KUET_ThunderBolt Team Notebook
August 11, 2021

Muhiminul Islam Osim, Toufiq Imam Nuhash,
Sharif Minhazul Islam

Contents

1 Dynamic Programming

1.1 CDQ Divide and Conquer [21 lines]

```
void cdq(int l, int r) {
  if(l + 1 == r) return;
  int m = (l + r) >> 1;
  cdq(l, m); cdq(m, r);
  int a = l, b = m, sum = 0;
/*need to record the modifications on BIT in order to
    reset it. The complexity will be O(N^2) if we
    resetting it brute-forcely.*/
  vector<int> record;//temporary array for mergesort
  vector<Pt> tmp;
  while(a < m && b < r) {
    if(v[a].y > v[b].y) bit.update(v[a].z, 1), sum++,
        record.push_back(v[a].z),
        tmp.push_back(v[a++]);
    else ans[v[b].i] += sum - bit.query(v[b].z),
        tmp.push_back(v[b++]);
  }
  while(a < m) tmp.push_back(v[a++]);
  while(b < r) ans[v[b].i] += sum -
      bit.query(v[b].z), tmp.push_back(v[b++]);
  for(int i = l ; i < r ; ++i) v[i] = tmp[i - l];
  // reset BIT
  for(auto i : record) bit.update(i, -1);
  // release used memory
  vector<int> ().swap(record);
  vector<Pt> ().swap(tmp);
}
```

1.2 Convex Hull Trick [91 lines]

```
struct Hull_Static{
/*all m need to be decreasing order
if m is in increasing order then negate the
    m(like,add_line(-m,c)),
remember in query you have to negate the x also*/
  const ll inf=1000000000000000000;
  int min_or_max;//if min then 0 otherwise 1
  int pointer;/*keep track for the best line for
      previous query,requires all insert first*/
  vector<ll>M,C;//y=m*x+c;
  inline void clear(){
    min_or_max=0;//initially with minimum trick
    pointer=0;
    M.clear();
    C.clear();
  }
  Hull_Static(){clear();}
  Hull_Static(int _min_or_max){
    clear();
    this->min_or_max=_min_or_max;
  }
  bool bad_min(int idx1,int idx2,int idx3){
    return(C[idx3]-C[idx1])*(M[idx1]-M[idx2]) <
        (C[idx2]-C[idx1])*(M[idx1]-M[idx3]);
    return 1.0*(C[idx3]-C[idx1])*(M[idx1]-M[idx2]) <
        1.0*(C[idx2]-C[idx1])*(M[idx1]-M[idx3]);//for
        overflow
  }
  bool bad_max(int idx1,int idx2,int idx3){
    return(C[idx3]-C[idx1])*(M[idx1]-M[idx2]) >
        (C[idx2]-C[idx1])*(M[idx1]-M[idx3]);
    return 1.0*(C[idx3]-C[idx1])*(M[idx1]-M[idx2]) >
        1.0*(C[idx2]-C[idx1])*(M[idx1]-M[idx3]);//for
        overflow
  }
  bool bad(int idx1,int idx2,int idx3){
    if(!min_or_max)return bad_min(idx1,idx2,idx3);
    else return bad_max(idx1,idx2,idx3);
  }
  void add_line(ll m,ll c){/*add line where m is given
      in decreasing order
//if(M.size()>0 and M.back()==m)return;//same
    gradient,no need to add
```

```
above line added from tarango khan,this line cost me
    sevaral wa,but some code got ac with this*/
    M.push_back(m);
    C.push_back(c);
    while(M.size()>=3 and bad((int)M.size()-3,
        (int)M.size()-2,(int)M.size()-1)){
      M.erase(M.end()-2);
      C.erase(C.end()-2);
    }
  }
  ll getval(ll idx,ll x){
    return M[idx]*x+C[idx];
  }
  ll getminval(ll x){/*if queries are non-decreasing
      order*/
    while(pointer<(int)M.size()-1 and getval(pointer+
        1,x)<getval(pointer,x))pointer++;
    return M[pointer]*x+C[pointer];
  }
  ll getmaxval(ll x){
    while(pointer<(int)M.size()-1 and getval(pointer+
        1,x)>getval(pointer,x))pointer++;
    return M[pointer]*x+C[pointer];
  }
  ll getminvalternary(ll x){
    ll lo=0,hi=(ll)M.size()-1;
    ll ans=inf;/*change with problem*/
    while(lo<=hi){
      ll mid1=lo+(hi-lo)/3;
      ll mid2=hi-(hi-lo)/3;
      ll val1=getval(mid1,x);
      ll val2=getval(mid2,x);
      if(val1<val2){
        ans=min(ans,val2);
        hi=mid2-1;
      }
      else{
        ans=min(ans,val1);
        lo=mid1+1;
      }
    }
    return ans;
  }
  ll getmaxvalternary(ll x){
    ll lo=0,hi=(ll)M.size()-1;
    ll ans=-inf;/*change with problem*/
    while(lo<=hi){
      ll mid1=lo+(hi-lo)/3;
      ll mid2=hi-(hi-lo)/3;
      ll val1=getval(mid1,x);
      ll val2=getval(mid2,x);
      if(val1<val2){
        ans=max(ans,val2);
        lo=mid1+1;
      }
      else{
        ans=max(ans,val1);
        hi=mid2-1;
      }
    }
    return ans;
  }
};
```

1.3 Divide and Conquer DP [26 lines]

```
ll G,L;///total group,cell size
ll dp[8001][801],cum[8001];
ll C[8001];///value of each cell
inline ll cost(ll l,ll r){
  return(cum[r]-cum[l-1])*(r-l+1);
}
void fn(ll g,ll st,ll ed,ll r1,ll r2){
  if(st>ed)return;
  ll mid=(st+ed)/2,pos=-1;
  dp[mid][g]=inf;
  for(int i=r1;i<=r2;i++){
    ll tcost=cost(i,mid)+dp[i-1][g-1];
    if(tcost<dp[mid][g]){
        dp[mid][g]=tcost,pos=i;
    }
  }
  fn(g,st,mid-1,r1,pos);
  fn(g,mid+1,ed,pos,r2);
}
int main(){
  for(int i=1;i<=L;i++)
    cum[i]=cum[i-1]+C[i];
  for(int i=1;i<=L;i++)
    dp[i][1]=cost(1,i);
  for(int i=2;i<=G;i++)fn(i,1,L,1,L);
}
```

1.4 Grundy-1 [38 lines]

```
/*Initially there are n piles.
A pile is formed by some cells.
Alice starts the game and they alternate turns.
In each tern a player can pick any pile and divide it
    into two unequal piles.
If a player cannot do so,he/she loses the game.*/
#define Size 100005
int N,A[105],DP[10005];
int call(int cur){
  if(cur<=2)return 0;
  if(DP[cur]!=-1)return DP[cur];
  vector<int>grundy;
  for(int d1=1;d1<=cur/2;d1++){
    int d2=cur-d1;
    if(d1!=d2){
      grundy.pb(call(d1)^ call(d2));
    }
  }
  make_unique(grundy);
  if(grundy[0]!=0)return DP[cur]=0;
  int f=0;
  for(int i=1;i<grundy.size();i++){
    if(grundy[i]==f)continue;
    f++;
    if(grundy[i]!=f)return DP[cur]=f;
  }
  return DP[cur]=grundy[grundy.size()-1]+1;
}
int main(){
  mems(DP,-1);
  sf("%d",&N);
  int res=0;
  for(int i=0;i<N;i++){
    sf("%d",&A[i]);
    res=res^call(A[i]);
  }
  if(res==0)pf("Bob\n");
  else pf("Alice\n");
}
```

1.5 Grundy-2 [24 lines]

```
/*Initially there are n piles.
A pile is formed by some stones.
Alice starts the game and they alternate turns.
In each tern a player can pick any pile and remove
    some stones.
At least 1 and at most half of stones on that pile.
```

```
If a player cannot do so,he/she loses the game.*/
#define Size 100005
int N,A[1005],DP[10005];
int call(int cur){
  if(cur%2==0)return cur/2;
  while(cur%2!=0)cur/=2;
  return cur/2;
}
int main(){
  mems(DP,-1);
  sf("%d",&N);
  int res=0;
  for(int i=0;i<N;i++){
    sf("%d",&A[i]);
    res=res ^ call(A[i]);
  }
  if(res==0)pf("Bob\n");
  else pf("Alice\n");
}
```

1.6 Knuth Optimization [32 lines]

```
/*It is applicable where recurrence is in the form :
dp[i][j]=mini<k<j{dp[i][k]+dp[k][j]}+C[i][j]
condition for applicability is:
A[i,j-1]<=A[i,j]<=A[i+1,j]
Where,
A[i][j]-the smallest k that gives optimal answer,like-
dp[i][j]=dp[i-1][k]+C[k][j]
C[i][j]-given cost function
also applicable if: C[i][j]satisfies the following 2
    conditions:
C[a][c]+C[b][d]<=C[a][d]+C[b][c],a<=b<=c<=d
C[b][c]<=C[a][d],a<=b<=c<=d
reduces time complexity from O(n^3)to O(n^2)*/
for(int s=0;s<=k;s++)//s-length(size)of substring
  for(int l=0;l+s<=k;l++){//l-left point
    int r=l+s;//r-right point
    if(s<2){
      res[l][r]=0;//DP base-nothing to break
      mid[l][r]=l;/*mid is equal to left border*/
      continue;
    }
    int mleft=mid[l][r-1];/*Knuth's trick: getting
        bounds on m*/
    int mright=mid[l+1][r];
    res[l][r]=inf;
    for(int m=mleft;m<=mright;m++){/*iterating for m
        in the bounds only*/
      int64 tres=res[l][m]+res[m][r]+(x[r]-x[l]);
      if(res[l][r]>tres){//relax current solution
        res[l][r]=tres;
        mid[l][r]=m;
      }
    }
  }
int64 answer=res[0][k];
```

1.7 LIS O(nlogn) with full path [17 lines]

```
int num[MX],mem[MX],prev[MX],array[MX],res[MX],maxlen;
void LIS(int SZ,int num[]){
  CLR(mem),CLR(prev),CLR(array),CLR(res);
  int i,k;
  maxlen=1;
  array[0]=-inf;
  RFOR(i,1,SZ+1)  array[i]=inf;
  prev[0]=-1,mem[0]=num[0];
  FOR(i,SZ){
    k=lower_bound(array,array+maxlen+1,num[i])-array;
    if(k==1)  array[k]=num[i],mem[k]=i,prev[i]=-1;
    else array[k]=num[i],mem[k]=i,prev[i]=mem[k-1];
    if(k>maxlen)  maxlen=k;
  }
  k=0;
  for(i=mem[maxlen];i!=-1;i=prev[i])res[k++]=num[i];
}
```

1.8 Sibling DP [26 lines]

```
/*/dividing tree into min group such that each group
    cost not exceed k*/
ll n,k,dp[mx][mx];
vector<pair<ll,ll>>adj[mx];///must be rooted tree
ll sibling_dp(ll par,ll idx,ll remk){
  if(remk<0)return inf;
  if(adj[par].size()<idx+1)return 0;
  ll u=adj[par][idx].first;
  if(dp[u][remk]!=-1)
    return dp[u][remk];
  ll ret=inf,under=0,sibling=0;
  if(par!=0){//creating new group
    under=1+dfs(u,0,k);
    sibling=dfs(par,idx+1,remk);
    ret=min(ret,under+sibling);
  }
  //divide the current group
  ll temp=remk-adj[par][idx].second;
  for(ll chk=temp;chk>=0;chk--){
    ll siblingk=temp-chk;
    under=0,sibling=0;
    under=dfs(u,0,chk);
    sibling=dfs(par,idx+1,siblingk);
    ret=min(ret,under+sibling);
  }
  return dp[u][remk]=ret;
}
```

2 Graph

2.1 2-SAT [52 lines]

```
struct TwoSat{
  #define vi vector<int>
  int N;
  vector<vi> gr;
  vi values; // 0 = false, 1 = true

  TwoSat(int n = 0) : N(n), gr(2*n) {}
  int addVar() { // (optional)
    gr.emplace_back();
    gr.emplace_back();
    return N++;
  }
  void either(int f, int j){
    f = max(2*f, -1-2*f);
    j = max(2*j, -1-2*j);
    gr[f].push_back(j^1);
    gr[j].push_back(f^1);
  }
  void setValue(int x) { either(x, x);}
  void atMostOne(const vi& li){//(optional)
    if (sz(li) <= 1) return;
    int cur = ~li[0];
    rep(i,2,sz(li)) {
      int next = addVar();
      either(cur, ~li[i]);
      either(cur, next);
      either(~li[i], next);
      cur = ~next;
    }
    either(cur, ~li[1]);
  }
  vi val, comp, z; int time = 0;
  int dfs(int i){
```

```
    int low = val[i] = ++time, x; z.push_back(i);
    for(int e : gr[i]) if (!comp[e])
      low = min(low, val[e] ?: dfs(e));
    if (low == val[i]) do {
      x = z.back(); z.pop_back();
      comp[x] = low;
      if (values[x>>1] == -1)
        values[x>>1] = x&1;
    } while (x != i);
    return val[i] = low;
  }
  bool solve() {
    values.assign(N, -1);
    val.assign(2*N, 0); comp = val;
    rep(i,0,2*N) if (!comp[i]) dfs(i);
    rep(i,0,N) if (comp[2*i] == comp[2*i+1]) return 0;
    return 1;
  }
};
```

2.2 Articulation Point and Bridge [23 lines]

```
int dfsRoot,rootChildren;
vector<int>dfs_num,dfs_low,dfs_parent,
    articulation_vertex;
int dfsNumberCounter;
vector<pair<int,int>>AdjList[maxn];
void articulationPointAndBridge(int u){
  dfs_low[u]=dfs_num[u]=dfsNumberCounter++;
      //dfs_low[u]<=dfs_num[u]
  for(int j=0;j<(int)AdjList[u].size();j++){
    pair<int,int>v=AdjList[u][j];
    if(dfs_num[v.first]==0){//a tree edge
      dfs_parent[v.first]=u;
      if(u==dfsRoot)
        rootChildren++;//special case if u is a root
      articulationPointAndBridge(v.first);
      if(dfs_low[v.first]>=dfs_num[u])//for
          articulation point
        articulation_vertex[u]=1;/*store this
            information first*/
      if(dfs_low[v.first]>dfs_num[u])//for bridge
        printf("Edge(%d,%d)is a bridge\n",u,v.first);
      dfs_low[u]=min(dfs_low[u],dfs_low[v.first]);
          //update dfs_low[u]
    }
    else if(v.first!=dfs_parent[u])/*a back edge and
        not direct cycle*/
      dfs_low[u]=min(dfs_low[u],dfs_num[v.first]);
          //update dfs_low[u]
  }
}
```

