最小逆序对

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define lson l,m,rt<<1
#define rson m+1,r,rt<<1 | 1
#define root 1,n,1
const int N=1e5+2;
const int inf=0x3f3f3f3f;
int SUM[4*N],MAX[4*N],MIN[4*N],add[4*N],change[4*N];
void PushUp(int rt) {
    SUM[rt]=SUM[rt<<1]+SUM[rt<<1 | 1];
}


void build(int l,int r,int rt) {    //
    if (l==r) {
        SUM[rt]=0;
        return ;
    }
    int m=(l+r)>>1;
    build(lson);      //ππΩ®◊Û◊" ˜
    build(rson);      //ππΩ®""◊" ˜
    PushUp(rt);       //ππΩ®ÕÍ◊Û""¡Ωø≈◊" ˜∫Û£¨∏¸¬µ±«∞◊" ˜
}

void point_assign(int p,int val,int l,int r,int rt) {
    if (l==r) {
        SUM[rt]=val;
        MAX[rt]=val;
        MIN[rt]=val;
        return ;
    }
    int m=(l+r)>>1;
    if (p<=m) point_assign(p,val,lson);
    else      point_assign(p,val,rson);
    PushUp(rt);
}

int query_SUM(int L,int R,int l,int r,int rt) {
    if (L<=l && r<=R) {
        return SUM[rt];
    }
    int m=(l+r)>>1;
    int res=0;
    if (L<=m) res+=query_SUM(L,R,lson);
    if (R>m)  res+=query_SUM(L,R,rson);
    return res;
}

void solve() {
    int n;
    cin>>n;
    build(root);
```

```
}

int main() {
    //    freopen("datain.txt","r",stdin);
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    while(cin>>n){
        int x[n];
        build(1,n,1);
        int sum=0;
        for (int i=0;i<n;i++){
            cin>>x[i];
            x[i]++;
            sum+=query_SUM(x[i],n,1,n,1);
            point_assign(x[i],1,1,n,1);
        }
        int ans=sum;
        for (int i=0;i<n;i++){
            sum+=(n-x[i]-x[i]+1);
            ans=min(ans,sum);
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

区间最长匹配括号数

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cstdio>
#include <vector>
using namespace std;
typedef long long LL;
#define ls (rt<<1)
#define rs (rt<<1|1)
#define lson ls,l,m
#define rson rs,m+1,r
#define root 1,1,n
const int N=1e6+5;
const int inf=0x3f3f3f3f;
int SUM[N<<2],MAX[N<<2],MIN[N<<2],XOR[N<<2],assign[N<<2];
void setvalue(int rt,int l,int r,int val){
    assign[rt]=val;
    int tmp=(r-l+1)*val;
    SUM[rt]=tmp;
    MAX[rt]=max(tmp,0);
    MIN[rt]=min(tmp,0);
    XOR[rt]=0;
}
void Reversal(int rt) {
    if (assign[rt]){
        assign[rt] *= -1;
        SUM[rt]    *= -1;
        XOR[rt]=0;
    }else{
        XOR[rt]       ^=  1;
        SUM[rt]    *= -1;
    }
    int a=MAX[rt],b=MIN[rt];
    MAX[rt]=-b;
    MIN[rt]=-a;
}
void PushUp(int rt) {
    SUM[rt]= SUM[ls] + SUM[rs];
    MAX[rt]=max( MAX[ls] , SUM[ls]+MAX[rs]);
    MIN[rt]=min( MIN[ls] , SUM[ls]+MIN[rs]);
}
void PushDown(int rt,int l,int r) {
    int m=(l+r)>>1;
    if(assign[rt]) {
        setvalue(lson,assign[rt]);
        setvalue(rson,assign[rt]);
        assign[rt]=0;
    }
    if (XOR[rt]){
        Reversal(ls);
        Reversal(rs);
        XOR[rt]=0;
    }
}
void build(int rt,int l,int r){
    XOR[rt]=assign[rt]=0;
    if (l==r){
```

```
        char ch;
        cin>>ch;
        if (ch=='(') setvalue(rt,l,r,-1);
        else         setvalue(rt,l,r,1);
        return ;
    }
    int m=(l+r)>>1;
    build(lson);
    build(rson);
    PushUp(rt);
}

void update(int L,int R,int val,int rt,int l,int r) {
    if (L<=l && r<=R) {
        if (val==2) Reversal(rt);
        else         setvalue(rt,l,r,val);
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    if(L<=m) update(L,R,val,lson);
    if(R>m) update(L,R,val,rson);
    PushUp(rt);
}

void query(int L,int R,int rt,int l,int r,int &_sum,int &_max) {
    if (L<=l && r<=R) {
        _sum=SUM[rt];
        _max=MAX[rt];
        return ;
    }
    PushDown(rt,l,r);
    int m=(l+r)>>1;
    int lsum=0,rsum=0,lmax=0,rmax=0;
    if (L<=m) query(L,R,lson,lsum,lmax);
    if (R>m)  query(L,R,rson,rsum,rmax);
    _sum=lsum+rsum;
    _max=max(lmax,lsum+rmax);
    return ;
}

void solve() {
    int n;
    cin>>n;
    build(root);
    int Q;
    cin>>Q;
    while(Q--){
        string order;
        cin>>order;
        int L,R;
        cin>>L>>R;
        L++;
        R++;
        if (order[0]=='q'){
            int _sum=0,_max=0;
            query(L,R,root,_sum,_max);
```