```
/*Day 85 coding Statement :

You are given an undirected graph with N nodes
(numbered 1 through N) and M edges. Each edge connects two distinct nodes.
However, there may be multiple edges connecting the same pairs of nodes, and they are
considered to be distinct edges.
A lowercase English letter is written in each node.
You are also given a string S with length L.
A beautiful path is a sequence of L−1 edges such that there is a sequence of L nodes with
the following properties:

for each valid i, the i-th edge connects the i-th and (i+1)-th of these nodes
for each valid i, the i-th character of S is written in the i-th of these nodes
There are no other restrictions — a path may visit nodes or edges any number of times in
any order.

Determine the number of beautiful paths in the graph. Since the answer can be very large,
compute it modulo (10^9)+7.*/
import java.util.*;
import java.lang.*;
import java.io.*;
class Main
{ static ArrayList<Integer> tree[];
 static int f[][];
 public static void main(String[] args) {
 Scanner input=new Scanner(System.in);
 int t=input.nextInt();
 while (t-->0){
 int n=input.nextInt();
 int m=input.nextInt();
 int l=input.nextInt();
 String s=input.next();
 char a[]=input.next().toCharArray();
 tree=new ArrayList[n+1];
 for (int i = 0; i <=n ; i++) {
 tree[i]=new ArrayList<>();
 }
 int x[]=new int[m];
 for (int i = 0; i <m ; i++) {
 x[i]=input.nextInt();
 }
 f=new int[n+1][n+1];
 for (int i = 0; i <m ; i++) {
 int y=input.nextInt();

 tree[x[i]].add(y);
 tree[y].add(x[i]);
 f[x[i]][y]++;
f[y][x[i]]++;
}
long res=0;
dp=new Long[n+2][22];
for (int i = 1; i <=n ; i++) {
res+= dfs(i,0,s,l,a);
res%=mod;
}
boolean allsame=true;
for (int i = 1; i <l ; i++) {
if (s.charAt(i)!=s.charAt(i-1)) allsame=false;
}
if (allsame){
long temp=0;
dp2=new Long[n+1][n+1];
boolean v[][]=new boolean[n+1][n+1];
for (int i = 1; i <=n ; i++) {
for (int c:tree[i]) {
if (v[i][c]) continue;
if (a[i-1]==a[c-1]) {
v[i][c]=true;
v[c][i]=true;
temp+=power(f[i][c],l-1,mod);
temp%=mod;

}
}
}
System.out.println((res-temp+mod)%mod);
}else {
System.out.println(res);
}
}
}
static Long dp[][];
static Long dp2[][];
static long mod= (long) (1e9+7);
private static long dfs2(int i, int j, String s, int l, char a[],int k) {
if (j==l-1 ){
if (s.charAt(j)!=a[i-1]) return 0;
return 1;
}
if (s.charAt(j)!=a[i-1]) return 0;
if (dp2[i][j]!=null) return dp2[i][j];
long ans=0;
for (int c:tree[i]) {
if (c!=k) continue;
ans+=dfs2(k, j+1, s, l, a,i)%mod;
```

```
 ans%=mod;
 }
 return dp2[i][j]=ans%mod;
 }
 static long power(long x,
 long y, long p)
 {
 long res = 1;
 x = x % p;
 while (y > 0)
 {
 if ((y & 1) > 0)
 res = (res * x) % p;
 y = y >> 1;
 x = (x * x) % p;
 }
 return res;
 }
 private static long dfs(int i, int j, String s, int l, char[] a) {
 if (j==l-1 ){
 if (s.charAt(j)!=a[i-1]) return 0;

 return 1;
 }
 if (s.charAt(j)!=a[i-1]) return 0;
 if (dp[i][j]!=null) return dp[i][j];
 long ans=0;
 for (int c:tree[i]) {
 ans+=((dfs(c, j+1, s, l, a)))%mod;
 ans%=mod;
 }
 return dp[i][j]=ans%mod;
 }
}
```