2.3 Bipartite Matching [40 lines]

```
ll n,m;
vector<ll>g[55];
ll lt[55],rt[55];
bool visited[55];
bool dfs(ll u){
  if(visited[u])return false;
  visited[u]=true;
  for(ll i=0;i<g[u].size();i++){
    ll v=g[u][i];
    if(rt[v]==-1){
      rt[v]=u,lt[u]=v;
      return true;
    }
  }
  for(ll i=0;i<g[u].size();i++){
    ll v=g[u][i];
    if(dfs(rt[v])){
      rt[v]=u,lt[u]=v;
      return true;
    }
  }
  return false;
}
ll match(){
  memset(lt,-1,sizeof lt);
  memset(rt,-1,sizeof rt);
  bool done=false;
  while(!done){
    done=true;
    memset(visited,false,sizeof visited);
    for(ll i=1;i<=n;i++){
      if(lt[i]==-1 and dfs(i)){
        done=false;
      }
    }
  }
  ll ret=0;
  for(ll i=1;i<=n;i++)ret+=(int)lt[i]!=-1;
  return ret;
}
```

2.4 Centroid Decomposition [49 lines]

```
ll n,subsize[mx];
vector<ll>adj[mx];
char ans[mx];
bool brk[mx];
void calculatesize(ll u,ll par){
  subsize[u]=1;
  for(ll i=0;i<(ll)adj[u].size();i++){
    ll v=adj[u][i];
    if(v==par or brk[v]==true)continue;
    calculatesize(v,u);
    subsize[u]+=subsize[v];
  }
}
ll getcentroid(ll u,ll par,ll n){
  ll ret=u;
  for(ll i=0;i<(ll)adj[u].size();i++){
    ll v=adj[u][i];
    if(v==par or brk[v]==true)continue;
    if(subsize[v]>(n/2)){
      ret=getcentroid(v,u,n);
      break;
    }
  }
  return ret;
}
void decompose(ll u,char rank){
  calculatesize(u,-1);
  ll c=getcentroid(u,-1,subsize[u]);
  brk[c]=true;
  ans[c]=rank;
  for(ll i=0;i<(ll)adj[c].size();i++){
    ll v=adj[c][i];
    if(brk[v]==true)continue;
    decompose(v,rank+1);
  }
}
int main(){
  scanf("%lld",&n);
  for(ll i=0;i<n-1;i++){
    ll a,b;
    scanf("%lld %lld",&a,&b);
    adj[a].push_back(b);
    adj[b].push_back(a);
  }
  decompose(1,'A');
```

```
  for(ll i=1;i<=n;i++){
    printf("%c",ans[i]);
  }
}
```

2.5 Dinic [61 lines]

```
class edge {
public:
    int a, b, cap, flow;
    edge(int _a, int _b, int _cap, int _flow) {
        a = _a; b = _b, cap = _cap;
        flow = _flow;
    }
};
vector<edge> e; vector<int> g[maxn];
int s, t, d[maxn], q[maxn], ptr[maxn];
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;
    memset(d, -1, sizeof d);
    q[qt++] = s, d[s] = 0;
    while (qh < qt and d[t] == -1) {
        int v = q[qh++];
        for (int i = 0; i < (int)g[v].size(); i++) {
            int id = g[v][i], to = e[id].b;
            if (d[to] == -1 and 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;
    while (1) {
        if (!bfs()) break;
        memset(ptr, 0, sizeof ptr);
        while (int pushed = dfs(s, inf)) {
            flow += pushed;
            if (pushed == 0) break;
        }
    }
    return flow;
}
```

2.6 Directed MST [39 lines]

```
/*Complexity: O(N*M),where N is the number of
    nodes,and M the number of edges*/
struct Edge{int x,y,w;};
int dmst(int N,vector<Edge>E,int root){
  const int oo=1e9;
  vector<int>cost(N),back(N),label(N),bio(N);
  int ret=0;
  for(;;){
    REP(i,N)cost[i]=oo;
    for(auto e : E){
      if(e.x==e.y)continue;
      if(e.w<cost[e.y])cost[e.y]=e.w,back[e.y]=e.x;
    }
    cost[root]=0;
    REP(i,N)if(cost[i]==oo)return-1;
    REP(i,N)ret+=cost[i];
    int K=0;
    REP(i,N)label[i]=-1;
    REP(i,N)bio[i]=-1;
    REP(i,N){
      int x=i;
      for(;x!=root&&bio[x]==-1;x=back[x])bio[x]=i;
      if(x!=root && bio[x]==i){
        for(;label[x]==-1;x=back[x])label[x]=K;
       ++K;
      }
    }
    if(K==0)break;
    REP(i,N)if(label[i]==-1)label[i]=K++;
    for(auto &e : E){
      int xx=label[e.x];
      int yy=label[e.y];
      if(xx!=yy)e.w-=cost[e.y];
      e.x=xx,e.y=yy;
    }
    root=label[root];
    N=K;
  }
  return ret;
}
```

2.7 Heavy Light Decomposition [73 lines]

```
/*Heavy Light Decomposition
Build Complexity O(n)
Query Complexity O(lg^2 n)
Call init()with number of nodes
It's probably for the best to not do"using namespace
    hld"*/
namespace hld {
  //N is the maximum number of nodes
  /*par,lev,size corresponds to
      parent,depth,subtree-size*/
  //head[u]is the starting node of the chain u is in
  //in[u]to out[u]keeps the subtree indices
  const int N=100000+7;
  vector<int>g[N];
  int par[N],lev[N],head[N],size[N],in[N],out[N];
  int cur_pos,n;
  //returns the size of subtree rooted at u
  /*maintains the child with the largest subtree at
      the front of g[u]*/
  //WARNING: Don't change anything here specially with
      size[]if Jon Snow
  int dfs(int u,int p){
    size[u]=1,par[u]=p;
    lev[u]=lev[p]+1;
    for(auto &v : g[u]){
      if(v==p)continue;
      size[u]+=dfs(v,u);
```

```
    if(size[v]>size[g[u].front()]){
      swap(v,g[u].front());
    }
  }
  return size[u];
}
//decomposed the tree in an array
//note that there is no physical array here
void decompose(int u,int p){
  in[u]=++cur_pos;
  for(auto &v : g[u]){
    if(v==p)continue;
    head[v]=(v==g[u].front()? head[u]: v);
    decompose(v,u);
  }
  out[u]=cur_pos;
}
//initializes the structure with _n nodes
void init(int _n,int root=1){
  n=_n;
  cur_pos=0;
  dfs(root,0);
  head[root]=root;
  decompose(root,0);
}
//checks whether p is an ancestor of u
bool isances(int p,int u){
  return in[p]<=in[u]and out[u]<=out[p];
}
//Returns the maximum node value in the path u-v
ll query(int u,int v){
  ll ret=-INF;
  while(!isances(head[u],v)){
    ret=max(ret,seg.query(1,1,n,in[head[u]],in[u]));
    u=par[head[u]];
  }
  swap(u,v);
  while(!isances(head[u],v)){
    ret=max(ret,seg.query(1,1,n,in[head[u]],in[u]));
    u=par[head[u]];
  }
  if(in[v]<in[u])swap(u,v);
  ret=max(ret,seg.query(1,1,n,in[u],in[v]));
  return ret;
}
//Adds val to subtree of u
void update(int u,ll val){
  seg.update(1,1,n,in[u],out[u],val);
}
};
```

2.8 K'th Shortest path [40 lines]

```
int m,n,deg[MM],source,sink,K,val[MM][12];
struct edge{
  int v,w;
}adj[MM][500];
struct info{
  int v,w,k;
  bool operator<(const info &b)const{
    return w>b.w;
  }
};
priority_queue<info,vector<info>>Q;
void kthBestShortestPath(){
  int i,j;
  info u,v;
  for(i=0;i<n;i++)
    for(j=0;j<K;j++)val[i][j]=inf;
  u.v=source,u.k=0,u.w=0;
  Q.push(u);
  while(!Q.empty()){
    u=Q.top();
    Q.pop();
    for(i=0;i<deg[u.v];i++){
      v.v=adj[u.v][i].v;
      int cost=adj[u.v][i].w+u.w;
      for(v.k=u.k;v.k<K;v.k++){
        if(cost==inf)break;
        if(val[v.v][v.k]>cost){
          swap(cost,val[v.v][v.k]);
          v.w=val[v.v][v.k];
          Q.push(v);
          break;
        }
      }
      for(v.k++;v.k<K;v.k++){
        if(cost==inf)break;
        if(val[v.v][v.k]>cost)swap(cost,
            val[v.v][v.k]);
      }
    }
  }
}
```

2.9 LCA [20 lines]

```
void build(){
  for(int i=1;i<LOGN;i++){
    for(int j=0;j<n;j++){
      if(parent[i-1][j]!=-1)
        parent[i][j]=parent[i-1][parent[i-1][j]];
    }
  }
}
ll LCA(ll u,ll v){
  if(depth[u]<depth[v])swap(u,v);
  ll diff=depth[u]-depth[v];
  for(int
      i=0;i<LOGN;i++)if((diff>>i)&1)u=parent[i][u];
  if(u==v)return u;
  for(int i=LOGN-1;i>=0;i--){
    if(parent[i][u]!=parent[i][v]){
      u=parent[i][u],v=parent[i][v];
    }
  }
  return parent[0][u];
}
```

2.10 Min cost Max flow [87 lines]

```
struct node{
  ll u,v,cap,cost;
  node(ll u1,ll v1,ll cap1,ll cost1){
    u=u1,v=v1,cap=cap1,cost=cost1;
  }
};
vector<node>e;
vector<ll>adj[3010];
ll dis[3010],par[3010];
bool vis[3010];
void add_node(ll u,ll v,ll cap,ll cost){
  node e1=node(u,v,cap,cost);
  node e2=node(v,u,0,-cost);
  adj[u].push_back(e.size());
  e.push_back(e1);
  adj[v].push_back(e.size());
  e.push_back(e2);
}
bool spfa(ll s,ll t,ll n){
  queue<ll>q;
  for(ll i=0;i<3001;i++){
    par[i]=-1;
```

```
    dis[i]=inf,vis[i]=false;
  }
  q.push(s);
  dis[s]=0,vis[s]=1;
  while(q.size()){
    ll u=q.front();
    q.pop();
    vis[u]=1;
    for(ll i=0;i<adj[u].size();i++){
      ll id=adj[u][i];
      ll to=e[id].v;
      ll temp=dis[u]+e[id].cost;
      if(e[id].cap && dis[to]>temp){
        par[to]=id;
      }
    }
  }
  return(dis[t]!=inf);
}
pair<ll,ll>mincostmaxflow(ll s,ll t,ll
    n){///Source,Sink,Total node
  ll mincost=0,maxflow=0;
  while(spfa(s,t,n)){
    ll flow=inf+1,id=par[t];
    while(id!=-1){
      flow=min(flow,e[id].cap);
      id=par[e[id].u];
    }
    id=par[t];
    while(id!=-1){
      e[id].cap-=flow;
      e[id^1].cap+=flow;
      id=par[e[id].u];
    }
    mincost+=dis[t]*flow;
    maxflow+=flow;
  }
  return make_pair(maxflow,mincost);
}      dis[to]=temp;
        q.push(to);
        vis[to]=1;
        par[to]=id;
      }
    }
  }
  return(dis[t]!=inf);
}
pair<ll,ll>mincostmaxflow(ll s,ll t,ll
    n){///Source,Sink,Total node
  ll mincost=0,maxflow=0;
  while(spfa(s,t,n)){
    ll flow=inf+1,id=par[t];
    while(id!=-1){
      flow=min(flow,e[id].cap);
      id=par[e[id].u];
    }
    id=par[t];
    while(id!=-1){
      e[id].cap-=flow;
      e[id^1].cap+=flow;
      id=par[e[id].u];
    }
    mincost+=dis[t]*flow;
    maxflow+=flow;
  }
  return make_pair(maxflow,mincost);
}
```

2.11 SACK [50 lines]

```
int sz[maxn];
void getsz(int v,int p){
  sz[v]=1;
  for(auto u : g[v])
    if(u!=p){
      getsz(u,v);
      sz[v]+=sz[u];
    }
}
//SACK O(nlog^2n)
map<int,int>*cnt[maxn];
void dfs(int v,int p){
  int mx=-1,bigChild=-1;
  for(auto u : g[v])
   if(u!=p){
     dfs(u,v);
     if(sz[u]>mx)mx=sz[u],bigChild=u;
   }
  if(bigChild!=-1)cnt[v]=cnt[bigChild];
  else cnt[v]=new map<int,int>();
 (*cnt[v])[col[v]]++;
  for(auto u : g[v])
   if(u!=p && u!=bigChild){
     for(auto x :*cnt[u])
      (*cnt[v])[x.first]+=x.second;
   }
}
//SACK-O(nlogn)
vector<int>*vec[maxn];
int cnt[maxn];
void dfs(int v,int p,bool keep){
  int mx=-1,bigChild=-1;
  for(auto u : g[v])
   if(u!=p&&sz[u]>mx)mx=sz[u],bigChild=u;
  for(auto u : g[v])
   if(u!=p && u!=bigChild)dfs(u,v,0);
  if(bigChild!=-1)
    dfs(bigChild,v,1),vec[v]=vec[bigChild];
  else vec[v]=new vector<int>();
  vec[v]->push_back(v);cnt[col[v]]++;
  for(auto u : g[v])
   if(u!=p && u!=bigChild)
     for(auto x :*vec[u]){
       cnt[col[x]]++;
       vec[v]->push_back(x);
     }
/*in this step*vec[v]contains all of the subtree of
    vertex v.*/
  if(keep==0)
    for(auto u:*vec[v])cnt[col[u]]--;
}
```

3 Misc

3.1 ICPC Template [48 lines]

```
#include <bits/stdc++.h>
using namespace std;
#define eni(x) template < class c > typename \
enable_if<sizeof dud<c>(0) x 1,\
debug&>::type operator<<(c i) {
template < class c > struct rge { c b, e; };
template < class c > rge<c> range(c i, c j) { return
    rge<c>{i, j}; }
template < class c > auto dud(c* x) -> decltype(cerr
    << *x, 0);
template < class c > char dud(...);
struct debug {
#ifdef LOCAL
~debug() { cerr << endl; }
eni(!=) cerr << boolalpha << i; return * this; }
```

```
eni(==) return * this << range(begin(i), end(i)); }
template < class c, class b > debug & operator <<(pair
    < b, c > d) {
  return * this << "(" << d.first << ", " << d.second
      << ")";
}
template < class c > debug & operator <<(rge<c> d) {
  *this << "[";
  for (auto it = d.b; it != d.e; ++it)
  *this << ", " + 2 * (it == d.b) << *it;
  return * this << "]";
}
#else
template < class c > debug & operator <<(const c&) {
    return * this; }
#endif
};
#define imie(...) " [" << #__VA_ARGS__ ": " <<\
   (__VA_ARGS__) << "] "
#define ll long long
#define pb push_back
#define ff first
#define ss second
#define pll pair<ll,ll>
#define pii pair<int,int>
#define all(v) v.begin(),v.end()
#define mod 1000000007
#define mx 100005
mt19937 myrand(time(0));

int main() {
  int tst;
  scanf("%d", &tst);
  for(int t = 1; t <= tst; t++) {
    printf("Case %d: ", t);
  }
  return 0;
}
```

4 Math

4.1 Big Sum [13 lines]

```
ll bigsum(ll a, ll b, ll m) {
  if (b == 0) return 0;
  ll sum; a %= m;
  if (b & 1) {
    sum = bigsum((a * a) % m, (b - 1) / 2, m);
    sum = (sum + (a * sum) % m) % m;
    sum = (1 + (a * sum) % m) % m;
  } else {
    sum = bigsum((a * a) % m, b / 2, m);
    sum = (sum + (a * sum) % m) % m;
  }
  return sum;
}
```

