Bugged Coders

1 Data sctrucutres

1.1 Segment tree

```
int nums[]={1,3,4,5,7};
struct segmentTree{
    int l, r,sum;
    segmentTree *nodeLeft,*nodeRight;
    segmentTree(int a, int b){
        l=a;
        r=b;
        int m=(l+r)/2;
        if(l!=r){
            nodeLeft=new segmentTree(l,m);
            nodeRight=new segmentTree(m+1,r);
            sum=nodeLeft->sum+nodeRight->sum;
        }
        else sum=nums[l];
    }
    int query(int a, int b){
        if(b<l || a>r) return 0;
        if(a<=l && r<=b) return sum;
        return nodeLeft->query(a,b)+nodeRight->query(a,b);
    }
    void update(int pos, int v){
        if(l!=r){
            int m=(l+r)/2;
            if(pos<=m) nodeLeft->update(pos,v);
            else nodeRight->update(pos,v);
            sum=nodeLeft->sum+nodeRight->sum;
        }
        else sum=v;
    }
};
```

1.2 Segment tree- Lazzy Propagation

```
int nums[]={1,3,5,7,9,11};
struct segmentTree{
    int l, r,sum,lazy;
    segmentTree *nodeLeft,*nodeRight;
    segmentTree(int a, int b){
        l=a;
        r=b;
        int m=(l+r)/2;
        lazy=0;
        if(l!=r){
            nodeLeft=new segmentTree(l,m);
            nodeRight=new segmentTree(m+1,r);
            sum=nodeLeft->sum+nodeRight->sum;
        }
        else sum=nums[l];
    }
    int query(int a, int b){
        if(nodeLeft!=nullptr && lazy!=0) nodeLeft->lazy=lazy;
        if(nodeRight!=nullptr && lazy!=0) nodeRight->lazy=lazy;
        sum+=(r-l+1)*lazy;lazy=0;
        if(b<l || a>r) return 0;
        if(a<=l && r<=b) return sum;
        return nodeLeft->query(a,b)+nodeRight->query(a,b);
    }
    int update(int a, int b, int v){
        int increment=0;
        if(b<l || a>r) return 0;
        if(a<=l && r<=b){
            if(nodeLeft!=nullptr) nodeLeft->lazy+=lazy;
            if(nodeRight!=nullptr) nodeRight->lazy+=lazy;
            increment=(r-l+1)*v;
            sum+=increment;
            return increment;
        }
        increment=nodeLeft->update(a,b,v)+nodeRight->update(a,b,v);
        sum+=increment;
        return increment;
    }
};
```

1.3 Disjoin Set

```
//Se usa para detectar cyclos en un grafo no dirigido convexo & en el
    algoritmo de Krustal.
vector<pair<int,int>>ds;
```

```cpp
void init(int n){
    ds.assign(n+1,{-1,0});
}
int find(int x){
    if(-1==ds[x].first) return x;
    return ds[x].first=find(ds[x].first);
}
bool unionDs(int x, int y){
    int px=find(x),py=find(y);
    int &rx=ds[px].second,&ry=ds[py].second;
    if(px==py) return false;
    else{
        if(rx>ry) ds[py].first=px;
        else{
            ds[px].first=py;
            if(rx==ry) ry+=1;
        }
    }
    return true;
}
```

1.4 Sparce Table

```cpp
//Se usa para RMQ porque se puede hacer en O(1), no acepta updates
vector<int>lg;
vector<vector<int>>st;
int *nums;
void init(int n){
    int logn=(int) log2(n)+1;
    lg.assign(n+1,0);
    st.assign(logn,vector<int>(n+1));
    for(int i=0;i<n;i++) st[0][i]=nums[i];
    lg[1]=0;
    for(int i=2;i<=n;i++) lg[i]=lg[i/2]+1;
    for(int i=1;i<logn;i++)
        for(int j=0;j+(1<<i)<n;j++)st[i][j]=min(st[i-1][j],st[i-1][j
            +(1<<(i-1))]);
}
int query(int a,int b){
    int logn=lg[(b-a+1)];
    cout<<st[logn][a]<<endl;
    return min(st[logn][a],st[logn][b-(1<<logn)+1]);
}
```

1.5 Treap

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef struct Node *pitem;
struct Node{
    int x,y;
    pitem l,r;
    Node(int v) x(v),y(rand()),l(nullptr),r(nullptr);
};
vector<int> rank;//1)optiona 2)intialize this array n+1
pair<pitem,pitem> split(pitem root,int value){
    pitem b=root->r;
    if(!root) return {nullptr,nullptr};
    if(root->x==value){
        pitem b=root->r;
        root->r=nullptr;
    }
    else{
        if(root->x>value) return split(root->l,value);
        else return split(root->r,value);
    }
    return {root,b};
}
void leftRotation(pitem x,int value){
    pitem y,a,b,c;
    tie(x,y)=split(x,value);
    tie(y,c)=split(y,y->y);
    a=x->l;b=y->l;
    x->r=c;x->l=y;
    y->l=a;y->r=b;
    swap(x->x,y->x);swap(x->y,y->y);
}
void rightRotation(pitem x,int value){
    pitem y=x->l,a,b,c;
    tie(x,c)=split(x,value);
    tie(y,b)=split(y,y->y);
    a=y->l;
    x->r=y;
    y->l=b;y->r=c;
    x->l=a;
    swap(x->x,y->x);swap(x->y,y->y);
}
```

