Pipe Network  
Our company has set up a pipe network for fuel distribution in factory. In that network, a fuel tank will be place at a location in the factory. From there, fuel will be distributed to other place by the pipe network.  
There are total 7 kinds of pipe, marked from 1-7 as below figures:  
  
Only the connected pipes can allow fuel pass through them. Two pipes are connected if they have a common end-point. For example :  
- In this case, fuel can flow from A to B  
  
- But in this case, fuel can not flow from A to B (Pipe in A does not have any common end-point with pipe in B)  
  
However, the pump used to distribute fuel has limit in its performance. Depend on its kind, the pump can only pump fuel as far as its limit. For example, let say the limit of pump is 3, that mean from start location, fuel can only flow to the farest place that 3 steps away (include start point).  
  
Given the map of pipe network, the location of the fuel tank (start position), the limit of pump, write a program to calculate the total number of pipes that fuel can flow to.  
  
Example 1: In this case, the height of the map (N) is 5, the length of of the map (M) is 6, location of the fuel tank is (2,1), the limit of the pump is 3  
- First fuel is pumped to location (2,1)  
- Second, fuel flows to location (2,0) and (2,2)  
- Third, fuel flows to location (1,2) and (2,3)  
As the limit of pump is reached, the fuel can not flow any further. Hence, the total number of pipes that fuel can flow to is 5.  
  
  
  
Example 2: In this case, the height of the map (N) is 5, the length of of the map (M) is 6, location of the fuel tank is (2,2), the limit of the pump is 6. The answer is 15.  
  
  
[Constraints]  
- The fuel tank is alway placed at a location where a pipe exist  
  
[Input]  
- The number of test case T (T <= 50)  
- In each TC, the first row will give :  
+ The size of the map N x M (5 <= N, M <= 50)  
+ The location of fuel tank (start position) (0-based index)  
+ The limit of the pump P (1 <= P <= 20)  
- Detail of the map will be given at next N rows, the value C of each cell represent the pipes (total 7 kind of pipes, 0 <= C <= 7), value 0 mean there is no pipe at that location.  
5 => Total 5 test cases  
5 6 2 1 3 => The size of the map is 5 x 6, starting point is (2,1), the limit of pump is 3  
0 0 5 3 6 0 => The first row of the map  
0 0 2 0 2 0 => The second row of the map  
3 3 1 3 7 0  
0 0 0 0 0 0  
0 0 0 0 0 0  
5 6 2 2 6  
3 0 0 0 0 3  
2 0 0 0 0 6  
1 3 1 1 3 1  
2 0 2 0 0 2  
0 0 4 3 1 1  
10 10 4 3 9  
0 0 0 0 0 0 0 0 0 0  
0 0 0 7 5 0 5 0 0 0  
0 0 3 2 2 6 0 0 0 0  
0 4 7 2 2 2 7 0 0 4  
0 3 0 1 1 2 2 0 0 5  
0 5 6 1 1 1 1 6 2 5  
7 4 1 2 0 0 4 6 0 0  
5 3 1 7 0 2 2 6 5 7  
7 3 2 1 1 7 1 0 2 7  
3 4 0 0 4 0 5 1 0 1  
...  
  
[Output]  
- The total number of pipes that fuel can flow to  
Case #1  
5  
Case #2  
15  
Case #3  
29

[#include](tg://search_hashtag?hashtag=include) <iostream>  
[#define](tg://search_hashtag?hashtag=define) SIZE 50  
  
using namespace std;  
int Arr[SIZE][SIZE];  
int Visited[SIZE][SIZE];  
int M, N, P;  
int Start[2]; //   
int dX[4] = {0, -1, 0, 1}; // 1  
int dY[4] = {-1, 0, 1, 0}; // 0 2  
// 3  
int Q[1000000][3];  
int rear, front;  
  
int main()  
{  
int tc,TestCase, output = 0;  
  
freopen("Text.txt", "r", stdin);  
freopen("Text1.txt", "w", stdout);  
cin>>TestCase;  
  
for(tc=0; tc<TestCase; tc++)  
{  
output = 0;  
cin >> N >> M >> Start[0] >> Start[1] >> P;  
for(int i = 0; i < N; i++){  
for(int j = 0; j < M; j++){  
cin >> Arr[i][j];  
Visited[i][j] = 0;  
}  
}   
  
rear = 0;   
front = 0;  
  