4.2 CRT [9 lines]

```
int CRT(){
  int prod=1,res=0;
  for(int i=0;i<k;i++)prod*=mod[i];
  for(int i=0;i<k;i++){
    int pp=prod/mod[i];
    res+=rem[i]*modInv(pp,mod[i])*pp;
  }
  return(res % prod);
}
```

4.3 Euler's Totient [24 lines]

```
int eulerPhi(int n){
  int result=n;
  for(int i=2;i*i<=n;i++){
    if(n % i==0){
      result-=result/i;
      while(n % i==0)n/=i;
    }
  }
  if(n>1)result-=result/n;
  return result;
}
void phi_sieve(){
  for(int i=1;i<MX;i++){
    tot[i]=i;
    if(i%2==0)tot[i]-=tot[i]/2;
  }
  for(int i=3;i<MX;i+=2){
    if(tot[i]==i){
      for(int j=i;j<MX;j+=i){
        tot[j]-=tot[j]/i;
      }
    }
  }
}
```

4.4 FFT [54 lines]

```
typedef complex<double> CD;
const double PI = acos(-1);
struct FFT{
  int N;
  vector<int> perm;
  void precalculate(){
    perm.resize(N);
    perm[0] = 0;
    for (int k=1; k<N; k<<=1) {
      for (int i=0; i<k; i++) {
        perm[i] <<= 1;
        perm[i+k] = 1 + perm[i];
      }
    }
  }
  void fft(vector<CD> &v, bool invert = false){
    if (v.size() != perm.size()) {
      N = v.size();
      assert(N && (N&(N-1)) == 0);
      precalculate();
    }
    for (int i=0; i<N; i++)
      if (i < perm[i])
        swap(v[i], v[perm[i]]);

    for (int len = 2; len <= N; len <<= 1){
      double angle = 2 * PI / len;
      if (invert) angle = -angle;
      CD factor = polar(1.0, angle);
      for (int i=0; i<N; i+=len) {
        CD w(1);
        for (int j=0; j<len/2; j++){
          CD x = v[i+j], y = w * v[i+j+len/2];
          v[i+j] = x+y;
          v[i+j+len/2] = x-y;
          w *= factor;
        }
      }
    }
    if (invert) for (CD &x : v) x/=N;
  }
  vector<CD> multiply(vector<CD> fa, vector<CD> fb){
    int n = 1;
    while (n < fa.size()+ fb.size())  n<<=1;
    fa.resize(n);
    fb.resize(n);
```

```
    fft(fa);
    fft(fb);
    for (int i=0; i<n; i++) fa[i] *= fb[i];
    fft(fa, true);
    return fa;
  }
};
```

4.5 Gauss Jordan [35 lines]

```
const double EPS=1e-9; const int INF=2;//means inf sol
//returns how many solutions. ans has solutions
int gauss(vector<vector<double>>a,vector<double>&ans){
  int n = (int)a.size(); // number of equations
  int m = (int)a[0].size() - 1; // number of variables
  vector<int> where(m, -1); // solutions column
  for (int col=0, row=0; col<m && row<n; ++col) {
    int sel = row;
    for (int i=row; i<n; ++i)
      if (abs(a[i][col]) > abs(a[sel][col])) sel = i;
    if (abs(a[sel][col]) < EPS) continue;
    for (int i=col; i<=m; ++i)
      swap(a[sel][i], a[row][i]);
    where[col] = row;
    for (int i=0; i<n; ++i)
      if (i != row) {
        double c = a[i][col] / a[row][col];
        for (int j=col; j<=m; ++j)
          a[i][j] -= a[row][j] * c;
      }
    ++row;
  }
  ans.assign(m, 0);
  for (int i=0; i<m; ++i)
    if (where[i] != -1)
      ans[i] = a[where[i]][m] / a[where[i]][i];
  for (int i=0; i<n; ++i) {
    double sum = 0;
    for (int j=0; j<m; ++j) sum += ans[j] * a[i][j];
    if (abs(sum - a[i][m]) > EPS) return 0;//no sol
  }
  for (int i=0; i<m; ++i)
    if (where[i] == -1) return INF; //inf solutions
  return 1; // has one solutions
}
```

4.6 Inverse Mod Upto N [6 lines]

```
void ModularInverse_0_N(ll N){
  mdInv[1]=1,m=MOD;
  for(ll i=2;i<=N;i++)
    mdInv[i]=(-(m/i)*mdInv[m%i])%m+m;
}
/*a^{\phi(m)} \equiv 1 \mod m where a and m is coprime*/
```

4.7 Linear Diophatine [12 lines]

```
/*x'=x+(k*B/g),y'=y-(k*A/g);infinite soln
if A=B=0,C must equal 0 and any x,y is solution;
if A|B=0,(x,y)=(C/A,k)|(k,C/B)*/
bool LDE(int A,int B,int C,int*x,int*y){
  int g=gcd(A,B);
  if(C%g!=0)return false;
  int a=A/g,b=B/g,c=C/g;
  extended_gcd(a,b,x,y);//ax+by=1
  if(g<0)a*=-1;b*=-1;c*=-1;//Ensure gcd(a,b)=1
 *x*=c;*y*=c;//ax+by=c
  return true;//Solution Exists
}
```

4.8 Matrix Exponentiation [65 lines]

```
//Matrix Expo Optimization:
/*We can precalculate every power 2^i of base matrix
    for every i.
Then for mat^x we can multiply every mat^(2^i)where i
    th bit is on in x.*/
struct matrix{
  ll mat[2][2];
  int row,col;
  matrix(){
    memset(mat,0,sizeof mat);
  }
  matrix(int a,int b){
    row=a,col=b;
    memset(mat,0,sizeof mat);
  }
  matrix operator*(const matrix &p)const{
    assert(col==p.row);
    matrix temp;
    temp.row=row,temp.col=p.col;
    for(int i=0;i<temp.row;i++){
      for(int j=0;j<temp.col;j++){
        ll sum=0;
        for(int k=0;k<col;k++){
          sum+=((mat[i][k]%MOD)*
              (p.mat[k][j]%MOD))%MOD;
          sum%=MOD;
        }
        temp.mat[i][j]=sum;
      }
    }
    return temp;
  }
  matrix operator+(const matrix &p)const{
    assert(row==p.row && col==p.col);
    matrix temp;
    temp.row=row,temp.col=col;
    for(int i=0;i<temp.row;i++){
      for(int j=0;j<temp.col;j++)
        temp.mat[i][j]=((mat[i][j]%MOD)+
            (p.mat[i][j]%MOD))%MOD;;
    }
    return temp;
  }
  matrix identity(){
    matrix temp;
    temp.row=row,temp.col=col;
    for(int i=0;i<row;i++)
      temp.mat[i][i]=1;
    return temp;
  }
  matrix pow(ll pow){
    matrix temp=(*this);
    matrix ret=(*this).identity();
    while(pow){
      if(pow % 2==1)
        ret=ret*temp;
      temp=temp*temp;
      pow/=2;
    }
    return ret;
  }
  void show(){
    for(int i=0;i<row;i++){
      for(int j=0;j<col;j++)
        printf("%lld",mat[i][j]);
      printf("\n");
    }
  }
};
```

4.9 Miller Robbin [28 lines]

```
ll mulmod(ll a,ll b,ll mo){
  ll q=((ld)a*(ld)b/(ld)mo);
  ll res=a*b-mo*q;
  return((res % mo)+mo)%mo;
}
bool miller(ll a,ll d,ll p){
  ll x=bigmod(a,d,p);
  if(x==1 || x==p-1)return 1;
  while(d!=p-1) {
    x=mulmod(x,x,p);
    d*=2;
    if(x==1)return 0;
    if(x==p-1)return 1;
  }
  return 0;
}
bool isPrimes(ll p){
  if(p<2) return 0;
  if(p==2)return 1;
  if(p!=2 && p%2==0)return 0;
  ll d=p-1;
  while(d % 2==0) d=d/2;
  for(ll i=1;i<20;i++){
    ll a=abs(rand()%(p-2))+2;
    if(!miller(a,d,p))return 0;
  }
  return 1;
}
```

4.10 Mobius Function [15 lines]

```
int mob[N];
void mobius() {
  for(int i = 1; i < N; i++) mob[i] = 3;
  mob[1] = 1;
  for (int i = 2; i < N; i++){
    if (mob[i] == 3){
      mob[i] = -1;
      for (int j = 2 * i; j < N; j += i)
        mob[j] = (mob[j] == 3 ? -1 : mob[j] * (-1));
      if (i < N / i)
        for (int j = i * i; j < N; j += i * i)
          mob[j] = 0;
    }
  }
}
```

4.11 Mod Int [26 lines]

```
template <const int32_t MOD>
struct modint {
  int32_t value;
  modint() = default;
  modint(int32_t value_) : value(value_) {}
  inline modint<MOD> operator + (modint<MOD> other)
      const { int32_t c = this->value + other.value;
      return modint<MOD>(c >= MOD ? c - MOD : c); }
  inline modint<MOD> operator - (modint<MOD> other)
      const { int32_t c = this->value - other.value;
      return modint<MOD>(c <  0 ? c + MOD : c); }
  inline modint<MOD> operator * (modint<MOD> other)
      const { int32_t c = (int64_t)this->value *
      other.value % MOD; return modint<MOD>(c < 0 ? c
      + MOD : c); }
  inline modint<MOD> & operator += (modint<MOD> other)
      { this->value += other.value; if (this->value
      >= MOD) this->value -= MOD; return *this; }
  inline modint<MOD> & operator -= (modint<MOD> other)
      { this->value -= other.value; if (this->value <
      0) this->value += MOD; return *this; }
  inline modint<MOD> & operator *= (modint<MOD> other)
      { this->value = (int64_t)this->value *
      other.value % MOD; if (this->value < 0)
      this->value += MOD; return *this; }
  inline modint<MOD> operator - () const { return
      modint<MOD>(this->value ? MOD - this->value :
      0); }
  modint<MOD> pow(uint64_t k) const { modint<MOD> x =
      *this, y = 1; for (; k; k >>= 1) { if (k & 1) y
      *= x; x *= x; } return y; }
  modint<MOD> inv() const { return pow(MOD - 2); }  /*
      MOD must be a prime*/
  inline modint<MOD> operator /  (modint<MOD> other)
      const { return *this *  other.inv(); }
  inline modint<MOD> operator /= (modint<MOD> other)
      { return *this *= other.inv(); }
  inline bool operator == (modint<MOD> other) const {
      return value == other.value; }
  inline bool operator != (modint<MOD> other) const {
      return value != other.value; }
  inline bool operator < (modint<MOD> other) const {
      return value < other.value; }
  inline bool operator > (modint<MOD> other) const {
      return value > other.value; }
};
template <int32_t MOD> modint<MOD> operator * (int64_t
    value, modint<MOD> n) { return modint<MOD>(value)
    * n; }
template <int32_t MOD> modint<MOD> operator * (int32_t
    value, modint<MOD> n) { return modint<MOD>(value %
    MOD) * n; }
template <int32_t MOD> istream & operator >> (istream
    & in, modint<MOD> &n) { return in >> n.value; }
template <int32_t MOD> ostream & operator << (ostream
    & out, modint<MOD> n) { return out << n.value; }
using mint = modint<mod>;
```

4.12 Mod Inverse [16 lines]

```
int gcdExt(int a, int b, int &x, int &y) {
    if (a == 0) {
        x = 0, y = 1;
        return b;
    }
    int x1, y1;
    int gcd = gcdExt(b % a, a, x1, y1);
    x = y1 - (b / a) * x1;
    y = x1;
    return gcd;
}
int modInv(int a, int m) {
    int x, y; //if g==1 Inverse doesn't exist
    int g = gcdExt(a, m, x, y);
    return (x % m + m) % m;
}
```

4.13 NTT [133 lines]

```
namespace ntt{
  struct num{
    double x, y;
    num() { x = y = 0; }
    num(double x, double y) : x(x), y(y) {}
  };
  inline num operator+(num a, num b){ return num(a.x
      + b.x, a.y + b.y); }
  inline num operator-(num a, num b){ return num(a.x
      - b.x, a.y - b.y); }
  inline num operator*(num a, num b){ return num(a.x
      * b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
  inline num conj(num a) { return num(a.x, -a.y); }
  int base = 1;
  vector<num> roots = {{0, 0}, {1, 0}};
```

```
vector<int> rev = {0, 1};
const double PI = acosl(-1.0);
void ensure_base(int nbase){
  if (nbase <= base) return;
  rev.resize(1 << nbase);
  for (int i = 0; i < (1 << nbase); i++)
    rev[i] = (rev[i >> 1] >> 1) + ((i & 1) << (nbase
        - 1));
  roots.resize(1 << nbase);
  while (base < nbase){
    double angle = 2 * PI / (1 << (base + 1));
    for (int i = 1 << (base - 1); i < (1 << base);
        i++){
      roots[i << 1] = roots[i];
      double angle_i = angle * (2 * i + 1 - (1 <<
          base));
      roots[(i << 1) + 1] = num(cos(angle_i),
          sin(angle_i));
    }
    base++;
  }
}
void fft(vector<num> &a, int n = -1){
  if (n == -1) n = a.size();
  assert((n & (n - 1)) == 0);
  int zeros = __builtin_ctz(n);
  ensure_base(zeros);
  int shift = base - zeros;
  for (int i = 0; i < n; i++)
    if (i < (rev[i] >> shift))
      swap(a[i], a[rev[i] >> shift]);
  for (int k = 1; k < n; k <<= 1){
    for (int i = 0; i < n; i += 2 * k){
      for (int j = 0; j < k; j++){
        num z = a[i + j + k] * roots[j + k];
        a[i + j + k] = a[i + j] - z;
        a[i + j] = a[i + j] + z;
      }
    }
  }
}
vector<num> fa, fb;
vector<int> multiply(vector<int> &a, vector<int>
    &b){
  int need = a.size() + b.size() - 1;
  int nbase = 0;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int)fa.size()) fa.resize(sz);
  for (int i = 0; i < sz; i++){
    int x = (i < (int)a.size() ? a[i] : 0);
    int y = (i < (int)b.size() ? b[i] : 0);
    fa[i] = num(x, y);
  }
  fft(fa, sz);
  num r(0, -0.25 / sz);
  for (int i = 0; i <= (sz >> 1); i++){
    int j = (sz - i) & (sz - 1);
    num z = (fa[j] * fa[j] - conj(fa[i] * fa[i])) *
        r;
    if (i != j) fa[j] = (fa[i] * fa[i] - conj(fa[j]
        * fa[j])) * r;
    fa[i] = z;
  }
  fft(fa, sz);
  vector<int> res(need);
  for (int i = 0; i < need; i++) res[i] = fa[i].x +
      0.5;
  return res;
}
vector<int> multiply(vector<int> &a, vector<int>
    &b, int m, int eq = 0){
  int need = a.size() + b.size() - 1;
  int nbase = 0;
  while ((1 << nbase) < need) nbase++;
  ensure_base(nbase);
  int sz = 1 << nbase;
  if (sz > (int)fa.size()) fa.resize(sz);
  for (int i = 0; i < (int)a.size(); i++){
    int x = (a[i] % m + m) % m;
    fa[i] = num(x & ((1 << 15) - 1), x >> 15);
  }
  fill(fa.begin() + a.size(), fa.begin() + sz,
      num{0, 0});
  fft(fa, sz);
  if (sz > (int)fb.size()) fb.resize(sz);
  if (eq) copy(fa.begin(), fa.begin() + sz,
      fb.begin());
  else {
    for (int i = 0; i < (int)b.size(); i++){
      int x = (b[i] % m + m) % m;
      fb[i] = num(x & ((1 << 15) - 1), x >> 15);
    }
    fill(fb.begin() + b.size(), fb.begin() + sz,
        num{0, 0});
    fft(fb, sz);
  }
  double ratio = 0.25 / sz;
  num r2(0, -1), r3(ratio, 0), r4(0, -ratio), r5(0,
      1);
  for (int i = 0; i <= (sz >> 1); i++){
    int j = (sz - i) & (sz - 1);
    num a1 = (fa[i] + conj(fa[j]));
    num a2 = (fa[i] - conj(fa[j])) * r2;
    num b1 = (fb[i] + conj(fb[j])) * r3;
    num b2 = (fb[i] - conj(fb[j])) * r4;
    if (i != j) {
      num c1 = (fa[j] + conj(fa[i]));
      num c2 = (fa[j] - conj(fa[i])) * r2;
      num d1 = (fb[j] + conj(fb[i])) * r3;
      num d2 = (fb[j] - conj(fb[i])) * r4;
      fa[i] = c1 * d1 + c2 * d2 * r5;
      fb[i] = c1 * d2 + c2 * d1;
    }
    fa[j] = a1 * b1 + a2 * b2 * r5;
    fb[j] = a1 * b2 + a2 * b1;
  }
  fft(fa, sz);
  fft(fb, sz);
  vector<int> res(need);
  for (int i = 0; i < need; i++){
    ll aa = fa[i].x + 0.5;
    ll bb = fb[i].x + 0.5;
    ll cc = fa[i].y + 0.5;
    res[i] = (aa + ((bb % m) << 15) + ((cc % m) <<
        30)) % m;
  }
  return res;
}
vector<int> square(vector<int> &a, int m){
  return multiply(a, a, m, 1);
}
}; // namespace ntt
using namespace ntt;
```

