提高组3

回文

题解

暴力枚举左上角和右下角 DP，复杂度 $O(n^4)$ 。

观察到只有 $x1+y1+x2+y2=n+m+2$ 的状态才有用，因此我们枚举 $x1,y1,x2$ 即可得知 $y2$ ，复杂度降为 $O(n^3)$ 。

标准代码

C++

```cpp
#include<cstdio>
#include<cstring>
const int mod=993244853;
int n,m;
int dp[2][505][505],lab[505][505];
char s[505][505];
int main() {
    //freopen("palin.in","r",stdin);
    //freopen("palin.out","w",stdout);
    scanf("%d%d",&n,&m);
    int i,ii;
    for(i=1;i<=n;i++)
        scanf("%s",s[i]+1);
    for(i=1;i<=n&&i<=m;i++)dp[0][i][i]=1;
    for(i=1;i<=n;i++)
        for(ii=1;ii<=m;ii++)
            lab[i][ii]=lab[i-1][ii+1]+1;
    int cr=0,pr,sl=(n+m+1)>>1,sr=(n+m+2)-sl;
    for(;sl>1;) {
        //printf("%d %d\n",sl,sr);
        cr=!cr,pr=!cr;
        memset(dp[cr],0,sizeof dp[cr]);
        for(int i=1;i<=n&&i<=sl;i++) {
            int yi=sl-i;
            if(yi>m)continue;
            for(int ii=i;ii<=n&&ii<=sr;ii++) {
                int yii=sr-ii;
                if(yii>m||yii<yi)continue;
                if(s[i][yi]==s[ii][yii]) {
                    if((i+1==ii&&yi==yii)||(i==ii&&yi+1==yii))
                        dp[cr][lab[i][yi]][lab[ii][yii]]=1;
                    else dp[cr][lab[i][yi]][lab[ii][yii]]=
                    (
                        0ll+
                        dp[pr][lab[i+1][yi]][lab[ii][yii-1]]+
                        dp[pr][lab[i+1][yi]][lab[ii-1][yii]]+
                        dp[pr][lab[i][yi+1]][lab[ii][yii-1]]+
                        dp[pr][lab[i][yi+1]][lab[ii-1][yii]]
                    )%mod;

                    //printf("(%d,%d)-(%d,%d) => %d\n",i,yi,ii,yii,dp[cr][lab[i][yi]][lab[ii][yii]]);    }
                }
            }
```

```
44          sl--,sr++;
45      }
46      printf("%d\n",dp[cr][1][1]);
47  }
48
49
```

快速排序

题解

观察给出操作的性质：

首先，如果所有被排序数都非 nan，那么必然是按升序排列。

否则，我们考虑第一个数 x_1 ：

如果 x_1 是 nan，那相当于把一个 nan 置于最前，然后排序其余的数。

否则，相当于把所有剩下的 $< x_1$ 的数从小到大放在 x_1 之前，并放置 x_1 。

不难通过预先给所有非 nan 的数排序来均摊 $O(1)$ 完成此操作。

复杂度 $O(nlogn)$ 。

标准代码

C++

```cpp
#include<cstdio>
#include<algorithm>
const int Inf=1e9+7;
int a[1<<20],b[1<<20],bs,bl,c[1<<20],ct;
void exec() {
    int n;
    scanf("%d",&n),bs=0,bl=1,ct=0;
    register int i;
    for(i=1;i<=n;i++) {
        char s[12];
        scanf("%s",s);
        if(*s!='n')sscanf(s,"%d",a+i),b[++bs]=a[i];
        else a[i]=-1;
    }
    std::sort(b+1,b+bs+1),b[bs+1]=Inf;
    for(i=1;i<=n;i++) {
        if(a[i]==-1) c[++ct]=-1;
        else {
            if(b[bl]>a[i])continue;
            while(b[bl]<a[i])c[++ct]=b[bl],bl++;
            c[++ct]=b[bl],bl++;
        }
    }
    for(i=1;i<=n;i++) {
        if(~c[i])printf("%d%c",c[i]," \n"[i==n]);
        else printf("nan%c"," \n"[i==n]);
    }
}
int main() {
    //freopen("qsort.in","r",stdin);
    //freopen("qsort.out","w",stdout);
    int T;
    scanf("%d",&T);
    while(T--) exec();
}
```

混乱邪恶

题解

从 MO 高联的二试改来的构造题。

不妨设 n 为偶数。若 n 为奇数，我们考虑加入一个 $a_{n+1}=0$ ，归约到偶数的情况。

我们将 a 排序，并构造 $d_i = a_{2i} - a_{2i-1}$ 。

可以发现 $\sum d_i \leq m - \frac{n}{2} < n$ 。

我们尝试对于每个 d_i 分配一个 e_i 使得 $\sum d_i e_i = 0$ ，这样便可以构造出一组满足条件的 c_i 。

我们不难归纳得出，若 n 个正整数 $d_1, d_2, \ldots, d_n$ 的和为偶数且小于 $2n$ ，则必存在一种方案：

$n = 1$ 显然成立。

对于 $n = k$ ，若 $d_i = 1$ ，显然成立。

若存在 $di > 1$ ，我们考虑将 d 的最大值 $maxd$ 和最小值 $mind$ 删除，并加入 $maxd - mind$ 。

不难发现总和减少了 $2mind$ ，即至少 2 ，且最小值仍然非零。问题归约到 $n' = n - 1$ 的情况。

因此我们证明了一定存在合法的构造方案，并能成功给出一种构造。

复杂度 $O(nlogn)$ 。

标准代码

C++

```cpp
#include<cstdio>
#include<vector>
#include<queue>
#include<algorithm>
int n,m;
int a[1<<20];
int rnk[1<<20];
int c[1<<20];
bool vis[1<<20];
int ls[1<<20],rs[1<<20],v[1<<20],tt;
class cmp1{public:inline bool operator()(int x,int y){return v[x]>v[y];}};
class cmp2{public:inline bool operator()(int x,int y){return v[x]<v[y];}};
std::priority_queue<int,std::vector<int>,cmp1>Q1;// vmin
std::priority_queue<int,std::vector<int>,cmp2>Q2;// vmax
void dfs(int p,int C) {
    if(p<=0){c[-p]=C;return;}
    dfs(ls[p],-C),dfs(rs[p],C);
}
void solve() {
    for(register int i=1;i<=tt;i++)Q1.push(i),Q2.push(i);
    for(;;) {
        int t=Q2.top();Q2.pop();
        //printf("%d\n",v[t]);
        while(vis[t]){t=Q2.top();Q2.pop();}
        if(v[t]==1) {
            int C=1;
            for(register int i=1;i<=tt;i++)
                if(!vis[i])dfs(i,C),C=-C;
            return;
        }
        int s=Q1.top();Q1.pop();
        while(vis[s]){s=Q1.top();Q1.pop();}

     vis[s]=vis[t]=1,v[++tt]=v[t]-v[s],ls[tt]=s,rs[tt]=t,Q1.push(tt),Q2.push(tt);
    }
}
int main() {
    //freopen("chaoticevil.in","r",stdin);
    //freopen("chaoticevil.out","w",stdout);
    scanf("%d%d",&n,&m);
    register int i;
    for(i=1;i<=n;i++)scanf("%d",a+i),rnk[i]=i;
    std::sort(rnk+1,rnk+n+1,[](int x,int y){return a[x]<a[y];});
    for(i=n;i>0;i-=2)
```

```
            ls[++tt]=-rnk[i-1],rs[tt]=-rnk[i],v[tt]=a[rnk[i]]-a[rnk[i-1]];
        solve();
        puts("NP-Hard solved");
        for(i=1;i<=n;i++)printf("%d ",c[i]);
        puts("");
    }


```

校门外歪脖树上的鸽子

题解

观察一次操作 $[l,r]$ 选中节点的特征，问题可以转化为若干次以下操作：

1. 给定一条链，遍历链上的每个节点，如果这个节点为右儿子，那么给其父亲的左子树上打一个标记；（或如果这个节点为左儿子，那么给其父亲的右子树上打一个标记）
2. 给定一条链，遍历链上的每个节点，如果这个节点为右儿子，那么答案加上父亲的左子树大小乘以左子树的标记，并求答案之和。（或反之）

将原树树链剖分，每条重链维护两棵线段树，分别维护与这条链相邻的左子树和右子树。（若一个点的左子树不在链上则这个点维护左子树，否则这个点维护右子树）

容易发现跳重链的过程只会影响某一棵线段树的一个区间，相当于在线段树上做一个区间加。

跳轻链的过程难以在线段树上表示，但由于轻边只会跳 $O(logn)$ 次，可以暴力维护。

查询和修改几乎是对称的，这里不再赘述。

复杂度 $O(nlog^2n)$ 。

标准代码

C++

```cpp
#include<cstdio>
#include<vector>
typedef long long ll;
int n,m,rt;
int ls[1<<20],rs[1<<20],sz[1<<20],fa[1<<20],ds[1<<20],le[1<<20],de[1<<20];
ll s[1<<20];
void init(int p) {
    de[p]=de[fa[p]]+1;
    if(p<=n){sz[p]=le[p]=1;return;}
    init(ls[p]),init(rs[p]),sz[p]=sz[ls[p]]+sz[rs[p]];
    if(sz[ls[p]]>sz[rs[p]])ds[p]=ls[p];
    else ds[p]=rs[p];
    le[p]=le[ds[p]]+1;
}
int dfn[1<<20],tp[1<<20],szdl[1<<20],szdr[1<<20],tot;
struct segtree {
    std::vector<int>siz;
    std::vector<ll>sum,spc,tag;
    int l,r;
    void __segtree_init(int L,int R,int*a,int p) {
        sum[p]=tag[p]=0;
        if(L==R){siz[p]=a[L];return;}
        int M=(L+R)>>1;
        __segtree_init(L,M,a,p<<1),__segtree_init(M+1,R,a,p<<1|1);
        siz[p]=siz[p<<1]+siz[p<<1|1];
    }
    void init(int L,int R,int*szlist) {
        l=L,r=R;
        spc.resize(R-L+1,0);
        siz.resize((R-L+1)<<2);
        sum.resize((R-L+1)<<2);
        tag.resize((R-L+1)<<2);
        __segtree_init(L,R,szlist,1);
    }
    void push(int p,ll d){sum[p]+=d*1ll*siz[p],tag[p]+=d;}
    void pushd(int p){push(p<<1,tag[p]),push(p<<1|1,tag[p]),tag[p]=0;}
    void pushu(int p){sum[p]=sum[p<<1]+sum[p<<1|1];}
    void __segtree_upd(int L,int R,int d,int p,int pl,int pr) {
        if(L>R||L>pr||pl>R)return;
        if(L<=pl&&pr<=R)return push(p,d);
        int pm=(pl+pr)>>1;
        pushd(p),__segtree_upd(L,R,d,p<<1,pl,pm),
        __segtree_upd(L,R,d,p<<1|1,pm+1,pr),pushu(p);
    }
```

```
45      void upd(int L,int R,int d){__segtree_upd(L,R,d,1,l,r);}
46      ll __segtree_ask(int L,int R,int p,int pl,int pr) {
47          if(L>R||L>pr||pl>R)return 0;
48          if(L<=pl&&pr<=R)return sum[p];
49          int pm=(pl+pr)>>1;
50          pushd(p);
51          return __segtree_ask(L,R,p<<1,pl,pm)+__segtree_ask(L,R,p<<1|1,pm+1,pr);
52      }
53      ll ask(int L,int R){return __segtree_ask(L,R,1,l,r);}
54
55  }LST[1<<20],RST[1<<20];
56  void split(int p) {
57      if(ds[fa[p]]!=p)tp[p]=p;
58      else tp[p]=tp[fa[p]];
59      dfn[p]=++tot;
60      if(ds[p])split(ds[p]),split(ls[p]+rs[p]-ds[p]);
61      if(ds[p]!=ls[p])szdl[dfn[p]]=sz[ls[p]];
62      if(ds[p]!=rs[p])szdr[dfn[p]]=sz[rs[p]];
63      if(ds[fa[p]]!=p)
64
65  LST[p].init(dfn[p],dfn[p]+le[p]-1,szdl),RST[p].init(dfn[p],dfn[p]+le[p]-1,szdr);
66  ll FULL;
67  void updL(int p,int d) {
68      for(;p;) {
69          LST[tp[p]].upd(dfn[tp[p]],dfn[p]-1,d);
70          p=tp[p];
71          bool sig=(p==rs[fa[p]]);
72          p=fa[p];
73          if(sig)LST[tp[p]].spc[dfn[p]-dfn[tp[p]]]+=d*1ll*sz[ls[p]];
74      }
75  }
76  void updR(int p,int d) {
77      for(;p;) {
78          RST[tp[p]].upd(dfn[tp[p]],dfn[p]-1,d);
79          p=tp[p];
80          bool sig=(p==ls[fa[p]]);
81          p=fa[p];
82          if(sig)RST[tp[p]].spc[dfn[p]-dfn[tp[p]]]+=d*1ll*sz[rs[p]];
83      }
84  }
85  ll askL(int p) {
86      ll ret=0;
87      for(;p;) {
```

```
        ret+=LST[tp[p]].ask(dfn[tp[p]],dfn[p]-1);
        p=tp[p];
        bool sig=(p==rs[fa[p]]);
        p=fa[p];
        if(sig)ret+=LST[tp[p]].spc[dfn[p]-dfn[tp[p]]];
    }
    return ret;
}
ll askR(int p) {
    ll ret=0;
    for(;p;) {
        ret+=RST[tp[p]].ask(dfn[tp[p]],dfn[p]-1);
        p=tp[p];
        bool sig=(p==ls[fa[p]]);
        p=fa[p];
        if(sig)ret+=RST[tp[p]].spc[dfn[p]-dfn[tp[p]]];
    }
    return ret;
}
void upd(int L,int R,int d) {
    if(L==1&&R==n){FULL+=d;return;}
    if(L==1)return updL(R+1,d);
    if(R==n)return updR(L-1,d);
    L--,R++;
    for(;;) {
        if(tp[L]==tp[R]) {
            if(de[L]>de[R])
                RST[tp[L]].upd(dfn[R]+1,dfn[L]-1,d);
            if(de[R]>de[L])
                LST[tp[L]].upd(dfn[L]+1,dfn[R]-1,d);
            return;
        }
        if(de[tp[L]]>=de[tp[R]]) {
            RST[tp[L]].upd(dfn[tp[L]],dfn[L]-1,d);
            L=tp[L];
            bool sig=(L==ls[fa[L]]);
            L=fa[L];

            if(sig)if(tp[L]!=tp[R])RST[tp[L]].spc[dfn[L]-dfn[tp[L]]]+=d*1ll*sz[rs[L]];
        }
        else {
            LST[tp[R]].upd(dfn[tp[R]],dfn[R]-1,d);
            R=tp[R];
            bool sig=(R==rs[fa[R]]);
```

```
            R=fa[R];

            if(sig)if(tp[L]!=tp[R])LST[tp[R]].spc[dfn[R]-dfn[tp[R]]]+=d*1ll*sz[ls[R]];
        }
    }
}
ll ask(int L,int R) {
    if(L==1&&R==n)return n*1ll*FULL;
    if(L==1)return askL(R+1);
    if(R==n)return askR(L-1);
    ll ret=0;
    L--,R++;
    for(;;) {
        //printf("%d %d %lld\n",L,R,ret);
        if(tp[L]==tp[R]) {
            if(de[L]>de[R])
                ret+=RST[tp[L]].ask(dfn[R]+1,dfn[L]-1);
            if(de[R]>de[L])
                ret+=LST[tp[L]].ask(dfn[L]+1,dfn[R]-1);
            return ret;
        }
        if(de[tp[L]]>=de[tp[R]]) {
            ret+=RST[tp[L]].ask(dfn[tp[L]],dfn[L]-1);
            L=tp[L];
            bool sig=(L==ls[fa[L]]);
            L=fa[L];
            if(sig)if(tp[L]!=tp[R])ret+=RST[tp[L]].spc[dfn[L]-dfn[tp[L]]];
        }
        else {
            ret+=LST[tp[R]].ask(dfn[tp[R]],dfn[R]-1);
            R=tp[R];
            bool sig=(R==rs[fa[R]]);
            R=fa[R];
            if(sig)if(tp[L]!=tp[R])ret+=LST[tp[R]].spc[dfn[R]-dfn[tp[R]]];
        }
    }
}
int main() {
    //freopen("pigeons.in","r",stdin);
    //freopen("pigeons.out","w",stdout);
    scanf("%d%d",&n,&m);
    int i;
    for(i=n+1;i<(n<<1);i++)
        scanf("%d%d",ls+i,rs+i),fa[ls[i]]=fa[rs[i]]=i;
```

```
    for(i=1;i<(n<<1);i++)
        if(!fa[i])rt=i;
    init(rt),split(rt);
    for(i=1;i<=m;i++) {
        int op,l,r,d;
        scanf("%d%d%d",&op,&l,&r);
        if(op==1)scanf("%d",&d),upd(l,r,d);
        else printf("%lld\n",ask(l,r));
    }
}


```