p1, x) / y0;
    cdouble H = G * G - eval(p2, x) - y0;
        cdouble R = sqrt(cdouble(n - 1)* (H * cdouble(n)
- G * G));
        cdouble D1 = G + R;
        cdouble D2 = G - R;
cdouble a = cdouble(n) / (cmp(D1, D2) > 0 ? D1:D2);
        x -= a;
        if (cmp(a, 0) == 0) break;
    }
    return x;
}
vector<cdouble>find_all_roots(const poly &p) {
    vector<cdouble> res;
    poly q = p;
    while (q.size() > 2) {
        cdouble z(rand() / double(RAND_MAX), rand() /
double(RAND_MAX));
        z = find_one_root(q, z);
        z = find_one_root(p, z);
        q = horner(q, z).first;
        res.push_back(z);
    }
    res.push_back(-q[0] / q[1]);
    return res;
}
int main( int argc, char* argv[] ) {
    poly p;
//x^3-8x^2 - 13x + 140 = (x+4)(x-5)(x-7)
    p.push_back(140);
    p.push_back(-13);
    p.push_back(-8);
    p.push_back(1);
    vector<cdouble> roots = find_all_roots(p);
for(size_t i=0; i < roots.size(); i++) {
        if (abs(roots[i].real()) < EPS) roots[i]-=
cdouble(roots[i].real(), 0);
        if (abs(roots[i].imag()) < EPS) roots[i]-=
cdouble(0, roots[i].imag());
     cout << setprecision(3) << roots[i] << endl;
    }
    return 0;
}
```

CHINESE REMAINDER THEOREM: smallest solution to given array of p[i] and r[i].

```
int64 x, y, n, p[N], r[N];
void exgcd(int64 a, int64 b) {
  if(!b) { x = 1, y = 0; return; }
  exgcd(b, a%b); int64 tx = x; x = y; y = tx-a/b*y;
}
int64 chinese_remainder() {
  int64 D = 1, ans = 0;
  forl(i, 0, n) D *= p[i];
  forl(i, 0, n) {
    int64 m = D/p[i];
    exgcd(m, p[i]);
    ans = (ans+r[i]*m*x)%D;
  }
  return (ans%D+D)%D;
}
int main() {
    REP(i, n) cin >> p[i] >> r[i];
    cout << chinese_remainder() << endl;
  return 0;
}
```

Generalized Diophantine equation solver

Linear Diophantine Equation Solver for Any Number of Variables. Given the linear equation with integer coefficients a[i]:
a[0]*x_0 + a[1]*x_1 + . . . + a[n-1]*x_n-1 = c. It is assumed that no coefficient is 0. Determines whether integer solutions exist. If they do, provides a specific solution for the variables in the array sol.(sol is viewed as an n-dimensional vector). Then bas[0], bas[1], . . ., bas[n-1] are basis vectors for the integer solutions for the homogeneous equation: a[0]*x_0 + a[1]*x_1 + . . . + a[n-1]*x_n-1 = 0 (Each bas[i] is an n-vector). Thus, all solutions are given by: (x_0, . . ., x_n-1) = sol + t_0*bas[0] + . . . + t_{n-1}*bas[n-1] where t_0, . . ., t_{n-1} are arbitrary integers.

```
int gcd(int a, int b, int *s, int *t) {
    int r, r1, r2, a1, a2, b1, b2, q;
    int A = a, B = b;
// unnecessary if a, b >= 0
    if (a < 0) {
        *s *= -1;
        r = gcd(-a, b, s, t);
        return r;
    }
    if (b < 0) {
        r = gcd(a, -b, s, t);
        *t *= -1;
        return r;
    }
    a1 = b2 = 1;
    a2 = b1 = 0;
    while (b) {
        assert(a1*A + a2*B == a);
        q = a / b;
        r = a % b;
        r1 = a1 - q*b1;
        r2 = a2 - q*b2;
        a = b; a1 = b1; a2 = b2;
        b = r; b1 = r1; b2 = r2;
    }
    *s = a1; *t = a2;
    assert(a >= 0);
    return a;
}
int gcd_solve(int a[M], int numV, int sol[M], int bas[M-1]
[M]) {
    int ret, d, v, i;
    if (numV == 2) {
 ret = gcd(a[0], a[1], &sol[0], &sol[1]);
        bas[0][0] = a[1]/ret;
        bas[0][1] = -a[0]/ret;
    } else if (numV > 2) {
    d = gcd_solve(a, numV-1, sol, bas);
ret = gcd(d, a[numV-1],&v,&sol[numV-1]);
        for (i=0; i<numV-1; i++) {
bas[numV-2][i] = a[numV-1]/ret * sol[i];
            sol[i] *= v;
        }
```

```
        bas[numV-2][numV-1] = -d/ret;
    } else exit(1);
    return ret; }
int diophantine(int a[M], int numV, int c, int sol[M],
int bas[M-1][M]) {
    int i,j,d,q;
    for( i=0; i<numV; i++ )
        assert( a[i] != 0 );
    for( i=0; i<numV-1; i++ )
      for( j=0; j<numV; j++ )
            bas[i][j] = 0;
    d = gcd_solve(a,numV,sol,bas);
    if( c % d == 0 ) {
        q = c/d;
        for( i=0; i<numV; i++ )
            sol[i] *= q;
        return 1;
    }
}
return 0;
```

Simplex Algorithm:

Simplex algorithm on augmented matrix a of dimension (m+1)x(n+1) returns 1 if feasible, 0 if not feasible, -1 if unbounded. returns solution in b[] in original var order, max(f) in ret form: maximize sum_j(a_mj*x_j)-a_mn s.t. sum_j(a_ij*x_j)<=a_in standard form.
To convert into standard form:
1. if exists equality constraint, then replace by both >= and <=
2. if variable x doesn't have nonnegativity constraint, then replace by difference of 2 variables like x1-x2, where x1>=0, x2>=0
3. for a>=b constraints, convert to-a<=-b
note: watch out for -0.0 in the solution, algorithm may cycle eps = 1e-7 may give wrong answer, 1e-10 is better

```
void pivot(int m, int n, double a[maxm][maxn], int
B[maxm], int N[maxn], int r, int c) {
    int i, j;
    swap(N[c], B[r]);
    a[r][c]=1/a[r][c];
    for (j=0; j<=n; j++)if (j!=c)
            a[r][j]*=a[r][c];
    for (i=0; i<=m; i++)if (i!=r) {
          for (j=0; j<=n; j++)if (j!=c)
             a[i][j]-=a[i][c]*a[r][j];
            a[i][c] = -a[i][c]*a[r][c];
        }
}
int feasible(int m, int n, double a[maxm][maxn], int
B[maxm], int N[maxn]) {
    int r, c, i;
    double p, v;
    while (1) {
        for (p=inf, i=0; i<m; i++)
          if (a[i][n]<p)
                p=a[r=i][n];
        if (p>-eps) return 1;
        for (p=0, i=0; i<n; i++)
            if (a[r][i]<p)
                p=a[r][c=i];
        if (p>-eps) return 0;
        p = a[r][n]/a[r][c];
        for (i=r+1; i<m; i++)
            if (a[i][c]>eps) {
                v = a[i][n]/a[i][c];
                if (v<p)
                    r=i, p=v;
            }
        pivot(m, n, a, B, N, r, c);
    }
}
int simplex(int m, int n, double a[maxm][maxn], double
b[maxn], double& ret) {
    int B[maxm], N[maxn], r, c, i;
    double p, v;
    for (i=0; i<n; i++)
        N[i]=i;
    for (i=0; i<m; i++)
        B[i]=n+i;
    if (!feasible(m, n, a, B, N))
        return 0;
    while (1) {
        for (p=0, i=0; i<n; i++)
          if (a[m][i]>p) p=a[m][c=i];
        if (p<eps) {
            for (i=0; i<n; i++)
              if (N[i]<n) b[N[i]]=0;
        for (i=0; i<m; i++) if (B[i]<n)
                    b[B[i]]=a[i][n];
            ret = -a[m][n];
            return 1;
        }
        for (p=inf, i=0; i<m; i++)
          if (a[i][c]>eps) {
                v = a[i][n]/a[i][c];
                if (v<p) p=v, r=i;
            }
        if (p==inf) return -1;
        pivot(m, n, a, B, N, r, c);
    }
}
```

FFT:

```
typedef complex < double > base ;
void fft ( vector < base > & a, bool invert ) {
    int n = ( int ) a. size ( ) ;
    for ( int i=1,j = 0 ; i < n ; ++ i ){
        int bit = n >> 1 ;
        for ( ; j >= bit ; bit >>= 1 )
            j -= bit ;
        j += bit ;
        if ( i < j )
            swap ( a [ i ] , a [ j ] ) ;
    }
    for(int len=2;len<=n;len <<= 1 ) {
        double ang = 2 * PI / len * (invert?-1:1) ;
        base wlen(cos(ang ),sin(ang));
        for(int i=0;i < n ; i += len ) {
            base w ( 1 ) ;
            for(int j=0;j<len/2;++j){
  base u=a[i+j],v = a [ i + j + len / 2 ] * w ;
                a [ i + j ] = u + v ;
          a [ i + j + len / 2 ] = u - v ;
                w *= wlen ;
            }
        }
    }
    if ( invert )
        for ( int i = 0 ; i < n ; ++ i )
            a [ i ] /= n ;
}
int multiply(const vector<int>&a,const vector<int>&b){
vector<base>fa(a.begin(),a.end()),fb(b.begin(),b.end() );
    size_t n = 1 ;
    while (n<max(a.size(),b.size() ) )  n <<= 1 ;
    n <<= 1 ;
    fa.resize ( n ) ,  fb. resize ( n ) ;
    fft(fa,false ) ,  fft ( fb, false ) ;
    for ( size_t i = 0 ; i < n ; ++ i )
        fa [ i ] *= fb [ i ] ;
    fft ( fa, true ) ; int ret=0;
    for ( size_t i = 0 ; i < n ; ++ i ){
        int x=(int)(fa[i].real() + 0.5 );
        if(x>0) ret++;
    }
    return ret; }
```

GEOMETRY

Point in a Polygon:

```
bool PointInPolygon(const vector<PT> &p, PT q) {
  bool c = 0;
  for (int i = 0; i < p.size(); i++){
    int j = (i+1)%p.size();
    if ((p[i].y <= q.y && q.y < p[j].y ||
      p[j].y <= q.y && q.y < p[i].y) &&
      q.x < p[i].x + (p[j].x - p[i].x) * (q.y -
p[i].y) / (p[j].y - p[i].y))
      c = !c;
  }
  return c;  }
```

Point on polygon:

```
bool PointOnPolygon(const vector<PT> &p, PT q) {
  for (int i = 0; i < p.size(); i++)
    if (dist2(ProjectPointSegment(p[i],
p[(i+1)%p.size()], q), q) < EPS)
      return true;
    return false;
}
```

Triangle area from Medians:

```
double triAreaFromMedians( double ma, double mb, double
     mc ){
    double x = 0.5 * ( ma + mb + mc );
  double a=x*( x - ma ) * ( x - mb ) * ( x - mc );
    if( a < 0.0 ) return -1.0;
    else return sqrt( a ) * 4.0 / 3.0;
}
```

Greater Circle(latitude, longitude):

