1269 - Consecutive Sum:

Now you are given an array of n integers, indexed from 0 to n-1, you have to find two indices i j in the array (0 ≤ i ≤ j < n), such that the xor summation of all integers between indices i and j in the array, is maximum. And you also have to find two indices, p q in the array (0 ≤ p ≤ q < n), such that the xor summation of all integers between indices p and q in the array, is minimum. You only have to report the maximum and minimum integers.

In short, Given an array of integers, we have to find the sub array whose XOR is maximum and minimum.

Each case starts with a line containing an integer n (1 ≤ n ≤ 50000). The next line contains n space separated non-negative integers, denoting the integers of the given array. Each integer fits into a 32 bit signed integer.

Sample Code(Dynamic Trie):

```
#define mx 100005
#define ll long long
ll a[mx+5],sx[mx+5];
string ss[mx+5];

struct node{
    bool endmark;
    int dx;
    node *next[2];
    node(){
        endmark = false;
        dx = 0;
        next[0] = next[1] = NULL;
    }
}*root;

string DecToBin(int n,int msz)
{
    string tm;
    while(n!=0){
        tm += (n&1)+'0';
        n = n>>1;
    }
    while(tm.size()<msz) tm+='0';

    reverse(tm.begin(),tm.end());
    return tm;
}
```

```
void Insert(string s,int x)
{
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        if(cur->next[id]==NULL)
            cur->next[id] = new node();
        cur = cur->next[id];
    }
    cur->endmark = true;
    cur->dx = x;
}
int queryMax(string s)
{
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        int nx = 1-id;

        if(cur->next[id]==NULL && cur->next[nx]==NULL) return 0;
        else if(cur->next[nx]==NULL) cur = cur->next[id];
        else cur = cur->next[nx];
    }
    return cur->dx;
}
int queryMin(string s)
{
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        int nx = 1-id;

        if(cur->next[id]==NULL && cur->next[nx]==NULL) return 0;
        else if(cur->next[id]==NULL) cur = cur->next[nx];
        else cur = cur->next[id];
    }
    return cur->dx;
}

void del(node *cur)
{
    for(int i=0; i<2; i++){
        if(cur->next[i])
            del(cur->next[i]);
    }
    delete(cur);
}
```

```
int main()
{
    int tt; scanf("%d",&tt);

    for(int ks=1; ks<=tt; ks++)
    {
        root = new node();

        int n; scanf("%d",&n);
        ll maxx = -1,minn=1000000000000000;
        for(int i=1; i<=n; i++){
            scanf("%lld",&a[i]);
            maxx = max(maxx,a[i]);
            minn = min(minn,a[i]);
        }

        int msz;
        if(maxx==0)msz=1;
        else msz = floor(log2(maxx))+1;

        sx[0]=0;

        for(int i=1; i<=n; i++){
            sx[i] = sx[i-1]^a[i];
            maxx = max(maxx,sx[i]);
            minn = min(minn,sx[i]);
            ss[i] = DecToBin(sx[i],msz);
        }

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

            int tx = queryMax(ss[i]);
            ll xr = sx[i]^sx[tx];
            maxx = max(maxx,xr);

            int ty = queryMin(ss[i]);
            ll yr = sx[i]^sx[ty];
            minn = min(minn,yr);

            Insert(ss[i],i);
        }

        printf("Case %d: %lld %lld\n",ks,maxx,minn);

        del(root);
    }
    return 0;
}
```

Sample Code(Static Trie):

```
#define MX 50007
#define LL long long int
int avail;
LL a[MX];

struct node
{
    int child[2];
    node(){
        child[0] = child[1] = -1;
    }
}tree[34*MX];;

void Insert(LL n)
{
    int ptr = 0;
    for (int i=32; i>=0; i--)
    {
        int id = (n & (1LL<<i))>0;

        if (tree[ptr].child[id] == -1){
            tree[ptr].child[id] = ++avail;
            tree[avail] = node();
        }
        ptr = tree[ptr].child[id];
    }
}

LL findMax(LL n)
{
    int ptr = 0;
    LL ans = 0;

    for (int i=32; i>=0; i--)
    {
        int id = (n&(1LL<<i))==0;

        if (tree[ptr].child[id] != -1){
            ans |= (1LL<<i);
            ptr = tree[ptr].child[id];
        }
        else{
            ptr = tree[ptr].child[(id+1)%2];
        }
    }
    return ans;
}
```

```
LL findMin(LL n)
{
    int ptr = 0; LL ans = 0;

    for (int i=32; i>=0; i--)
    {
        int id = (n&(1LL<<i))>0;
        if (tree[ptr].child[id] != -1){
            ptr = tree[ptr].child[id];
        }
        else{
            ans |= (1LL<<i);
            ptr = tree[ptr].child[(id+1)%2];
        }
    }
    return ans;
}
int main()
{
    int t; scanf("%d",&t);
    for(int ks=1; ks<=t; ks++)
    {
        int n; scanf("%d",&n);
        for (int i=1; i<=n; i++) scanf("%d",&a[i]);

        avail = 0;
        tree[0] = node();

        LL XOR = 0;  LL maxx = 0;  LL minn = (1LL<<50);

        Insert(0);

        for (int i=1; i<=n; i++)
        {
            XOR ^= a[i];

            LL mx = findMax(XOR);
            LL mn = findMin(XOR);

            maxx = max(maxx,mx);
            minn = min(minn,mn);

            Insert(XOR);
        }
        printf("Case %d: %lld %lld\n",ks,maxx,minn);
    }
    return 0;
}
```