4.14 No of Digist in Fact(n) in base B [7 lines]

```
ll NoOfDigitInNFactInBaseB(ll N,ll B){
  ll i;
  double ans=0;
  for(i=1;i<=N;i++)ans+=log(i);
  ans=ans/log(B),ans=ans+1;
  return(ll)ans;
}
```

4.15 OpenClose Interval DP [16 lines]

```
int n,k,dp[205][205][1005];
int ara[205];///ensure sorted(maybe :p)
int func(int idx,int g,int tk){
  if(g<0 or tk<0)return 0;
  if(idx==n+1){
    if(tk>=0 and g==0)return 1;
    else return 0;
  }
  if(dp[idx][g][tk]!=-1)return dp[idx][g][tk];
  ll ret=0;
  ret+=func(idx+1,g+1,tk-g*(ara[idx]-ara[idx-1]));
      /*open grp*/
  ret+=((ll)g*(ll)func(idx+1,g-1,tk-g*(ara[idx]-
      ara[idx-1])));/*close grp*/
  ret+=((ll)g*(ll)func(idx+1,g,tk-g*(ara[idx]-
      ara[idx-1])));/*cur grp*/
  ret+=func(idx+1,g,tk-g*(ara[idx]-ara[idx-1]));
      /*open and close grp*/
  return dp[idx][g][tk]=(int)ret;
}
```

4.16 Pollard Rho [17 lines]

```
ll mult(ll a, ll b, ll mod){
  return ((__int128)a*b)%mod;
}
ll f(ll x, ll c, ll mod){
  return (mult(x,x,mod)+c)%mod;
}
ll rho(ll n){
  if(n%2==0) return 2;
  ll x=myrand()%n+1,y=x,c=myrand()%n+1,g=1;
  while(g==1){
    x=f(x,c,n);
    y=f(y,c,n);
    y=f(y,c,n);
    g=__gcd(abs(x-y),n);
  }
  return g;
}
```

4.17 SOD Upto N [16 lines]

```
ll SOD_UpTo_N(ll N){
  ll i,j,ans=0;///upto N in Sqrt(N)
  for(i=1;i*i<=N;i++){
    j=N/i;
    ans+=((j*(j+1))/2)-(((i-1)*i)/2);
    ans+=((j-i)*i);
  }
  return ans;
}
ll SODUptoN(ll N){
  ll res=0,u=sqrt(N);
  for(ll i=1;i<=u;i++)
    res+=(N/i)-i;
  res*=2,res+=u;
  return res;
}
```

4.18 Segmented Sieve [24 lines]

```
ll SegmentedSieve(ll low,ll high){
  ll i=0,j,start,ans=0;
  memset(isPrime,true,sizeof isPrime);
  if(low % 2==1)i++;
  for(i;i<=high-low;i+=2)isPrime[i]=false;
  for(i=3;i<=sqrt(high)+1;i+=2){
    start=max((ll)ceil(low/i),1ll);
    if(start==1)start++;
    start=start*i;
    if(start<low) start+=i;
    for(j=start-low;j<=high-low;j+=i)
      isPrime[j]=false;
  }
  if(low==1){
    isPrime[0]=false;
    isPrime[1]=true;
  }
  if(low==2)isPrime[0]=true;
  for(i=0;i<=high-low;i++){
    if(isPrime[i])
      ans++;
  }
  return ans;
}
```

4.19 nCr DP [11 lines]

```
ll NCR(){
  for(ll i=0;i<1003;i++)DP[i][0]=1;
  for(ll i=1;i<1003;i++)
    for(ll j=1;j<1003;j++)
     DP[i][j]=DP[i-1][j]+DP[i-1][j-1];
}
/*star and bars theorem
divide n stars into k partitions,
where each partition is positive
nCr(n-1,k-1);
nCr(n+k-1,k-1);//non negative */
```

5 Data Structure

5.1 2D BIT [16 lines]

```
int n,m,bit[MX][MX];//bit[n][m]
/*for a specific rectangle:
query(rx,ry)-query(lx-1,ry)-query(rx,ly-1)
+query(lx-1,ly-1),where lx<=ly & rx<=ry*/
int query(int x,int y){
  int res=0;
  for(x=x+1;x>0;x-=x&-x)
    for(int py=y+1;py>0;py-=py&-py)
    res+=bit[x][py];
  return res;
}
void update(int x,int y,int val){
  for(x=x+1;x<=n;x+=x&-x)
    for(int py=y+1;py<=m;py+=py&-py)
    bit[x][py]+=val;
}
```

5.2 MO's Algorithm [34 lines]

```
int n,m,block,arr[maxn],ans[maxn];
struct query{
  int l,r,id;
  bool operator<(const query &q)const {
  int a=l/block, b=q.l/block;
    if(a!=b) return l<q.l;
    return l/block%2?r<q.r:r>q.r;
  }
}q[maxn];
int curr=0,cnt[maxn];
void add(int i){
  int x=arr[i];if(x>=maxn)return;
  if(cnt[x]==x)curr--;
  cnt[x]++;
  if(cnt[x]==x)curr++;
}
```

```
void remove(int i){
  int x=arr[i];if(x>=maxn)return;
  if(cnt[x]==x)curr--;
  cnt[x]--;
  if(cnt[x]==x)curr++;
}
void MO(){
  block=sqrt(n)+1;
  sort(q,q+m);
  int l=0,r=-1;curr=0;
  for(int i=0;i<m;i++){
    while(l>q[i].l)add(--l);
    while(r<q[i].r)add(++r);
    while(l<q[i].l)remove(l++);
    while(r>q[i].r)remove(r--);
    ans[q[i].id]=curr;
  }
}
```

5.3 Merge Sort Tree [28 lines]

```
class MergeSortTree {
  int n; vector<vector<int>> tree;
  MergeSortTree(vector<int> const& a) {
    n = a.size(); tree.resize(4 * n);
    build_tree(1, 0, n - 1, a);
  }
  void build_tree(int at, int b, int e, vector<int>
      const& a) {
    if (b == e) {
      tree[at].push_back(a[b]);
      return;
    }
    int mid = (b + e) >> 1;
    build_tree(at + at, b, mid, a);
    build_tree(at + at + 1, mid + 1, e, a);
    merge(tree[at + at].begin(), tree[at + at].end(),
        tree[at + at + 1].begin(), tree[at + at +
        1].end(), back_inserter(tree[at]));
  }
  int countLessEqual(int l, int r, int x) {
    return query(1, 0, n - 1, l, r, x);
  }
  int query(int at, int b, int e, int l, int r,int k){
    if (e < l || b > r) return 0;
    if (b >= l && e <= r) {
      return upper_bound(tree[at].begin(),
          tree[at].end(), k) - tree[at].begin();
    }
    int mid = (b + e) >> 1;
    return query(at + at, b, mid, l, r, k) + query(at
        + at + 1, mid + 1, e, l, r, k);
  }
};
```

5.4 PBDS [10 lines]

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/detail/standard_policies.hpp>
using namespace __gnu_pbds;
using namespace std;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update>ordered_set;
    /*less_equal->multiset, problem with erase*/
ordered_set X; X.insert(1);
*X.find_by_order(0)//1
 X.order_of_key(1)//0
(end(X)==X.find_by_order(6))
```

5.5 Persistent Segment Tree [44 lines]

```
//To update in a version copy the root,
//then make update in the version. like-
//root[y]=root[x];
//update(root[y],...)
const int maxn=1e5+10;
struct node {
  int l,r,sum;
}t[maxn*20];
int root[maxn],a[maxn],n,m,k,idx,M;
void update(int &nd,int l,int r,int &i){
  t[++idx]=t[nd];
 ++t[nd=idx].sum;//+=v;
  if(l==r)return;
  int m=l+r>>1;
  if(i<=m)update(t[nd].l,l,m,i);
  else update(t[nd].r,m+1,r,i);
}
int query(int nd,int l,int r,int &i,int &j){
  if(r<i || l>j)return 0;
  if(i<=l && r<=j)return t[nd].sum;
  int m=l+r>>1;
  return query(t[nd].l,l,m,i,j)
 +query(t[nd].r,m+1,r,i,j);
}
//a=root[r],b=root[l-1]
int lesscnt(int a,int b,int l,int r,int k){
  if(r<=k)return t[a].sum-t[b].sum;
  int m=l+r>>1;
  if(k<=m)return lesscnt(t[a].l,t[b].l,l,m,k);
  else return lesscnt(t[a].l,t[b].l,l,m,k)+
    lesscnt(t[a].r,t[b].r,m+1,r,k);
}
int kthnum(int a,int b,int l,int r,int k){
  if(l==r)return l;
  int cnt=t[t[a].l].sum-t[t[b].l].sum;
  int m=l+r>>1;
  if(cnt>=k)return kthnum(t[a].l,t[b].l,l,m,k);
  else return kthnum(t[a].r,t[b].r,m+1,r,k-cnt);
}
void init(int v=1){
  t[0].l=t[0].r=t[0].sum=0;
  for(int i=1;i<=n;++i)
    update(root[i]=root[i-1],0,M,a[i]);
}
```

5.6 Segment Tree [42 lines]

```
struct lazySegTree{
  #define maxn 200000
  int tree[8*maxn],lazy[8*maxn];
  void init(){
    memset(tree,0,sizeof tree);
    memset(lazy,0,sizeof lazy);
  }
  void merge(ll pos, ll l, ll r){
    if(!lazy[pos]) return;
    tree[pos]+=lazy[pos];
    if(l!=r){
      lazy[pos*2]+=lazy[pos];
      lazy[pos*2+1]+=lazy[pos];
    }
    lazy[pos]=0;
  }
  void update(ll pos, ll l, ll r, ll L, ll R, ll
      value){
    merge(pos,l,r);
    if(l>R || r<L || L>R) return;
    if(l>=L && r<=R){
      tree[pos]+=value;
      if(l!=r){
        lazy[pos*2]+=value;
        lazy[pos*2+1]+=value;
      }
```

```
    return;
  }
  ll mid=(l+r)/2;
  update(pos*2,l,mid,L,R,value);
  update(pos*2+1,mid+1,r,L,R,value);
  tree[pos]=tree[pos*2]+tree[pos*2+1];
}
ll query(ll pos, ll l, ll r, ll L, ll R){
  merge(pos,l,r);
  if(l>R || r<L) return 0;
  if(l>=L && r<=R) return tree[pos];
  ll mid=(l+r)/2;
  ll q1=query(pos*2,l,mid,L,R);
  ll q2=query(pos*2+1,mid+1,r,L,R);
  return q1+q2;
}
};
```

5.7 Treap-Implicit [319 lines]

```
#define nl '\n'
mt19937 rnd(chrono::steady_clock::now().
time_since_epoch().count()); //this line goes up
const int N = 3e5 + 9; const int mod = 1e9 + 7;
struct treap {
  /*This is an implicit treap which investigates here
      on an array*/
  struct node {
    int val, sz, prior, lazy, sum,mx,mn,repl;
    bool repl_flag,rev;
    node *l, *r, *par;
    node() {
      lazy = 0; rev = 0;
      sum=0; val = 0;
      sz = 0; mx=0;
      mn=0; repl=0;
      repl_flag=0; prior = 0;
      l = NULL; r = NULL;
      par = NULL;
    }
    node(int _val) {
      val = _val; sum = _val;
      mx=_val; mn=_val;
      repl=0; repl_flag=0;
      rev = 0; lazy = 0;
      sz = 1; prior = rnd();
      l = NULL; r = NULL;
      par = NULL;
    }
  };
  typedef node* pnode;
  pnode root;
  map<int,pnode>position;/*positions of all the values
  clearing the treap*/
  void clear() {
    root = NULL; position.clear();
  }
  treap() {
    clear();
  }
  int size(pnode t) {
    return t ? t->sz : 0;
  }
  void update_size(pnode &t) {
    if(t) t->sz = size(t->l) + size(t->r) + 1;
  }
  void update_parent(pnode &t) {
    if(!t) return;
    if(t->l) t->l->par = t;
    if(t->r) t->r->par = t;
  }
  //add operation
  void lazy_sum_upd(pnode &t) {
    if( !t or !t->lazy )    return;
    t->sum += t->lazy * size(t);
    t->val += t->lazy;
    t->mx += t->lazy;
    t->mn += t->lazy;
    if( t->l )  t->l->lazy += t->lazy;
    if( t->r )  t->r->lazy += t->lazy;
    t->lazy = 0;
  }
  //replace update
  void lazy_repl_upd(pnode &t) {
    if( !t or !t->repl_flag )   return;
    t->val = t->mx = t->mn = t->repl;
    t->sum = t->val * size(t);
    if( t->l ) {
      t->l->repl = t->repl;
      t->l->repl_flag = true;
    }
    if( t->r ) {
      t->r->repl = t->repl;
      t->r->repl_flag = true;
    }
    t->repl_flag = false;
    t->repl = 0;
  }
  ///reverse update
  void lazy_rev_upd(pnode &t) {
    if( !t or !t->rev ) return;
    t->rev = false;
    swap(t->l, t->r);
    if( t->l )  t->l->rev ^= true;
    if( t->r )  t->r->rev ^= true;
  }
  /*reset the value of current node assuming it now
  represents a single element of the array*/
  void reset(pnode &t) {
    if(!t) return;
    t->sum = t->val;
    t->mx = t->val;
    t->mn = t->val;
  }
  /*combine node l and r to form t by updating
      corresponding queries*/
  void combine(pnode &t, pnode l, pnode r) {
    if(!l) {
      t = r;
      return;
    }
    if(!r) {
      t = l;
      return;
    }
    /*Beware!!!Here t can be equal to l or r anytime
    i.e. t and (l or r) is representing same node
    so operation is needed to be done carefully
    e.g. if t and r are same then after
    t->sum=l->sum+r->sum operation,
    r->sum will be the same as t->sum
    so BE CAREFUL*/
    t->sum = l->sum + r->sum;
    t->mx = max(l->mx,r->mx);
    t->mn = min(l->mn,r->mn);
  }
  ///perform all operations
  void operation(pnode &t) {
    if( !t )    return;
```

