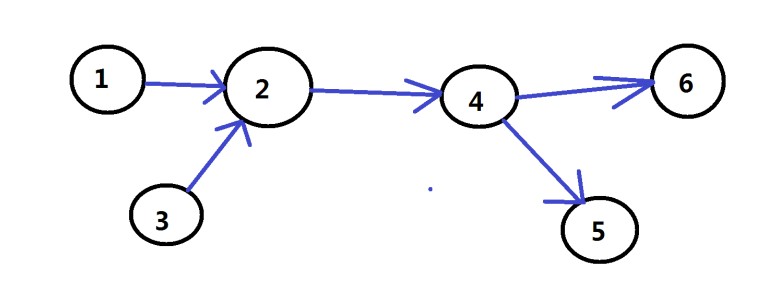
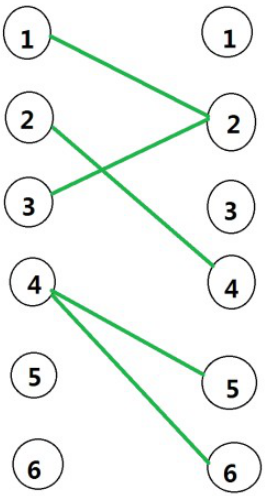
# **[有向图最小路径覆盖](https://www.cnblogs.com/jianglangcaijin/p/5989907.html)**

（1）什么是有向图G的最小路径覆盖？首先，图G必须是有向无环的。路径覆盖就是在图G中找出一些路径，每条路径从起点走到终点并且标记中间经过的点。最后，每个点只被标记一次。选出的这些路径组成路径覆盖。如果找出最少的路径成为一个路径覆盖，则称为最小路径覆盖。



在上图中,以下两个均是路径覆盖：

a、<1,2> <4,6> <3> <5>

b、<1,2,4,6> <3> <5>

但是<1,2,4,6> <3,2,4,6>则不是一个路径覆盖，因为2,4被标记两次。在上面两个覆盖中b为最小覆盖，|b|=3,而|a|=4。（注意一个单独的节点也是一个路径）

（2）由原图G构造对应的二分图S，将原图G中的每个点i拆成两个点i1和i2，i1和i2属于S。i1组成二分图的X集合，i2组成Y集合。若原图G中有边<i,j>，则在S中有边<i1,j2>，则上面的图可以得到如下二分图：

（3）定理证明：如果匹配数为零，那么P中不存在有向边，于是显然有最小路径覆盖=|G|－最大匹配数=|G|。在匹配数为0的基础上，即S中一条边没有，如果在S中增加一条匹配边(i1,j2)，那么在图G的路径覆盖中就存在一条由i连接j的边，也就是说i与j 在一条路径上，于是路径覆盖数就可以减少一个；如此继续增加匹配边，每增加一条，路径覆盖数就减少一条；直到匹配边不能继续增加时，路径覆盖数也不能再减少了；但是这里只 是说明了每条匹配边对应于路径覆盖中的一条路径上的一条连接两个点之间的有向边；下面来说明一个路径覆盖中的每条连接两个顶点之间的有向边对应于一条匹配 边。与前面类似，对于路径覆盖中的每条连接两个顶点之间的每条有向边<i,j>，我们可以在匹配图中对应作一条连接i1与j2的边， 显然这样做出来二分图中的边均为匹配，即连接的X和Y中的两个点之前都未匹配过，否则匹配过的那个点对应原图时就被覆盖两次，与路径覆盖矛盾。这就说明匹配边与路径覆盖图中连接两顶点之间边的一一对应关系，那么也就说明了前面的公式成立，即S中每增加一条匹配边，路径覆盖数减少1.

## **BZOJ 2150 部落战争 最小路径覆盖 二分图最大匹配**

题目大意：给出一张地图，一个军队要征战整个土地。一块土地只能经过一次，有X的地方不能走，军队只会走R\*C个格子，只会向下走，问最少需要多少军队能够征战所有的土地。

思路：这个是前几天考试的题，今天居然发现时BZ的原题，还好当时A掉了。。。

看到每个土地只能经过一次就想到了网络流什么的，再一想想好像是最小路径覆盖啊，然后拆点，建图，Hungary，二分图最小路径覆盖=点数-最大匹配，没了。。

CODE：

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

#define MAX 20010

using namespace std;

int m,n,r,c;

char src[200][200];

int num[200][200],cnt;

int head[MAX],total;

int \_next[MAX],aim[MAX];

bool v[MAX];

int paired[MAX];

inline void Add(int x,int y)

{

\_next[++total] = head[x];

aim[total] = y;

head[x] = total;

}

bool Hungary(int x)

{

for(int i = head[x]; i; i = \_next[i]) {

if(!v[aim[i]]) {

v[aim[i]] = true;

if(!paired[aim[i]] || Hungary(paired[aim[i]])) {

paired[aim[i]] = x;

return true;

}

}

}

return false;

}

int main()

{

cin >> m >> n >> r >> c;

for(int i = 1; i <= m; ++i) {

scanf("%s",src[i] + 1);

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

if(src[i][j] == '.')

num[i][j] = ++cnt;

}

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

for(int j = 1; j <= n; ++j) {

if(!num[i][j]) continue;

if(i + r > 0 && j + c > 0 && num[i + r][j + c])

Add(num[i][j],num[i + r][j + c] + cnt);

if(i + c > 0 && j + r > 0 && num[i + c][j + r])

Add(num[i][j],num[i + c][j + r] + cnt);

if(i + r > 0 && j - c > 0 && num[i + r][j - c])

Add(num[i][j],num[i + r][j - c] + cnt);

if(i + c > 0 && j - r > 0 && num[i + c][j - r])

Add(num[i][j],num[i + c][j - r] + cnt);

}

int ans = 0;

for(int i = 1; i <= cnt; ++i) {

memset(v,false,sizeof(v));

ans += Hungary(i);

}

cout << cnt - ans << endl;

return 0;

}