python:

```
import sys
from collections import defaultdict
def bfs(g,n,s):
    dist=[0]*n
    distance=0
    if(n>=1):
      Q=[s]
      seen=set([s])
      while Q!=[]:
        k=Q.pop(0)
        i=0
        while i<n:
         if(g[k][i]==1):
          if(i not in seen):
           Q=Q+[i]
           dist[i]=dist[k]+1
           seen.add(i)
         i+=1
    return dist
T=input()
while T:
    T-=1
    n=input()
    graph=[]
    for k in range(n):
        graph+=[[int(x) for x in list(raw_input())]]
    q=input()
```

```
    for k in range(q):
        v,m=[int(x) for x in raw_input().split()]
        for l in range(m):
            d,t=[int(x) for x in raw_input().split()]
            graph[d-1][t-1]=(graph[d-1][t-1]+1)%2
        lis=bfs(graph,n,v-1)
        print sum(lis)
```

---

c:

```
#include<stdio.h>
#include<math.h>



int main(){
    int t;
    scanf("%d",&t);
    int n;
    char a[1000];
    int tempN;
    int i,j,k;
    int q;
    int v,m;
    int A,B;
    int result;
    while(t--){

        scanf("%d",&n);
        int matrix[n][n];
```

```
int distanceMatrix[n][n];
tempN=0;

while(tempN<n){
   scanf("%s",a);
   for(i=0;i<n;i++){
      if(a[i]=='0')
         matrix[tempN][i]=-1;
      else
         matrix[tempN][i]=1;
   }
       tempN++;
}
scanf("%d",&q);
while(q--){
   result=0;
   scanf("%d%d",&v,&m);
   v=v-1;
   while(m--){
      scanf("%d%d",&A,&B);
      A=A-1;
      B=B-1;
      if(matrix[A][B]==1){
         matrix[A][B]=-1;
      }
      else{
         matrix[A][B]=1;
      }
   }
```

```
        for(i=0;i<n;i++){
            for(j=0;j<n;j++){
                distanceMatrix[i][j]=matrix[i][j];
            }
        }
        for(k=0;k<n;k++){
            for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                    if(i==j){
                        continue;
                    }
                    if( distanceMatrix[i][k]!=-1 && distanceMatrix[k][j]!=-1 && (
distanceMatrix[i][k]+distanceMatrix[k][j] < distanceMatrix[i][j] || distanceMatrix[i][j]==-1)  ){
                        distanceMatrix[i][j]=distanceMatrix[i][k]+distanceMatrix[k][j];
                    }
                }
            }
        }
        for(i=0;i<n;i++){
            if(i==v)
                continue;
            if(distanceMatrix[v][i]!=-1)
                result+=distanceMatrix[v][i];
        }
        printf("%d\n",result);
      }
   }
   return 0;
```

}