```
    reset(t);
    lazy_rev_upd(t->l);
    lazy_rev_upd(t->r);
    lazy_repl_upd(t->l);
    lazy_repl_upd(t->r);
    lazy_sum_upd(t->l);
    lazy_sum_upd(t->r);
    combine(t, t->l, t);
    combine(t, t, t->r);
  }
  /*split node t in l and r by key k
so first k+1 elements(0,1,2,...k) of the array from
    node t
will be split in left node and rest will be in right
    node*/
  void split(pnode t, pnode &l, pnode &r, int k, int
      add = 0) {
    if(t == NULL) {
      l = NULL;
      r = NULL;
      return;
    }
    lazy_rev_upd(t);
    lazy_repl_upd(t);
    lazy_sum_upd(t);
    int idx = add + size(t->l);
    if(idx <= k) split(t->r, t->r, r, k, idx + 1), l
        = t;
    else split(t->l, l, t->l, k, add), r = t;
    update_parent(t);
    update_size(t);
    operation(t);
  }
  //merge node l with r in t
  void merge(pnode &t, pnode l, pnode r) {
    lazy_rev_upd(l);
    lazy_rev_upd(r);
    lazy_repl_upd(l);
    lazy_repl_upd(r);
    lazy_sum_upd(l);
    lazy_sum_upd(r);
    if(!l) {
      t = r;
      return;
    }
    if(!r) {
      t = l;
      return;
    }
    if(l->prior > r->prior) merge(l->r, l->r, r), t =
        l;
    else merge(r->l, l, r->l), t = r;
    update_parent(t);
    update_size(t);
    operation(t);
  }
  /*insert val in position a[pos]
  so all previous values from pos to last will be
    right shifted*/
  void insert(int pos, int val) {
    if(root == NULL) {
      pnode to_add = new node(val);
      root = to_add;
      position[val] = root;
      return;
    }
    pnode l, r, mid;
    mid = new node(val);
    position[val] = mid;
    split(root, l, r, pos - 1);
    merge(l, l, mid);
    merge(root, l, r);
  }
  /*erase from qL to qR indexes
  so all previous indexes from qR+1 to last will be
    left shifted qR-qL+1 times*/
  void erase(int qL, int qR) {
    pnode l, r, mid;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
    merge(root, l, r);
  }
  /*returns answer for corresponding types of query*/
  int query(int qL, int qR) {
    pnode l, r, mid;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
    int answer = mid->sum;
    merge(r, mid, r);
    merge(root, l, r);
    return answer;
  }
  /*add val in all the values from a[qL] to a[qR]
      positions*/
  void update(int qL, int qR, int val) {
    pnode l, r, mid;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
    lazy_repl_upd(mid);
    mid->lazy += val;
    merge(r, mid, r);
    merge(root, l, r);
  }
  /*reverse all the values from qL to qR*/
  void reverse(int qL, int qR) {
    pnode l, r, mid;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
    mid->rev ^= 1;
    merge(r, mid, r);
    merge(root, l, r);
  }
  /*replace all the values from a[qL] to a[qR] by v*/
  void replace(int qL, int qR,int v) {
    pnode l, r, mid;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
    lazy_sum_upd(mid);
    mid->repl_flag=1;
    mid->repl=v;
    merge(r, mid, r);
    merge(root, l, r);
  }
  /*it will cyclic right shift the array k times
  so for k=1, a[qL]=a[qR] and all positions from ql+1
    to qR will
  have values from previous a[qL] to a[qR-1]
  if you make left_shift=1 then it will to the
    opposite*/
  void cyclic_shift(int qL, int qR, int k,bool
      left_shift=0) {
    if(qL == qR) return;
    k %= (qR - qL + 1);
    pnode l, r, mid, fh, sh;
    split(root, l, r, qL - 1);
    split(r, mid, r, qR - qL);
```

```
    if(left_shift==0) split(mid, fh, sh, (qR - qL + 1)
        - k - 1);
    else split(mid,fh,sh,k-1);
    merge(mid, sh, fh);
    merge(r, mid, r);
    merge(root, l, r);
  }
  bool exist;
  //returns index of node curr
  int get_pos(pnode curr, pnode son = nullptr) {
    if(exist==0) return 0;
    if(curr==NULL) {
      exist=0;
      return 0;
    }
    if(!son) {
      if(curr == root) return size(curr->l);
      else return size(curr->l) + get_pos(curr->par,
          curr);
    }
    if(curr == root) {
      if(son == curr->l) return 0;
      else return size(curr->l) + 1;
    }
    if(curr->l == son) return get_pos(curr->par,
        curr);
    else return get_pos(curr->par, curr) +
        size(curr->l) + 1;
  }
  /*returns index of the value
  if the value has multiple positions then it will
  return the last index where it was added last time
  returns -1 if it doesn't exist in the array*/
  int get_pos(int value) {
    if(position.find(value)==position.end()) return
        -1;
    exist=1;
    int x=get_pos(position[value]);
    if(exist==0) return -1;
    else return x;
  }
  //returns value of index pos
  int get_val(int pos) {
    return query(pos,pos);
  }
  //returns size of the treap
  int size() {
    return size(root);
  }
  //inorder traversal to get indexes chronologically*
  void inorder(pnode cur) {
    if(cur==NULL) return;
    operation(cur);
    inorder(cur->l);
    cout<<cur->val<<' ';
    inorder(cur->r);
  }
  //print current array values serially
  void print_array() {
    inorder(root);
    cout<<nl;
  }
  bool find(int val) {
    if(get_pos(val)==-1) return 0;
    else return 1;
  }
};
treap t;
///Beware!!!here treap is 0-indexed
// use in main
  int i,j,k,n,m,l,r,q;
  for(i=0; i<10; i++) t.insert(i,i*10);
  t.cyclic_shift(4,5,1);
  t.update(2,5,1);
  t.replace(2,5,100);
  t.reverse(2,9);
  t.replace(2,5,200);
  cout<<t.query(0,7)<<nl;
  t.cyclic_shift(2,3,2,1);
  cout<<t.get_pos(20)<<nl;
  t.erase(2,2);
  cout<<t.find(30)<<nl;
  //t.print_array();
```

6 String

6.1 Aho-Corasick [124 lines]

```
const int NODE=3000500;///Maximum Nodes
const int LGN=30;      ///Maximum Number of Tries
const int MXCHR=53;    ///Maximum Characters
const int MXP=5005;    ///
struct node {
  int val;
  int child[MXCHR];
  vector<int>graph;
  void clear(){
    CLR(child,0);
    val=0;
    graph.clear();
  }
}Trie[NODE+10];
int maxNodeId,fail[NODE+10],par[NODE+10];
int nodeSt[NODE+10],nodeEd[NODE+10];
vlong csum[NODE+10],pLoc[MXP];
void resetTrie(){
  maxNodeId=0;
}
int getNode(){
  int curNodeId=++maxNodeId;
  Trie[curNodeId].clear();
  return curNodeId;
}
inline void upd(vlong pos){
  csum[pos]++;
}
inline vlong qry(vlong pos){
  vlong res=csum[pos];
  return res;
}
struct AhoCorasick {
  int root,size,euler;
  void clear(){
    root=getNode();
    size=euler=0;
  }
  inline int getname(char ch){
    if(ch=='-')return 52;
    else if(ch>='A' && ch<='Z')return 26+(ch-'A');
    else return(ch-'a');
  }
  void addToTrie(string &s,int id){
  //Add string s to the Trie in general way
    int len=SZ(s),cur=root;
    FOR(i,0,len-1){
      int c=getname(s[i]);
      if(Trie[cur].child[c]==0){
        int curNodeId=getNode();
        Trie[curNodeId].val=c;
        Trie[cur].child[c]=curNodeId;
```

```
    }
    cur=Trie[cur].child[c];
  }
  pLoc[id]=cur;
  size++;
}
void calcFailFunction(){
  queue<int>Q;
  Q.push(root);
  while(!Q.empty()){
    int s=Q.front();
    Q.pop();
  //Add all the children to the queue:
    FOR(i,0,MXCHR-1){
      int t=Trie[s].child[i];
      if(t!=0){
        Q.push(t);
        par[t]=s;
      }
    }
    if(s==root){/*Handle special case when s is
        root*/
      fail[s]=par[s]=root;
      continue;
    }
//Find fall back of s:
    int p=par[s],f=fail[p];;
    int val=Trie[s].val;
/*Fall back till you found a node who has got val as
   a child*/
    while(f!=root && Trie[f].child[val]==0){
      f=fail[f];
    }
    fail[s]=(Trie[f].child[val]==0)? root :
        Trie[f].child[val];
//Self fall back not allowed
    if(s==fail[s]){
      fail[s]=root;
    }
    Trie[fail[s]].graph.push_back(s);
  }
}
void dfs(int pos){
  ++euler;
  nodeSt[pos]=euler;
  for(auto x: Trie[pos].graph){
    dfs(x);
  }
  nodeEd[pos]=euler;
}
//Returns the next state
int goTo(int state,int c){
  if(Trie[state].child[c]!=0){/*No need to fall
      back*/
    return Trie[state].child[c];
  }
//Fall back now:
  int f=fail[state];
  while(f!=root && Trie[f].child[c]==0){
    f=fail[f];
  }
  int res=(Trie[f].child[c]==0)?
      root:Trie[f].child[c];
  return res;
}
/*Iterate through the whole text and find all the
   matchings*/
void findmatching(string &s){
  int cur=root,idx=0;
  int len=SZ(s);
  while(idx<len){
    int c=getname(s[idx]);
    cur=goTo(cur,c);
    upd(nodeSt[cur]);
    idx++;
  }
 }
}acorasick;
```

6.2 Double Hasing [50 lines]

```
struct simplehash {
    int len;
    long long base, mod;
    vector<int> P, H, R;
    simplehash() {}
    simplehash(const char* str, long long b, long long
        m) {
        base = b, mod = m, len = strlen(str);
        P.resize(len + 4, 1), H.resize(len + 3, 0),
            R.resize(len + 3, 0);
        for (int i = 1; i <= len + 3; i++)
            P[i] = (P[i - 1] * base) % mod;
        for (int i = 1; i <= len; i++)
            H[i] = (H[i - 1] * base + str[i - 1] +
                1007) % mod;
        for (int i = len; i >= 1; i--)
            R[i] = (R[i + 1] * base + str[i - 1] +
                1007) % mod;
    }
    inline int range_hash(int l, int r) {
        int hashval = H[r + 1] - ((long long)P[r - l
            + 1] * H[l] % mod);
        return (hashval < 0 ? hashval + mod :
            hashval);
    }
    inline int reverse_hash(int l, int r) {
        int hashval = R[l + 1] - ((long long)P[r - l
            + 1] * R[r + 2] % mod);
        return (hashval < 0 ? hashval + mod :
            hashval);
    }
};
struct stringhash {
    simplehash sh1, sh2;
    stringhash() {}
    stringhash(const char* str) {
        sh1 = simplehash(str, 1949313259, 2091573227);
        sh2 = simplehash(str, 1997293877, 2117566807);
    }
    long long concate(stringhash& B , int l1 , int r1
        , int l2 , int r2) {
        int len1 = r1 - l1+1 , len2 = r2 - l2+1;
        long long x1 = sh1.range_hash(l1, r1) ,
        x2 = B.sh1.range_hash(l2, r2);
        x1 = (x1 * B.sh1.P[len2]) % 2091573227;
        long long newx1 = (x1 + x2) % 2091573227;
        x1 = sh2.range_hash(l1, r1);
        x2 = B.sh2.range_hash(l2, r2);
        x1 = (x1 * B.sh2.P[len2]) % 2117566807;
        long long newx2 = (x1 + x2) % 2117566807;
        return (newx1 << 32) ^ newx2;
    }
    inline long long range_hash(int l, int r) {
        return ((long long)sh1.range_hash(l, r) << 32)
            ^ sh2.range_hash(l, r);
    }
    inline long long reverse_hash(int l, int r) {
```

```
        return ((long long)sh1.reverse_hash(l, r) <<
            32) ^ sh2.reverse_hash(l, r);
    }
};
```

6.3 KMP [23 lines]

```
char P[maxn],T[maxn];
int b[maxn],n,m;
void kmpPreprocess(){
  int i=0,j=-1;
  b[0]=-1;
  while(i<m){
    while(j>=0 and P[i]!=P[j])
      j=b[j];
      i++;j++;
      b[i]=j;
  }
}
void kmpSearch(){
  int i=0,j=0;
  while(i<n){
    while(j>=0 and T[i]!=P[j])
      j=b[j];
      i++;j++;
    if(j==m){
      //pattern found at index i-j
    }
  }
}
```

6.4 Palindromic Tree [30 lines]

```
struct PalindromicTree{
  int n,idx,t;
  vector<vector<int>> tree;
  vector<int> len,link;
  string s; // 1-indexed
  PalindromicTree(string str){
    s="$"+str;
    n=s.size();
    len.assign(n+5,0);
    link.assign(n+5,0);
    tree.assign(n+5,vector<int>(26,0));
  }
  void extend(int p){
    while(s[p-len[t]-1]!=s[p]) t=link[t];
    int x=link[t],c=s[p]-'a';
    while(s[p-len[x]-1]!=s[p]) x=link[x];
    if(!tree[t][c]){
      tree[t][c]=++idx;
      len[idx]=len[t]+2;
      link[idx]=len[idx]==1?2:tree[x][c];
    }
    t=tree[t][c];
  }
  void build(){
    len[1]=-1,link[1]=1;
    len[2]=0,link[2]=1;
    idx=t=2;
    for(int i=1;i<n;i++) extend(i);
  }
};
```

6.5 Suffix Array [78 lines]

