

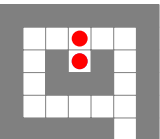
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



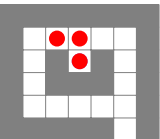
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) ⇒ mark[(3, 3)]=EXPL
call sp.r((4, 3)) ⇒ mark[(4, 3)]=EXPL
call sp.r((4, 2)) ⇒ mark[(4, 2)]=EXPL
call sp.r((4, 1)) ⇒ mark[(4, 1)]=EXPL
call sp.r((4, 2)) ⇒ mark[(4, 2)]=EXPL ⇒ return -1
call sp.r((3, 1)) ⇒ mark[(3, 1)]=EXPL
call sp.r((2, 1)) ⇒ mark[(2, 1)]=EXPL
call sp.r((1, 1)) ⇒ mark[(1, 1)]=EXPL
call sp.r((1, 2)) ⇒ mark[(1, 2)]=EXPL
call sp.r((1, 1)) ⇒ mark[(1, 1)]=EXPL ⇒ return -1
call sp.r((1, 3)) ⇒ mark[(1, 3)]=EXPL
call sp.r((1, 2)) ⇒ mark[(1, 2)]=EXPL ⇒ return -1
call sp.r((1, 4)) ⇒ mark[(1, 4)]=EXPL
call sp.r((1, 3)) ⇒ mark[(1, 3)]=EXPL ⇒ return -1
call sp.r((1, 5)) ⇒ mark[(1, 5)]=EXPL
call sp.r((1, 4)) ⇒ mark[(1, 4)]=EXPL ⇒ return -1
call sp.r((0, 5)) ⇒ mark[(0, 5)]=EXPL
call sp.r((-1, 5)) ⇒ mark[(-1, 5)]=EXIT ⇒ return 0
mark[(0, 5)]=(-1, 5) ⇒ return 1
mark[(1, 5)]=(-1, 5) ⇒ return 2
mark[(1, 4)]=(-1, 5) ⇒ return 3
mark[(1, 3)]=(-1, 4) ⇒ return 4
mark[(1, 2)]=(-1, 3) ⇒ return 5
mark[(1, 1)]=(-1, 2) ⇒ return 6
mark[(2, 1)]=(-1, 1) ⇒ return 7
mark[(3, 1)]=(-2, 1) ⇒ return 8
mark[(4, 1)]=(-3, 1) ⇒ return 9
mark[(4, 2)]=(-4, 1) ⇒ return 10
mark[(4, 3)]=(-4, 2) ⇒ return 11
mark[(3, 3)]=(-4, 3) ⇒ return 12
```



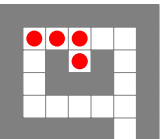
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



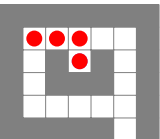
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)=(0, 5) => return 2
mark[(1, 4)=(1, 5) => return 3
mark[(1, 3)=(1, 4) => return 4
mark[(1, 2)=(1, 3) => return 5
mark[(1, 1)=(1, 2) => return 6
mark[(2, 1)=(1, 1) => return 7
mark[(3, 1)=(2, 1) => return 8
mark[(4, 1)=(3, 1) => return 9
mark[(4, 2)=(4, 1) => return 10
mark[(4, 3)=(4, 2) => return 11
mark[(3, 3)=(4, 3) => return 12
```



dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

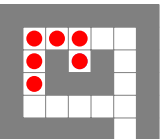
```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)=(0, 5) => return 2
mark[(1, 4)=(1, 5) => return 3
mark[(1, 3)=(1, 4) => return 4
mark[(1, 2)=(1, 3) => return 5
mark[(1, 1)=(1, 2) => return 6
mark[(2, 1)=(1, 1) => return 7
mark[(3, 1)=(2, 1) => return 8
mark[(4, 1)=(3, 1) => return 9
mark[(4, 2)=(4, 1) => return 10
mark[(4, 3)=(4, 2) => return 11
mark[(3, 3)=(4, 3) => return 12
```



```

call sp_r((3, 3)) ⇒ mark[(3, 3)]=EXPL
call sp_r((4, 3)) ⇒ mark[(4, 3)]=EXPL
call sp_r((4, 2)) ⇒ mark[(4, 2)]=EXPL
call sp_r((4, 1)) ⇒ mark[(4, 1)]=EXPL
call sp_r((4, 2)) ⇒ mark[(4, 2)]=EXPL ⇒ return -1
call sp_r((3, 1)) ⇒ mark[(3, 1)]=EXPL
call sp_r((2, 1)) ⇒ mark[(2, 1)]=EXPL
call sp_r((1, 1)) ⇒ mark[(1, 1)]=EXPL
call sp_r((1, 2)) ⇒ mark[(1, 2)]=EXPL
call sp_r((1, 1)) ⇒ mark[(1, 1)]=EXPL ⇒ return -1
call sp_r((1, 3)) ⇒ mark[(1, 3)]=EXPL
call sp_r((1, 2)) ⇒ mark[(1, 2)]=EXPL ⇒ return -1
call sp_r((1, 4)) ⇒ mark[(1, 4)]=EXPL
call sp_r((1, 3)) ⇒ mark[(1, 3)]=EXPL ⇒ return -1
call sp_r((1, 5)) ⇒ mark[(1, 5)]=EXPL
call sp_r((1, 4)) ⇒ mark[(1, 4)]=EXPL ⇒ return -1
call sp_r((0, 5)) ⇒ mark[(0, 5)]=EXPL
call sp_r((-1, 5)) ⇒ mark[(-1, 5)]=EXIT ⇒ return 0
mark[(0, 5)]=(-1, 5) ⇒ return 1
mark[(1, 5)]=(-1, 5) ⇒ return 2
mark[(1, 4)]=(-1, 5) ⇒ return 3
mark[(1, 3)]=(-1, 4) ⇒ return 4
mark[(1, 2)]=(-1, 3) ⇒ return 5
mark[(1, 1)]=(-1, 2) ⇒ return 6
mark[(2, 1)]=(-1, 1) ⇒ return 7
mark[(3, 1)]=(-1, 1) ⇒ return 8
mark[(4, 1)]=(-1, 1) ⇒ return 9
mark[(4, 2)]=(-1, 1) ⇒ return 10
mark[(4, 3)]=(-1, 2) ⇒ return 11
mark[(3, 3)]=(-1, 3) ⇒ return 12

```



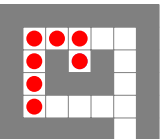
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



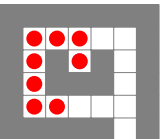
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```

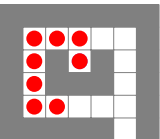
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
    call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
    mark[(0, 5)]=(-1, 5) => return 1
    mark[(1, 5)=(0, 5) => return 2
    mark[(1, 4)=(1, 5) => return 3
    mark[(1, 3)=(1, 4) => return 4
    mark[(1, 2)=(1, 3) => return 5
    mark[(1, 1)=(1, 2) => return 6
    mark[(2, 1)=(1, 1) => return 7
    mark[(3, 1)=(2, 1) => return 8
    mark[(4, 1)=(3, 1) => return 9
    mark[(4, 2)=(4, 1) => return 10
    mark[(4, 3)=(4, 2) => return 11
    mark[(3, 3)=(4, 3) => return 12
```



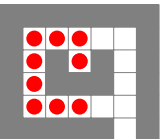
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 0)) => mark[(4, 0)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



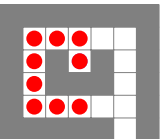
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



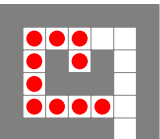
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)=(0, 5) => return 2
mark[(1, 4)=(1, 5) => return 3
mark[(1, 3)=(1, 4) => return 4
mark[(1, 2)=(1, 3) => return 5
mark[(1, 1)=(1, 2) => return 6
mark[(2, 1)=(1, 1) => return 7
mark[(3, 1)=(2, 1) => return 8
mark[(4, 1)=(3, 1) => return 9
mark[(4, 2)=(4, 1) => return 10
mark[(4, 3)=(4, 2) => return 11
mark[(3, 3)=(4, 3) => return 12
```



