实质就是判断有几个联通子集，400ms的时限。可以用BFS或者DFS或者并查集，效率当然是并查集最高。

我用BFS暴力320ms过了，练习了C++ STL的QUEUE，代码如下：

**[cpp]** [view plain](http://blog.csdn.net/sup_heaven/article/details/8451696) [copy](http://blog.csdn.net/sup_heaven/article/details/8451696)

[在CODE上查看代码片](https://code.csdn.net/snippets/177199)

1. #include<iostream>
2. #include<queue>
3. **using** **namespace** std;
4. **int** rel[1005][1005];
5. **int** main(){
6. **int** i,j,k,N,M,K;
7. **int** from,to,occ;
8. **int** vst[1005];
9. cin>>N>>M>>K;
10. **for**(i=1;i<=N;i++){
11. **for**(j=1;j<=N;j++){
12. rel[i][j]=0;
13. }
14. }
15. **for**(i=1;i<=M;i++){
16. cin>>from>>to;
17. rel[from][to]=rel[to][from]=1;
18. }
19. **for**(i=1;i<=K;i++){
20. cin>>occ;
21. **for**(j=1;j<=N;j++){
22. vst[j]=0;
23. }
24. vst[occ]=1;
25. **int** cnt=0;
26. **for**(j=1;j<=N;j++){
27. **if**(!vst[j]){  //若没有被标记；
28. vst[j]=1;  //则标记
29. cnt++;   //计数器++
30. queue<**int**>q;  //创建队列q
31. q.push(j);  //j进队列
32. **while**(!q.empty()){  //当queue不空时循环
33. **int** tmp=q.front();
34. q.pop();    创建一个队列头的拷贝
35. **for**(k=1;k<=N;k++){
36. **if**(!vst[k]&&rel[tmp][k]){
37. vst[k]=1;
38. q.push(k);
39. }
40. }
41. }
42. }
44. }
45. cout<<cnt-1<<endl;
46. }
47. **return** 0;
48. }  //总结思路：事实上是先在图上去掉点，再求联通分支的数量，然后因为链接任意两个联通分支只需要一条边，所以需要的边数是联通分支-1。

DFS，150ms暴力通过

**[cpp]** [view plain](http://blog.csdn.net/sup_heaven/article/details/8451696) [copy](http://blog.csdn.net/sup_heaven/article/details/8451696)

[在CODE上查看代码片](https://code.csdn.net/snippets/177199)

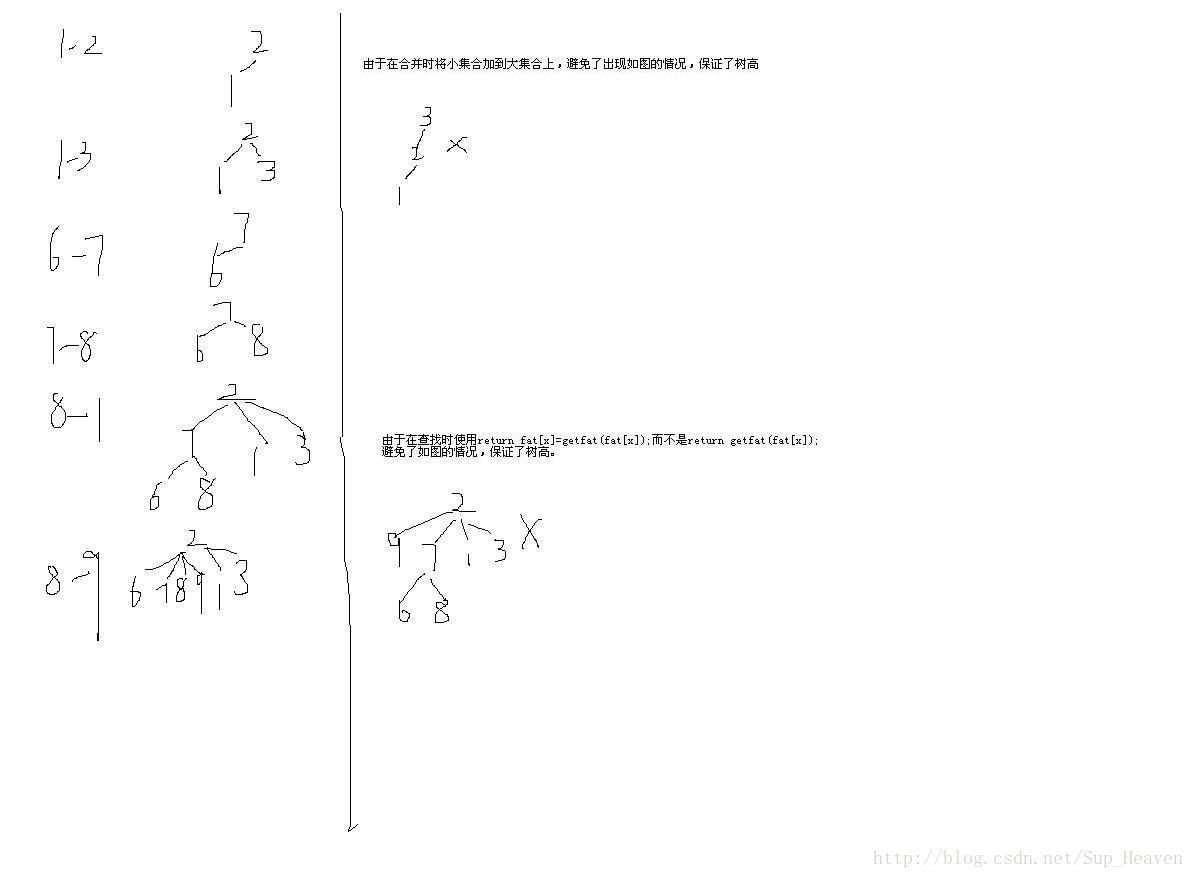
1. #include<stdio.h>
2. #define max\_city 1005
3. **int** map[max\_city][max\_city],mark[max\_city];
4. **int** n,m,k,repair\_city,destroy\_city;
5. **void** dfs(**int** cur){
6. **int** i;
7. **for**(i=1;i<=n;i++){
8. **if**(map[cur][i]&&cur!=destroy\_city&i!=destroy\_city&&!mark[i]){
9. mark[i]=1;
10. dfs(i);
11. }
12. }
13. }
14. **int** main(){
15. **int** i,j;
16. **int** from,to,city;
17. scanf("%d %d %d",&n,&m,&k);
18. **for**(i=1;i<=n;i++){
19. **for**(j=1;j<=n;j++){
20. map[i][j]=0;
21. }
22. }
23. **for**(i=0;i<m;i++){
24. scanf("%d %d",&from,&to);
25. map[from][to]=map[to][from]=1;
26. }
27. **while**(k){
28. scanf("%d",&destroy\_city);
29. repair\_city=0;
30. **for**(i=1;i<=n;i++){
31. mark[i]=0;
32. }
33. **for**(i=1;i<=n;i++){
34. **if**(!mark[i]&&i!=destroy\_city){
35. repair\_city++;
36. }
37. dfs(i);
38. }
39. printf("%d\n",repair\_city-1);
40. k--;
41. }
42. }

