Mục lục

1 Bài A.

1.1 Sub 1: $A_i \le 5$

Ta sinh mọi hoán vị độ dài i thỏa mãn để tìm ra $F(i)$ tốt nhất. Việc phía sau, ta chỉ cần tính tổng của mảng là được.

1.2 Sub 2: $n = 1, A_i \le 10^6$

Nhận xét: gọi B_i là vị trí bit 1 lớn nhất của i khi i được biểu diện hệ nhị phân ($1 \le i \le A_1$), nhận thấy,ta cần sắp xếp hoán vị để xóa được $max(B_i)$ tức xóa được cái vị trí bit 1 lớn nhất. Tuy nhiên, việc này là không thể vì khi sắp xếp sẽ luôn tồn tại ít nhất một phần tử có bit lớn nhất đứng cạnh phần tử có bit nhỏ hơn, chỉ khi toàn bộ B_i bằng nhau mới xảy ra chuyện đấy và như ta nhận thấy số 0 là số duy nhất có $B_0 = 0$ nên mảng B luôn có ít nhất 2 giá trị khác nhau.
Từ đó, ban đầu ta cho toàn bộ số có cùng $max(B_i)$ đứng chung với nhau ở đầu dãy và số tiếp theo đó ta sẽ đặt một số (thỏa mãn vị trí nào có bit 1 của số cuối dãy thì số sắp được thêm vào cũng có bit 1 tại đó và ngược lại bit 0 cũng thế chỉ khác ở chỗ là vị trí bit 1 xa nhất thì tại đó số được thêm vào nhận bit 0) khi đấy, ta lấy xor ta chỉ giữ lại bit 1 cao nhất tức là giữ lại $max(B_i)$ thôi và hay nói cách khác $max(B_i)$ chính là đáp án của bài.

1.3 Sub 3: Không có ràng buộc

Từ sub 2 ta nhận thấy với mỗi A_i ta chỉ cần tìm $max(B_j)$ tức là tìm bit 1 lớn nhất trong các số từ 1 đến A_i, mà ta cũng biết rằng bit thứ i mà là bit 1 thì ta cũng có thể biểu diễn thập phân là 2^i. Từ đó bài toán đưa về với mỗi i tìm 2^x lớn nhất sao cho $2^x < A_i$, giá trị đó cũng chính là $F(A_i)$!!

1.4 Code mẫu

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
int n,x,sum,v;
signed main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>n;
    //n--;
    while(n--){
        cin>>x;
        x--;
        v=1;
        while(v<=x) v*=2;
        v/=2;
        sum+=v;
    }
    cout<<sum;