```
double greatCircle( double laa, double loa, double lab,
     double lob ){
    double PI = acos( -1.0 ), R = 6378.0;
    double u[3] = { cos( laa ) *sin(loa), cos( laa ) *
cos( loa ), sin( laa ) };
    double v[3] = { cos( lab ) *sin(lob), cos( lab ) *
cos( lob ), sin( lab ) };
    double dot = u[0]*v[0] + u[1]*v[1] + u[2]*v[2];
    bool flip = false;
    if( dot < 0.0 ){
        flip = true;
 for(int i=0; i < 3; i++ ) v[i] = -v[i];
    }
    double cr[3] = { u[1]*v[2] - u[2]*v[1], u[2]*v[0] -
u[0]*v[2], u[0]*v[1] - u[1]*v[0] };
    double theta = asin( sqrt( cr[0]*cr[0] + cr[1]*cr[1]
+ cr[2]*cr[2] ) );
    double len = theta * R;
    if( flip ) len = PI * R - len;
    return len;
}
```

Polygon area + centroid:

```
double ComputeSignedArea(const vector<PT> &p) {
  double area = 0;
  for(int i = 0; i < p.size(); i++) {
    int j = (i+1) % p.size();
    area += p[i].x*p[j].y - p[j].x*p[i].y;
  }
  return area / 2.0;
}
double ComputeArea(const vector<PT> &p){
  return fabs(ComputeSignedArea(p));
}
PT ComputeCentroid(const vector<PT> &p){
  PT c(0,0);
double scale = 6.0*ComputeSignedArea(p);
  for (int i = 0; i < p.size(); i++){
    int j = (i+1) % p.size();
    c = c + (p[i]+p[j])*(p[i].x*p[j].y - p[j].x*p[i].y);
  }
  return c / scale;
}
```

Union of n circles:

```
struct Point{
    double x,y;
    Point(double a=0.0,double b=0.0){x=a;y=b;}
    Point operator+(const Point&a)const{return
Point(x+a.x,y+a.y);}
    Point operator-(const Point&a)const{return Point(x-
a.x,y-a.y);}
    Point operator*(const double&a)const{return
Point(x*a,y*a);}
    Point operator/(const double&a)const{return
Point(x/a,y/a);}
    double operator*(const Point&a)const{return x*a.y-
y*a.x;}
    double operator/(const Point&a)const{return sqrt((a.x-
x)*(a.x-x)+(a.y-y)*(a.y-y));}
    double operator%(const Point&a)const{return
x*a.x+y*a.y;}
}po[1005];
double r[1005];
inline int sgn(double x) {return fabs(x)<eps?0:(x>0.0?1:-
1);}
pair<double,bool>ARG[2005];
double cir_uni(Point c[],double r[],int n){
    double sum=0.0,sum1=0.0,d,p1,p2,p3;
    for(int i=0;i<n;i++){
        bool f=1;
        for(int j=0;f&&j<n;j++)
        if(i!=j&&sgn(r[j]-r[i]-c[i]/c[j])!=-1)f=0;
        if(!f)swap(r[i],r[--n]),  swap(c[i--],c[n]);
    }
    for(int i=0;i<n;i++){
        int k=0,cnt=0;
        for(int j=0;j<n;j++)
            if(i!=j&&sgn((d=c[i]/c[j])-r[i]-r[j])<=0){
                p3=acos((r[i]*r[i]+d*d-r[j]*r[j])/
(2.0*r[i]*d));
   p2=atan2(c[j].y-c[i].y,c[j].x-c[i].x);
                p1=p2-p3;p2=p2+p3;
        if(sgn(p1+pi)==-1)p1+=2*pi,cnt++;
         if(sgn(p2-pi)==1)p2-=2*pi,cnt++;
                ARG[k++]=make_pair(p1,0);
                ARG[k++]=make_pair(p2,1);
            }
        if(k){sort(ARG,ARG+k);
            p1=ARG[k-1].first-2*pi;
            p3=r[i]*r[i];
            for(int j=0;j<k;j++){
                p2=ARG[j].first;
                if(cnt==0){
            sum+=(p2-p1-sin(p2-p1))*p3;
                    sum1+=(c[i]
+Point(cos(p1),sin(p1))*r[i])*(c[i]
+Point(cos(p2),sin(p2))*r[i]); }
       p1=p2;ARG[j].second?cnt--:cnt++;
            }     }
        else sum+=2*pi*r[i]*r[i];
    }
    return (sum+fabs(sum1))*0.5; }
```

CONVEX HULL

```
typedef pair<long, long> point;
bool cw(const point &a, const point &b, const point &c) {
    return (b.first - a.first) * (c.second - a.second) -
(b.second - a.second) * (c.first - a.first) < 0;}
vector<point> convexHull(vector<point> p) {
    int n = p.size();
    if (n <= 1)return p;
    sort(p.begin(), p.end());
    int cnt = 0;
    vector<point> q(n * 2);
    for (int i = 0; i < n; q[cnt++] = p[i++])
     for (; cnt >= 2 && !cw(q[cnt - 2], q[cnt - 1], p[i]);
--cnt);
    for (int i = n - 2, t = cnt; i >= 0; q[cnt++] = p[i--])
```

```
    for (; cnt > t && !cw(q[cnt - 2], q[cnt - 1],
p[i]); --cnt);
    q.resize(cnt - 1 - (q[0] == q[1]));
    return q;
}
```

CIRCLE CIRCLE TANGENT LINES:

```
bool operator<(line a, line b){
    return a.a==b.a?a.b<b.b:a.a<b.a;
}
point getPoint(circle C, double a){
   return point(C.p.x+cos(a)*C.r,C.p.y+sin(a)*C.r);
}
int getTangents(circle A,circle B,vector<line> &v){
    point ta[5],tb[5];
    point *a = ta,*b = tb;
    if(A.r<B.r){
        swap(A,B); swap(a,b);
    }
    int cnt = 0;
    double d2 = sqr(A.p.distance(B.p));
    double rdiff = A.r-B.r;
    double rsum = A.r+B.r;
    if(dblcmp(d2-rdiff*rdiff)<0) return 0;
    point va = B.p-A.p;
    double base = atan2(va.y,va.x);
   if(dblcmp(d2)==0&&dblcmp(A.r-B.r)==0) return -1;
    if(dblcmp(d2-rdiff*rdiff)==0){
        a[cnt] = getPoint(A,base);
        b[cnt++] = getPoint(B,base);
        v.push_back(line(ta[0],tb[0]));
        return 1;
    }
    double ang = acos((A.r-B.r)/sqrt(d2));
    a[cnt] = getPoint(A,base+ang);b[cnt++] =
getPoint(B,base+ang);
    a[cnt] = getPoint(A,base-ang);b[cnt++] =
getPoint(B,base-ang);
    if(dblcmp(d2-rsum*rsum)==0){
        a[cnt] = getPoint(A,base);b[cnt++] =
getPoint(B,PI+base);
    }else if(dblcmp(d2-rsum*rsum)>0){
        double ang = acos((A.r+B.r)/sqrt(d2));
        a[cnt] = getPoint(A,base+ang);b[cnt++] =
getPoint(B,PI+base+ang);
        a[cnt] = getPoint(A,base-ang);b[cnt++] =
getPoint(B,PI+base-ang);
    }
    for(int i = 0;i<cnt;i++) v.pb(line(ta[i],tb[i]));
    return cnt;
}
```

CLOSEST PAIR:

```
const int N = int(1e5) + 10;
const LL INF = 1LL << 60;
struct Point {
    LL x,y;
} point[N];
int n;
int tmpt[N];
bool cmpxy(const Point& a, const Point& b) {
    if (a.x != b.x)
        return a.x < b.x;
    return a.y < b.y;
}
bool cmpy(const int& a, const int& b) {
    return point[a].y < point[b].y;
}
LL dis2(int i, int j) {
    return (point[i].x - point[j].x) * (point[i].x –
point[j].x) + (point[i].y - point[j].y) * (point[i].y -
point[j].y);
}
LL sqr(LL x) {
    return x * x;
}
LL Closest_Pair(int left, int right) {
    LL d = INF;
    if (left == right)
        return d;
    if (left + 1 == right)
        return dis2(left, right);
    int mid = (left + right) >> 1;
    LL d1 = Closest_Pair(left, mid);
    LL d2 = Closest_Pair(mid + 1, right);
    d = min(d1, d2);
    int i, j, k = 0;
    for (i = left; i <= right; i++) {
        if (sqr(point[mid].x - point[i].x) <= d)
            tmpt[k++] = i;
    }
    sort(tmpt, tmpt + k, cmpy);
    for (i = 0; i < k; i++) {
        for (j = i + 1; j < k && sqr(point[tmpt[j]].y -
point[tmpt[i]].y) < d;j++) {
            LL d3 = dis2(tmpt[i], tmpt[j]);
            if (d > d3) d = d3;
        }
    }
    return d;}
```

Incircle of triangle:

```
r = area / s
I.x = (A.x*a + B.x*b + C.x) / (a+b+c)
I.y = (A.y*a + B.y*b + C.y) / (a+b+c)
```

Pick's Theorem:

```
I = area + 1 – B/2,
where I = number of points inside
B = number of points on the border.
```

POLYGON Intersection:

```
#define EPS 1e-8
// 1 for clockwise ordering of points, -1 for counter-
clockwise
#define ORDER 1
struct Point {
  double x, y;
  bool operator < (const Point &p) const {
  return (y+EPS<p.y)||((y-EPS<p.y)&&(x+EPS<p.x));
  }
  bool operator == (const Point &p) const {
    return !(*this < p) && !(p < *this);
  }
};
struct Triangle {
  Point p[3];
};
inline double cross(const Point &a, const Point &b,
                    const Point &c, const Point &d){
  return ((b.x - a.x)*(d.y - c.y) - (d.x - c.x)*(b.y -
a.y));
}
int leftRight(const Point &a, const Point &b, const Point
&p){
  double res = cross(a, b, a, p);
  if (res > EPS) return -1;
  else if (res < -EPS) return 1;
  return 0;
}
int isConcave(Point &a, Point &b, Point &c){
  return (ORDER*leftRight(a, b, c) <= 0);
}
int isInsideTriangle(Point &a, Point &b, Point &c, Point
&p){
  int r1 = leftRight(a, b, p);
  int r2 = leftRight(b, c, p);
  int r3 = leftRight(c, a, p);
  return ((ORDER*r1 >= 0) && (ORDER*r2 >= 0) && (ORDER*r3
>= 0));
}
```