```
struct SuffixArray {
vector<int> p, c, rank, lcp;
vector<vector<int>> st;
SuffixArray(string const& s) {
  build_suffix(s + char(1));
  p.erase(p.begin());
  build_rank(p.size());
  build_lcp(s);
  build_sparse_table(lcp.size());
}
void build_suffix(string const& s) {
  int n = s.size();
  const int MX_ASCII = 256;
  vector<int> cnt(max(MX_ASCII, n), 0);
  p.resize(n); c.resize(n);
  for (int i = 0; i < n; i++) cnt[s[i]]++;
  for (int i=1; i<MX_ASCII; i++) cnt[i]+=cnt[i-1];
  for (int i = 0; i < n; i++) p[--cnt[s[i]]] = i;
  c[p[0]] = 0;
  int classes = 1;
  for (int i = 1; i < n; i++) {
    if (s[p[i]] != s[p[i-1]]) classes++;
    c[p[i]] = classes - 1;
  }
  vector<int> pn(n), cn(n);
  for (int h = 0; (1 << h) < n; ++h) {
    for (int i = 0; i < n; i++) {
      pn[i] = p[i] - (1 << h);
      if (pn[i] < 0) pn[i] += n;
    }
    fill(cnt.begin(), cnt.begin() + classes, 0);
    for (int i = 0; i < n; i++) cnt[c[pn[i]]]++;
    for (int i=1; i<classes; i++) cnt[i]+=cnt[i-1];
    for (int i=n-1;i>=0;i--) p[--cnt[c[pn[i]]]]=pn[i];
    cn[p[0]] = 0; classes = 1;
    for (int i = 1; i < n; i++) {
      pair<int, int> cur = {c[p[i]], c[(p[i] + (1 <<
          h)) % n]};
      pair<int, int> prev = {c[p[i-1]], c[(p[i-1] + (1
          << h)) % n]};
      if (cur != prev) ++classes;
      cn[p[i]] = classes - 1;
    }
    c.swap(cn);
  }
}
void build_rank(int n) {
  rank.resize(n, 0);
  for (int i = 0; i < n; i++) rank[p[i]] = i;
}
void build_lcp(string const& s) {
  int n = s.size(), k = 0;
  lcp.resize(n - 1, 0);
  for (int i = 0; i < n; i++) {
    if (rank[i] == n - 1) {
      k = 0;
      continue;
    }
    int j = p[rank[i] + 1];
    while (i + k < n && j + k < n && s[i+k] == s[j+k])
      k++;
    lcp[rank[i]] = k;
    if (k) k--;
  }
}
void build_sparse_table(int n) {
  int lim = __lg(n);
  st.resize(lim + 1, vector<int>(n)); st[0] = lcp;
  for (int k = 1; k <= lim; k++)
    for (int i = 0; i + (1 << k) <= n; i++)
      st[k][i] = min(st[k - 1][i], st[k - 1][i + (1 <<
          (k - 1))]);
}
int get_lcp(int i) { return lcp[i]; }
int get_lcp(int i, int j) {
  if (j < i) swap(i, j);
  j--; /*for lcp from i to j we don't need last lcp*/
```

```
  int K = __lg(j - i + 1);
  return min(st[K][i], st[K][j - (1 << K) + 1]);
}
};
```

6.6 Trie [28 lines]

```
const int maxn=100005;
struct Trie{
  int next[27][maxn];
  int endmark[maxn],sz;
  bool created[maxn];
  void insertTrie(string& s){
    int v=0;
    for(int i=0;i<(int)s.size();i++){
      int c=s[i]-'a';
      if(!created[next[c][v]]){
        next[c][v]=++sz;
        created[sz]=true;
      }
      v=next[c][v];
    }
    endmark[v]++;
  }
  bool searchTrie(string& s){
    int v=0;
    for(int i=0;i<(int)s.size();i++){
      int c=s[i]-'a';
      if(!created[next[c][v]])
        return false;
      v=next[c][v];
    }
    return(endmark[v]>0);
  }
};
```

6.7 Z-Algorithm [19 lines]

```
void compute_z_function(const char*S,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;
      Z[i]=R-L,--R;
    }
    else{
      int k=i-L;
      if(Z[k]<R-i+1)Z[i]=Z[k];
      else{
        L=i;
        while(R<N && S[R-k]==S[R])++R;
        Z[i]=R-L,--R;
      }
    }
  }
}
```

7 Geometry

7.1 Geometry [384 lines]

```
namespace Geometry
{
  #define M_PI(acos(-1.0))
  double eps=1e-8;
  typedef double T;//coordinate point type
  struct pt //Point
  {
    T x,y;
    pt(){}
    pt(T _x,T _y):x(_x),y(_y){}
    pt operator+(pt p){
      return{x+p.x,y+p.y};
    }
    pt operator-(pt p){
      return{x-p.x,y-p.y};
    }
    pt operator*(T d){
      return{x*d,y*d};
    }
    pt operator*(pt d){/*I added for General linear
        transformation,not sure about that function*/
      return{x*d.x,y*d.y};
    }
    pt operator/(T d){
      return{x/d,y/d};/*only for floating point*/
    }
    pt operator/(pt d){/*I added for General linear
        transformation,not sure about that function*/
      return{x/d.x,y/d.y};
    }
    bool operator<(const pt& p)const {
      if(x!=p.x)
        return x<p.x;
      return y<p.y;
    }
    bool operator==(pt b){
      return x==b.x && y==b.y;
    }
    bool operator!=(pt b){
      return!(*(this)==b);
    }
    friend ostream& operator<<(ostream& os,const pt
        p){
      return os<<"("<<p.x<<","<<p.y<<")";
    }
    friend istream& operator>>(istream& is,pt &p){
      is>>p.x>>p.y;
      return is;
    }
};
 T sq(pt p){
   return p.x*p.x+p.y*p.y;
 }
 double Abs(pt p){
   return sqrtl(sq(p));
 }
 pt translate(pt v,pt p){ /*To translate an object by
     a vector v*/
   return p+v;
 }
 pt scale(pt c,double factor,pt p){/*To scale an
     object by a certain ratio factor around a
     center*/
   return c+(p-c)*factor;
 }
 pt rot(pt p,double a){/*To rotate a point by angle
     a*/
   return{p.x*cos(a)-p.y*sin(a),p.x*sin(a)+p.y*
       cos(a)};
 }
 pt perp(pt p){/*To rotate a point 90 degree*/
   return{-p.y,p.x};
 }
 pt linearTransfo(pt p,pt q,pt r,pt fp,pt fq){/*so
     far don't know about that function*/
   return fp+(r-p)*(fq-fp)/(q-p);
 }
 T dot(pt v,pt w){
   return v.x*w.x+v.y*w.y;
 }
 bool isPerp(pt v,pt w){
   return dot(v,w)==0;
```

```
}
double angle(pt v,pt w){/*Find the smallest angle of
    two vector*/
  double cosTheta=dot(v,w)/Abs(v)/Abs(w);
  return acos(max(-1.0,min(1.0,cosTheta)));
}
T cross(pt v,pt w){
  return v.x*w.y-v.y*w.x;
}
T orient(pt a,pt b,pt c){
  return cross(b-a,c-a); /*if c is left side+ve,c is
      right side-ve,on line 0*/
}
bool inAngle(pt a,pt b,pt c,pt p){/*if p is in the
    angle*/
  assert(orient(a,b,c)!=0);
  if(orient(a,b,c)<0)
    swap(b,c);
  return orient(a,b,p)>=0 && orient(a,c,p)<=0;
}
double orientedAngle(pt a,pt b,pt c){/*the actual
    angle from ab to ac*/
  if(orient(a,b,c)>=0)
      return angle(b-a,c-a);
  else
      return 2*M_PI-angle(b-a,c-a);
}
///line
struct line{
  pt v;
  T c;
  line(){}
  line(pt p,pt q){/*From points P and Q*/
    v=(q-p),this->c=cross(v,p);
  }
  line(T a,T b,T c){/*From equation ax+by=c*/
    v=pt(b,-a),this->c=c;
  }
  line(pt v,T c){/*From direction vector v and
      offset c*/
    this->v=v,this->c=c;
  }
  double getY(double x){/*self made,not sure if it
      is okay*/
    assert(v.x!=0);
    double ret=(double)(c+v.y*x)/v.x;
    return ret;
  }
  double getX(double y){/*self made,not sure if it
      is okay*/
    assert(v.y!=0);
    double ret=(double)(c-v.x*y)/-v.y;
    return ret;
  }
  T side(pt p){/*which side a point is*/
     return cross(v,p)-c;
  }
  double dist(pt p){/*point to line dist*/
    return abs(side(p))/Abs(v);
  }
  double sqDist(pt p){/*square dist*/
    return side(p)*side(p)/(double)sq(v);
  }
  line perpThrough(pt p){/*perpendicular line with
      point p*/
      return line(p,p+perp(v));
  }
  bool cmpProj(pt p,pt q){/*compare function to sort
      points on a line*/
      return dot(v,p)<dot(v,q);
  }
  line translate(pt t){/*translate with vector t*/
      return line(v,c+cross(v,t));
  }
  line shiftLeft(double dist){/*translate with
      distance dist*/
      return line(v,c+dist*Abs(v));
  }
  pt proj(pt p){
      return p-perp(v)*side(p)/sq(v);
  }
  pt refl(pt p){
      return p-perp(v)*2*side(p)/sq(v);
  }
};
bool areParallel(line l1,line l2){
  return(l1.v.x*l2.v.y==l1.v.y*l2.v.x);
}
bool areSame(line l1,line l2){
  return areParallel(l1,l2)and(l1.v.x*l2.c==l2.v.x*
      l1.c)and(l1.v.y*l2.c==l2.v.y*l1.c);
}
bool inter(line l1,line l2,pt& out){
  T d=cross(l1.v,l2.v);
  if(d==0)return false;
  out=(l2.v*l1.c-l1.v*l2.c)/d;
  return true;
}
line intBisector(line l1,line l2,bool interior){/*if
    change sign then returns the other one*/
  assert(cross(l1.v,l2.v)!=0);
  double sign=interior?1:-1;
  return line(l2.v/Abs(l2.v)+l1.v*sign/Abs(l1.v),
          l2.c/Abs(l2.v)+l1.c*sign/Abs(l1.v));
}
//segment
bool inDisk(pt a,pt b,pt p){/*check weather point p
    is in diameter AB*/
  return dot(a-p,b-p)<=0;
}
bool onSegment(pt a,pt b,pt p){/*check weather point
    p is in segment AB*/
  return orient(a,b,p)==0 and inDisk(a,b,p);
}
bool properInter(pt a,pt b,pt c,pt d,pt& i){
  double oa=orient(c,d,a),
         ob=orient(c,d,b),
         oc=orient(a,b,c),
         od=orient(a,b,d);
//Proper intersection exists iff opposite signs
  if(oa*ob<0 and oc*od<0){
    i=(a*ob-b*oa)/(ob-oa);
    return 1;
  }
  return 0;
}
/*To create sets of points we need a comparison
    function*/
struct cmpX{
  bool operator()(pt a,pt b){
      return make_pair(a.x,a.y)<make_pair(b.x,b.y);
  }
};
set<pt,cmpX>inters(pt a,pt b,pt c,pt d){
  pt out;
  if(properInter(a,b,c,d,out))
    return{out};
```

```
  set<pt,cmpX>s;
  if(onSegment(c,d,a))s.insert(a);
  if(onSegment(c,d,b))s.insert(b);
  if(onSegment(a,b,c))s.insert(c);
  if(onSegment(a,b,d))s.insert(d);
  return s;
}
bool LineSegInter(line l,pt a,pt b,pt& out){
  if(l.side(a)*l.side(b)>eps)return 0;
  return inter(l,line(a,b),out);
}
double segPoint(pt a,pt b,pt p){/*returns distance
    from a point p to segment AB*/
  if(a!=b){
      line l(a,b);
      if(l.cmpProj(a,p)and l.cmpProj(p,b))
        return l.dist(p);
  }
  return min(Abs(p-a),Abs(p-b));
}
double segSeg(pt a,pt b,pt c,pt d){/*returns
    distance from a segment AB to segment CD*/
  pt dummy;
  if(properInter(a,b,c,d,dummy))return 0;
  return min(min(min(segPoint(a,b,c),segPoint(a,b,
      d)),segPoint(c,d,a)),segPoint(c,d,b));
}
/*int latticePoints(pt a,pt b){
  // requires int representation
  return __gcd(abs(a.x-b.x),abs(a.y-b.y))+1;
} // A = i+(b/2)-1; here
  A=area,i=pointsinside,b=pointsonline
Polygon*/
bool isConvex(vector<pt>&p){
  bool hasPos=0,hasNeg=0;
  for(int i=0,n=p.size();i<n;i++){
    int o=orient(p[i],p[(i+1)%n],p[(i+2)%n]);
    if(o>0)hasPos=1;
    if(o<0)hasNeg=true;
  }
  return!(hasPos and hasNeg);
}
double areaTriangle(pt a,pt b,pt c){
  return abs(cross(b-a,c-a))/2.0;
}
double areaPolygon(const vector<pt>&p){
  double area=0.0;
  for(int i=0,n=p.size();i<n;i++){
    area+=cross(p[i],p[(i+1)%n]);
  }
  return fabs(area)/2.0;
}
bool pointInPolygon(const vector<pt>&p,pt
    q){/*returns true if pt q is in polygon p*/
  bool c=false;
  for(int i=0,n=p.size();i<n;i++){
    int j=(i+1)%p.size();
    if((p[i].y<=q.y and q.y<p[j].y or p[j].y<=q.y
        and q.y<p[i].y)and
      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;
}
ll is_point_in_convex(vector<pt>& p, pt &x) { //
    O(log n)
    ll n = p.size(); /*this function from
        YouKnowWho*/
    if (n < 3) return 1;
    ll a =orient(p[0], p[1], x), b = orient(p[0],
        p[n - 1], x);
    if (a < 0 || b > 0) return 1;
    ll l = 1, r = n - 1;
    while (l + 1 < r) {
        int mid = l + r >> 1;
        if (orient(p[0], p[mid], x) >= 0) l = mid;
        else r = mid;
    }
    ll k = orient(p[l], p[r], x);
    if (k <= 0) return -k;
    if (l == 1 && a == 0) return 0;
    if (r == n - 1 && b == 0) return 0;
    return -1;
}
pt centroidPolygon(vector<pt>&p){/*from rezaul,i
    don't know about that*/
  pt c(0,0);
  double scale=6.0*areaPolygon(p);
//  if(scale<eps)return c;
  for(int i=0,n=p.size();i<n;i++){
    int j=(i+1)%n;
    c=c+(p[i]+p[j])*cross(p[i],p[j]);
  }
  return c/scale;
}
///Circle
pt circumCenter(pt a,pt b,pt c){/*return the center
    of the circle go through point a,b,c*/
  b=b-a,c=c-a;
  assert(cross(b,c)!=0);
  return a+perp(b*sq(c)-c*sq(b))/cross(b,c)/2;
}
bool circle2PtsRad(pt p1,pt p2,double r,pt& c){
  double d2=sq(p1-p2);
  double det=r*r/d2-0.25;
  if(det<0.0)return false;
  double h=sqrt(det);
  c.x=(p1.x+p2.x)*0.5+(p1.y-p2.y)*h;
  c.y=(p1.y+p2.y)*0.5+(p2.x-p1.x)*h;
  return true;
}
int circleLine(pt c,double r,line l,pair<pt,pt>&
    out){/*circle line intersection*/
  double h2=r*r-l.sqDist(c);
  if(h2<0)return 0;/*the line doesn't touch the
      circle;*/
  pt p=l.proj(c);
  pt h=l.v*sqrt(h2)/Abs(l.v);
  out=make_pair(p-h,p+h);
  return 1+(h2>0);
}
int circleCircle(pt c1,double r1,pt c2,double
    r2,pair<pt,pt>& out){/*circle circle
    intersection*/
  pt d=c2-c1;
  double d2=sq(d);
  if(d2==0){//concentric circles
    assert(r1!=r2);
    return 0;
  }
  double pd=(d2+r1*r1-r2*r2)/2;
  double h2=r1*r1-pd*pd/d2;//h ^ 2
  if(h2<0)return 0;
  pt p=c1+d*pd/d2,h=perp(d)*sqrt(h2/d2);
  out=make_pair(p-h,p+h);
  return 1+h2>0;
```

