qnmbql的11.15日考试总结

1 Reverse

题目：小G有一个长度为n的01串T ，其中只有TS = 1，其余位置都是0。现在小G可以进行若干次以下操作： 选择一个长度为K的连续子串（ K是给定的常数），翻转这个子串。
对于每个i, i ∈ [1, n]，小G想知道最少要进行多少次操作使得Ti = 1. 特别的，有m个“禁止位置”，你需要保证在操作过程中1始终不在任何一个禁止位置上。

考场思路：大家都想到了把这道题转化成最短路问题，但仔细想想其实没有必要。因为不管是怎么样的最短路算法，都无法避免一个点的扩展其实有很多次是无效的（扩展的点早已被扩展），所以 说这样的最短路算法实质上与暴力区别不大，甚至不如$01bfs$。于是我考场就敲了个暴力，然而 还敲挂了，值得反思。

正解：可以发现转移到的点一定是一段区间内的奇数或者偶数点，于是一种简单的优化方法是在bfs时开两个set维护当前有哪些奇数点和偶数点还未被bfs到，转移时直接在set上$lowerbound$，就不会访问已经bfs到过的点了。$O(nlogn)$

代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
#include<algorithm>
#include<queue>
#include<set>
using namespace std;
const int MAXN=100005;
const int INF=99999999;
int n,kk,m,s;
int dis[MAXN];
int q[2*MAXN],head=1,tail=1;
set<int> odd,even;
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("reverse.in","r",stdin);
    freopen("reverse.out","w",stdout);
    cin>>n>>kk>>m>>s;
    for(int i=1;i<=m;i++){
        int t; cin>>t;
        dis[t]=INF;
    }

    for(int i=1;i<=n;i++){
        if(dis[i]!=INF && i!=s) {
            dis[i]=-1;
            if(i&1) odd.insert(i);
            else even.insert(i);
        }
    }
```

```
    dis[s]=0;
    q[tail]=s; tail++;

    while(head<tail){
        int x=q[head]; head++;
        int l=max(1,x-kk+1),r=min(n,x+kk-1);
         l=l+(l+kk-1)-x; r=r+(r-kk+1)-x;
         //cout<<"++";
         if(l&1){
            for(set<int>::iterator it=odd.lower_bound(l), j;it != odd.end() &&
*it<=r; it=j){ j=it; ++j;
                int v=*it;
                dis[v]=dis[x]+1;
                q[tail]=v; tail++;
                odd.erase(it);
            }
         }
         else{
            for(set<int>::iterator it=even.lower_bound(l), j;it != even.end() &&
*it<=r; it=j){ j=it; ++j;
                int v=*it;
                dis[v]=dis[x]+1;
                q[tail]=v; tail++;
                even.erase(it);
            }
         }
    }

