1

考虑从初始坐标向上下左右走，加上初始位置自己就是答案。

```
#include<bits/stdc++.h>
using namespace std;
string s[200];int n,m,x,y,ans=1;
int main(){
    cin>>n>>m>>x>>y;
    for(int i=0;i<n;i++)cin>>s[i];
    for(int i=x;i<n;i++)
    if(s[i][y-1]=='#')break;
    else ans++;
    for(int i=y;i<m;i++)
    if(s[x-1][i]=='#')break;
    else ans++;
    for(int i=x-2;i>=0;i--)
    if(s[i][y-1]=='#')break;
    else ans++;
    for(int i=y-2;i>=0;i--)
    if(s[x-1][i]=='#')break;
    else ans++;
    cout<<ans<<endl;
}
```

2

对所有序列进行排序，相同的序列会排在一起，如果一个序列在排完序之后与其上一位相同，则在统计不同的序列时就不能将其统计进去。

```
#include<bits/stdc++.h>
using namespace std;
vector<int>ai[300000];
bool operator<(vector<int>a,vector<int>b){
    if(a.size()!=b.size())return a.size()<b.size();
    for(int i=0;i<a.size();i++)
    if(a[i]!=b[i])return a[i]<b[i];
    return 0;
}
bool operator==(vector<int>a,vector<int>b){
    if(a.size()!=b.size())return 0;
    for(int i=0;i<a.size();i++)
```

```
    if(a[i]!=b[i])return 0;
    return 1;
}
int n,ans;
int main(){
    scanf("%d",&n);ans=n;
    for(int i=1;i<=n;i++){
        int a;scanf("%d",&a);
        for(int j=1;j<=a;j++){
            int b;scanf("%d",&b);
            ai[i].push_back(b);
        }
    }
    sort(ai+1,ai+1+n);
    for(int i=2;i<=n;i++){
        if(ai[i]==ai[i-1])ans--;
    }
    printf("%d\n",ans);
}
```

3

对于一个序列中的任意一位，在另一个序列中二分查找找到与自己最接近的数字求差，最后统计最小值即可。（正解应该是双指针，但时间复杂度类似）

```
#include<bits/stdc++.h>
using namespace std;
vector<int>ai,bi;
int n,m,ans=2e9;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++){
        int a;scanf("%d",&a);
        ai.push_back(a);
    }
    for(int i=1;i<=m;i++){
        int a;scanf("%d",&a);
        bi.push_back(a);
    }
    ai.push_back(2e9);
    bi.push_back(2e9);
    sort(ai.begin(),ai.end());
```

```
    sort(bi.begin(),bi.end());
    for(int i:bi)if(i!=2e9)ans=min(ans,*(lower_bound(ai.begin(),ai.end
    for(int i:ai)if(i!=2e9)ans=min(ans,*(lower_bound(bi.begin(),bi.end
    printf("%d\n",ans);
}
```

4

对于每一个节点我们求它可以到达哪些节点我们可以在$O(m)$的时间复杂度下获取，而 m 的规模只有 2k ， 所以对于 n 个节点只需要求一次dfs即可，将最终答案累加就是我们要求的答案。

```
#include<bits/stdc++.h>
using namespace std;
vector<int>touch[3000];
int ans,st[3000];
int n,m;
void dfs(int a){
    if(st[a])return ;
    ans++;st[a]=1;
    for(int i:touch[a])
    dfs(i);
}
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        int a,b;scanf("%d%d",&a,&b);
        touch[a].push_back(b);
    }
    for(int i=1;i<=n;i++){
        memset(st,0,sizeof(st));
        dfs(i);
    }
    printf("%d\n",ans);
}
```

5

将字符串按照字典序排序，那么前缀与自己最相似的就是自己的上一位和下一位中的其中之一，所以只需要暴力与这两位进行比较即可。（正解应该是

字典树，时间复杂度明显优于此做法）

```
#include<bits/stdc++.h>
using namespace std;
struct node{string s;int i;} s[2000000];
bool operator<(node a,node b){return a.s<b.s;}
int n,ans[2000000];
int main(){
    cin>>n;
    for(int i=1;i<=n;i++)
    cin>>s[i].s,s[i].i=i;
    sort(s+1,s+1+n);
    for(int i=2;i<=n;i++){
        int res=0;
        for(int j=0;j<min(s[i-1].s.size(),s[i].s.size());j++)
        if(s[i-1].s[j]!=s[i].s[j])break;else res++;
        ans[s[i].i]=max(ans[s[i].i],res);
        ans[s[i-1].i]=max(ans[s[i-1].i],res);
    }
    for(int i=1;i<=n;i++)
    cout<<ans[i]<<endl;
}
```