```
void triangulate(list<Point> &P, vector<Triangle> &T){
  list<Point>::iterator a, b, c, q;
  Triangle t;
  T.clear();
  if (P.size() < 3) return;
  for (a=b=P.begin(), c=++b, ++c; c != P.end(); a=b, c=+
+b, ++c) {
    if (!isConcave(*a, *b, *c)) {
      for (q = P.begin(); q != P.end(); q++) {
    if (q == a) {
      ++q;
      ++q;
      continue;
    }
    if (isInsideTriangle(*a, *b, *c, *q)) break;
      }
      if (q == P.end()) {
    t.p[0] = *a; t.p[1] = *b; t.p[2] = *c;
    T.push_back(t);
    P.erase(b);
    b = a;
    if (b != P.begin()) b--;
      }
    }
  }
}
int isectLineSegs(Point &a, Point &b, Point &c, Point
&d, Point &p){
  double r, s, dn, n1, n2;
  n1 =  cross(c, d, c, a);
  n2 = -cross(a, b, a, c);
  dn =  cross(a, b, c, d);
  if ((dn > EPS) || (dn < -EPS)) {
    r = n1/dn; s = n2/dn;
    if ((-EPS < r) && (r < 1+EPS) &&
    (-EPS < s) && (s < 1+EPS)) {
      p.x = a.x + r*(b.x - a.x);
      p.y = a.y + r*(b.y - a.y);
      return 1;
    }
  }
  return 0;
}
double areaPoly(vector<Point> &P){
  double area = 0.0;
  vector<Point>::iterator p, q;
  for (p = P.end()-1, q = P.begin(); q != P.end();
       p = q++) {
    area += p->x*q->y - p->y*q->x;
  }
  return area/2.0;
}
Point P0;
bool radialLessThan(const Point &a,const Point &b){
  return (ORDER == leftRight(P0, a, b));
}
double isectAreaTriangles(Triangle &a,Triangle &b){
  vector<Point> P;
  vector<Point>::iterator s, e, ne;
  Point p;
  Triangle T[2] = { a, b };
  int i, j, r, t, u, v;
  double area = 0;
  P.clear();
  for(r=1, t=0; t < 2; r = t++) for(i = 2, j = 0; j < 3;
i = j++){
if(isInsideTriangle(T[r].p[0],T[r].p[1],T[r].p[2],T[t].p
[i]))
      P.push_back(T[t].p[i]);
    for (u = 2, v = 0; v < 3; u = v++)
if(isectLineSegs(T[t].p[i],T[t].p[j],T[r].p[u],T[r].p[v]
,p))
      P.push_back(p);
  }
  if (!P.empty()) {
    s = P.begin();e = P.end();
    sort(s, e);
    ne = unique(s, e);
    P.erase(ne, e);
    if (P.size() >= 3) {
      P0 = P[0];
      sort(s+1, ne, radialLessThan);
      area = areaPoly(P);
    }
  }
  return area;
}
double isectAreaGpoly(list<Point> &P, list<Point> &Q){
  double area = 0.0;
  vector<Triangle> S, T;
  vector<Triangle>::iterator s, t;
  triangulate(P, S);
  triangulate(Q, T);
  for (s = S.begin(); s != S.end(); s++)
    for (t = T.begin(); t != T.end(); t++)
      area += isectAreaTriangles(*s, *t);
  return -ORDER*area;
}
```

3d polygon area:

```
double area_heron(double a, double b, double c){
  double s = (a+b+c)/2.0;
  if(a > s || b > s || c > s) return -1;
  return sqrt(s*(s-a)*(s-b)*(s-c));
}
double dist3d(Point a, Point b){return sqrt(SQR(a.x-b.x)
+SQR(a.y-b.y)+SQR(a.z-b.z));
}
double area_poly3d(Point *p, int n){
  int i; double total = 0;
  for(i = 2; i < n; i++)
    total += area_heron(dist3d(p[0], p[i]),
             dist3d(p[0], p[i-1]),
             dist3d(p[i], p[i-1]));
  return total;
}
```

3d polygon:

```
ld dot(const p3d &a, const p3d &b){
    return a.x*b.x + a.y*b.y + a.z*b.z;
}
p3d cross(const p3d &a, const p3d &b){
    return p3d(a.y*b.z - a.z*b.y, a.z*b.x - a.x*b.z,
a.x*b.y - a.y*b.x);
}
struct face {
    int p1, p2, p3;
    face(int a, int b, int c) {
        p1 = a, p2 = b, p3 = c;
    }
};
int t, n;
p3d data[1024];
bool above(const face &f, const p3d p) {
    p3d normal = cross(data[f.p2] - data[f.p1], data[f.p3]
- data[f.p2]);
    return dot(normal, p - data[f.p1]) > eps;
}
ld area(const face &f) {
    return abs(cross(data[f.p2] - data[f.p1], data[f.p3] -
data[f.p1]));
}
ld volume(const face &f) {
    p3d normal = cross(data[f.p2] - data[f.p1], data[f.p3]
- data[f.p2]);
    normal = normal / abs(normal);
    ld h = dot(normal, data[f.p1]);
```

```
    return area(f)*h/6;
}
int main() { // outputs area/volume of the CH in 3D
    cin >> n;
    for (int i = 0; i < n; i++) cin >> data[i].x >>
data[i].y >> data[i].z;
        for (int i = n - 1; i >= 0; i--) data[i] =
data[i] - data[0];
        for (int i = 1; i < n; i++)
            if (abs(data[i] - data[0]) > eps) {
                swap(data[i], data[1]);  break;
            }
        for (int i = 2; i < n; i++)
  if (abs(cross(data[1] - data[0], data[i] - data[0])) >
eps) swap(data[i], data[2]);  break;
        for (int i = 3; i < n; i++) {
            p3d normal = cross(data[1] - data[0],
data[2] - data[0]);
  if (abs(dot(normal, data[0] - data[i])) > eps)
                swap(data[i], data[3]);  break;
        }
        vector < face > f;
        f.push_back(face(0, 1, 2));
        f.push_back(face(2, 1, 0));
        for (int i = 3; i < n; i++) {
            vector < face > next;
            set < int > edge;
            for (int j = 0; j < (int)f.size(); j++)
                if (above(f[j], data[i])) {
         edge.insert((f[j].p1 << 10) | f[j].p2);
         edge.insert((f[j].p2 << 10) | f[j].p3);
         edge.insert((f[j].p3 << 10) | f[j].p1);
                }
            else next.push_back(f[j]);
            for (__typeof(edge.begin()) it =
edge.begin(); it != edge.end(); it++)
   if(!edge.count(((*it & 1023)<<10)|(*it>>10)))
   next.push_back(face(*it >> 10, *it & 1023, i));
         f = next;
        }
        ld ar = 0; ld vo = 0;
        for (int i = 0; i < (int)f.size(); i++) ar +=
area(f[i]), vo += volume(f[i]);
        cout << ar/2 << " " << vo << endl;
    }
    return 0;
}
GEOMETRY TEMPLATE:
const double eps=1e-8;
const double pi=acos(-1.0);
const double inf=1e20;
const int maxp=1111;
int dblcmp(double d)
{
    if (fabs(d)<eps)return 0;
    return d>eps?1:-1;
}
inline double sqr(double x){return x*x;}
struct point{
    double x,y;
    point()    {         }
    point(double _x,double _y){x = _x; y = _y; }
    bool operator==(point a)const{
return dblcmp(a.x-x) == 0 && dblcmp(a.y - y) == 0;
    }
    bool operator<(point a)const{
        return dblcmp(a.x - x) == 0 ? dblcmp(y - a.y) <
0 : x < a.x;
    }
    point operator-(point a)const{
        return point(x-a.x, y-a.y);
    }
    double len(){   return hypot(x, y);   }
    double len2() {   return x * x + y * y;   }
    double distance(point p){return hypot(x - p.x, y -
p.y); }
    point add(point p){return point(x+p.x,y+p.y);}
    point sub(point p){return point(x-p.x,y-p.y);}
    point mul(double b){return point(x*b, y*b);}
    point div(double b){return point(x/b,y/b);}
    double dot(point p) {return x*p.x+y*p.y;     }
    double det(point p) {   return x*p.y-y*p.x;
}
    double rad(point a,point b){
        point p=*this;
        return
fabs(atan2(fabs(a.sub(p).det(b.sub(p))),a.sub(p).dot(b.sub(
p))));
    }
    point trunc(double r){
        double l=len();
        if (!dblcmp(l))return *this;
        r/=l;
        return point(x*r,y*r);
    }
    point rotleft(){return point(-y,x);         }
    point rotright(){   return point(y,-x);     }
    point rotate(point p,double angle){
        point v=this->sub(p);
        double c=cos(angle),s=sin(angle);
        return point(p.x+v.x*c-v.y*s,p.y+v.x*s+v.y*c);
    }
};
/*relation()                          - 0 if collinear, 1
if ccw, 2 if cw
pointonseg(point p) - returns 1 if point is on segment
parallel(line v) - returns 1 if they are parallel
segcrossseg(line v) - returns 0 if does not intersect
returns 1 if non-standard intersection, returns 2 if
intersects
segcrossseg_inside(line v)    - returns 1 if intersects
strictly inside, else 0
linecrossseg(line v)                - v is line
linecrossline(line v)         - 0 if parallel, 1 if
coincides, 2 if intersects
lineprog(point p)-returns projected point p on ab line
symmetrypoint(point p)- returns reflection point of p over
ab */
struct line{
    point a,b;
    line() { }
    line(point _a,point _b){ a=_a; b=_b;   }
    bool operator==(line v){ return (a==v.a)&&(b==v.b);
}
    line(point p,double angle){
        a=p;
        if (dblcmp(angle-pi/2)==0){
            b=a.add(point(0,1));
        }else{
            b=a.add(point(1,tan(angle)));
        }
    }
    line(double _a,double _b,double _c){//ax+by+c=0
        if (dblcmp(_a)==0){
            a=point(0,-_c/_b);
            b=point(1,-_c/_b);
        }else if (dblcmp(_b)==0){
            a=point(-_c/_a,0);
            b=point(-_c/_a,1);
        }else{
            a=point(0,-_c/_b);
            b=point(1,(-_c-_a)/_b);
        }
    }
    void adjust() {if(b<a)swap(a,b); }
```

