D. Olya and Energy Drinks

time limit per test 2 seconds
memory limit per test 256 megabytes
input standard input
output standard output

Olya loves energy drinks. She loves them so much that her room is full of empty cans from energy drinks.

Formally, her room can be represented as a field of $n \times m$ cells, each cell of which is empty or littered with cans.

Olya drank a lot of energy drink, so now she can run k meters per second. Each second she chooses one of the four directions (up, down, left or right) and runs from 1 to k meters in this direction. Of course, she can only run through empty cells.

Now Olya needs to get from cell (x_1, y_1) to cell (x_2, y_2). How many seconds will it take her if she moves optimally?

It's guaranteed that cells (x_1, y_1) and (x_2, y_2) are empty. These cells can coincide.

Input

The first line contains three integers n, m and k $(1 \le n, m, k \le 1000)$ — the sizes of the room and Olya's speed.

Then n lines follow containing m characters each, the i-th of them contains on j-th position "`#`", if the cell (i, j) is littered with cans, and "`.`" otherwise.

The last line contains four integers x_1, y_1, x_2, y_2 $(1 \le x_1, x_2 \le n,\ 1 \le y_1, y_2 \le m)$ — the coordinates of the first and the last cells.

Output

Print a single integer — the minimum time it will take Olya to get from (x_1, y_1) to (x_2, y_2).

If it's impossible to get from (x_1, y_1) to (x_2, y_2), print `-1`.

Examples

input Copy

```
3 4 4
....
###.
....
1 1 3 1
```

output Copy

```
3
```

input Copy

```
3 4 1
....
###.
....
1 1 3 1
```

output Copy

```
8
```

input Copy

```
2 2 1
.#
#.
1 1 2 2
```

output Copy

```
-1
```

Note

In the first sample Olya should run 3 meters to the right in the first second, 2 meters down in the second second and 3 meters to the left in the third second.

In second sample Olya should run to the right for 3 seconds, then down for 2 seconds and then to the left for 3 seconds.

Olya does not recommend drinking energy drinks and generally believes that this is bad.

```cpp
#include<bits/stdc++.h>
using namespace std;
#define IO ios_base::sync_with_stdio(false); cin.tie(NULL)
string st[1005];
int n,m,k,sx,sy,dx,dy;
int fx[]= {-1,+0,+0,+1};
int fy[]= {+0,-1,+1,+0};
int vis[1005][1005],cost[1005][1005];
struct data{ int x,y,v; };
queue<data>qq;
int valid(int tx,int ty){
    if(tx>=0&&tx<n&&ty>=0&&ty<m&&st[tx][ty]=='.') return 1;
    return 0;
}
int BFS(){
    data nd; nd.x=sx; nd.y=sy; nd.v=0;
    vis[sx][sy]=1;
    cost[sx][sy]=0;
    qq.push(nd);

    while(!qq.empty()) {
        data u = qq.front(); qq.pop();
        int x,y,v; x = u.x; y = u.y; v = u.v;

        if(x==dx && y == dy) return v;

        for(int j=0; j<4; j++){
            for(int i=1; i<=k; i++) {
                int tx,ty;
                tx = (i*fx[j]) + x;
                ty = (i*fy[j]) + y;

                if(valid(tx,ty)==1) {
                    if(vis[tx][ty]==0) {
                        data tm; tm.x = tx; tm.y = ty; tm.v = v+1;
                        cost[tx][ty]=v+1;
                        vis[tx][ty]=1;
                        qq.push(tm);
                    }
                    else {
                        if(cost[tx][ty]<cost[x][y]+1)break;
                    }
                }
                else{
                    break;
                }
            }
        }
    }
    return -1;
}
int main(){
    IO;
    cin>>n>>m>>k;
    for(int i=0; i<n; i++) cin>>st[i];
    cin>>sx>>sy>>dx>>dy;
    sx--; sy--; dx--; dy--;
    int ans = BFS();
    printf("%d\n",ans);
    return 0;
}
```