总共n个蜡烛，m个被吹灭了，我们每次只能吹灭隔壁已经有蜡烛被吹灭了的蜡烛。

那么我们可以根据m个被吹灭的蜡烛为隔板，把n-m个待吹灭的蜡烛分成m+1段（可能存在空的段）

对于最左侧和最右侧的那两段只能不停吹灭最右侧/最左侧的蜡烛，对于这一段蜡烛来说只有一种吹灭的情况。而对于剩下的m-1段来说，从左侧吹灭或者从右侧吹灭都是可行的，根据排列组合，如果这一段总共有x个蜡烛，那么总共可行的情况是2^(x-1)，因为每次我们都有左右2种取法，除了最后一次只剩下一个蜡烛的情况。

接着就是这m+1段蜡烛之间的先后顺序了。我们注意到每一段的蜡烛是绑定在一起的，所以其实我们可以依次往可能的空位里面放这m+1段。

剩下还有temp=n-m个蜡烛没吹灭，第一段蜡烛的个数为x1，那么第一段蜡烛所在顺序取法就是组合数C（x1,temp），放完第一段后，剩余空位变成temp-x1个，第二段蜡烛的个数为x2，那么第二段蜡烛所在顺序取法就是组合数C（x2,temp-x1）........

以上就结论完毕，算就是了。

此题数据范围较小，可以利用杨辉三角预处理出可能需要的组合数，或者也可以用乘法逆元直接算组合数。

第一份代码：杨辉三角预处理

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

**using** **namespace** std;

**typedef** **long** **long** ll;

**const** **int** maxn=1007;

**const** **int** mod=1e9+7;

**int** a[maxn];

ll fac[maxn],c[maxn][maxn];

ll posmod(ll a){

**return** (a%mod+mod)%mod;

}

**void** calfac(){//fac[i]表示2^{i-1}次方，0除外

    fac[0]=fac[1]=1;

**for**(**int** i=2;i<maxn;i++)

        fac[i]=posmod(fac[i-1]\*2);

}

**void** calc(){//递推求组合数

    c[0][0]=1;

**for**(**int** i=1;i<maxn;i++){

        c[i][0]=1;

**for**(**int** j=1;j<=i;j++)

            c[i][j]=posmod(c[i-1][j]+c[i-1][j-1]);

    }

}

**int** main(){

    #ifdef LOCAL

    freopen("in.txt","r",stdin);

    freopen("out.txt","w",stdout);

    #endif // LOCAL

    ios::sync\_with\_stdio(**false**);

**int** x,y;

    calfac();

    calc();

**while**(cin>>x>>y){

**for**(**int** i=1;i<=y;i++)

            cin>>a[i];

        sort(a+1,a+y+1);

**int** left=x-y;

        ll ans=1;

**for**(**int** i=1;i<=y;i++){//最后一个区间肯定是C\_{x}^{x}的情况，就省去了

            ans=posmod(ans\*c[left][a[i]-a[i-1]-1]);

            left-=a[i]-a[i-1]-1;

        }

**for**(**int** i=2;i<=y;i++)

            ans=posmod(ans\*fac[a[i]-a[i-1]-1]);

        cout<<ans<<endl;

    }

}

第二份代码：乘法逆元

#include<bits/stdc++.h>

#define ll long long

#define IOS ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

**using** **namespace** std;

**const** ll maxn=1e3+7;

**const** ll mod=1e9+7;

ll num[maxn];

ll qpow(ll a,ll b)//快速幂

{

    ll ret=1;

**while**(b>0)

    {

**if**(b&1) ret=ret\*a%mod;

        a=a\*a%mod;

        b>>=1;

    }

**return** ret;

}

ll cas[maxn];

**int** main()

{

    IOS;

    cas[0]=cas[1]=1;

**for**(ll i=2;i<=1e3;i++) cas[i]=cas[i-1]\*i%mod;

    ll n,m;

    cin>>n>>m;

    num[0]=0;

**for**(ll i=1;i<=m;i++) cin>>num[i];

    sort(num+1,num+1+m);

    ll ans=1;

    ll temp=n-m;

**for**(ll i=1;i<=m;i++)

    {

        ans=ans\*cas[temp]%mod\*qpow(cas[temp-(num[i]-num[i-1]-1)],mod-2)%mod\*qpow(cas[num[i]-num[i-1]-1],mod-2)%mod;//乘法逆元求取组合数C（num[i]-num[i-1]-1,temp）

        temp-=num[i]-num[i-1]-1;

    }

**for**(ll i=2;i<=m;i++)

    {

        ans=ans\*qpow(2,num[i]-num[i-1]-2)%mod;

    }

    cout<<ans<<endl;

}