    return 0;
}
```

2 Bài B:

2.1 sub 1: n ≤ 40

cơ bản có thể chia thành 4 subtask nhỏ hơn:
1. $n \leq 10$, sinh mọi hoán vị độ dài n và tìm hoán vị tốt nhất.
2. $n \leq 20$, nhận thấy ta chia n số thành 2 nhóm có tổng bằng nhau đặt là nhóm A và nhóm B, giả sử nhóm A là nhóm có số 1, việc của ta chỉ cần quay lui nhị phân và tìm các giá trị của nhóm A và đương nhiên các giá trị nhóm B sẽ là các số còn lại không nằm trong nhóm A.
3. $n \leq 30$, Ta cải tiến sub trên bằng thuật quay lui chia tập để tăng tốc độ chạy hơn , quay lui n/2 số đầu tiên và lưu các tập quay lui được lẫn cả phần tổng của tập đó, sau đó quay lui n/2 số còn lại mỗi tập quay lui ra được nếu tổng còn thiếu một lượng bao nhiêu thì ta sẽ tìm xem trong tập trước đó có tồn tại dãy có tổng đúng bằng phần còn thiếu không. điều này ta có thể làm bằng tìm kiếm nhị phân nên độ phức tạp tương đối an toàn.
4. $n \leq 40$, ta sử dụng sub 3 để làm rồi sau đó sử dụng mảng hằng để lưu kết quả (vì từ $31 -> 40$ chỉ có 6 số có đáp án.

2.2 sub 2: không ràng buộc gì thêm

Ta cũng sẽ đi tìm các phần tử cho tập A. ta thực hiện duyệt i từ 1 đến n, ta sẽ cố gắng nhét phần tử i vào nhóm A vì nếu nhét được thì dãy hoán vị ta nhận được chắc chắn sẽ nhỏ hơn về từ điển so với khi ta không nhét phần tử i vào nhóm mà thay bằng các số lớn hơn.
Vậy, đầu tiên ta có sum là tổng còn thiếu để nhóm A có tổng đúng bằng một nửa, ta giả sử sẽ thêm i vào nhóm A thì ta cần kiểm tra từ $i+1 -> n$ có thể tạo ra được tổng $sum - i$ hay không. Ta chỉ kiểm tra $i+1 -> n$ mà không cần thêm các phần tử trước i chưa được cho vào A là vì cơ bản khi ta thử nhét i và các số còn lại không thể tạo ra $sum - i$ thì nghĩa là i chắc chắn thuộc nhóm B nên ta đã xác nhận được nhóm cho các số trước i việc bốc nó vào để kiểm tra có tạo ra $sum - i$ không là vô lý.
Bài toán lúc này chỉ còn việc giải quyết là kiểm tra các số trong đoạn $l -> r$ có tạo ra tổng bằng s được không. ta nhận thấy nếu ta chỉ bốc k số trong đoạn thì có nhận xét sau, đặt a là tổng bé nhất có thể tạo ra nếu chỉ bốc k số, b là tổng lớn nhất nếu chỉ bốc k số. Nhận thấy các tổng từ $1 -> a-1$ và $b+1 -> oo$ không thể tạo ra được, còn các số từ $a -> b$ ta đều tạo ra được, vì giả sử nếu ta có ($a \leq x < b$), x được phân tích dưới dạng $x = p_1 + p_2 + p_3 + ... + p_k$ để tạo ra $x+1$ ta chỉ việc bốc một giá trị p_i bất kỳ và tăng thêm 1 sẽ luôn tồn tại ít nhất một giá trị p_i để ta tăng thêm 1 mà không sợ các giá trị sẽ trùng nhau (vì dễ hiểu tổng bé nhất là bốc k thằng bé nhất trong mảng, tổng lớn nhất là bốc k thằng lớn nhất trong mảng, nên nếu x chưa phải lớn nhất ta có thể đổi thằng lớn nhất được chọn hiện tại +1 nếu không được ta chuyển sang thằng lớn nhì +1, không được thì chọn lớn 3..., và vì x chưa phải tổng lớn nhất nên phải tồn tại một thằng trong k thằng không phải là số thuộc k số cuối của đoạn).
Vậy từ đó, nhận thấy nếu chọn k số ta tạo ra được các tổng liên tiếp từ $a -> b$, vậy để kiểm

tra có tạo ra s hay không ta cần tìm một số nguyên k nào đó thỏa mãn x nằm trong $a->b$, ta sẽ thực hiện tìm kiếm nhị phân tìm số k lớn nhất sao cho giá trị a tương ứng của nó nhỏ hơn hoặc bằng x và ta tìm thêm b của nó nếu $a->b$ chứa x thì ta xác nhận có thể tạo ra x.

2.3 Code mẫu

```
        #include<bits/stdc++.h>
using namespace std;
#define int long long
const int N=1e6+10;

int n,t,sum,pos,l,r,mid,vt;
vector<int> ds;
bool vs[N];
bool check(int a,int b){
    if(a==0) return true;
    if(a<0) return false;
    l=b;
    r=n;
    vt=-1;
    while(l<=r){
        mid=(l+r)/2;
        if((mid-b+1)*(mid+b)/2<=a){
            vt=mid;
            l=mid+1;
        }
        else{
            r=mid-1;
        }
    }
    if(vt==-1) return false;
    if(vt-b+1==1){
        return vt<=a && a<=n;
    }
    else return (vt-b+1)*(vt+b)/2<=a && a<=max(0ll,(n-(vt-b+1)+1+n)*(vt-b+1)/2);
}
signed main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>t;
    while(t--){
        cin>>n;
        sum=n*(n+1)/4;
        for(int i=1; i<=n; i++){
            if(check(sum-i,i+1) && sum){
                sum-=i;
                //ds.push_back(i);
                vs[i]=true;
                cout<<i<<' ';
```