而用并查集100ms通过，代码如下：

**[cpp]** [view plain](http://blog.csdn.net/sup_heaven/article/details/8451696) [copy](http://blog.csdn.net/sup_heaven/article/details/8451696)

[在CODE上查看代码片](https://code.csdn.net/snippets/177199)

1. #include<stdio.h>
2. #define MAXN 1000
3. #define MAXE MAXN\*(MAXN+1)/2
4. **struct** Node{
5. **int** from;
6. **int** to;
7. }node[MAXE];
8. **int** fat[MAXN],siz[MAXN];
9. **int** getfat(**int** x){
10. **if**(fat[x]==x) **return** x;
11. **else** **return** fat[x]=getfat(fat[x]);
12. }
13. **void** Union(**int** a,**int** b){
14. **int** fa=getfat(a);
15. **int** fb=getfat(b);
16. **if**(fa!=fb){
17. **if**(siz[fa]<=siz[fb]){
18. fat[fa]=fb;
19. siz[fb]+=siz[fa];
20. }
21. **else**{
22. fat[fb]=fa;
23. siz[fa]+=siz[fb];
24. }
25. }
26. }
27. **int** main(){
28. **int** i,j,cut;
29. **int** n,m,k;
30. scanf("%d %d %d",&n,&m,&k);
31. **for**(i=0;i<m;i++){
32. scanf("%d %d",&node[i].from,&node[i].to);
33. }
34. **for**(i=0;i<k;i++){
35. **for**(j=1;j<=n;j++){
36. fat[j]=j;
37. siz[j]=1;
38. }
39. scanf("%d",&cut);
40. **for**(j=0;j<m;j++){
41. **if**(node[j].from==cut||node[j].to==cut) **continue**;
42. **if**(fat[node[j].from]==fat[node[j].to]) **continue**;
43. Union(node[j].from,node[j].to);
44. }
45. **int** cnt=0;
46. **for**(j=1;j<=n;j++){
47. **if**(j!=cut&&fat[j]==j){
48. cnt++;
49. }
50. }
51. printf("%d\n",cnt-1);
52. }
53. **return** 0;
54. }



#include<cmath>

#include<cstdio>

#include<iostream>

#include<string>

#include<algorithm>

#include<queue>

using namespace std;

int map[1005][1005], mark[1005],cnt=0;

int N, M, K, target;

void cipher(int tar);

int main() {

cin >> N >> M >> K;

for (int i = 0; i < M; i++)

{

int from, to;

cin >>from>>to;

map[from][to] = map[to][from] = 1;

}

for (int i = 0; i < K; i++)

{

cin >> target;

cipher(target);

cout << cnt - 1<<endl;

}

}

void cipher(int tar)

{

queue<int>a;

cnt = 0;

for (int i = 1; i <= N; i++)

mark[i] = 0;

mark[tar] = 1;

for (int i = 1; i <= N; i++)

{

if (mark[i] == 0)

{

mark[i] = 1;

a.push(i);

cnt++;

while (!a.empty()) {

int temp;

temp = a.front();

a.pop();

for (int j = 1; j <= N;j++)

if (mark[j] == 0 && map[temp][j])

{

mark[j] = 1;

a.push(j);

}

}

}

}

}