    for(int i=1;i<=n;i++){
        if(dis[i]==INF || (dis[i]==0 && i!=s)){
            cout<<-1<<" "; continue;
        }
        cout<<dis[i]<<" ";
    }
    return 0;
}
```

T2 Silhouette

题目：有一个n × n的网格，在每个格子上堆叠了一些边长为1的立方体。现在给出这个三维几何体的正视图和左视图，求有多少种与之符合的堆叠立方体的方案。两种方案被认为是不同的，当且仅　当某个格子上立方体的数量不同。输出答案对10^9+7取模的结果。

考场思路：搜索——基于每一列和每一行的最大限制条件

正解：不妨先将A、B排序，显然这样并不会有什么影响。如果A的最大值不等于B的最大值则答案是0，否则一定有解。显然有$x_{i,j}\leq min(A_i,B_j)$。从大到小枚举每个A;B中出现的值s，每次确定所有$min(A_i,B_j)=s$的位置的值，我们先来考虑s为最大值的情况，此时我们要确定的位置构成了一个矩形。可以发现我们要解决这样一个子问题：一个$a*b$的矩形，每个位置的值在$[0,s]$中，且每行每列的最大值均s,需要计算有多少取值的方案。

考虑容斥，设$f(i)$为至少有i行的限制不满足条件时的方案迷需要保证每一列都满足条件遴，

那么$f(i)=C_a^i(s^i*((s+1)^{a-i}-s_{a-i}))^b$方案数就是$\sum_{i=0}^{a}(-1)^if(i)$。$s$不是最大值时，我们每次要确定的位置可能是一个$L$形，仍然可以用相同的方法容斥。

L型时：$f(i)=C_i^a(s^i((s+1)^{a+c-i}-s^{a+c-i}))^b((s+1)^{a-i}s^i)^d$ 由于需要快速幂，

复杂度为$O(nlogn)$

代码：

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
#include<vector>
#define ll long long
using namespace std;
const int p=1000000007;
const int MAXN=100005;
int n,a[MAXN],b[MAXN],s[2*MAXN];
ll jc[MAXN],jc_inv[MAXN];
ll ans=1;
ll qsm(ll a,ll b){
    ll val=1;
    for(;b;b>>=1){
        if(b&1){
            val=val*a%p;
        }
        a=a*a%p;
    }
    return val;
}
ll cc(ll x,ll y){
    return jc[x]*jc_inv[y]%p*jc_inv[x-y]%p;
}
ll lucas(ll x,ll y){
//  if(x<y) return 0;
    if(y==0) return 1;
    return lucas(x/p,y/p)*cc(x%p,y%p)%p;
}
ll f(ll a,ll b,ll c,ll d,ll i,ll s){
    return lucas(a,i)*qsm(qsm(s,i)*(qsm(s+1,a+c-i)-qsm(s,a+c-
i)+p)%p,b)%p*qsm(qsm(s+1,a-i)*qsm(s,i)%p,d)%p;
}
ll alrea,alreb;
ll work(ll a,ll b,ll c,ll d,ll s){
    ll val=0;
    for(int i=0;i<=a;i++){
        ll ff=f(a,b,c,d,i,s);
        if(i&1) val=(val-ff+p)%p;
        else val=(val+ff)%p;
    }
    return val;
}
int main(){
    freopen("silhouette.in", "r", stdin);
    freopen("silhouette.out", "w", stdout);
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    jc[0]=1; jc_inv[0]=1;
    for(int i=1;i<MAXN;i++){
```

```
        jc[i]=jc[i-1]*i%p;
    }
    jc_inv[MAXN-1]=qsm(jc[MAXN-1],p-2);
    for(int i=MAXN-1;i>=1;i--) jc_inv[i-1]=jc_inv[i]*i%p;
    cin>>n; alrea=alreb=n+1;
    for(int i=1;i<=n;i++){
        cin>>a[i]; s[i]=a[i];
    }
    for(int i=1;i<=n;i++){
        cin>>b[i]; s[i+n]=b[i];
    }
    sort(a+1,a+n+1); sort(b+1,b+n+1);

    if(a[n]!=b[n]){cout<<0; return 0;   }
    sort(s+1,s+2*n+1);
    int num=unique(s+1,s+2*n+1)-(s+1);
    int cnta=n,cntb=n;
    for(int i=num;i>=1;i--){

        while(a[cnta-1]==s[i] && cnta-1) cnta--;
        //cout<<"++";
        while(b[cntb-1]==s[i] && cntb-1) cntb--;

        ans=(ans*work(alrea-cnta,alreb-cntb,n-alrea+1,n-alreb+1,s[i])%p);
        alrea=cnta; alreb=cntb;
    }
    cout<<ans;
}
```

T3 seat

题目：有n + 2个座位等距地排成一排，从左到右编号为0至n + 1。最开始时0号以及n + 1号座位上已经坐了一个小G，接下来会有n个小G依次找一个空座位坐下。由于小G们坐得太近就容易互相搏弈，每个小G会找一个当前离最近的小G距离最远的座位坐下。如果有多个备选的座位，这个小G 会等概率选择其中一个。给出n，求第i个坐下的小G坐在j号座位的概率, 对P 取模。具体来说，如 果答案化为最简分数可以表示a/b,你需要输出a × b−1，其中b−1 = bP-2 (mod P).

