

Deltix Round, Summer 2021 (Div. 1 + Div. 2)

[比赛链接](#)

E. Equilibrium

题意

给定两个序列 $\{A\}, \{B\}$ 每次询问一个区间 $[l, r]$

每次操作可以在 $[l, r]$ 中选择若干对位置 $(a_1, b_1), (a_2, b_2) \dots (a_k, b_k)$ 满足 $a_1 < b_1 < a_2 < b_2 < \dots < a_k < b_k$

然后对每对位置 (i, j) 令 $A_i \text{ gets } A_i + 1, B_j \text{ gets } B_j + 1$ 问使得 $A_i = B_i (\forall i \in [l, r])$ 的最小操作数。

题解

令 $C_i = B_i - A_i$ 于是上述操作的每对位置 (i, j) 影响为 $C_i \text{ gets } C_i - 1, C_j \text{ gets } C_j + 1$ 最终需要使得 $C_i = 0$

不难发现 $\sum_{i=l}^r C_i \neq 0$ 时必然无解，因为每次操作不改变 C_i 之和。

另外 $C_i = 0$ 等价于前缀和等于 0 ，然后上述操作的每次位置 (i, j) 对前缀和的影响为 $i \sim j-1$ 的前缀和减一，其余不变。

于是 $C[l, r]$ 的最小前缀和小于 0 时也必然无解。然后每次操作可以等效为选择 $[l, r-1]$ 的若干段前缀和减一，于是最小操作等于最大前缀和。

考虑线段树维护，时间复杂度 $O((n+q)\log n)$

```
const int MAXN=1e5+5;
struct Node{
    LL sum,max_pre,min_pre;
    Node(int v=0){
        sum=max_pre=min_pre=v;
    }
    Node operator + (const Node &b)const{
        Node c;
        c.sum=sum+b.sum;
        c.max_pre=max(max_pre,sum+b.max_pre);
        c.min_pre=min(min_pre,sum+b.min_pre);
        return c;
    }
}s[MAXN<<2];
int a[MAXN],lef[MAXN<<2],rig[MAXN<<2];
void build(int k,int L,int R){
```

```
lef[k]=L, rig[k]=R;
int M=L+R>>1;
if(L==R){
    s[k]=Node(a[M]);
    return;
}
build(k<<1,L,M);
build(k<<1|1,M+1,R);
s[k]=s[k<<1]+s[k<<1|1];
}
Node query(int k,int L,int R){
    if(L<=lef[k]&&rig[k]<=R)
        return s[k];
    int mid=lef[k]+rig[k]>>1;
    if(mid>=R)
        return query(k<<1,L,R);
    else if(mid<L)
        return query(k<<1|1,L,R);
    else
        return query(k<<1,L,R)+query(k<<1|1,L,R);
}
int main(){
    int n=read_int(),q=read_int();
    _rep(i,1,n)
        a[i]=-read_int();
    _rep(i,1,n)
        a[i]+=read_int();
    build(1,1,n);
    while(q--){
        int l=read_int(),r=read_int();
        Node ans=query(1,l,r);
        if(ans.sum!=0||ans.min_pre<0)
            puts("-1");
        else
            enter(ans.max_pre);
    }
    return 0;
}
```

F. Sports Betting

题意

给定 n 个队伍的能力值，队伍间两两进行一场比赛，队伍 i 战胜队伍 j 的概率为 $\frac{a_i}{a_i+a_j}$

定义间接战胜：如果队伍 i 战胜队伍 b_1 ， b_1 战胜 $b_2 \cdots b_k$ 战胜 j ，则称 i 间接战胜 j

称一个队伍为冠军队伍当且仅当该队直接战胜或间接战胜所有其他队伍。问冠军队伍的期望个数。

题解

如果队伍 i 战胜队伍 j 则从点 i 向点 j 连一条边。

设 $U = \{1, 2, \dots, n\}$ 不难发现，假设冠军队伍集合为 S 当且仅当 S 为强连通分量且 S 与 $U - S$ 的所有边方向为 $S \rightarrow U - S$