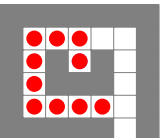
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



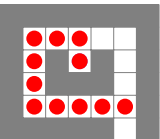
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



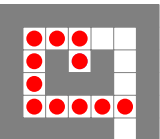
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



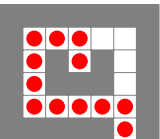
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```

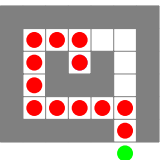
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self, start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



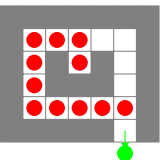
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)=(0, 5) => return 2
mark[(1, 4)=(1, 5) => return 3
mark[(1, 3)=(1, 4) => return 4
mark[(1, 2)=(1, 3) => return 5
mark[(1, 1)=(1, 2) => return 6
mark[(2, 1)=(1, 1) => return 7
mark[(3, 1)=(2, 1) => return 8
mark[(4, 1)=(3, 1) => return 9
mark[(4, 2)=(4, 1) => return 10
mark[(4, 3)=(4, 2) => return 11
mark[(3, 3)=(4, 3) => return 12
```



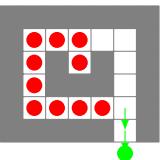
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

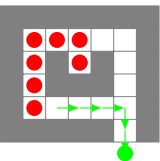
```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)=(0, 5) => return 2
mark[(1, 4)=(1, 5) => return 3
mark[(1, 3)=(1, 4) => return 4
mark[(1, 2)=(1, 3) => return 5
mark[(1, 1)=(1, 2) => return 6
mark[(2, 1)=(1, 1) => return 7
mark[(3, 1)=(2, 1) => return 8
mark[(4, 1)=(3, 1) => return 9
mark[(4, 2)=(4, 1) => return 10
mark[(4, 3)=(4, 2) => return 11
mark[(3, 3)=(4, 3) => return 12
```



21/30



22/30



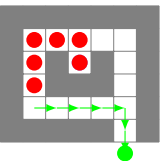
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



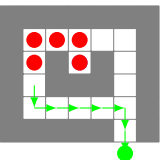
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```

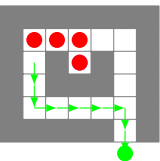
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



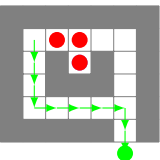
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



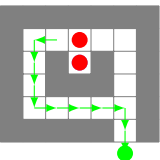
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



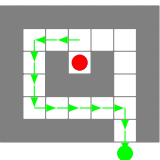
dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) => mark[(3, 3)]=EXPL
call sp.r((4, 3)) => mark[(4, 3)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL
call sp.r((4, 1)) => mark[(4, 1)]=EXPL
call sp.r((4, 2)) => mark[(4, 2)]=EXPL => return -1
call sp.r((3, 1)) => mark[(3, 1)]=EXPL
call sp.r((2, 1)) => mark[(2, 1)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL
call sp.r((1, 1)) => mark[(1, 1)]=EXPL => return -1
call sp.r((1, 3)) => mark[(1, 3)]=EXPL
call sp.r((1, 2)) => mark[(1, 2)]=EXPL => return -1
call sp.r((1, 4)) => mark[(1, 4)]=EXPL
call sp.r((1, 3)) => mark[(1, 3)]=EXPL => return -1
call sp.r((1, 5)) => mark[(1, 5)]=EXPL
call sp.r((1, 4)) => mark[(1, 4)]=EXPL => return -1
call sp.r((0, 5)) => mark[(0, 5)]=EXPL
call sp.r((-1, 5)) => mark[(-1, 5)]=EXIT => return 0
mark[(0, 5)]=(-1, 5) => return 1
mark[(1, 5)]=(-1, 5) => return 2
mark[(1, 4)]=(-1, 5) => return 3
mark[(1, 3)]=(-1, 5) => return 4
mark[(1, 2)]=(-1, 5) => return 5
mark[(1, 1)]=(-1, 5) => return 6
mark[(2, 1)]=(-1, 5) => return 7
mark[(3, 1)]=(-1, 5) => return 8
mark[(4, 1)]=(-1, 5) => return 9
mark[(4, 2)]=(-1, 5) => return 10
mark[(4, 3)]=(-1, 5) => return 11
mark[(3, 3)]=(-1, 5) => return 12
```