```
    double length() {return a.distance(b); }
    double angle(){
        double k=atan2(b.y-a.y,b.x-a.x);
        if (dblcmp(k)<0)k+=pi;
        if (dblcmp(k-pi)==0)k-=pi;
        return k;
    }
    int relation(point p){
        int c=dblcmp(p.sub(a).det(b.sub(a)));
        if (c<0)return 1;
        if (c>0)return 2;
        return 3;
    }
    bool pointonseg(point p){
        return
dblcmp(p.sub(a).det(b.sub(a)))==0&&dblcmp(p.sub(a).dot(p
.sub(b)))<=0;
    }
    bool parallel(line v){
    return dblcmp(b.sub(a).det(v.b.sub(v.a)))==0;
    }
    int segcrossseg(line v){
        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
    int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
    int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
        if ((d1^d2)==-2&&(d3^d4)==-2)return 2;
        return
(d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
    }
    int segcrossseg_inside(line v){
if(v.pointonseg(a)||v.pointonseg(b)||pointonseg(v.a) ||
pointonseg(v.b)) return 0;
        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
     int d3=dblcmp(v.b.sub(v.a).det(a.sub(v.a)));
     int d4=dblcmp(v.b.sub(v.a).det(b.sub(v.a)));
        if ((d1^d2)==-2&&(d3^d4)==-2)return 1;
        return
(d1==0&&dblcmp(v.a.sub(a).dot(v.a.sub(b)))<=0||
d2==0&&dblcmp(v.b.sub(a).dot(v.b.sub(b)))<=0||
d3==0&&dblcmp(a.sub(v.a).dot(a.sub(v.b)))<=0||
d4==0&&dblcmp(b.sub(v.a).dot(b.sub(v.b)))<=0);
    }
    int linecrossseg(line v){//*this seg v line
        int d1=dblcmp(b.sub(a).det(v.a.sub(a)));
        int d2=dblcmp(b.sub(a).det(v.b.sub(a)));
        if ((d1^d2)==-2)return 2;
        return (d1==0||d2==0);
    }
    int linecrossline(line v){
if ((*this).parallel(v)) return v.relation(a)==3;
        return 2;
    }
    point crosspoint(line v){
        double a1=v.b.sub(v.a).det(a.sub(v.a));
        double a2=v.b.sub(v.a).det(b.sub(v.a));
        return point((a.x*a2-b.x*a1)/(a2-a1),(a.y*a2-
b.y*a1)/(a2-a1));
    }
    double dispointtoline(point p){
        return fabs(p.sub(a).det(b.sub(a)))/length();
    }
    double dispointtoseg(point p){
        if (dblcmp(p.sub(b).dot(a.sub(b)))<0||
dblcmp(p.sub(a).dot(b.sub(a)))<0){
            return min(p.distance(a),p.distance(b));
        }
        return dispointtoline(p);
}
    point lineprog(point p){
        return
a.add(b.sub(a).mul(b.sub(a).dot(p.sub(a))/b.sub(a).len2()))
;
    }
    point symmetrypoint(point p){
        point q=lineprog(p);
        return point(2*q.x-p.x,2*q.y-p.y);
    }
};
/*relation(point p)                -0 outside, 1 on
circumference, 2 inside circle
relationseg(line v)              -0 outside, 1 on
circumference, 2 inside circle
relationline(line v)             -0 outside, 1 on
circumference, 2 inside circle
getcircle(point a,point b,double r,circle&c1,circle&c2)
 -returns two circle c1,c2 through points a,b of radius
r,returns 0 for nor circle
getcircle(line u,point q,double r1,circle &c1,circle &c2)-
returns two circle c1,c2 which is tangent to line u, goes
through
 point q and has radius r1, returns 0 for no circle ,1 if
c1=c2 ,2 if c1!=c2
getcircle(line u,line v,double r1,circle &c1,circle
&c2,circle &c3,circle &c4)-returns 4 circles which is
tangent to line u,v has radius r1.
areatriangle(point a,point b)         -intersection area of
circle and triangle of point a,b,p
*/
struct circle {
    point p;
    double r;
    circle(){  }
    circle(point _p,double _r):p(_p),r(_r){  };
    circle(double x,double y,double _r):
p(point(x,y)),r(_r){};
    circle(point a,point b,point c){
p=line(a.add(b).div(2),a.add(b).div(2).add(b.sub(a).rotleft
())).crosspoint(line(c.add(b).div(2),c.add(b).div(2).add(b.
sub(c).rotleft())));
        r=p.distance(a);
    }
    circle(point a,point b,point c,bool t){
        line u,v;
        double m=atan2(b.y-a.y,b.x-a.x),n=atan2(c.y-
a.y,c.x-a.x);
        u.a=a;
u.b=u.a.add(point(cos((n+m)/2),sin((n+m)/2)));
        v.a=b;
        m=atan2(a.y-b.y,a.x-b.x),n=atan2(c.y-b.y,c.x-b.x);
v.b=v.a.add(point(cos((n+m)/2),sin((n+m)/2)));
        p=u.crosspoint(v);
        r=line(a,b).dispointtoseg(p);
    }
    void input(){p.input();scanf("%lf",&r); }
    void output() { printf("%.2lf %.2lf
%.2lf\n",p.x,p.y,r); }
    bool operator==(circle v){
        return ((p==v.p)&&dblcmp(r-v.r)==0);
    }
    bool operator<(circle v)const{
        return ((p<v.p)||(p==v.p)&&dblcmp(r-v.r)<0);
    }
    double area(){return pi*sqr(r);}
    double circumference(){return 2*pi*r;}
    int relation(point b){
        double dst=b.distance(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
        return 0;}
    int relationseg(line v){
```

```
        double dst=v.dispointtoseg(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
        return 0;
    }
    int relationline(line v){
        double dst=v.dispointtoline(p);
        if (dblcmp(dst-r)<0)return 2;
        if (dblcmp(dst-r)==0)return 1;
        return 0;
    }
    int getcircle(point a,point b,double
r,circle&c1,circle&c2){
        circle x(a,r),y(b,r);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if (!t)return 0;
        c1.r=c2.r=r;
        return t;
    }
    int getcircle(line u,point q,double r1,circle
&c1,circle &c2){
        double dis=u.dispointtoline(q);
        if (dblcmp(dis-r1*2)>0)return 0;
        if (dblcmp(dis)==0){
    c1.p=q.add(u.b.sub(u.a).rotleft().trunc(r1));
c2.p=q.add(u.b.sub(u.a).rotright().trunc(r1));
            c1.r=c2.r=r1;
            return 2;
        }
        line
u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),
u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
        line
u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),
u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
        circle cc=circle(q,r1);
        point p1,p2;
        if (!
cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
        c1=circle(p1,r1);
        if (p1==p2){c2=c1;return 1; }
        c2=circle(p2,r1);
        return 2;
    }
    int getcircle(line u,line v,double r1,circle
&c1,circle &c2,circle &c3,circle &c4){
        if (u.parallel(v))return 0;
        line
u1=line(u.a.add(u.b.sub(u.a).rotleft().trunc(r1)),
u.b.add(u.b.sub(u.a).rotleft().trunc(r1)));
        line
u2=line(u.a.add(u.b.sub(u.a).rotright().trunc(r1)),
u.b.add(u.b.sub(u.a).rotright().trunc(r1)));
        line
v1=line(v.a.add(v.b.sub(v.a).rotleft().trunc(r1)),
v.b.add(v.b.sub(v.a).rotleft().trunc(r1)));
        line
v2=line(v.a.add(v.b.sub(v.a).rotright().trunc(r1)),
v.b.add(v.b.sub(v.a).rotright().trunc(r1)));
        c1.r=c2.r=c3.r=c4.r=r1;
        c1.p=u1.crosspoint(v1); c2.p=u1.crosspoint(v2);
        c3.p=u2.crosspoint(v1); c4.p=u2.crosspoint(v2);
        return 4;
    }
    int getcircle(circle cx,circle cy,double
r1,circle&c1,circle&c2){
        circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
        int t=x.pointcrosscircle(y,c1.p,c2.p);
        if (!t)return 0;
        c1.r=c2.r=r1;
        return t;
    }
    int pointcrossline(line v,point &p1,point &p2){
        if (!(*this).relationline(v))return 0;
        point a=v.lineprog(p);
        double d=v.dispointtoline(p);
        d=sqrt(r*r-d*d);
        if (dblcmp(d)==0){  p1=a; p2=a; return 1;}
        p1=a.sub(v.b.sub(v.a).trunc(d));
        p2=a.add(v.b.sub(v.a).trunc(d));
        return 2;
    }
    int relationcircle(circle v){
        double d=p.distance(v.p);
        if (dblcmp(d-r-v.r)>0)return 5;
        if (dblcmp(d-r-v.r)==0)return 4;
        double l=fabs(r-v.r);
     if (dblcmp(d-r-v.r)<0&&dblcmp(d-l)>0)return 3;
        if (dblcmp(d-l)==0)return 2;
        if (dblcmp(d-l)<0)return 1;
    }
int pointcrosscircle(circle v,point &p1,point &p2){
        int rel=relationcircle(v);
        if (rel==1||rel==5)return 0;
        double d=p.distance(v.p);
        double l=(d+(sqr(r)-sqr(v.r))/d)/2;
        double h=sqrt(sqr(r)-sqr(l));

p1=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotleft().trunc
(h)));

p2=p.add(v.p.sub(p).trunc(l).add(v.p.sub(p).rotright().trun
c(h)));
        if (rel==2||rel==4)return 1;
        return 2;
    }
    int tangentline(point q,line &u,line &v){
        int x=relation(q);
        if (x==2)return 0;
        if (x==1){
            u=line(q,q.add(q.sub(p).rotleft()));
            v=u; return 1;  }
        double d=p.distance(q);
        double l=sqr(r)/d;
        double h=sqrt(sqr(r)-sqr(l));
u=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotleft().tru
nc(h))));
v=line(q,p.add(q.sub(p).trunc(l).add(q.sub(p).rotright().tr
unc(h))));
        return 2;
    }
    double areacircle(circle v){
        int rel=relationcircle(v);
        if (rel>=4)return 0.0;
        if (rel<=2)return min(area(),v.area());
        double d=p.distance(v.p);
        double hf=(r+v.r+d)/2.0;
      double ss=2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
      double a1=acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
        a1=a1*r*r;
    double a2=acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
        a2=a2*v.r*v.r;
        return a1+a2-ss;
    }
    double areatriangle(point a,point b){
 if (dblcmp(p.sub(a).det(p.sub(b))==0))return 0.0;
        point q[5];
        int len=0;
        q[len++]=a;
        line l(a,b);  point p1,p2;
        if (pointcrossline(l,q[1],q[2])==2){
if (dblcmp(a.sub(q[1]).dot(b.sub(q[1])))<0)q[len++]=q[1];
if (dblcmp(a.sub(q[2]).dot(b.sub(q[2])))<0)q[len++]=q[2];
        }
```

```
        q[len++]=b;
if(len==4&&(dblcmp(q[0].sub(q[1]).dot(q[2].sub(q[1])))>0
))swap(q[1],q[2]);
        double res=0;
        int i;
        for (i=0;i<len-1;i++){
            if (relation(q[i])==0||relation(q[i+1])==0){
                double arg=p.rad(q[i],q[i+1]);
                res+=r*r*arg/2.0;
            }
            else
        res+=fabs(q[i].sub(p).det(q[i+1].sub(p))/2.0);
        }
        return res;
    }
};
/*relationpoint(point q)       - returns 3 if q is a
vertex, 2 if on a side, 1 if inside, 0 if outside
relationline(line u)            - returns 1 if there is
some intersection, 0 if no intersection, 2 if intersect
at corner
relationcircle(circle c)        - returns 0 if outside
circle, 1 if tangent, 2 if inside */
struct polygon{
    int n;
    point p[maxp];
    line l[maxp];
    void input(int _n){
        n=_n;
        for (int i=0;i<n;i++)   p[i].input();
    }
    void add(point q)   {   p[n++]=q;  }
    void getline(){
        for (int i=0;i<n;i++)
            l[i]=line(p[i],p[(i+1)%n]);
    }
    struct cmp{
        point p;
        cmp(const point &p0){p=p0;}
  bool operator()(const point &aa,const point &bb){
            point a=aa,b=bb;
            int d=dblcmp(a.sub(p).det(b.sub(p)));
            if (d==0)
                return dblcmp(a.distance(p)-
b.distance(p))<0;
            return d>0;
        }
    };
    void norm(){
        point mi=p[0];
        for (int i=1;i<n;i++)mi=min(mi,p[i]);
        sort(p,p+n,cmp(mi));
    }
    bool isconvex(){
        bool s[3];
        memset(s,0,sizeof(s));
        int i,j,k;
        for (i=0;i<n;i++){
            j=(i+1)%n;
            k=(j+1)%n;
 s[dblcmp(p[j].sub(p[i]).det(p[k].sub(p[i])))+1]=1;
            if (s[0]&&s[2])return 0;
        }
        return 1;
    }
    int relationline(line u){
        int i,k=0;
        getline();
        for (i=0;i<n;i++){
            if (l[i].segcrossseg(u)==2)return 1;
            if (l[i].segcrossseg(u)==1)k=1;
        }
        if (!k)return 0;
        vector<point>vp;
        for (i=0;i<n;i++){
            if (l[i].segcrossseg(u)){
                if (l[i].parallel(u)){
                    vp.pb(u.a);
                    vp.pb(u.b);
                    vp.pb(l[i].a);
                    vp.pb(l[i].b);
                    continue;
                }
                vp.pb(l[i].crosspoint(u));
            }
        }
        sort(vp.begin(),vp.end());
        int sz=vp.size();
        for (i=0;i<sz-1;i++){
            point mid=vp[i].add(vp[i+1]).div(2);
            if (relationpoint(mid)==1)return 1;
        }
        return 2;
    }
    void convexcut(line u,polygon &po){
        int i;
        int &top=po.n;
        top=0;
        for (i=0;i<n;i++){
  int d1=dblcmp(p[i].sub(u.a).det(u.b.sub(u.a)));
            int
d2=dblcmp(p[(i+1)%n].sub(u.a).det(u.b.sub(u.a)));
            if (d1>=0)po.p[top++]=p[i];
            if (d1*d2<0)po.p[top++]
=u.crosspoint(line(p[i],p[(i+1)%n]));
        }
    }
    bool getdir(){
        double sum=0;
        int i;
        for (i=0;i<n;i++)
            sum+=p[i].det(p[(i+1)%n]);
        if (dblcmp(sum)>0)return 1;
        return 0;
    }
    double areacircle(circle c){
        int i,j,k,l,m;
        double ans=0;
        for (i=0;i<n;i++){
            int j=(i+1)%n;
            if
(dblcmp(p[j].sub(c.p).det(p[i].sub(c.p)))>=0)
                ans+=c.areatriangle(p[i],p[j]);
            else ans-=c.areatriangle(p[i],p[j]);
        }
        return fabs(ans);
    }
    int relationcircle(circle c){
        getline();
        int i,x=2;
        if (relationpoint(c.p)!=1)return 0;
        for (i=0;i<n;i++){
            if (c.relationseg(l[i])==2)return 0;
            if (c.relationseg(l[i])==1)x=1;
        }
        return x;
    }
    void find(int st,point tri[],circle &c){
        if (!st) c=circle(point(0,0),-2);
        if (st==1) c=circle(tri[0],0);
        if (st==2)
c=circle(tri[0].add(tri[1]).div(2),tri[0].distance(tri[1])/
2.0);
        if (st==3) c=circle(tri[0],tri[1],tri[2]);  }
```