考场思路：无，果断跳过。

正解：一个结论是，对于任意一个人，他坐下时离最近的人的距离是固定的，不随前面的人的 决策而改变。这样我们可以将所有人按坐下时的距离分成若干层。另一个结论是，无论之前 每一层如何决策，轮到这一层时,空区间的长度构成的可重集也是固定的。对于最后一层特殊处理,接下来均默认不是最后一层。对于之前的每一层，考虑在哪些空区间中央坐下可使得距离最 大，其中可能会有一些长度为奇数的区间和一些长度为偶数的区间，而对于每个人来说，坐在任 意一个奇数的区间的中央的概率都是相等的，偶数同理。那么我们只需要知道，每个人有多大的 概率坐在一个奇/偶数区间的中央。考虑dp,$dp(i,j)$表示这一层已经坐下i个人之后，还有j个长度 为偶数的区间的概率，转移只需考虑当前这个人坐了哪类区间即可。dp之后容易算出之前要求的 概率。区间长度为奇数时位置是固定的；考虑区间长度为偶数的情况，此时会出现两个位置可 供选择，但无论选择哪一个，都会将区间划分成长度为$n/2$和$n/2-1$ 的两段。因此这两种选择具有对称性，我们只需要假定选择其中的一个，算出这种情况下之后的层的答案，即可对称地推 出另一种情况下的答案。瓶颈在利用对称性推答案的地方，复杂度$O(n^2logn)$

T4 Ancient

题目：给定$q,n(1\le q,n\le 10^9)$,计算$q^{\sum_{d|n}C_n^d}mod 999911659$

考场思路：暴力枚举约数，计算值。$lucas$定理优化计算组合数。止步于此。

正解：用欧拉定理的推论得：

$$q^{\sum d|n\,C_n^d}\ mod\ 999911659=q^{\sum d|n\,C_n^d\ mod\ 999911658}\ mod\ 999911659$$

那么关键计算$\sum_{d|n}\ C_n^d\ mod\ 999911658$.直接Lucas$Lucas$绝对挂，那么尝试把模数缩小再合并

将999911658因数分解，可得$999911658=2\times 3\times 4679\times 35617$那么把模数缩小，枚举$n$的因数$d$，然后运用$Lucas$定理把$C_n^d$算出来，分别计算出$\sum_{d|n}\ C_n^d$对2,3,4679,35617四个质数取模的结果，记为a_1,a_2,a_3,a_4 最后，用中国剩余定理求解一下方程组：

$$\begin{cases} x\equiv a_1(\mathrm{mod}\,2) \\ x\equiv a_2(\mathrm{mod}\,3) \\ x\equiv a_3(\mathrm{mod}\,4679) \\ x\equiv a_4(\mathrm{mod}\,35617) \end{cases}$$

然后就得到了最小的非负整数解x，之后用快速幂求一下就得到答案.

代码：

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
#include<algorithm>
#define ll long long
using namespace std;
const int md=999911658;
ll p[5]={0,2,3,4679,35617};
ll jc[40005];
const int MAXN=100005;
ll n,g;
ll m[10],a[5];
void ask_jc(int x){
    jc[0]=jc[1]=1;
    for(int i=2;i<=x;i++) jc[i]=(jc[i-1]*i)%x;
    return ;
}
ll qsm(ll a,ll b,ll p){
    ll val=1;
    for(;b;b>>=1){
        if(b&1){
            val=val*a%p;
        }
        a=a*a%p;
    }
```

```
    return val;
}
ll yz[MAXN],cnt;
ll cc(ll m,ll n,ll p){
    if(m<n) return 0;
    return jc[m]*qsm(jc[n]*jc[m-n]%p,p-2,p)%p;
}
ll lucas(ll m,ll n,ll p){
    if(m==0) return 1;
    if(m<n) return 0;
    return 1ll*lucas(m/p,n/p,p)*cc(m%p,n%p,p);
}
ll ans;
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    freopen("ancient.in","r",stdin);
    freopen("ancient.out","w",stdout);
    cin>>n>>g;
    if(!(g%(md+1))){cout<<0; return 0;}
    for(int i=1;i*i<=n;i++){
        if(n%i==0){
            yz[++cnt]=i;
            if(i!=(n/i)){
                yz[++cnt]=(n/i);
            }
        }
    }
//  for(int i=1;i<=cnt;i++){
//      cout<<yz[i]<<" ";
//  }
    for(int i=1;i<=4;i++){
        ask_jc(p[i]);
        for(int j=1;j<=cnt;j++){

            a[i]+=lucas(n,yz[j],p[i]);
            //cout<<"++";
            a[i]%=p[i];
        }
    }

    for(int i=1;i<=4;i++){
        ans+=(a[i]*(md/p[i])%md*qsm(md/p[i],p[i]-2,p[i]))%md;
        ans%=md;
    }
    cout<<qsm(g,ans,md+1);
}
```

总结

数学专题，学会推式子，以及对于所学知识在题目中的灵活运用是很重要的。同时，考场暴力是一定不能写挂的！！