dirs=[down,up,left,right]

EXPL ●

EXIT ●

```
def somepath2exit(self,start):
    mark = dict()
    def somepath2exit_rec(sq):
        if self.isexit(sq):
            mark[sq] = EXIT
            return 0
        if sq in mark and mark[sq] == EXPL:
            return -1
        mark[sq] = EXPL
        pathlen = -1
        for next_sq in self.neighbors(sq):
            pathlen = somepath2exit_rec(next_sq)
            if pathlen >= 0:
                mark[sq] = next_sq
                pathlen += 1
                break
        return pathlen
    pathlen = somepath2exit_rec(start)
    if pathlen < 0: return None
    return pathlen, mark
```

```
call sp.r((3, 3)) ⇒ mark[(3, 3)]=EXPL
call sp.r((4, 3)) ⇒ mark[(4, 3)]=EXPL
call sp.r((4, 2)) ⇒ mark[(4, 2)]=EXPL
call sp.r((4, 1)) ⇒ mark[(4, 1)]=EXPL
call sp.r((4, 2)) ⇒ mark[(4, 2)]=EXPL ⇒ return -1
call sp.r((3, 1)) ⇒ mark[(3, 1)]=EXPL
call sp.r((2, 1)) ⇒ mark[(2, 1)]=EXPL
call sp.r((1, 1)) ⇒ mark[(1, 1)]=EXPL
call sp.r((1, 2)) ⇒ mark[(1, 2)]=EXPL
call sp.r((1, 1)) ⇒ mark[(1, 1)]=EXPL ⇒ return -1
call sp.r((1, 3)) ⇒ mark[(1, 3)]=EXPL
call sp.r((1, 2)) ⇒ mark[(1, 2)]=EXPL ⇒ return -1
call sp.r((1, 4)) ⇒ mark[(1, 4)]=EXPL
call sp.r((1, 3)) ⇒ mark[(1, 3)]=EXPL ⇒ return -1
call sp.r((1, 5)) ⇒ mark[(1, 5)]=EXPL
call sp.r((1, 4)) ⇒ mark[(1, 4)]=EXPL ⇒ return -1
call sp.r((0, 5)) ⇒ mark[(0, 5)]=EXPL
call sp.r((-1, 5)) ⇒ mark[(-1, 5)]=EXIT ⇒ return 0
mark[(0, 5)]=(-1, 5) ⇒ return 1
mark[(1, 5)]=(0, 5) ⇒ return 2
mark[(1, 4)]=(-1, 5) ⇒ return 3
mark[(1, 3)]=(-1, 4) ⇒ return 4
mark[(1, 2)]=(-1, 3) ⇒ return 5
mark[(1, 1)]=(-1, 2) ⇒ return 6
mark[(2, 1)]=(-1, 1) ⇒ return 7
mark[(3, 1)]=(-2, 1) ⇒ return 8
mark[(4, 1)]=(-3, 1) ⇒ return 9
mark[(4, 2)]=(-4, 1) ⇒ return 10
mark[(4, 3)]=(-4, 2) ⇒ return 11
mark[(3, 3)]=(-4, 3) ⇒ return 12
```



30/30