```
    void solve(int cur,int st,point tri[],circle &c){
        find(st,tri,c);
        if (st==3)return;
        int i;
        for (i=0;i<cur;i++){
            if (dblcmp(p[i].distance(c.p)-c.r)>0){
                tri[st]=p[i];
                solve(i,st+1,tri,c);
            }
        }
    }
    circle mincircle(){
        random_shuffle(p,p+n);
        point tri[4];
        circle c;
        solve(n,0,tri,c);
        return c;   }    };
struct point3{
    double x,y,z;
    point3(){  }
    point3(double _x,double _y,double _z):
    x(_x),y(_y),z(_z)   {  };
    bool operator==(point3 a){
        return dblcmp(a.x-x)==0&&dblcmp(a.y-
y)==0&&dblcmp(a.z-z)==0;
    }
    bool operator<(point3 a)const {
        return dblcmp(a.x-x)==0?dblcmp(y-a.y)==0?
dblcmp(z-a.z)<0:y<a.y:x<a.x;
    }
    double len(){ return sqrt(len2()); }
    double len2(){return x*x+y*y+z*z; }
    double distance(point3 p){
        return sqrt((p.x-x)*(p.x-x)+(p.y-y)*(p.y-y)+
(p.z-z)*(p.z-z));
    }
    point3 add(point3 p){   return
point3(x+p.x,y+p.y,z+p.z);}
point3 sub(point3 p){return point3(x-p.x,y-p.y,z-p.z);}
point3 mul(double d){return point3(x*d,y*d,z*d);}
    point3 div(double d){return point3(x/d,y/d,z/d);}
    double dot(point3 p){return x*p.x+y*p.y+z*p.z;}
    point3 det(point3 p){
 return point3(y*p.z-p.y*z,p.x*z-x*p.z,x*p.y-p.x*y);
    }
    double rad(point3 a,point3 b){
        point3 p=(*this);
        return acos(a.sub(p).dot(b.sub(p))/
(a.distance(p)*b.distance(p)));
    }
    point3 trunc(double r){
        r/=len();
        return point3(x*r,y*r,z*r);
    }
};
struct line3{
    point3 a,b;
    line3(){  }
    line3(point3 _a,point3 _b){a=_a;b=_b;}
    bool operator==(line3 v){  return
(a==v.a)&&(b==v.b);    }
    double length(){return a.distance(b);}
    bool pointonseg(point3 p){
        return
dblcmp(p.sub(a).det(p.sub(b)).len())==0&&dblcmp(a.sub(p)
.dot(b.sub(p)))<=0;
    }
    double dispointtoline(point3 p){
        return
b.sub(a).det(p.sub(a)).len()/a.distance(b);
    }
    double dispointtoseg(point3 p){
        if (dblcmp(p.sub(b).dot(a.sub(b)))<0||
dblcmp(p.sub(a).dot(b.sub(a)))<0){
            return min(p.distance(a),p.distance(b));
        }
        return dispointtoline(p);
    }
    point3 lineprog(point3 p){
        return
a.add(b.sub(a).trunc(b.sub(a).dot(p.sub(a))/b.distance(a)))
;
    }
    point3 rotate(point3 p,double ang){
        if
(dblcmp((p.sub(a).det(p.sub(b)).len()))==0)return p;
        point3 f1=b.sub(a).det(p.sub(a));
        point3 f2=b.sub(a).det(f1);
        double
len=fabs(a.sub(p).det(b.sub(p)).len()/a.distance(b));
        f1=f1.trunc(len);f2=f2.trunc(len);
        point3 h=p.add(f2);
        point3 pp=h.add(f1);
        return
h.add((p.sub(h)).mul(cos(ang*1.0))).add((pp.sub(h)).mul(sin
(ang*1.0)));
    }
};
```

POINT IN A POLYGON O(nlgn):

```
typedef complex<long long> pnt;
template<typename T>int size(const T &t){
    return t.size();
}
LL dot(const pnt &a, const pnt &b){
    return real(conj(a) * b);
}
LL cross(const pnt &a, const pnt &b){
    return imag(conj(a) * b);
}
LL cross(const pnt &a, const pnt &b, const pnt &c){
    return cross(b - a, c - a);
}
struct compare_xy{
    bool operator()(const pnt &a, const pnt &b) const{
        if (a.imag() != b.imag())
            return a.imag() < b.imag();
        else
            return a.real() < b.real();
    }
};
static inline int wrap(int x, int H){
    return x >= H ? x - H : x;
}
static int score(const vector<pnt> &h, vector<pnt> &s){
    int H = size(h); int bl = 0;
    int tr = max_element(h.begin(), h.end(), compare_xy())
- h.begin();
    int br = bl; int tl = tr;
    if (h[bl].imag() == h[tr].imag()){
        br = tr;
        tl = bl;
    }
    else{
      while (h[br + 1].imag() == h[bl].imag())br++;
        while (h[wrap(tl + 1, H)].imag() == h[tr].imag())
tl = wrap(tl + 1, H);
    }
    sort(s.begin(), s.end(), compare_xy());
    int r = 0; int l = h.size() - 1; int total = 0;
    for (size_t i = 0; i < s.size(); i++){
        const pnt &p = s[i];
        bool hit = false;
        if (p.imag() >= h[0].imag() && p.imag() <=
   h[tr].imag()){
            if (p.imag() == h[bl].imag())
```

```
                hit = p.real() >= h[bl].real() &&
p.real() <= h[br].real();
            else if (p.imag() == h[tr].imag())
                hit = p.real() >= h[tl].real() &&
p.real() <= h[tr].real();
            else {
           while (p.imag() >= h[r + 1].imag()) r++;
                while (p.imag() >= h[l].imag())l--;
      hit = (cross(h[r], h[wrap(r + 1, H)], p) >= 0
      && cross(h[l], h[wrap(l + 1, H)], p) >= 0);
            }
        }
        total += hit;
    }
    return total;
}
```

3d geo:

```
Point closest_pt_iline(Point a, Point b, Point p) {
  double along = dot(b-a,p-a)/lng2(b-a);
  return (b-a)*along + a;
}
Point closest_pt_seg(Point a, Point b, Point p) {
  double along;
  if (lng2(b-a) < EPS) return a;
  along = dot(b-a,p-a)/lng2(b-a);
  if (along < 0) along = 0;
  if (along > 1) along = 1;
  return (b-a)*along + a;
}
/* Planes are commonly represented in one of two ways:
   as a vector normal (perpendicular) to the plane and a
   single point on the plane, or as three non-collinear
   points on the plane. */
Point closest_pt_plane(Point norm, Point a, Point p) {
  Point res = cross(cross(norm,p-a),norm);
  if (lng2(res) < EPS) return a;
  return res*dot(res,p-a)/lng2(res);
}
Point closest_pt_plane(Point a, Point b, Point c, Point
p) {
  Point norm;
  norm = cross(b-a,c-a);
  assert(lng2(norm) > EPS);  // collinearity
  return closest_pt_plane(norm,a,p);
}
/* Given a sphere and an infinite line, determine if the
two
   intersect, and if so, find the points of
intersection.  If the
   line is tangent to the sphere, the points will be the
same */
int sphere_iline_isect(Point c, double r, Point a, Point
b,
                       Point *p, Point *q) {
  Point vec, mid = closest_pt_iline(a,b,c);
  if (lng2(c-mid) > r*r) return 0;
  vec = (a-b)*sqrt((r*r - lng2(c-mid))/lng2(a-b));
  *p = mid + vec;
  *q = mid - vec;
  return 1;
}
/* Often we are given a set of points in 3D space, and
we know
   that they all lie in a certain plane.  We can
translate
   that problem into a 2D problem using this code.
   Select three of the points to define your plane.  The
points
   must be distinct and not collinear.  This code will
find the
   position of another point, p, in reference to an
arbitrary,
   but deterministic set of axes defined by the first
three points.
   This code just assumes that the points are actually
coplanar. */
Point to_plane(Point a, Point b, Point c, Point p) {
  Point norm, ydir, xdir, res;
  norm = cross(b-a,c-a);
  assert(lng2(norm) > EPS);  // collinearity
  xdir = (b-a)/lng(b-a); // create orthonormal vectors
  ydir = cross(norm,xdir);
  ydir = ydir/lng(ydir);
  res.x = dot(p-a,xdir);
  res.y = dot(p-a,ydir);
  res.z = 0;
  return res;
}
/* given two lines in 3D space, find distance of closest
approach */
double line_line_dist(Point a, Point b, Point c, Point d)
{
  Point perp = cross(b-a,d-c);
  if (lng2(perp) < EPS)  /* parallel */
    perp = cross(b-a,cross(b-a,c-a));
  if (lng2(perp) < EPS) return 0; /* coincident */
  return fabs(dot(a-c,perp))/lng(perp);
}
/* this is the same as line_line_dist, but it also returns
   the points of closest approach */
double closest_approach(Point a, Point b, Point c, Point d,
                        Point *p, Point *q) {
  double s = dot(d-c,b-a), t = dot(a-c,d-c);
  double num, den, tmp;
  den = lng2(b-a)*lng2(d-c) - s*s;
  num = t*s - dot(a-c,b-a)*lng2(d-c);
  if (fabs(den) < EPS) { /* parallel */
    *p = a;
    *q = (d-c)*t/lng2(d-c) + c;
    if (fabs(s) < EPS) *q = a;  /* coincident */
  } else {  /* skew */
    tmp = num/den;
    *p = a + (b-a)*tmp;
    *q = c + (d-c)*(t + s*tmp)/lng2(d-c);
  }
  return lng(*p-*q);
}
/* is the point p on the infinite line ab? */
int on_iline(Point a, Point b, Point p) {
  return (lng2(p-closest_pt_iline(a,b,p)) < EPS);
}
/* is the point p on the segment ab? */
int on_seg(Point a, Point b, Point p) {
  return (lng(a-p) + lng(p-b) - lng(a-b) < EPS);
}
/* Given a plane and a line ab, determine if the two
intersect,
   and if so, find the single point of intersection */
int plane_iline_isect(Point norm, Point ori, Point a,
Point b, Point *p) {
  double along, den = dot(norm,b-a);
  if (fabs(den) < EPS) { /* parallel */
    if (lng2(cross(ori-a,b-a)) < EPS) return -1; /*
coincident */
    return 0;  /* non-intersecting */
  }
  along = dot(norm,ori-a)/den;
  /* if you want to intersect a plane with a finite
segment,
     check that (along <= 1 && along => 0) */
  *p = a + along*(b-a);
  return 1;
}
```

Polar angle of points:

