牛客练习赛83

比赛链接

F - 音游家的谱面(Hard version)

题意

给定 n 条轨道和一个含有 m 个音符的谱，每个音符出现的轨道为 p_i

玩家有两个手指，一开始分别位于轨道 1 和轨道 n 的底部，且两个手指每秒可以向左右移动一格。

玩家的任务是在音符恰好到达底部时用手指敲击音符。

现要求构造每个音符到达底部的时刻，使得谱中每个音符依次出现的时刻不早于上一个音符，且玩家可以顺利完成任务。

要求最小化最后一个音符达到底部的时刻，如果有多种方案，任意输出一种即可$(n,m\le 5000)$

题解

设 $f(i,j)$ 表示第 i 个音符到达底部且一只手位于 p_i 且另一只手位于 j 的最小时刻，于是有状态转移

$$ f(i,j)\gets f(i-1,k)+|p_i-p_{i-1}|(|j-k|\le |p_i-p_{i-1}|) $$

$$ f(i,j)\gets f(i-1,k)+|p_i-k|(|j-p_{i-1}|\le |p_i-k|) $$

暴力做法时间复杂度 $O(n^2m)$方案输出用回溯即可。

考虑线段树优化，枚举 k 然后更新特定范围的 $f(i,j)$线段树只需要维护区间最值操作和单点查询操作，时间复杂度 $O(nm\log n)$

该复杂度实际上已经可能可以通过本题(如果卡常技巧优秀)，但实际上存在进一步优化。

不难看出第一种转移式可以用单调队列优化。

第二种转移式预处理出 $t=0\sim n-1$ 时 $\min(f(i-1,k)+|p_i-k|(t\le |p_i-k|))$ 的结果，然后枚举 j 然后根据 $|j-p_{i-1}|$ 直接查询即可。

时间复杂度优化为 $O(nm)$

教训：本人一开始想出的转移式如下所示。

$$ f(i,j)= \min(f(i-1,k)+\min(\max(|p_i-p_{i-1}|,|j-k|),\max(|p_i-k|,|j-p_{i-1}|))) $$

只能用线段树做到$O(nm\log n)$实现复杂且没法进一步优化，以后应该在填表法比较复杂时考虑刷表法。

```
const int MAXN=5005,inf=1e9;
int dp[MAXN][MAXN],pre[MAXN][MAXN],p[MAXN],res[MAXN];
pair<int,int> que[MAXN],temp[MAXN];
int main()
{
    int n=read_int(),m=read_int();
    _rep(i,1,m)p[i]=read_int();
    _rep(i,0,m)_rep(j,1,n)dp[i][j]=inf;
    p[0]=1;
    dp[0][n]=0;
    _rep(i,1,m){
        int head=0,tail=1,lim=abs(p[i]-p[i-1]);
        _rep(j,1,lim){
            while(head>=tail&&que[head].first>=dp[i-1][j])head--;
            que[++head]=make_pair(dp[i-1][j],j);
        }
        _rep(j,1,n){
            if(j+lim<=n){
                while(head>=tail&&que[head].first>=dp[i-1][j+lim])head--;
                que[++head]=make_pair(dp[i-1][j+lim],j+lim);
            }
            if(head>=tail&&que[tail].second+lim<j)tail++;
            dp[i][j]=que[tail].first+lim;
            pre[i][j]=que[tail].second;
        }
        temp[n]=make_pair(inf,0);
        for(int lim=n-1;lim>=0;lim--){
            int lpos=p[i]-lim,rpos=p[i]+lim;
            temp[lim]=temp[lim+1];
            if(lpos>=1)
            temp[lim]=min(temp[lim],make_pair(dp[i-1][lpos]+abs(p[i]-
lpos),lpos));
            if(rpos<=n)
            temp[lim]=min(temp[lim],make_pair(dp[i-1][rpos]+abs(p[i]-
rpos),rpos));
        }
        _rep(j,1,n){
            int lim=abs(j-p[i-1]);
            if(dp[i][j]>temp[lim].first){
                dp[i][j]=temp[lim].first;
                pre[i][j]=temp[lim].second;
            }
        }
    }
    int ans=inf;
    _rep(i,1,n)
    ans=min(ans,dp[m][i]);
    _rep(i,1,n){
        if(ans==dp[m][i]){
            int pos=i;
```

```
            for(int j=m;j;j--){
                res[j]=dp[j][pos];
                pos=pre[j][pos];
            }
            break;
        }
    }
    _rep(i,1,m)
    space(res[i]);
    return 0;
}
```