```
}
int tangents(pt c1,double r1,pt c2,double r2,bool
    inner,vector<pair<pt,pt>>&out){
  if(inner)r2=-r2;/*returns tangent(the line which
      touch a circle in one point)of two circle*/
  pt d=c2-c1;/*the same code can be used to find the
      tangent to a circle passing through a point by
      setting r2 to 0*/
  double dr=r1-r2,d2=sq(d),h2=d2-dr*dr;
  if(d2==0 or h2<0){
    assert(h2!=0);
    return 0;
  }
  for(int sign :{-1,1}){
      pt v=pt(d*dr+perp(d)*sqrt(h2)*sign)/d2;
      out.push_back(make_pair(c1+v*r1,c2+v*r2));
  }
  return 1+(h2>0);
}
//Convex Hull-Monotone Chain
pt H[100000+5];
vector<pt>monotoneChain(vector<pt>&points){
  sort(points.begin(),points.end());
  vector<pt>ret;
  ret.clear();
  int st=0;
  for(int i=0,sz=points.size();i<sz;i++){
    while(st>=2 and
        orient(H[st-2],H[st-1],points[i])<0)st--;
    H[st++]=points[i];
  }
  int taken=st-1;
  for(int i=points.size()-2;i>=0;i--){
    while(st>=taken+2 and
        orient(H[st-2],H[st-1],points[i])<0)st--;
    H[st++]=points[i];
  }
  for(int i=0;i<st;i++)ret.push_back(H[i]);
  return ret;
}
//Convex Hull-Monotone Chain from you_know_who
int sz;
vector<pt> monotoneChain(vector<pt> &v) {
    if(v.size()==1) return v;
    sort(v.begin(), v.end());
    vector<pt> up(2*v.size()+2), down(2*v.size()+2);
    int szup=0, szdw=0;
    for(int i=0;i<v.size();i++) {
        while(szup>1 && orient(up[szup-2],
            up[szup-1], v[i])>=0)
            szup--;
        while(szdw>1 && orient(down[szdw-2],
            down[szdw-1], v[i])<=0)
            szdw--;
        up[szup++]=v[i];
        down[szdw++]=v[i];
    }
    if(szdw>1) szdw--;
    reverse(up.begin(), up.begin()+szup);
    for(int i=0;i<szup-1;i++) down[szdw++]= up[i];
    if(szdw==2 && down[0].x==down[1].x &&
        down[0].y==down[1].y)
        szdw--;
    sz = szdw;
    return down;
}
double cosA(double a,double b,double c){
    double val=b*b+c*c-a*a;
    val/=(2*b*c);
    return acos(val);
  }
  double triangle(double a,double b,double c){
    double s=(a+b+c)/2;
    return sqrtl(s*(s-a)*(s-b)*(s-c));
  }
}
using namespace Geometry;
```

7.2 Rotation Matrix [39 lines]

```
struct { double x; double y; double z; } Point;
double rMat[4][4];
double inMat[4][1] = {0.0, 0.0, 0.0, 0.0};
double outMat[4][1] = {0.0, 0.0, 0.0, 0.0};
void mulMat() {
  for(int i = 0; i < 4; i++ ){
    for(int j = 0; j < 1; j++){
      outMat[i][j] = 0;
      for(int k = 0; k < 4; k++)
        outMat[i][j] += rMat[i][k] * inMat[k][j];
    }
  }
}
void setMat(double ang, double u, double v, double w){
  double L = (u * u + v * v + w * w);
  ang = ang * PI / 180.0; /*converting to radian
      value*/
  double u2 = u*u; double v2 = v*v; double w2 = w*w;
  rMat[0][0]=(u2+(v2+w2)*cos(ang))/L;
  rMat[0][1]=(u*v*(1-cos(ang))-w*sqrt(L)*sin(ang))/L;
  rMat[0][2]=(u*w*(1-cos(ang))+v*sqrt(L)*sin(ang))/L;
  rMat[0][3]=0.0;
  rMat[1][0]=(u*v*(1-cos(ang))+w*sqrt(L)*sin(ang))/L;
  rMat[1][1]=(v2+(u2+w2)*cos(ang))/L;
  rMat[1][2]=(v*w*(1-cos(ang))-u*sqrt(L)*sin(ang))/L;
  rMat[1][3]=0.0;
  rMat[2][0]=(u*w*(1-cos(ang))-v*sqrt(L)*sin(ang))/L;
  rMat[2][1]=(v*w*(1-cos(ang))+u*sqrt(L)*sin(ang))/L;
  rMat[2][2]=(w2 + (u2 + v2) * cos(ang)) / L;
  rMat[2][3]=0.0; rMat[3][0]=0.0; rMat[3][1]=0.0;
  rMat[3][2]=0.0; rMat[3][3]=1.0;
}
/*double ang;
  double u, v, w; //points = the point to be rotated
  Point point, rotated; //u,v,w=unit vector of line
  inMat[0][0] = points.x; inMat[1][0] = points.y;
  inMat[2][0] = points.z; inMat[3][0] = 1.0;
  setMat(ang, u, v, w); mulMat();
  rotated.x = outMat[0][0]; rotated.y = outMat[1][0];
  rotated.z = outMat[2][0];*/
```

Theoretical Computer Science Cheat Sheet

Definitions

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

Series

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

In general:

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

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

Geometric series:

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

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

Harmonic series:

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

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

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. $\left\langle\!\left\langle{n \atop n}\right\rangle\!\right\rangle = 0 \quad \text{for } n \ne 0$
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. $\sum_{k=0}^{n}\left\langle\!\left\langle{n \atop k}\right\rangle\!\right\rangle = \frac{(2n)!}{2^n}$
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

Calculus Cont.

62. $\int \frac{dx}{x\sqrt{x^2-a^2}} = \frac{1}{a}\arccos\frac{a}{|x|}, \quad a > 0,$
63. $\int \frac{dx}{x^2\sqrt{x^2\pm a^2}} = \mp\frac{\sqrt{x^2\pm a^2}}{a^2x},$
64. $\int \frac{x\,dx}{\sqrt{x^2\pm a^2}} = \sqrt{x^2\pm a^2},$
65. $\int \frac{\sqrt{x^2\pm a^2}}{x^4}\,dx = \mp\frac{(x^2+a^2)^{3/2}}{3a^2x^3},$
66. $\int \frac{dx}{ax^2+bx+c} = \begin{cases}\frac{1}{\sqrt{b^2-4ac}}\ln\left|\frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}}\right|, & \text{if } b^2 > 4ac,\\ \frac{2}{\sqrt{4ac-b^2}}\arctan\frac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2 < 4ac,\end{cases}$
67. $\int \frac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases}\frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c}\right|, & \text{if } a > 0,\\ \frac{1}{\sqrt{-a}}\arcsin\frac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a < 0,\end{cases}$
68. $\int \sqrt{ax^2+bx+c}\,dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$
69. $\int \frac{x\,dx}{\sqrt{ax^2+bx+c}} = \frac{\sqrt{ax^2+bx+c}}{a} - \frac{b}{2a}\int\frac{dx}{\sqrt{ax^2+bx+c}},$
70. $\int \frac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases}\frac{-1}{\sqrt{c}}\ln\left|\frac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x}\right|, & \text{if } c > 0,\\ \frac{1}{\sqrt{-c}}\arcsin\frac{bx+2c}{|x|\sqrt{b^2-4ac}}, & \text{if } c < 0,\end{cases}$
71. $\int x^3\sqrt{x^2+a^2}\,dx = (\tfrac{1}{3}x^2 - \tfrac{2}{15}a^2)(x^2+a^2)^{3/2},$
72. $\int x^n\sin(ax)\,dx = -\frac{1}{a}x^n\cos(ax) + \frac{n}{a}\int x^{n-1}\cos(ax)\,dx,$
73. $\int x^n\cos(ax)\,dx = \frac{1}{a}x^n\sin(ax) - \frac{n}{a}\int x^{n-1}\sin(ax)\,dx,$
74. $\int x^n e^{ax}\,dx = \frac{x^n e^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,dx,$
75. $\int x^n\ln(ax)\,dx = x^{n+1}\left(\frac{\ln(ax)}{n+1} - \frac{1}{(n+1)^2}\right),$
76. $\int x^n(\ln ax)^m\,dx = \frac{x^{n+1}}{n+1}(\ln ax)^m - \frac{m}{n+1}\int x^n(\ln ax)^{m-1}\,dx.$

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:

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

$x^{\underline{1}} =$	$x^{\underline{1}}$	$=$	$x^{\overline{1}}$
$x^{\underline{2}} =$	$x^2 + x^{\underline{1}}$	$=$	$x^{\overline{2}} - x^{\overline{1}}$
$x^{\underline{3}} =$	$x^3 + 3x^2 + x^{\underline{1}}$	$=$	$x^{\overline{3}} - 3x^{\overline{2}} + x^{\overline{1}}$
$x^{\underline{4}} =$	$x^4 + 6x^3 + 7x^2 + x^{\underline{1}}$	$=$	$x^{\overline{4}} - 6x^{\overline{3}} + 7x^{\overline{2}} - x^{\overline{1}}$
$x^{\underline{5}} =$	$x^5 + 15x^4 + 25x^3 + 10x^2 + x^{\underline{1}}$	$=$	$x^{\overline{5}} - 15x^{\overline{4}} + 25x^{\overline{3}} - 10x^{\overline{2}} + x^{\overline{1}}$
$x^{\overline{1}} =$	x^1	$x^{\underline{1}} =$	x^1
$x^{\overline{2}} =$	$x^2 + x^1$	$x^{\underline{2}} =$	$x^2 - x^1$
$x^{\overline{3}} =$	$x^3 + 3x^2 + 2x^1$	$x^{\underline{3}} =$	$x^3 - 3x^2 + 2x^1$
$x^{\overline{4}} =$	$x^4 + 6x^3 + 11x^2 + 6x^1$	$x^{\underline{4}} =$	$x^4 - 6x^3 + 11x^2 - 6x^1$
$x^{\overline{5}} =$	$x^5 + 10x^4 + 35x^3 + 50x^2 + 24x^1$	$x^{\underline{5}} =$	$x^5 - 10x^4 + 35x^3 - 50x^2 + 24x^1$

- $\varphi(n) = \sum_{d\mid n} d^k\mu(\frac{n}{d})$
- $\sum_{d\mid n}\frac{\mu^2(d)}{\varphi(d)} = \frac{n}{\varphi(n)}$
- $\sum_{\substack{1\le k\le n\\(k,n)=1}} k = \frac{1}{2}n\varphi(n) \quad \text{for } n > 1$

$$\frac{\varphi(n)}{n} = \frac{\varphi(\mathrm{rad}(n))}{\mathrm{rad}(n)} \quad \text{where,}$$

$$\mathrm{rad}(n) = \prod_{\substack{p\mid n\\ p \text{ prime}}} p$$

- $\left\lfloor\frac{n}{\varphi(n)}\right\rfloor$ is periodic. 1,2,1,2,1,3,1,2,1,2,1,3...

$$\sum_{\substack{1\le k\le n\\ \gcd(k,n)=1}} \gcd(k-1,n) = \varphi(n)d(n)$$

where d(n) is number of divisors.

same equation for gcd(ak-1,n) where a and n are coprime.

- for every n there is at least one other integer m ≠ n such that ϕ(m) = ϕ(n).

Divisor function

$$\sigma_x(n) = \sum_{d\mid n} d^x.$$

- **It is multiplicative** i.e. if $\gcd(a,b) = 1 \Longrightarrow \sigma_x(ab) = \sigma_x(a)\sigma_x(b)$.

coprime $(k+1)$-tuple together with n. It is a generalization of Euler's totient, $\phi(n) = J_1(n)$.

$$J_k(n) = n^k\prod_{p\mid n}\left(1 - \frac{1}{p^k}\right)$$

- $\sum_{d\mid n} J_k(d) = n^k$
- $\sum_{d\mid n}\varphi(d) = n$
- $\varphi(n) = \sum_{d\mid n}\mu(d)\cdot\frac{n}{d} = n\sum_{d\mid n}\frac{\mu(d)}{d}$
- $a \mid b \Longrightarrow \varphi(a) \mid \varphi(b)$
- $n \mid \varphi(a^n - 1)$ for $a, n > 1$
- $\varphi(mn) = \varphi(m)\varphi(n)\cdot\frac{d}{\varphi(d)}$ where $d = \gcd(m,n)$

Note the special cases

- $\varphi(2m) = \begin{cases}2\varphi(m) & \text{if } m \text{ is even}\\ \varphi(m) & \text{if } m \text{ is odd}\end{cases}$
- $\varphi(n^m) = n^{m-1}\varphi(n)$
- $\varphi(\mathrm{lcm}(m,n))\cdot\varphi(\gcd(m,n)) = \varphi(m)\cdot\varphi(n)$

Compare this to the formula

- $\mathrm{lcm}(m,n)\cdot\gcd(m,n) = m\cdot n$

(See least common multiple.)

- $\varphi(n)$ is even for $n \ge 3$. Moreover, if n has r distinct odd prime factors, $2^r \mid \varphi(n)$

- Any three consecutive Fibonacci numbers are pairwise coprime, which means that, for every n, $\gcd(F_n, F_{n+1}) = \gcd(F_n, F_{n+2}) = \gcd(F_{n+1}, F_{n+2}) = 1$.
- If p is a prime,

$$\begin{cases} p = 5 & \Rightarrow p \mid F_p, \\ p \equiv \pm 1 \pmod 5 & \Rightarrow p \mid F_{p-1} \\ p \equiv \pm 2 \pmod 5 & \Rightarrow p \mid F_{p+1} \end{cases}$$

- The only nontrivial square Fibonacci number is 144. Attila Pethő proved in 2001 that there is only a finite number of perfect power Fibonacci numbers.In 2006, Y. Bugeaud, M. Mignotte, and S. Siksek proved that 8 and 144 are the only such non-trivial perfect powers.
- If the members of the Fibonacci sequence are taken mod n, the resulting sequence is periodic with period at most 6n.

Sum of floors

$\sum_{i=1}^{n} \left\lfloor \frac{n}{i} \right\rfloor$=?