```
double polarAngle( P p ){
    if( fabs( p.x ) <= EPS && fabs( p.y ) <= EPS ) return
```

```
-1.0;
    if( fabs( p.x ) <= EPS ) return ( p.y > EPS ? 1.0 :
3.0 ) * acos( 0 );
    double theta = atan( 1.0 * p.y / p.x );
    if( p.x > EPS ) return( p.y >= -EPS ? theta : ( 4 *
acos( 0 ) + theta ) );
    return( 2 * acos( 0 ) + theta );
}
```

Miscellinous

Gomory-hu tree:

```
bool same[MAXV];
pair<vii, vvi> construct_gh_tree(flow_network &g) {
    int n = g.n, v;
    vii par(n, ii(0, 0)); vvi cap(n, vi(n, -1));
    for (int s = 1; s < n; s++) {
        int l = 0, r = 0;
        par[s].second = g.max_flow(s, par[s].first,
false);
        memset(d, 0, n * sizeof(int));
        memset(same, 0, n * sizeof(int));
        d[q[r++] = s] = 1;
        while (l < r) {
            same[v = q[l++]] = true;
            for (int i = g.head[v]; i != -1; i =
g.e[i].nxt)
                if (g.e[i].cap > 0 && d[g.e[i].v] == 0)
                d[q[r++] = g.e[i].v] = 1;
        }
        for (int i = s + 1; i < n; i++)
        if (par[i].first == par[s].first && same[i])
par[i].first = s;
        g.reset();
    }
    for (int i = 0; i < n; i++) {
        int mn = INF, cur = i;
        while (true) {
            cap[cur][i] = mn;
            if (cur == 0) break;
            mn = min(mn, par[cur].second), cur =
par[cur].first;
        }
    }
    return make_pair(par, cap);
}
int compute_max_flow(int s, int t, const pair<vii, vvi>
&gh) {
    if (s == t) return 0;
    int cur = INF, at = s;while (gh.second[at][t] == -1)
        cur = min(cur, gh.first[at].second), at =
gh.first[at].first;
    return min(cur, gh.second[at][t]);
}
```

FFT-mod:

```
/*
say the first array a and second array b. their len is
n. and the result will be posted in c.
so call conv(a,b,c,n); */
const int MAX=1000000+10;
const int Mod=1711276033;
const int alp=29;
namespace opt{
    int N;
    int a[MAX],b[MAX],w[MAX],tmp[MAX];
    int Pow(int a,int b){
        int k=1;
        for(;b;b/=2,a=(LL)a*a%Mod)
            if(b%2==1)
                k=(LL)k*a % Mod;
        return k;
    }
    void FFT(int *a,int S,int d){
        if(d==N) return ;
        FFT(a,S,d<<1);
        FFT(a,S+d,d<<1);
        for(int i=S,j=S,p=0;i<N/2;i+=d,j+=d<<1,p+=d){
            int G=a[j],K=a[j+d];
            tmp[i]=((LL)G+(LL)w[p]*K)%Mod;
            tmp[i+N/2]=((LL)G+(LL)w[p+N/2]*K)%Mod;
        }
        for(int i=S;i<N;i+=d) a[i]=tmp[i];
    }
    void conv(int *aa,int *bb,int *c,int n){
        for(int i=0;i<n;++i)
            a[i]=aa[i],b[i]=bb[i];
        if(n<=500){
            for(int i=0;i<n*2;++i)
                c[i]=0;
            for(int i=0;i<n;++i)
                for(int j=0;j<n;++j)
                    c[i+j]=( c[i+j]+(LL)a[i]*b[j] ) % Mod;
            for(int i=0;i<n;++i)
                a[i]=b[i]=0;
            return;
        }
        N=1;
        for(;N<n;N<<=1);
        N<<=1;
        w[0]=1;
        w[1]=Pow(alp, (Mod - 1) / N);
        for(int i=2;i<=N;++i)
            w[i]=(LL)w[i-1]*w[1]%Mod;
        FFT(a,0,1);
        FFT(b,0,1);
        for(int i=0;i<N;++i)
            a[i]=(LL)a[i]*b[i]%Mod;
        for(int i=0;i<N;++i)
            b[i]=a[i?(N-i):0];
        FFT(b,0,1);
        int InvN=Pow(N,Mod-2);
        for(int i=0;i<N;++i)
        {
            c[i]=(LL)b[i]*InvN%Mod;
            a[i]=b[i]=0;
        }
    }
} using namespace opt;
/*
used for calculating root of the mod value
*/
    vector<int> divisors(int p) {
        int q = p - 1;
        vector<int> div;
        for (int j = 2; j * j <= q; j++) {
            if (q % j == 0) {
                div.pb(j);
                while (q % j == 0)
                    q /= j;
            }
        }
        if (q != 1) div.pb(q);
        return div;
    }
    bool check(int e, int p, vector<int> divisor) {
        for (auto &d : divisor) {
          if (bigmod((LL)e, (LL)((p - 1) / d), (LL)p) == 1)
                return false;
        }
        return true;
    }
    int getRoot(int p) {
        int e = 2;
        vector < int > divisor = divisors(p);
        while (!check(e, p, divisor))
            e++;
        return e;
```

```
    }
```

Pythagorean triplets:

```
struct { int a, b, c; } ptrip;
int gen_triples(int n, ptrip *res) {
    int a, b, p, q, cnt = 0;
    for (p = 2; p*p < n; p++) {
        for (q = 1+p%2; q < p && p*p+q*q < n; q += 2) {
            if (gcd(p,q)!=1) continue ;
            a = p*p-q*q;
            b = 2*p*q;
         if (a < b) { res[cnt].a = a; res[cnt].b = b; }
         else       { res[cnt].a = b; res[cnt].b = a; }
            res[cnt++].c = p*p+q*q;
        }   }
    return cnt;
}
```

TRIANGLE: Circum Radius = a*b*c/(4*area)
In Radius = area/s, where s = (a+b+b)/2
length of median to side c = sqrt(2*(a*a+b*b)-c*c)/2
length of bisector of angle C = sqrt(ab[(a+b)*(a+b)-c*c])/(a+b)

ELLIPSE: Area = PI*a*b
Circumference = 4a *int(0,PI/2){sqrt(1-(k*sint)*(k*sint))}dt = 2*PI*sqrt((a*a+b*b)/2)
approx where k = sqrt((a*a-b*b)/a)
= PI*(3*(r1+r2)-sqrt[(r1+3*r2)*(3*r1+r2)])

SPHERICAL CAP: V = (1/3)*PI*h*h*(3*r-h)
Surface Area = 2*PI*r*h

SPHERICAL SECTOR: V = (2/3)*PI*r*r*h

SPHERICAL SEGMENT: V = (1/6)*PI*h*(3*a*a+3*b*b+h*h)

TORUS: V = 2*PI*PI*R*r*r

TRUNCATED CONIC: V = (1/3)*PI*h*(a*a+a*b+b*b)
Surface Area = PI*(a+b)*sqrt(h*h+(b-a)*(b-a))
= PI*(a+b)*l

PYRAMIDAL FRUSTRAM: (1/3)*h*(A1+A2+sqrt(A1*A2))
Two big primes: 1000000000000000507, 1000000000000000523

ALPHA-BETA PRUNING:

```
function alphabeta(node, depth, α, β, maximizingPlayer)
  if depth = 0 or node is a terminal node
       return the heuristic value of node
  if maximizingPlayer
       v := -∞
       for each child of node
           v := max(v, alphabeta(child, depth - 1, α, β,
FALSE))
           α := max(α, v)
           if β ≤ α
               break (* β cut-off *)
       return v
   else
       v := ∞
       for each child of node
           v := min(v, alphabeta(child, depth - 1, α, β,
TRUE))
           β := min(β, v)
           if β ≤ α
               break (* α cut-off *)
    return v
```

(Initial call *)*
alphabeta(origin, depth, -∞, +∞, TRUE)

number of lattice points in between a segment of (x1, y1) and (x2,y2) is gcd(abs(x1-x2), abs(y1-y2)) + 1.

a^(b^c) % m, gcd(a, m) = 1, first get k = (b^c) % phi(m) then a^(k) % m;

derangements : dr[i] = (i - 1) * (dr[i - 1] + dr[i - 2]); dr[0] = dr[2] = 1; dr[1] = 0;

number of ways to ditribute n chocolate among r persons is (n+r-1)C(r-1)

number of divisors :

10^5 -> 200

10^12 -> 6000

10^18 -> 100000

given n integers, find for x how many integers satisfy (Ai & x = x) => (0<=k<bit(n)) F[k][x] = F[k-1][x] + F[k-1][x | 1<<k]; if k th bit is 0 F[k][x] = F[k-1][x]; otherwise

Number of primitive root is phi(phi(n)).

MIN COST GRAPH MATCHING:

```
const int MAXN = 110;
const int INF = 0x3f3f3f3f;
int G[MAXN][MAXN]; int cnt_node;
int dis[MAXN]; int match[MAXN];
bool vis[MAXN]; int sta[MAXN],top;
bool dfs(int u){
    sta[top++] = u;
    if(vis[u])return true;
    vis[u] = true;
    for(int i = 0;i < cnt_node;i++)
        if(i != u && i != match[u] && !vis[i]){
            int t = match[i];
            if(dis[t] < dis[u] + G[u][i] - G[i][t]){
                dis[t] = dis[u] + G[u][i] - G[i][t];
                if(dfs(t))return true;
            }
        }
    top--;
    vis[u] = false;
    return false;
}
int P[MAXN];
int get_Match(int N){
    cnt_node = N;
    for(int i = 0;i < cnt_node;i++)P[i] = i;
    for(int i = 0;i < cnt_node;i += 2){
        match[i] = i+1;
        match[i+1] = i;
    }
    int cnt = 0;
    while(1){
        memset(dis,0,sizeof(dis));
        memset(vis,false,sizeof(vis));
        top = 0;
        bool update = false;
        for(int i = 0;i < cnt_node;i++)
            if(dfs(P[i])){
                update = true;
                int tmp = match[sta[top-1]];
                int j = top-2;
                while(sta[j] != sta[top-1]){
                    match[tmp] = sta[j];
                    swap(tmp,match[sta[j]]);
                    j--;
                }
                match[tmp] = sta[j];
                match[sta[j]] = tmp;
                break;
            }
        if(!update){
            cnt++;
            if(cnt >= 3)break;
            random_shuffle(P,P+cnt_node);
        }
    }
    int ans = 0;
    for(int i = 0;i < cnt_node;i++){
        int v = match[i];
        if(i < v)ans += G[i][v];
    }
    return ans;
}
int g[MAXN][MAXN]; // G[i][j] = -g[i][j]
```

Directed MST:

```
const int inf = 100000000 + 10;
struct edge{
    int u, v, w;
    edge() {}
    edge(int a,int b,int c) : u(a), v(b), w(c) {}
```

```
    bool operator < (const edge& o) const{
        if (u == o.u)
            if (v == o.v)
                return w < o.w;
            else
                return v < o.v;
        return u < o.u;
    }
};
vector<int>res;
vector<pair<pair<int,int>,int> >VVV;
int EDGE[10004];
int dmst(vector<edge> &edges, int root, int n){
    res.clear(); VVV.clear();
    int ans = 0;
    int cur_nodes = n;
    while (true){
   vector<int> lo(cur_nodes, inf), pi(cur_nodes, inf);
            for (int i = 0; i < edges.size(); ++i){
   int u = edges[i].u, v = edges[i].v, w = edges[i].w;
            if (w < lo[v] && u != v){
                lo[v] = w; pi[v] = u;
                EDGE[v]=i;
            }
        }
        lo[root] = 0;
        for (int i = 0; i < lo.size(); ++i){
            if (i == root) continue;
            if (lo[i] == inf) return -1;
        }
        int cur_id = 0;
    vector<int> id(cur_nodes, -1), mark(cur_nodes, -1);
        for (int i = 0; i < cur_nodes; ++i){
            ans += lo[i];
            res.push_back(EDGE[i]);
            if(lo[i]>0)
VVV.push_back(make_pair(make_pair(pi[i],i),lo[i]));
            int u = i;
        while (u != root && id[u] < 0 && mark[u] != i){
                mark[u] = i; u = pi[u];
            }
            if (u != root && id[u] < 0){   // Cycle
                for (int v = pi[u]; v != u; v = pi[v])
                    id[v] = cur_id;
                id[u] = cur_id++;
            }
        }
        if (cur_id == 0)break;
        for (int i = 0; i < cur_nodes; ++i)
            if (id[i] < 0) id[i] = cur_id++;
        for (int i = 0; i < edges.size(); ++i){
   int u = edges[i].u, v = edges[i].v, w = edges[i].w;
            edges[i].u = id[u];
            edges[i].v = id[v];
            if (id[u] != id[v])
                edges[i].w -= lo[v];
        }
        cur_nodes = cur_id;
        root = id[root];
    }
    return ans;
}
```

Tips:

1. When all else fails run randomized algo or bactrack or sqrt decomposition or hashing.
2. Shanks solution checking: b=a^k mod p, iff order(a) %order[b] == 0.
3. FFT can and will be usefull.
4. Another divide and conquer strategy is matrix expo like dp.
5. For bug fixing look out for regular mistakes made in the previous contests.

TEMPLATE:

```
#define sz(a) ((int)a.size())
#define repv(i,a) for(int i=0;i<sz(a);i++)
#define revv(i,a) for(int i=sz(a)-1;i>=0;i--)
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define rev(i,a,b) for(int i=a;i>=b;i--)
#define FOR(I,A,B) for(int I = (A); I < (B); ++I)
#define REP(I,N)   FOR(I,0,N)
#define all(a) a.begin(),a.end()
#define rall(a) a.rbegin(),a.rend()
#define pb(a) push_back(a)
#define mp(a,b) make_pair(a,b)
#define pi (2.0*acos(0.0))
#define PI (2.0*acos(0.0))
#define SET(a) memset(a,-1,sizeof a)
#define CLR(a) memset(a,0,sizeof a)
#define set0(ar)   memset(ar,0,sizeof ar)
#define setinf(ar) memset(ar,126,sizeof ar)
#define in(a,x,y) (a>=x && a<=y)
#define out(a,x,y) (!in(a,x,y))
#define eq(a,b) (fabs(a-b)<eps)
#define less(a,b) (a+eps<b)
#define great(a,b) (a>b+eps)
#define xx first
#define yy second
#define root 1
#define MAX(a) (*max_element(all(a)))
#define MIN(a) (*min_element(all(a)))
#define LB(a,x) (lower_bound(all(a),x)-a.begin())
#define UB(a,x) (upper_bound(all(a),x)-a.begin())

using namespace std;
typedef long long ll;
typedef long long LL;
typedef unsigned long long ull;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef vector<ll> vll;
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
typedef vector<pii> vii;

template< class T > inline T _sq(T a) { return a * a; }
template< class T, class X > inline T _pow(T a,X y) {T
z=1; rep(i,1,y){z*=a;} return z; }
template< class T > inline T _gcd(T a,T b)
{a=_abs(a);b=_abs(b); if(!b) return a; return _gcd(b,a%b);}
template< class T > inline T _lcm(T a,T b)
{a=_abs(a);b=_abs(b); return (a/_gcd(a,b))*b;}

template< class T, class X > inline bool getbit(T a, X i)
{ T t=1; return ((a&(t<<i))>0); }
template< class T, class X > inline T setbit(T a, X i) { T
t=1;return (a|(t<<i)); }
template< class T, class X > inline T resetbit(T a, X i) {
T t=1;return (a&(~(t<<i))); }
template< class T, class X > inline T togglebit(T a, X i)
{ T t=1;return (a^(t<<i)); }