6

拓扑排序带深度计算的模板题。

```
#include<bits/stdc++.h>
using namespace std;
int n,m,ans;
vector<int>touch[2000000];
int du[2000000],deep[2000000];
queue<int>q;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++){
        int a,b;scanf("%d%d",&a,&b);
        touch[a].push_back(b);
        du[b]++;
    }
    for(int i=1;i<=n;i++)
    if(du[i]==0)q.push(i);
```

```
    while(q.size()){
        int t=q.front();q.pop();
        for(int i:touch[t]){
            deep[i]=max(deep[i],deep[t]+1);
            du[i]--;if(du[i]==0)q.push(i);
            ans=max(ans,deep[i]);
        }
    }
    printf("%d\n",ans);
    ###  for(int i=1;i<=n;i++)
    ###  printf("(%d)\n",deep[i]);
}
```

7

在没有第一种操作时这个题目就是字符串哈希模板题，但是它还有单点修改操作，单点修改区间查询且处理的是前缀问题，只需要使用树状数组去帮忙维护即可。

```
#include<bits/stdc++.h>
using namespace std;
const int ni=2e6;
unsigned long long tr1[ni],tr2[ni],pwp[ni];
int n,m;string s;
const int pi=131;
void add1(int a,int b){
    for(int i=a;i<=n;i+=i&-i)
    tr1[i]+=b*pwp[i-a];
}
unsigned long long ask1(int a){
    unsigned long long res=0;
    for(int i=a;i;i-=i&-i)
    res+=tr1[i]*pwp[a-i];
    return res;
}
void add2(int a,int b){
    for(int i=a;i<=n;i+=i&-i)
    tr2[i]+=b*pwp[i-a];
}
unsigned long long ask2(int a){
    unsigned long long res=0;
    for(int i=a;i;i-=i&-i)
```

```
    res+=tr2[i]*pwp[a-i];
    return res;
}
int main(){
    cin>>n>>m>>s;s=' '+s;
    pwp[0]=1;
    for(int i=1;i<=n;i++)
    pwp[i]=pwp[i-1]*pi;
    for(int i=1;i<=n;i++)
    add1(i,s[i]),add2(n-i+1,s[i]);
    for(int h=1;h<=m;h++){
        int a;cin>>a;
        if(a==1){
            int x;char c;
            cin>>x>>c;
            add1(x,c-s[x]);
            add2(n-x+1,c-s[x]);
            s[x]=c;
        }else{
            int l,r;
            cin>>l>>r;
            unsigned a,b;
            a=ask1(r)-ask1(l-1)*pwp[r-l+1];
            b=ask2(n-l+1)-ask2(n-r)*pwp[r-l+1];
            printf(a==b?"Yes\n":"No\n");
        }
    }
}
```

8

一道很经典的dp题目，需要注意的是每个节点对后续节点的影响的处理。

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
int n,m,s;
const int ni=2e6;
struct wall{
    int a,b,f;
}w[ni];
int jc[ni],ny[ni];
bool operator<(wall a,wall b){
```

```
    if(a.a==b.a)return a.b<b.b;
    return a.a<b.a;
}
const int pi=1e9+7;
int ksm(int a,int b){
    int res=1;
    while(b){
        if(b&1)res=res*a%pi;
        a=a*a%pi;b>>=1;
    }
    return res;
}
int he(int a,int b){

### printf("<%d %d>",a,b);

    return jc[a+b]*ny[a]%pi*ny[b]%pi;

}
signed main(){
    cin>>n>>m>>s;
    w[++s]={n,m};w[0]={1,1,1};
    for(int i=1;i<=s;i++)
    cin>>w[i].a>>w[i].b;
    sort(w+1,w+1+s);
    jc[0]=1;ny[0]=1;
    for(int i=1;i<=n+m;i++)
    jc[i]=jc[i-1]*i%pi,
    ny[i]=ksm(jc[i],pi-2);
    for(int i=1;i<=s;i++){
        w[i].f=he(w[i].a-1,w[i].b-1);
        for(int j=1;j<i;j++)
        if(w[j].a<=w[i].a&&w[j].b<=w[i].b)
        w[i].f=(w[i].f-w[j].f*he(w[i].a-w[j].a,w[i].b-w[j].b)%pi+pi)%p
       ###  printf("(%d)",w[i].f);
    }
    cout<<w[s].f<<endl;
}
```