```
void insert(pitem root,int value){
    if(!root){
        root=new Node(value);
        return;
    }
    insert((root->x>value?root->l,root->r),value);
    if(root->l && root->l->y>root->y) leftRotation(root,root->y);
    if(root->r && root->r->y>root->y) leftRotation(root,root->y);
}
//optional
int dfs(pitem root){
    if(root->l) rank[root->x]+=dfs(root->l);
    if(root->r) rank[root->x]+=dfs(root->r);
    return rank[root->x]+=1;
}
```

2 DP

2.1 Digit DP

```
ll dp[20][20][3];
ll n,k,d;
vector<int>num;
ll bk(int i, int len, int t){
    if(len>k) return 0;
    if(i==n){
        if(len==k) return 1;
        return 0;
    }
    ll &res=dp[i][len][t];
    if(res!=-1) return res;
    res=0;
    int tope;
    if(t==0) tope=num[i];
    else tope=9;
    for(int j=0;j<=tope;j++){
        int newt=t;
        int newlen=len;
        if(t==0 && j<tope) newt=1;
        if(d==j) newlen++;
        if(newlen<=k)res+=bk(i+1,newlen,newt);
    }
    return res;
}
ll rep(int a){
    num.clear();
    while(a>0){
        num.push_back(a%10);
        a/=10;
    }
    reverse(num.begin(),num.end());
    n=num.size();
    memset(dp,-1,sizeof(dp));
    return bk(0,0,0);
}
```

3 Graph

3.1 Krustal

```
// Este algoritmo sirve para buscar MST de un grafo convexo no dirigido
vector<tuple<int,int,int>>edges;
int n;m;
//Insertar Disjoin set
int krustal(){
    sort(edges.begin(),edges.end());
    int res=0;
    for(int i=0;i<m;i++){
        int c,a,b;
        tie(c,a,b)=edges[i];
        if(unionDs(a,b)==false) continue;
        else res+=c;
    }
    return res;
}
```

3.2 Kosaraju's (SCC)

```
//Sirve para encontrar los SCC
struct Kosaraju{
    int s;
    vector<vector<int>> g,gr;
    vector<int> visited,ids,topologic_sort;
    Kosaraju(int n){
        s=n;
        g.assign(n+1,vector<int>());
        gr.assign(n+1,vector<int>());
        visited.assign(n+1,0);
        ids.assign(n+1,0);
    }
    void addEdge(int a,int b){
        g[a].push_back(b);
        gr[b].push_back(a);
    }
    void dfs(int u){
        if(visited[u]!=0) return;
        visited[u]=1;
        for(int node:g[u])dfs(node);
        topologic_sort.push_back(u);
    }
    void dfsr(int u,int id){
        if(visited[u]!=0) return;
        visited[u]=1;
        ids[u]=id;
        for(int node:gr[u])dfsr(node,id);
    }
    void algo(){
        for(int i=1;i<=s;i++) if(visited[i]==0) dfs(i);
        fill(visited.begin(),visited.end(),0);
        reverse(topologic_sort.begin(),topologic_sort.end());
        int id=0;
        for(int i=0;i<topologic_sort.size();i++){
            if(visited[topologic_sort[i]]==0)dfsr(topologic_sort[i],id++);
        }
    }
    int search(int node){
        return ids[node];
    }
};
```

3.3 2 Sat

```
//Se usa para los problams en los cuales tengamos dos dosible variables
struct twoSat{
    int s;
    vector<vector<int>> g,gr;
    vector<int> visited,ids,topologic_sort,val;
    twoSat(int n){
        s=n;
        g.assign(n*2+1,vector<int>());
        gr.assign(n*2+1,vector<int>());
        visited.assign(n*2+1,0);
        ids.assign(n*2+1,0);
        val.assign(n+1,0);
    }
    void addEdge(int a,int b){
        g[a].push_back(b);
        gr[b].push_back(a);
    }
    void addOr(int a,bool ba,int b,bool bb){
        addEdge(a+(ba?s:0),b+(bb?0:s));
        addEdge(b+(bb?s:0),a+(ba?0:s));
    }
    void addXor(int a,bool ba,int b,bool bb){
        addOr(a,ba,b,bb);
        addOr(a,!ba,b,!bb);
    }
    void addAnd(int a,bool ba,int b,bool bb){
        addXor(a,!ba,b,bb);
    }
    void dfs(int u){
        if(visited[u]!=0) return;
        visited[u]=1;
        for(int node:g[u])dfs(node);
        topologic_sort.push_back(u);
    }
    void dfsr(int u,int id){
        if(visited[u]!=0) return;
        visited[u]=1;
        ids[u]=id;
        for(int node:gr[u])dfsr(node,id);
    }
    bool algo(){
        for(int i=0;i<s*2;i++) if(visited[i]==0) dfs(i);
        fill(visited.begin(),visited.end(),0);
        reverse(topologic_sort.begin(),topologic_sort.end());
        int id=0;
        for(int i=0;i<topologic_sort.size();i++){
            if(visited[topologic_sort[i]]==0)dfsr(topologic_sort[i],id++);
        }
        for(int i=0;i<s;i++){
            if(ids[i]==ids[i+s]) return false;
            val[i]=(ids[i]>ids[i+s]?0:1);
        }
        return true;
    }
};
```

4 Strings

4.1 KMP

```
vector<int> kmp(string s){
    int n=s.size();
    vector<int>pi(n);
    for(int i=1;i<n;i++){
        int j=pi[i-1];
        while(j>0 && s[i]!=s[j])j=pi[j-1];
        if(s[i]==s[j]) j++;
        pi[i]=j;
    }
    return pi;
}
```

5 Math

5.1 Linear Sieve

```
//O(N) for find all the primes in the given range
bool is_compositive[10000000+1];
vector<int>primes;
void sieve(int n){
    primes.clear();
   fill(is_compositive,is_compositive+n,false);
   for(int i=2;i<=n;i++){
    if(!is_compositive[i]) primes.push_back(i);
    for(int j=0;j<primes.size() && primes[j]*i<=n;j++){
        is_compositive[i*primes[j]]=true;
        if(!(i%primes[j])) break;
    }
   }
}

    int n;cin>>n;
    sieve(n);
    cout<<primes.size()<<endl;
    for(int i=0;i<primes.size();i++){
        cout<<primes[i]<<endl;
    }
}
```

5.2 Euler Sieve

```
//this is a sieve for a euler funciton that given the number of coprime numbers of x but in a range
vector<int>sieve;
void eulerSieve(int n){
    sieve.clear();
    sieve.push_back(0);
    for(int i=1;i<=n;i++){
        sieve.push_back(i);
    }
    for(int i=2;i<=n;i++){
        if(sieve[i]==i)
            for(int j=i;j<=n;j+=i)sieve[j]-=(sieve[j]/i);
    }
}
```

5.3 Euler Sieve Gauss Reduction

```
// sum(pi(n)) of the divisors of n is equal to n
vector<int>sieve;
void eulerSieve(int n){
    sieve.clear();
    sieve.push_back(0);
    sieve.push_back(1);
    for(int i=2;i<=n;i++){
        sieve.push_back(i-1);
    }
    for(int i=2;i<=n;i++){
            for(int j=i*2;j<=n;j+=i)sieve[j]-=sieve[i];
    }
}
```

5.4 Mobius Sieve

```
/* f(x)=0 if has square prime factor
```

```
f(x)=1 if if is square-free and even
f(x)=-1 if is square-free and odd
properti the sum of function of divisors of x is equl to 0 if x>1*/
vector<int>sieve;
void ms(int n){
    sieve.assign(n+1,-1);
    sieve[1]=1;
    for(int i=2;i<=n;i++)
       for(int j=i*2;j<=n;j+=i)sieve[j]-=sieve[i];
}
```

6 Flows

6.1 Dinics

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
struct dinics{
    int m=0,n;
    ll maxFlow=1e18;
    vector<tuple<int,ll,ll>edge;
    vector<vector<int>>adj;
    vector<int>level,id;
    void init(int _n){
        n=_n;
        level.resize(n+1);
        index.resize(n+1);
        adj.resize(n+1);
    }
    void addEdge(int u,ll f){
        edge.push_back({u,f,0});
        adj[u].push_back(m)
        edge.push_back({v,f,0});
        adj[u].push_back(m+1);
        m+=2;
    }
    bool bfs(int s, int t){
        fill(level.begin(),level.end(),-1);
        queue<int>aux;
        aux.push(s);
        while(!aux.empty()){
            int v=aux.front();aux.pop();
            for(auto idx:adj[v]){
                auto &[u,c,f]=edge[idx];
                if(c-f<0 || level[idx]!=-1) continue;
                aux.push(u);
                level[u]=level[v]+1;
            }
        }
        return level[t]!=-1?1:0;
    }
    ll dfs(int u, ll f){
        if(f==0) return 0;
        for(auto &cdx=id[u];cdx<adj[u].size();cdx++){
            int idx=adj[u][cdx];
            auto &[v,c,f]=edge[idx];
            if(level[v]!=level[u]+1 || f-c<1) continue;
            ll res;
            if(!(res=dfs(v,min(c-f))) continue;
            auto &fr=get<2>edge[idx^1];
            f+=res;
            fr-=res;
            return res;
        }
        return 0;
    }
    ll maxFlow(int s,int t){
        ll f=0;
        while(bfs(s,t)){
            fill(id.begin(),id.end(),0);
            while(f+=dfs(s,maxFlow));
        }
        return f;
    }
}
```

7 Geometry

8 Others