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 section

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

```
3   void init(int n){
4       ds.assign(n+1,{-1,0});
5   }
6   int find(int x){
7       if(-1==ds[x].first) return x;
8       return ds[x].first=find(ds[x].first);
9   }
10  bool unionDs(int x, int y){
11      int px=find(x),py=find(y);
12      int &rx=ds[px].second,&ry=ds[py].second;
13      if(px==py) return false;
14      else{
15          if(rx>ry) ds[py].first=px;
16          else{
17              ds[px].first=py;
18              if(rx==ry) ry+=1;
19          }
20      }
21      return true;
22  }
```

1.4   Sparce Table

```
1   //Se usa para RMQ porque se puede hacer en O(1), no acepta updates
2   vector<int>lg;
3   vector<vector<int>>st;
4   int *nums;
5   void init(int n){
6       int logn=(int) log2(n)+1;
7       lg.assign(n+1,0);
8       st.assign(logn,vector<int>(n+1));
9       for(int i=0;i<n;i++) st[0][i]=nums[i];
10      lg[1]=0;
11      for(int i=2;i<=n;i++) lg[i]=lg[i/2]+1;
12      for(int i=1;i<logn;i++)
13          for(int j=0;j+(1<<i)<n;j++)st[i][j]=min(st[i-1][j],st[i-1][j
                +(1<<(i-1))]);
14  }
15  int query(int a,int b){
16      int logn=lg[(b-a+1)];
17      cout<<st[logn][a]<<endl;
18      return min(st[logn][a],st[logn][b-(1<<logn)+1]);
19  }
```

2   DP

2.1   Digit DP

```
1   ll dp[20][20][3];
2   ll n,k,d;
3   vector<int>num;
4   ll bk(int i, int len, int t){
5       if(len>k) return 0;
6       if(i==n){
7           if(len==k) return 1;
8           return 0;
9       }
10      ll &res=dp[i][len][t];
11      if(res!=-1) return res;
12      res=0;
13      int tope;
14      if(t==0) tope=num[i];
15      else tope=9;
16      for(int j=0;j<=tope;j++){
17          int newt=t;
18          int newlen=len;
19          if(t==0 && j<tope) newt=1;
20          if(d==j) newlen++;
21          if(newlen<=k)res+=bk(i+1,newlen,newt);
22      }
23      return res;
24  }
25  ll rep(int a){
26      num.clear();
27      while(a>0){
28          num.push_back(a%10);
29          a/=10;
30      }
31      reverse(num.begin(),num.end());
32      n=num.size();
33      memset(dp,-1,sizeof(dp));
34      return bk(0,0,0);
35  }
```

3 Graph

3.1 Krustal

```cpp
// 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;
}
```

4 Strings

4.1 KMP

```cpp
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

6 Geometry

7 Others