```
int32_t main()
{
  BeatMeScanf;
  int i,j,k,n,m;
  cin>>n;
  ///complexity O(sqrt(n))
  for (int i = 1, last; i <= n; i = last + 1) {
    last = n / (n / i);
    debug(i,last,n/i);
    ///n / x yields the same value for i <= x <= la.
  }
  return 0;
}
```

Mobius Function and Inversion

Notes

- For any positive integer n, define μ(n) as the sum of the primitive nth roots of unity. It has values in {−1, 0, 1} depending on the factorization of n into prime factors:
  - ✓ μ(n) = 1 if n is a square-free positive integer with an even number of prime factors.
  - ✓ μ(n) = −1 if n is a square-free positive integer with an odd number of prime factors.
  - ✓ μ(n) = 0 if n has a squared prime factor.

  Here, a root of unity, occasionally called a de Moivre number, is any complex number that gives 1 when raised to some positive integer power n.
  An nth root of unity, where n is a positive integer (i.e. n = 1,2,3,...), is a number z(maybe complex) satisfying the equation $z^n = 1$.
  An nth root of unity is said to be primitive if it is not a kth root of unity for some smaller k, that is if
  $z^n = 1 \quad \text{and} \quad z^k \neq 1 \text{ for } k = 1, 2, 3, \ldots, n-1.$
- It is a multiplicatuve function.
- $\sum_{d \mid n} \mu(d) = \begin{cases} 1 & \text{if } n = 1, \\ 0 & \text{if } n > 1. \end{cases}$

- $\sum_{i=1}^{n} [\gcd(i,n) = k] = \varphi(\frac{n}{k})$
- $\sum_{k=1}^{n} \gcd(k,n) = \sum_{d \mid n} d \cdot \varphi(\frac{n}{d})$
- $\sum_{k=1}^{n} \frac{1}{\gcd(k,n)} = \sum_{d \mid n} \frac{1}{d} \cdot \varphi\left(\frac{n}{d}\right) = \frac{1}{n} \sum_{d \mid n} d \cdot \varphi(d)$
- $\sum_{k=1}^{n} \frac{k}{\gcd(k,n)} = \frac{n}{2} \cdot \sum_{d \mid n} \frac{1}{d} \cdot \varphi\left(\frac{n}{d}\right) = \frac{n}{2} \cdot \frac{1}{n} \cdot \sum_{d \mid n} d \cdot \varphi(d)$
- $\sum_{k=1}^{n} \frac{n}{\gcd(k,n)} = 2 * \sum_{k=1}^{n} \frac{k}{\gcd(k,n)} - 1 \, , for \; n > 1$
-
- Given several integers, with integer x appears c_x times, and some fixed integer m. It is asked that how many integers that are co-prime to m.so,

$$\sum_{i=1}^{n} c_i [\gcd(i, m) = 1] = \sum_{d \mid m} \mu(d) \sum_{i=1}^{\lfloor n/d \rfloor} c_{id}$$

The classic version states that if g and f are arithmetic functions satisfying

$$g(n) = \sum_{d \mid n} f(d) \quad \text{for every integer } n \geq 1$$

then

$$f(n) = \sum_{d \mid n} \mu(d) g\left(\frac{n}{d}\right) \quad \text{for every integer } n \geq 1$$

-
- $\sum_{d \mid n} \mu(d) = [n = 1]$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} [\gcd(i,j) = 1] = \sum_{d=1}^{n} \mu(d) \left\lfloor \frac{n}{d} \right\rfloor^2$
- $\sum_{i=1}^{n} \sum_{j=1}^{n} \gcd(i,j) = \sum_{d=1}^{n} \varphi(d) \left\lfloor \frac{n}{d} \right\rfloor^2$
- if $F(n) = \prod_{d \mid n} f(d)$, then $f(n) = \prod_{d \mid n} F\left(\frac{n}{d}\right)^{\mu(d)}$

mobius function

```
int mob[N];
void mobius()
{
  for(int i=1;i<N;i++) mob[i]=3;
  mob[1]=1;
  for(int i=2;i<N;i++){
    if(mob[i]==3){
      mob[i]=-1;
      for(int j=2*i;j<N;j+=i) mob[j]=(mob[j]==3?-1:mob[j]*(-1));
      if(i<=(N-1)/i) for(int j=i*i;j<N;j+=i*i) mob[j]=0;
    }
  }
}
```

GCD and LCM

- $\gcd(a, 0) = a$
- $\gcd(a, b) = \gcd(b, a \bmod b)$
- Every common divisor of a and b is a divisor of gcd(a, b).
- If a divides the product b·c, and gcd(a, b) = d, then a/d divides c.
- If m is any integer, then gcd(a + m·b, b) = gcd(a, b)
- The gcd is a multiplicative function in the following sense: if a1 and a2 are relatively prime, then gcd(a1·a2, b) = gcd(a1, b)·gcd(a2, b).
- gcd(a, b)·lcm(a, b) = |a·b|
- gcd(a, lcm(b, c)) = lcm(gcd(a, b), gcd(a, c))
- lcm(a, gcd(b, c)) = gcd(lcm(a, b), lcm(a, c)).
- For non-negative integers a and b, where a and b are not both zero,

  $\gcd(n^a - 1, n^b - 1) = n^{\gcd(a,b)} - 1.$

- $r_8(n) = 16 \sum_{d \mid n} (-1)^{n+d} d^3$
-

Gauss Circle Theorem

- The Gauss circle problem is the problem of determining how many integer lattice points there are in a circle centered at the origin and with radius r.
- Since the equation of this circle is given in Cartesian coordinates by $x^2+y^2=r^2$, the question is equivalently asking how many pairs of integers *m* and *n* there are such that $m^2+n^2<=r^2$
- If the answer for a given *r* is denoted by *N*(*r*) then

$$N(r) = 1 + 4 \sum_{i=0}^{\infty} \left(\left\lfloor \frac{r^2}{4i+1} \right\rfloor - \left\lfloor \frac{r^2}{4i+3} \right\rfloor \right)$$

- A much simpler sum appears if the sum of squares function r2(n) is defined as the number of ways of writing the number n as the sum of two squares. Then

$$N(r) = \sum_{n=0}^{r^2} r_2(n).$$

3. Combinatorics

Notes

- $\sum_{0 \leq k \leq n} \binom{n-k}{k}$ =Fib$_{n+1}$
- $\binom{n}{k} = \binom{n}{n-k}$
- $\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$
- $k \binom{n}{k} = n \binom{n-1}{k-1}$
- $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}$
- $\sum_{i=0}^{n} \binom{n}{i} = 2^n$
- $\sum_{i \geq 0} \binom{n}{2i} = 2^{n-1}$
- $\sum_{i \geq 0} \binom{n}{2i+1} = 2^{n-1}$
- $\sum_{i=0}^{k} (-1)^i \binom{n}{i} = (-1)^k \binom{n-1}{k}$
- $\sum_{i=0}^{k} \binom{n+i}{i} = \binom{n+k+1}{k}$
- $\sum_{i=0}^{k} \binom{n+i}{n} = \binom{n+k+1}{k}$
- $\sum_{i=0}^{k} \binom{i}{n} = \binom{k+1}{n+1}$
- $1 \cdot \binom{n}{1} + 2 \cdot \binom{n}{2} + 3 \cdot \binom{n}{3} + \ldots + n \cdot \binom{n}{n} = n \cdot 2^{n-1}$
- $1^2 \cdot \binom{n}{1} + 2^2 \cdot \binom{n}{2} + 3^2 \cdot \binom{n}{3} + \ldots + n^2 \cdot \binom{n}{n} = (n + n^2) \cdot 2^{n-2}$
- Vandermonde's Identity: $\sum_{k=0}^{r} \binom{m}{k} \binom{n}{r-k} = \binom{m+n}{r}$
- Hockey-Stick Identity: $n, r \in \mathbb{N}, n > r, \sum_{i=r}^{n} \binom{i}{r} = \binom{n+1}{r+1}$
- $\sum_{i=0}^{k} \binom{k}{i}^2 = \binom{2k}{k}$
- $\sum_{k=0}^{n} \binom{n}{k} \binom{n}{n-k} = \binom{2n}{n}$
- $\sum_{k=q}^{n} \binom{n}{k} \binom{k}{q} = 2^{n-q} \binom{n}{q}$
- $\sum_{i=0}^{n} 3^i \binom{n}{i} = 4^i$
- $\sum_{i=0}^{n} \binom{2n}{i} = 2^{2n-1} + \frac{1}{2} \binom{2n}{n}$
- $\sum_{i=1}^{n} \binom{n}{i} \binom{n-1}{i-1} = \binom{2n-1}{n-1}$

- $\sum_{i=0}^{n}\binom{2n}{i}^2=\frac{1}{2}\left\{\binom{4n}{2n}+\binom{2n}{n}^2\right\}$
- An integer n ≥ 2 is prime if and only if all the intermediate binomial coefficients $\binom{n}{1},\binom{n}{2},\ldots,\binom{n}{n-1}$ are divisible by n.
- $\binom{n+k}{k}$ divides $\frac{lcm(n,n+1,\ldots,n+k)}{n}$
- Kummer's theorem states that for given integers n ≥ m ≥ 0 and a prime number p, the largest power of p dividing $\binom{n}{m}$ is equal to the number of carries when m is added to n − m in base p.
- Number of different binary sequences of length n such that no two 0's are adjacent=Fib_{n+1}
- Combination with repetition: Let's say we choose k elements from an n-element set, the order doesn't matter and each element can be chosen more than once. In that case, the number of different combinations is: $\binom{n+k-1}{k}$
- Number of ways to divide n different persons in n/k equal groups i.e. each having size k is $\binom{n-1}{k-1}$
- The number non-negative solution of the equation $x_1+x_2+x_3+\ldots+x_k=n$ is $\binom{n+k-1}{n}$
- Number of binary sequence of length n and with k '1' is $\binom{n}{k}$
- The number of ordered pairs (a, b) of binary sequences of length n, such that the distance between them is k, can be calculated as follows: $\binom{n}{k}\cdot 2^n$.
  The distance between a and b is the number of components that differs in a and b — for example, the distance between (0, 0, 1, 0) and (1, 0, 1, 1) is 2).

Catalan numbers

- ✓ $C_n=\frac{1}{n+1}\binom{2n}{n}$
- ✓ $C_0=1, C_1=1$ and $C_n=\sum_{k=0}^{n-1}C_kC_{n-1-k}$
- ✓ 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786
- ✓ Number of correct bracket sequence consisting of n opening and n closing brackets.
- ✓ The number of ways to completely parenthesize n+1 factors.
- ✓ The number of triangulations of a convex polygon with +2 sides (i.e. the number of partitions of polygon into disjoint triangles by using the diagonals).
- ✓ The number of ways to connect the 2n points on a circle to form n disjoint i.e. non-intersecting chords.
- ✓ The number of monotonic lattice paths from point (0,0) to point (n,n) in a square lattice of size n×n, which do not pass above the main diagonal (i.e. connecting (0,0) to (n,n)).
- ✓ Number of permutations of length n that can be stack sorted (i.e. it can be shown that the rearrangement is stack sorted if and only if there is no such index i<j<k, such that $a_k<a_i<a_j$).
- ✓ The number of non-crossing partitions of a set of n elements.
- ✓ The number of rooted full binary trees with n+1 leaves (vertices are not numbered). A rooted binary tree is full if every vertex has either two children or no children.
- ✓ The number of Dyck words of length 2n. A Dyck word is a string consisting of n X's and n Y's such that no initial segment of the string has more Y's than X's For example, the following are the Dyck words of length 6: XXXYYY XYXXYY XYXYXY XXYYXY XXYXYY.
- ✓ The number of different ways a convex polygon with n + 2 sides can be cut into triangles by connecting vertices with straight lines (a form of Polygon triangulation)
- ✓ Number of permutations of {1, ..., n} that avoid the pattern 123 (or any of the other patterns of length 3); that is, the number of permutations with no three-term increasing subsequence. For n = 3, these permutations are 132, 213, 231, 312 and 321. For n = 4, they are 1432, 2143, 2413, 2431, 3142, 3214, 3241, 3412, 3421, 4132, 4213, 4231, 4312 and 4321
- ✓ Number of ways to tile a stairstep shape of height n with n rectangles.
- ✓ $N(n,k)=\frac{1}{n}\binom{n}{k}\binom{n}{k-1}$
- ✓ The number of expressions containing *n* pairs of parentheses, which are correctly matched and which contain *k* distinct nestings. For instance, *N*(4, 2) = 6 as with four pairs of parentheses six sequences can be created which each contain two times the sub-pattern '()':

  ()(()()) (())(()) (()(())) ((()())) ((())()) (()()()

- ✓ The number of paths from (0, 0) to (2*n*, 0), with steps only northeast and southeast, not straying below the *x*-axis, with *k* peaks. And sum of all number of peaks is Catalan number.

Stirling numbers of the first kind

- ✓ The Stirling numbers of the first kind count permutations according to their number of cycles (counting fixed points as cycles of length one).
- ✓ S(n,k) counts the number of permutations of n elements with k disjoint cycles.
- ✓ $S(n,k)=(n-1)*S(n-1,k)+S(n-1,k-1),$ $where\ S(0,0)=1, S(n,0)=S(0,n)=0$
- ✓ $\sum_{k=0}^{n}S(n,k)=n!$

Stirling numbers of the second kind

- ✓ Stirling number of the second kind is the number of ways to partition a set of *n* objects into *k* non-empty subsets.
- ✓ $S(n,k)=k*S(n-1,k)+S(n-1,k-1),$ $where\ S(0,0)=1, S(n,0)=S(0,n)=0$
- ✓ $S(n,2)=2^{n-1}-1$
- ✓ S(n,k)*k! = number of ways to color n nodes using colors from 1 to k such that each color is used at least once.

Bell number

- ✓ Counts the number of partitions of a set.
- ✓ $B_{n+1}=\sum_{k=0}^{n}\binom{n}{k}*B_k$
- ✓ $B_n=\sum_{k=0}^{n}S(n,k)$,where S(n,k) is stirling number of second kind.
- ✓ The number of multiplicative partitions of a squarefree number with i prime factors is the i-th Bell number, B_i.
- ✓ If a deck of *n* cards is shuffled by repeatedly removing the top card and reinserting it anywhere in the deck (including its original position at the top of the deck), with exactly *n* repetitions of this operation, then there are n^n different shuffles that can be performed. Of these, the number that return the deck to its original sorted order is exactly B_n. Thus, the probability that the deck is in its original order after shuffling it in this way is B_n/n^n.

Lucas Theorem

- ✓ If p is prime the $\binom{p^a}{k}\equiv 0\ (mod\ p)$
- ✓ For non-negative integers m and n and a prime p, the following congruence relation holds:

$$\binom{m}{n}\equiv\prod_{i=0}^{k}\binom{m_i}{n_i}\ (mod\ p),$$

where,
$m=m_kp^k+m_{k-1}p^{k-1}+\ldots+m_1p+m_0,$
and
$n=n_kp^k+n_{k-1}p^{k-1}+\ldots+n_1p+n_0$
are the base *p* expansions of *m* and *n* respectively. This uses the convention that $\binom{m}{n}=0, when\ m<n.$

Derangement

- ✓ A derangement is a permutation of the elements of a set, such that no element appears in its original position.
- ✓ $d(n)=(n-1)*(d(n-1)+d(n-2)),$ $where\ d(0)=1, d(1)=0$
- ✓ $d(n)=\left\lfloor\frac{n!}{e}\right\rfloor, n\geq 1$

4. Burnside Lemma

The task is to count the number of different necklaces from n beads, each of which can be painted in one of the k colors. When comparing two necklaces, they can be rotated, but not reversed (i.e. a cyclic shift is permitted).
Solution: