比赛链接

补题情况

题目	蒋贤蒙	王赵安	王智彪
D	2	0	1
E	0	0	2
L	0	1	2

题解

D. Contest Strategy

题意

给定 n 道题以及完成每道题需要的时间。对固定读题顺序，一个队伍会先按顺序读 k 道题，然后完成读过的题中需要时间最小的题。

然后按顺序读下一道题(如果还有题没读的话)，然后再完成读过且未完成的题中需要时间最小的题，不断重复，直到完成所有题。

对所有的读题顺序 ($1\sim n$ 的排列)，问这个队伍的总罚时之和。

题解

首先假如按做题顺序每题的做题时间分别为 $t_1,t_2\cdots t_n$ 则总罚时为 $\sum_{i=1}^n (n+1-i)t_i$

将所有题按所需时间大小排序，同时设完成第 i 题需要的时间为 a_i 不难发现对于最后 $k-1$ 道题，第 i 题一定也是做题顺序中的第 i 题。

接下来只考虑前 $n-k+1$ 道题，设 $f(i,j)$ 表示第 i 道题在读完 j 题后已经被完成的总方案数。

不难发现第 i 道题在读完 j 题后已经被完成等价于读题顺序前 j 道题一定包含第 i 题且至少包含 $k-1$ 道需要时间大于 i 的题。

考虑枚举前 j 题中一定包含第 i 题且正好包含 p 道需要时间大于 i 的题的方案数，于是有

$$ f(i,j)=j!(n-j)!\sum_{p=k-1}^{j-1}{n-i\choose p}{i-1\choose j-p-1} $$

然后第 i 道题在读完 j 题后恰好被完成的方案就是 $f(i,j)-f(i,j-1)$ 此时对答案的贡献为

$$ (n+k-j)a_i\times (f(i,j)-f(i,j-1)) $$

时间复杂度 $O\left(n^3\right)$

```
const int mod=1e9+7,MAXN=305;
int a[MAXN],frac[MAXN],invf[MAXN];
```

```
int dp[MAXN][MAXN];
int quick_pow(int a,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*a%mod;
        a=1LL*a*a%mod;
        k>>=1;
    }
    return ans;
}
int C(int n,int m){
    if(n<m)return 0;
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
int main(){
    int n=read_int(),k=read_int();
    _rep(i,1,n)a[i]=read_int();
    sort(a+1,a+n+1);
    frac[0]=1;
    _for(i,1,MAXN)
    frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--)
    invf[i-1]=1LL*invf[i]*i%mod;
    LL ans=0;
    _rep(i,1,n-k+1){
        _rep(j,k,n){
            _for(t,k-1,j)
            dp[i][j]=(dp[i][j]+1LL*C(n-i,t)*C(i-1,j-t-1))%mod;
            dp[i][j]=1LL*dp[i][j]*frac[j]%mod*frac[n-j]%mod;
            ans=(ans+1LL*(dp[i][j]-dp[i][j-1])*(n+k-j)%mod*a[i])%mod;
        }
    }
    _rep(i,n-k+2,n)
    ans=(ans+1LL*frac[n]*a[i]%mod*(n+1-i))%mod;
    enter(ans);
    return 0;
}
```

E. Enclosure

题意

给定 k 个点，作为初始的区域，对于剩下的 $n-k$ 个点，我们选择一个加入到现在这个区域，求构成的新的区域最大面积（当然区域的面积就是凸包的面积了）。

数据范围： $3≤k,n≤100,100$ ，坐标绝对值不超过 10^{9} 。

题解

思路来的很快，刚看这道题的时候，显然先求一下 k 个点的凸包，然后对于剩下的 $n-k$ 个点，可能极角排序一下然后在哪二分一下？找到相当于是这个点到凸包的两条切线的交点，然后求一下面积，单个复杂度是 $O(logn)$ 的，然后 $n-k$ 个点，大概就是 $O(nlogn)$ 可以通过，事实证明，真的有队伍这么过，但是我当时 zz 了，并不知道在哪里找二分的点（听说是在凸包里找一个点作为极角排序的那个点，大概重心？开始胡诌 $\cdots$ ）。

来说说正解，正解是我推翻了上个想法之后两分钟就想到的，显然最大面积的点一定是总图形的凸包上的顶点，因为其他任意的点由几何知识可以知道到某边距离一定不大于这条边被大凸包“ 包围 ”的那两个点到这条边距离的最大值，然后求一下三角形面积就知道了。

对于最开始选的一个点，我们可以暴力找到上述两条切线的交点，一个设为 l 一个设为 r 然后我们对于大凸包上的点开始转，容易发现 l 和 r 也会跟着转（其实就是双指针啦），这样转一圈总复杂度是 $O(n)$ 的（应该没口胡错 $\times$ ）。至于面积，我们可以发现新增的面积肯定是 l 和 r 和放进来的点构成的三角形减去 l 到 r 直接构成的多边形，显然我们在转的时候可以动态维护 l 和 r 之间的这个多边形的面积，用三角剖分搞一下就可以了，然后对于三角形，直接求面积就可以了，这样单次维护面积的复杂度是 $O(1)$ 的，总复杂度还是 $O(n)$

然后这题细节很多，首先肯定要用 $long long$ 而不能用 $double$ 否则第二个点都过不去 $\times$ 。然后我暴力找的那个点我想让他找的准，所以对于小凸包（就是 k 个点构成的凸包）上的点，我直接 $continue$ 了，然后有可能真的没找到这个符合规范的点，就直接输出小凸包面积就可以了，因为忘了这茬，所以在第三个数据那里 re 了…然后就 ac 了，撒花~

```
int n,k;
int main() {
    scanf("%d %d",&n,&k);
    for(int i=0; i<k; i++) {
        scanf("%lld %lld",&P0.p[i].x,&P0.p[i].y);
        P0.p[i].id=i+1;//这里标记一下哪些是小凸包的点
        P01.p[i]=P0.p[i];
    }
    P0.n=k;
    P01.n=n;
    P0.Graham(po);
    Point ptmp;
    int p1,p2;
    for(int i=k; i<n; i++) {
        scanf("%lld %lld",&P01.p[i].x,&P01.p[i].y);
    }
    P01.Graham(po1);
    ll ans=po.getarea();//先求一下小凸包面积
    ll tans=0;
    int l=-1,r=-1;
    int pp=-100;
    for(int i=0;i<po1.n;i++) {
        if(po1.p[i].id) {
            continue;
        }
        pp=i;
        break;
```

```
    }
    if(pp==-100) {//这里要特判没有外面点的情况，也就是大小凸包重合，没有算下去的必要。
        printf("%lld.%d\n",ans>>1,ans%2==0?0:5);
        return 0;
    }
    for (int i=0; i<po.n;i++) {//暴力求第一个的l,r指针
        if ((l==-1)||(((po.p[i]-po1.p[pp])^(po.p[l]-po1.p[pp]))<=0)) {
            l=i;
        }
        if ((r==-1)||(((po.p[i]-po1.p[pp])^(po.p[r]-po1.p[pp]))>=0)) {
            r=i;
        }
    }
    ll ts=0;
    for(int i=l;i!=r;i=(i+1)%po.n) {
        ts+=((po.p[i])^(po.p[(i+1)%po.n]));//暴力算一下多边形面积
    }
    ts+=(po.p[r]^po.p[l]);//三角剖分算回来
tans=max(tans,labs((po1.p[pp]^po.p[l])+(po.p[r]^po1.p[pp])+(po.p[l]^po.p[r]
))-labs(ts));//先求这个点的最大值 三角形面积-多边形面积
    for(int i=pp+1;i<po1.n;i++) {
        while((!(po.p[r]==po1.p[i]))&&((po.p[(r+1)%po.n]-
po1.p[i])^(po.p[r]-po1.p[i]))>=0) {
            ts-=(po.p[r]^po.p[l]);//都是三角剖分
            ts+=(po.p[r]^po.p[(r+1)%po.n]);
            ts+=(po.p[(r+1)%po.n]^po.p[l]);
            r=(r+1)%po.n;
        }
        while((!(po.p[l]==po1.p[i]))&&((po.p[(l+1)%po.n]-
po1.p[i])^(po.p[l]-po1.p[i])) <= 0) {
            ts-=(po.p[r]^po.p[l]);//都是三角剖分
            ts-=(po.p[l]^po.p[(l+1)%po.n]);
            l=(l+1)%po.n;
            ts+=(po.p[r]^po.p[l]);
        }
tans=max(tans,labs((po.p[l]^po1.p[i])+(po1.p[i]^po.p[r])+(po.p[r]^po.p[l]))
-labs(ts));
    }
    printf("%lld.%d\n",(ans+tans)>>1,(ans+tans)%2==0?0:5);
    return 0;
}
```

L. Windy Path

题意

给定 n 个点，并且给一个操作序列 str 其中 $str[i]$ 可以为 L 也可以为 R 如果是 L 则需要你给的点的序列满足 $p[i],p[i+1],p[i+2]$ 是逆时针排序 R 则需要为顺时针，请给出满足要求的操

作序列。

题解

比赛的时候被这个题的题意吓到了，因为是打着计算几何幌子的其他题，结果真的是计算几何…，给我的队友们道歉 qwq 其实我们只需要满足每一个操作之前我们都准备好下一步一定能进行这样的操作就可以了，随便选第一个点，从第二个点开始，比如我下一步是 L 那我这一步就从一个点走到逆时针的第一个点（其实是沿着凸包走），这样我从这个方向出发，到剩下的任意一个点都是逆时针方向 R 的话就到顺时针的下一个点，这样再下一个操作走到任意一个位置都是顺时针。

然后第二个操作开始，就按照上面的规律，一直走下去就行了，这样是一定有解的。因为数据范围很小，时间很充裕，每一步都暴力求凸包就可以了。

```
char str[1001];
bool vis[1001];
int main() {
    scanf("%d",&PO.n);
    int nn=PO.n;
    for(int i=0; i<PO.n; i++) {
        scanf("%lf %lf",&PO.p[i].x,&PO.p[i].y);
        PO.p[i].id=i;
    }
    scanf("%s",str+1);
    PO.Graham(po);
    int pos1=po.p[0].id+1;
    printf("%d ",pos1);
    vis[pos1]=true;
    int laspos,lasid;
    for(int i=0; i<po.n; i++) {
        if(po.p[i].id==pos1-1) {
            laspos=i;
            lasid=pos1-1;
            break;
        }
    }
    for(int i=1; i<=nn-2; i++) {
//        for(int j=0; j<po.n; j++) {
//            po.p[j].output();
//        }
        int mmp=(str[i]=='L')?1:-1;
        int tid=po.p[(laspos+mmp+po.n)%po.n].id;
        printf("%d ",tid+1);
        vis[tid+1]=true;
        int tpos=0;
        for(int j=0; j<PO.n; j++) {
            if(PO.p[j].id==lasid) {
                tpos=j;
                break;
            }
        }
```

```
        PO.p[tpos]=PO.p[PO.n-1];
        PO.n--;
        PO.Graham(po);
        for(int j=0; j<po.n; j++) {
            if(po.p[j].id==tid) {
                laspos=j;
                lasid=tid;
                break;
            }
        }
    }
    for(int i=1; i<=nn; i++) {
        if(!vis[i]) {
            printf("%d\n",i);
            return 0;
        }
    }
    return 0;
}
```

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:contest8&rev=1627745053

Last update: **2021/07/31 23:24**