设 $G(S, T)$ 表示 S 与 T 的所有边方向为 $S \rightarrow T$ 的概率 $f(S)$ 为 S 构成强连通分量的概率，根据容斥定理，有

$$f(S) = 1 - \prod_{T \subset S, T \neq \emptyset} G(T, S - T) f(T)$$

最终答案为

$$\sum_{S \subseteq U} G(S, U - S) f(S) |S|$$

考虑如何计算 $G(S, T)$ 有

$$G(S, T) = \prod_{i \in S} \prod_{j \in T} \frac{a_i}{a_i + a_j}$$

显然可以暴力 $O(n^2)$ 计算单个 $G(S, T)$ 预处理 $F(i, T) = \prod_{j \in T} \frac{a_i}{a_i + a_j}$ 则 $G(S, T) = \prod_{i \in S} F(i, T)$ 可以 $O(n)$ 计算。

进一步，考虑将前 $\frac{n}{2}$ 个点染黑，其余点染白，记黑点集为 U_1 白点集为 U_2 于是有

$$G(S, T) = G(S \cap U_1, T \cap U_1) G(S \cap U_1, T \cap U_2) G(S \cap U_2, T \cap U_1) G(S \cap U_2, T \cap U_2)$$

考虑 $O(n^2)$ 预处理出下述四种情况

$$G_{\{0/1, 0/1\}}(S, T) = G(S, T) (S \subseteq U_1, T \subseteq U_2, T \subseteq U_1, T \subseteq U_2)$$

于是可以 $O(1)$ 计算 $G(S, T)$ 总时间复杂度为 $O(3^n)$

```
const int MAXN=15,MAXB=8,mod=1e9+7;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int f[1<<MAXN],a[MAXN],p[MAXN][MAXN],g[4][1<<MAXB][1<<MAXB];
void cal(int p1,int n1,int p2,int n2,int g[1<<MAXB][1<<MAXB]){
    static int temp[1<<MAXB][1<<MAXB];
    int s1=1<<n1,s2=1<<n2;
    _for(i,0,n1){
```

```
    _for(j,0,s2){
        temp[i][j]=1;
        _for(k,0,n2){
            if(j&(1<<k))
                temp[i][j]=1LL*temp[i][j]*p[i+p1][k+p2]%mod;
        }
    }
}
_for(i,0,s1)_for(j,0,s2){
    g[i][j]=1;
    _for(k,0,n1){
        if(i&(1<<k))
            g[i][j]=1LL*g[i][j]*temp[k][j]%mod;
    }
}
}
int cal2(int i,int j,int m){
    int mk=(1<<m)-1;
    LL t=1LL*g[0][i&mk][j&mk]*g[1][i&mk][j>>m]%mod;
    t=t*g[2][i>>m][j&mk]%mod*g[3][i>>m][j>>m]%mod;
    return t;
}
int cal3(int i){
    int ans=0;
    while(i){
        ans+=i&1;
        i>>=1;
    }
    return ans;
}
int main(){
    int n=read_int();
    if(n==1){
        puts("1");
        return 0;
    }
    _for(i,0,n)
        a[i]=read_int();
    _for(i,0,n){
        _for(j,0,n)
            p[i][j]=1LL*a[i]*quick_pow(a[i]+a[j],mod-2)%mod;
    }
    int m=n/2;
    cal(0,m,0,m,g[0]);
    cal(0,m,m,n-m,g[1]);
    cal(m,n-m,0,m,g[2]);
    cal(m,n-m,m,n-m,g[3]);
    int S=(1<<n)-1,ans=0;
    _rep(i,1,S){
        f[i]=1;
```

```
    for(int j=(i-1)&i;j!=i;j=(j-1)&i)
        f[i]=(f[i]-1LL*f[j]*cal2(j,i^j,m))%mod;
    ans=(ans+1LL*f[i]*cal2(i,i^S,m)%mod*cal3(i))%mod;
}
enter((ans+mod)%mod);
return 0;
}
```

From:

<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:cf_deltix_round_summer_2021&rev=1630394740 

Last update: **2021/08/31 15:25**