Q[0][0] = Start[0];  
Q[0][1] = Start[1];  
Q[0][2] = 1;  
Visited[Start[0]][Start[1]] = 1;  
output++;  
rear++;  
while(rear > front){  
int a,b,c;  
a = Q[front][0];  
b = Q[front][1];  
c = Q[front][2];  
front++;  
if(c >= P)  
break;  
  
  
for(int k = 0; k < 4; k++){  
int x = a + dX[k];  
int y = b + dY[k];  
if(x >= 0 && x < N && y >= 0 && y < M){  
if(k == 0 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 3 || Arr[a][b] == 6 || Arr[a][b] == 7) && (Arr[x][y] == 1 || Arr[x][y] == 3 || Arr[x][y] == 4 || Arr[x][y] == 5)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 1 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 2 || Arr[a][b] == 4 || Arr[a][b] == 7) && (Arr[x][y] == 1|| Arr[x][y] == 2 || Arr[x][y] == 5 || Arr[x][y] == 6)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 2 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 3 || Arr[a][b] == 4 || Arr[a][b] == 5) && (Arr[x][y] == 1|| Arr[x][y] == 3 || Arr[x][y] == 6 || Arr[x][y] == 7)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 3 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 2 || Arr[a][b] == 5 || Arr[a][b] == 6) && (Arr[x][y] == 1|| Arr[x][y] == 2 || Arr[x][y] == 4 || Arr[x][y] == 7)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
Visited[x][y] = 1;  
rear++;  
}  
}  
}  
}   
////////////////////////////////  
  
  
  
////////////////////////////////  
  
  
  
cout<< "Case #" << tc+1 <<"\n"<< rear << endl;  
}  
  
return 0;  
}

[#include](tg://search_hashtag?hashtag=include) <iostream>  
[#define](tg://search_hashtag?hashtag=define) SIZE 50  
  
using namespace std;  
int Arr[SIZE][SIZE];  
int Visited[SIZE][SIZE];  
int M, N, P;  
int Start[2]; //   
int dX[4] = {0, -1, 0, 1}; // 1  
int dY[4] = {-1, 0, 1, 0}; // 0 2  
// 3  
int Q[1000000][3];  
int rear, front;  
  
int main()  
{  
int tc,TestCase, output = 0;  
  
freopen("Text.txt", "r", stdin);  
freopen("Text1.txt", "w", stdout);  
cin>>TestCase;  
  
for(tc=0; tc<TestCase; tc++)  
{  
output = 0;  
cin >> N >> M >> Start[0] >> Start[1] >> P;  
for(int i = 0; i < N; i++){  
for(int j = 0; j < M; j++){  
cin >> Arr[i][j];  
Visited[i][j] = 0;  
}  
}   
  
rear = 0;   
front = 0;  
  
Q[0][0] = Start[0];  
Q[0][1] = Start[1];  
Q[0][2] = 1;  
Visited[Start[0]][Start[1]] = 1;  
output++;  
rear++;  
while(rear > front){  
int a,b,c;  
a = Q[front][0];  
b = Q[front][1];  
c = Q[front][2];  
front++;  
if(c >= P)  
break;  
  
  
for(int k = 0; k < 4; k++){  
int x = a + dX[k];  
int y = b + dY[k];  
if(x >= 0 && x < N && y >= 0 && y < M){  
if(k == 0 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 3 || Arr[a][b] == 6 || Arr[a][b] == 7) && (Arr[x][y] == 1 || Arr[x][y] == 3 || Arr[x][y] == 4 || Arr[x][y] == 5)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 1 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 2 || Arr[a][b] == 4 || Arr[a][b] == 7) && (Arr[x][y] == 1|| Arr[x][y] == 2 || Arr[x][y] == 5 || Arr[x][y] == 6)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 2 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 3 || Arr[a][b] == 4 || Arr[a][b] == 5) && (Arr[x][y] == 1|| Arr[x][y] == 3 || Arr[x][y] == 6 || Arr[x][y] == 7)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
rear++;  
Visited[x][y] = 1;  
}  
else if(k == 3 && Visited[x][y] == 0 && (Arr[a][b] == 1 || Arr[a][b] == 2 || Arr[a][b] == 5 || Arr[a][b] == 6) && (Arr[x][y] == 1|| Arr[x][y] == 2 || Arr[x][y] == 4 || Arr[x][y] == 7)){  
Q[rear][0] = x;  
Q[rear][1] = y;  
Q[rear][2] = c + 1;  
Visited[x][y] = 1;  
rear++;  
}  
}  
}  
}   
////////////////////////////////  
  
  
  
////////////////////////////////  
  
  
  
cout<< "Case #" << tc+1 <<"\n"<< rear << endl;  
}  
  
return 0;  
}