```
                //cout<<sum<<' ';
            }
        }
        for(int i=1; i<=n; i++){
            if(!vs[i]) cout<<i<<' ';
            vs[i]=false;
        }
        cout<<'\n';
        ds.clear();
    }
}
```

3 Bài C:

3.1 sub 1: n $\leq$ 100

Ta duyệt mọi cặp i,j và coi thử có đường đi giữa chúng hay không bằng cách for từ $i->j$ tìm min và max nếu thỏa điều kiện ta thêm cạnh i và j, phần còn lại ta chỉ việc BFS từ 1 đến n.

3.2 sub 2: n $\leq$ 1000

Ta cải tiến từ sub 1, việc tìm min và max ta sử dụng segment tree để tìm max, min dễ hơn

3.3 Sub 3: n $\leq$ 4500

Ta tối ưu hơn, gọi $ma_{i,j}$ và $mi_{i,j}$ là max và min từ $i->j$, nhận thấy ta có thể xây dựng mảng bằng dp range, $ma_{i,j} = max(ma_{i,j-1}, A_j)$ tương tự với tìm mi_i,j

3.4 Sub 4: không có ràng buộc

Nhận thấy việc xây dựng cạnh là không thể, ở bài toán này ta sẽ tham lam. Để tham lam, mình cần quan sát giả sử như đang đứng tại đỉnh là i, ta tìm vị trí j ($i < j$) sao cho $A_j < A_i$ và gần i nhất, vậy nhận thấy từ $i->j-1$ tất cả các $A_j >= A_i$, vì vậy nếu ta đứng tại i và muốn di chuyển thì chỉ có thể di chuyển vào các ô từ $i->j-1$. Giải pháp tốt nhất là ta sẽ nhảy vào ô có A_k lớn nhất với $i \leq k < j$ để nhảy vào, vì nếu ta không thể từ i nhảy đến các ô từ $k+1->j-1$ do rằng A_k là max của đoạn đấy. Tương tự, ở trường hợp trên ta đang giả sử A_i làm min nên ta cần xét thêm trường hợp giả sử A_i là max, ta cũng tìm j xa nhất sao cho $A_j > A_i$ vậy nếu đứng tại i ta chỉ nhảy được tới $i+1->j-1$ và ta cũng sẽ chọn A_k là min của đoạn đó và nhảy tới đó. Mặc dù có 2 trường hợp, nhưng vì đây là hoán vị, nên một trong hai j ta tìm được chắc chắn bằng $i+1$, tức là tại 1 đỉnh ta chỉ có 1 trong 2 trường hợp trên là hoạt động.

Đút kết, thuật toán, là tìm với mỗi ô i tìm 2 j xa nhất sao chó $A_i > A_j$ và $A_i < A_j$, ta có thể tìm bằng deque tịnh tiền, khi có 2 j ta tìm j nào $!= i+1$ ta sẽ chạy trường hợp đó, và trong 2 trường hợp trên đều cần tìm min/max của một đoạn vậy ta sẽ sử dụng segmentree để tìm cho nhanh !!.

3.5 code AC

```
#include<bits/stdc++.h>
using namespace std;

const long N=5e5+10;
long v,ans,pos,n,A[N],RMI[N],RMA[N],t,x,DMI[N*4],DMA[N*4],vt[N],k;
stack<long> ds;
```

```
void build(long id,long l,long r){
    if(l==r){
        DMI[id]=A[l];
        DMA[id]=A[l];
        return;
    }
    long mid=(l+r)>>1;
    build(id*2,l,mid);
    build(id*2+1,mid+1,r);
    DMI[id]=min(DMI[id*2],DMI[id*2+1]);
    DMA[id]=max(DMA[id*2],DMA[id*2+1]);
}
void getmin(long id,long l,long r,long u,long v2){
    if(u>r || v2<l) return;
    if(u<=l && r<=v2){
        v=min(v,DMI[id]);
        return;
    }
    long mid=(l+r)>>1;
    getmin(id*2,l,mid,u,v2);
    getmin(id*2+1,mid+1,r,u,v2);
}
void getmax(long id,long l,long r,long u,long v2){
    if(u>r || v2<l) return;
    if(u<=l && r<=v2){
        v=max(v,DMA[id]);
        return;
    }
    long mid=(l+r)>>1;
    getmax(id*2,l,mid,u,v2);
    getmax(id*2+1,mid+1,r,u,v2);
}
int main(){
    ios::sync_with_stdio(0);cin.tie(0);
    cin>>t;
    while(t--){
        cin>>n;
        for(long i=1; i<=n; i++) {cin>>A[i]; vt[A[i]]=i;}
        if(n==1) {cout<<0<<'\n'; continue;}
        build(1,1,n);
        ds.push(n);
        for(long i=n-1; i>=1; i--){
            while(!ds.empty()){
                v=ds.top();
                if(A[v]>A[i]) break;
                 ds.pop();
            }
            if(ds.empty()) RMA[i]=n+1;
```

```
            else RMA[i]=ds.top();
            ds.push(i);
        }
        while(!ds.empty()) ds.pop();
        ds.push(n);
        for(long i=n-1; i>=1; i--){
            while(!ds.empty()){
               v=ds.top();
               if(A[v]<A[i]) break;
                ds.pop();
            }
            if(ds.empty()) RMI[i]=n+1;
            else RMI[i]=ds.top();
            ds.push(i);
        }

        while(!ds.empty()) ds.pop();
        ans=0;
        pos=1;
        x=2;
        while(pos<n){
            if(RMI[pos]>RMA[pos]){
                v=A[x];
                getmax(1,1,n,x+1,RMI[pos]-1);
                x=vt[v]+1;
                pos=vt[v];
            }
            else{
                v=A[x];
                getmin(1,1,n,x+1,RMA[pos]-1);
                x=vt[v]+1;
                pos=vt[v];
            }
            ans+=1;
        }
        cout<<ans<<'\n';
    }
}
```