template< class T,class X, class Y > inline T _bigmod(T
n,X m,Y mod){ull ret=1, a = n%mod ; while(m){
if(m&1)ret=(ret*a)%mod; m>>=1; a=(a*a)%mod; }ret
%=mod;return (T)ret;}
template< class T, class Y > inline T _modinv(T n,Y mod)
{return _bigmod(n,mod-2,mod);}
```

Theoretical Computer Science Cheat Sheet

Definitions

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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"

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

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

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

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

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

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

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

Note that

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

Subtracting we find

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

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

Generating functions:

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

Example:

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

Multiply and sum:

$$\sum_{i\ge0} g_{i+1}x^i = \sum_{i\ge0} 2g_i x^i + \sum_{i\ge0} x^i.$$

We choose $G(x) = \sum_{i\ge0} x^i g_i$. Rewrite in terms of $G(x)$:

$$\frac{G(x) - g_0}{x} = 2G(x) + \sum_{i\ge0} x^i.$$

Simplify:

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

Solve for $G(x)$:

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

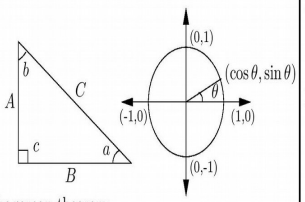
Expand this using partial fractions:

$$G(x) = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)$$
$$= x\left(2\sum_{i\ge0}2^i x^i - \sum_{i\ge0}x^i\right)$$
$$= \sum_{i\ge0}(2^{i+1} - 1)x^{i+1}.$$

So $g_i = 2^i - 1$.

Theoretical Computer Science Cheat Sheet

Trigonometry



Pythagorean theorem:

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

Definitions:

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

Area, radius of inscribed circle:

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

Identities:

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

Euler's equation:

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

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

Matrices

Multiplication:

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

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

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

2×2 and 3×3 determinant:

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

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

Permanents:

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

Hyperbolic Functions

Definitions:

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

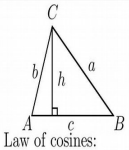
Identities:

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

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:

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

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

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

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

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

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

Fermat's theorem:

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

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

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

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

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

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

Wilson's theorem: n is a prime iff

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

Möbius inversion:

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

If

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

then

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

Prime numbers:

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

Graph Theory

Definitions:

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

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

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

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

Any planar graph has a vertex with degree ≤ 5.

Notation:

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

Geometry

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

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

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

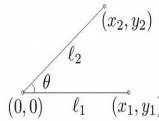
Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[\lvert x_1 - x_0 \rvert^p + \lvert y_1 - y_0 \rvert^p\right]^{1/p},$$
$$\lim_{p \to \infty} \left[\lvert x_1 - x_0 \rvert^p + \lvert y_1 - y_0 \rvert^p\right]^{1/p}.$$

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

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

Angle formed by three points:



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

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

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

Area of circle, volume of sphere:

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

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

Theoretical Computer Science Cheat Sheet

π

Wallis' identity:

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

Brouncker's continued fraction expansion:

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

Gregrory's series:

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

Newton's series:

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

Sharp's series:

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

Euler's series:

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

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

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

Partial Fractions

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

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

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

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

where

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

For a repeated factor:

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

where

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

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

Calculus

Derivatives:

1. $\frac{d(cu)}{dx} = c\frac{du}{dx}$,
2. $\frac{d(u+v)}{dx} = \frac{du}{dx} + \frac{dv}{dx}$,
3. $\frac{d(uv)}{dx} = u\frac{dv}{dx} + v\frac{du}{dx}$,
4. $\frac{d(u^n)}{dx} = nu^{n-1}\frac{du}{dx}$,
5. $\frac{d(u/v)}{dx} = \frac{v\left(\frac{du}{dx}\right) - u\left(\frac{dv}{dx}\right)}{v^2}$,
6. $\frac{d(e^{cu})}{dx} = ce^{cu}\frac{du}{dx}$,
7. $\frac{d(c^u)}{dx} = (\ln c)c^u\frac{du}{dx}$,
8. $\frac{d(\ln u)}{dx} = \frac{1}{u}\frac{du}{dx}$,
9. $\frac{d(\sin u)}{dx} = \cos u\frac{du}{dx}$,
10. $\frac{d(\cos u)}{dx} = -\sin u\frac{du}{dx}$,
11. $\frac{d(\tan u)}{dx} = \sec^2 u\frac{du}{dx}$,
12. $\frac{d(\cot u)}{dx} = \csc^2 u\frac{du}{dx}$,
13. $\frac{d(\sec u)}{dx} = \tan u \sec u\frac{du}{dx}$,
14. $\frac{d(\csc u)}{dx} = -\cot u \csc u\frac{du}{dx}$,
15. $\frac{d(\arcsin u)}{dx} = \frac{1}{\sqrt{1-u^2}}\frac{du}{dx}$,
16. $\frac{d(\arccos u)}{dx} = \frac{-1}{\sqrt{1-u^2}}\frac{du}{dx}$,
17. $\frac{d(\arctan u)}{dx} = \frac{1}{1+u^2}\frac{du}{dx}$,
18. $\frac{d(\operatorname{arccot} u)}{dx} = \frac{-1}{1+u^2}\frac{du}{dx}$,
19. $\frac{d(\operatorname{arcsec} u)}{dx} = \frac{1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
20. $\frac{d(\operatorname{arccsc} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
21. $\frac{d(\sinh u)}{dx} = \cosh u\frac{du}{dx}$,
22. $\frac{d(\cosh u)}{dx} = \sinh u\frac{du}{dx}$,
23. $\frac{d(\tanh u)}{dx} = \operatorname{sech}^2 u\frac{du}{dx}$,
24. $\frac{d(\coth u)}{dx} = -\operatorname{csch}^2 u\frac{du}{dx}$,
25. $\frac{d(\operatorname{sech} u)}{dx} = -\operatorname{sech} u \tanh u\frac{du}{dx}$,
26. $\frac{d(\operatorname{csch} u)}{dx} = -\operatorname{csch} u \coth u\frac{du}{dx}$,
27. $\frac{d(\operatorname{arcsinh} u)}{dx} = \frac{1}{\sqrt{1+u^2}}\frac{du}{dx}$,
28. $\frac{d(\operatorname{arccosh} u)}{dx} = \frac{1}{\sqrt{u^2-1}}\frac{du}{dx}$,
29. $\frac{d(\operatorname{arctanh} u)}{dx} = \frac{1}{1-u^2}\frac{du}{dx}$,
30. $\frac{d(\operatorname{arccoth} u)}{dx} = \frac{1}{u^2-1}\frac{du}{dx}$,
31. $\frac{d(\operatorname{arcsech} u)}{dx} = \frac{-1}{u\sqrt{1-u^2}}\frac{du}{dx}$,
32. $\frac{d(\operatorname{arccsch} u)}{dx} = \frac{-1}{|u|\sqrt{1+u^2}}\frac{du}{dx}$.

Integrals:

1. $\int cu\,dx = c\int u\,dx$,
2. $\int (u+v)\,dx = \int u\,dx + \int v\,dx$,
3. $\int x^n\,dx = \frac{1}{n+1}x^{n+1}, \quad n \neq -1$,
4. $\int \frac{1}{x}dx = \ln x$,
5. $\int e^x\,dx = e^x$,
6. $\int \frac{dx}{1+x^2} = \arctan x$,
7. $\int u\frac{dv}{dx}dx = uv - \int v\frac{du}{dx}dx$,
8. $\int \sin x\,dx = -\cos x$,
9. $\int \cos x\,dx = \sin x$,
10. $\int \tan x\,dx = -\ln|\cos x|$,
11. $\int \cot x\,dx = \ln|\cos x|$,
12. $\int \sec x\,dx = \ln|\sec x + \tan x|$,
13. $\int \csc x\,dx = \ln|\csc x + \cot x|$,
14. $\int \arcsin\frac{x}{a}dx = \arcsin\frac{x}{a} + \sqrt{a^2-x^2}, \quad a > 0$,

Theoretical Computer Science Cheat Sheet

Calculus Cont.

15. $\int \arccos\frac{x}{a}dx = \arccos\frac{x}{a} - \sqrt{a^2-x^2}, \quad a > 0$,
16. $\int \arctan\frac{x}{a}dx = x\arctan\frac{x}{a} - \frac{a}{2}\ln(a^2+x^2), \quad a > 0$,
17. $\int \sin^2(ax)dx = \frac{1}{2a}\big(ax - \sin(ax)\cos(ax)\big)$,
18. $\int \cos^2(ax)dx = \frac{1}{2a}\big(ax + \sin(ax)\cos(ax)\big)$,
19. $\int \sec^2 x\,dx = \tan x$,
20. $\int \csc^2 x\,dx = -\cot x$,
21. $\int \sin^n x\,dx = -\frac{\sin^{n-1}x\cos x}{n} + \frac{n-1}{n}\int \sin^{n-2}x\,dx$,
22. $\int \cos^n x\,dx = \frac{\cos^{n-1}x\sin x}{n} + \frac{n-1}{n}\int \cos^{n-2}x\,dx$,
23. $\int \tan^n x\,dx = \frac{\tan^{n-1}x}{n-1} - \int \tan^{n-2}x\,dx, \quad n \neq 1$,
24. $\int \cot^n x\,dx = -\frac{\cot^{n-1}x}{n-1} - \int \cot^{n-2}x\,dx, \quad n \neq 1$,
25. $\int \sec^n x\,dx = \frac{\tan x\sec^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \sec^{n-2}x\,dx, \quad n \neq 1$,
26. $\int \csc^n x\,dx = -\frac{\cot x\csc^{n-1}x}{n-1} + \frac{n-2}{n-1}\int \csc^{n-2}x\,dx, \quad n \neq 1$,
27. $\int \sinh x\,dx = \cosh x$,
28. $\int \cosh x\,dx = \sinh x$,
29. $\int \tanh x\,dx = \ln|\cosh x|$,
30. $\int \coth x\,dx = \ln|\sinh x|$,
31. $\int \operatorname{sech} x\,dx = \arctan\sinh x$,
32. $\int \operatorname{csch} x\,dx = \ln\left|\tanh\frac{x}{2}\right|$,
33. $\int \sinh^2 x\,dx = \frac{1}{4}\sinh(2x) - \frac{1}{2}x$,
34. $\int \cosh^2 x\,dx = \frac{1}{4}\sinh(2x) + \frac{1}{2}x$,
35. $\int \operatorname{sech}^2 x\,dx = \tanh x$,
36. $\int \operatorname{arcsinh}\frac{x}{a}dx = x\operatorname{arcsinh}\frac{x}{a} - \sqrt{x^2+a^2}, \quad a > 0$,
37. $\int \operatorname{arctanh}\frac{x}{a}dx = x\operatorname{arctanh}\frac{x}{a} + \frac{a}{2}\ln|a^2-x^2|$,
38. $\int \operatorname{arccosh}\frac{x}{a}dx = \begin{cases} x\operatorname{arccosh}\dfrac{x}{a} - \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a} > 0 \text{ and } a > 0, \\ x\operatorname{arccosh}\dfrac{x}{a} + \sqrt{x^2+a^2}, & \text{if } \operatorname{arccosh}\frac{x}{a} < 0 \text{ and } a > 0, \end{cases}$
39. $\int \frac{dx}{\sqrt{a^2+x^2}} = \ln\left(x + \sqrt{a^2+x^2}\right), \quad a > 0$,
40. $\int \frac{dx}{a^2+x^2} = \frac{1}{a}\arctan\frac{x}{a}, \quad a > 0$,
41. $\int \sqrt{a^2-x^2}\,dx = \frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}, \quad a > 0$,
42. $\int (a^2-x^2)^{3/2}dx = \frac{x}{8}(5a^2-2x^2)\sqrt{a^2-x^2} + \frac{3a^4}{8}\arcsin\frac{x}{a}, \quad a > 0$,
43. $\int \frac{dx}{\sqrt{a^2-x^2}} = \arcsin\frac{x}{a}, \quad a > 0$,
44. $\int \frac{dx}{a^2-x^2} = \frac{1}{2a}\ln\left|\frac{a+x}{a-x}\right|$,
45. $\int \frac{dx}{(a^2-x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2-x^2}}$,
46. $\int \sqrt{a^2 \pm x^2}\,dx = \frac{x}{2}\sqrt{a^2 \pm x^2} \pm \frac{a^2}{2}\ln\left|x + \sqrt{a^2 \pm x^2}\right|$,
47. $\int \frac{dx}{\sqrt{x^2-a^2}} = \ln\left|x + \sqrt{x^2-a^2}\right|, \quad a > 0$,
48. $\int \frac{dx}{ax^2+bx} = \frac{1}{a}\ln\left|\frac{x}{a+bx}\right|$,
49. $\int x\sqrt{a+bx}\,dx = \frac{2(3bx-2a)(a+bx)^{3/2}}{15b^2}$,
50. $\int \frac{\sqrt{a+bx}}{x}dx = 2\sqrt{a+bx} + a\int \frac{1}{x\sqrt{a+bx}}dx$,
51. $\int \frac{x}{\sqrt{a+bx}}dx = \frac{1}{\sqrt{2}}\ln\left|\frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}}\right|, \quad a > 0$,
52. $\int \frac{\sqrt{a^2-x^2}}{x}dx = \sqrt{a^2-x^2} - a\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|$,
53. $\int x\sqrt{a^2-x^2}\,dx = -\frac{1}{3}(a^2-x^2)^{3/2}$,
54. $\int x^2\sqrt{a^2-x^2}\,dx = \frac{x}{8}(2x^2-a^2)\sqrt{a^2-x^2} + \frac{a^4}{8}\arcsin\frac{x}{a}, \quad a > 0$,
55. $\int \frac{dx}{\sqrt{a^2-x^2}} = -\frac{1}{a}\ln\left|\frac{a+\sqrt{a^2-x^2}}{x}\right|$,
56. $\int \frac{x\,dx}{\sqrt{a^2-x^2}} = -\sqrt{a^2-x^2}$,
57. $\int \frac{x^2\,dx}{\sqrt{a^2-x^2}} = -\frac{x}{2}\sqrt{a^2-x^2} + \frac{a^2}{2}\arcsin\frac{x}{a}, \quad a > 0$,
58. $\int \frac{\sqrt{a^2+x^2}}{x}dx = \sqrt{a^2+x^2} - a\ln\left|\frac{a+\sqrt{a^2+x^2}}{x}\right|$,
59. $\int \frac{\sqrt{x^2-a^2}}{x}dx = \sqrt{x^2-a^2} - a\arccos\frac{a}{|x|}, \quad a > 0$,
60. $\int x\sqrt{x^2 \pm a^2}\,dx = \frac{1}{3}(x^2 \pm a^2)^{3/2}$,
61. $\int \frac{dx}{x\sqrt{x^2+a^2}} = \frac{1}{a}\ln\left|\frac{x}{a+\sqrt{a^2+x^2}}\right|$,

Theoretical Computer Science Cheat Sheet

Calculus Cont.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Finite Calculus

Difference, shift operators:

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

Fundamental Theorem:

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

Differences:

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

Sums:

$$\sum cu\,\delta x = c\sum u\,\delta x,$$
$$\sum(u+v)\,\delta x = \sum u\,\delta x + \sum v\,\delta x,$$
$$\sum u\Delta v\,\delta x = uv - \sum \mathrm{E}\,v\Delta u\,\delta x,$$
$$\sum x^{\underline{n}}\delta x = \tfrac{x^{\underline{n+1}}}{m+1}, \qquad \sum x^{\underline{-1}}\delta x = H_x,$$
$$\sum c^x\,\delta x = \tfrac{c^x}{c-1}, \qquad \sum\tbinom{x}{m}\delta x = \tbinom{x}{m+1}.$$

Falling Factorial Powers:

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

Rising Factorial Powers:

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

Conversion:

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

Theoretical Computer Science Cheat Sheet

Series

Taylor's series:

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

Expansions:

$$\begin{aligned}
\frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \cdots &&= \sum_{i=0}^{\infty} x^i, \\
\frac{1}{1-cx} &= 1 + cx + c^2x^2 + c^3x^3 + \cdots &&= \sum_{i=0}^{\infty} c^i x^i, \\
\frac{1}{1-x^n} &= 1 + x^n + x^{2n} + x^{3n} + \cdots &&= \sum_{i=0}^{\infty} x^{ni}, \\
\frac{x}{(1-x)^2} &= x + 2x^2 + 3x^3 + 4x^4 + \cdots &&= \sum_{i=0}^{\infty} ix^i, \\
x^k\frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) &= x + 2^nx^2 + 3^nx^3 + 4^nx^4 + \cdots &&= \sum_{i=0}^{\infty} i^nx^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_ix^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_ix^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_ix^i, \\
\frac{F_nx}{1-(F_{n-1}+F_{n+1})x-(-1)^nx^2} &= F_nx + F_{2n}x^2 + F_{3n}x^3 + \cdots &&= \sum_{i=0}^{\infty}F_{ni}x^i.
\end{aligned}$$

Ordinary power series:

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

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:

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

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

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

Convolution:

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

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