知識/知能プログラミング第3回 T322022 加藤 達也

深さ優先探索

```
from collections import deque

graph = []
parent = [0] * 8

def gen_graph():
    graph.append([])
    graph.append([2,3,4])
    graph.append([5,6])
    graph.append([2,4,7])
    graph.append([7])
    graph.append([6])
    graph.append([3,7])

def depth_first_search(start, goal):
    open = deque([start])
    closed = deque([])

    while open != []:
        n = open.popleft()
        if n == goal: return

        closed.append(n)
        for m in reversed(graph[n]):
            if m not in open and m not in closed:
                parent[m] = n
                open.appendleft(m)


def main():
    gen_graph()
    start, goal = 1,7
    depth_first_search(start, goal)

    n = goal
    print(n, end=' ')
    while n != start:
        n = parent[n]
        print("<-{0}".format(n), end = ' ')

if __name__ == "__main__":
    main()
```

実行結果

```
7 <-6 <-2 <-1
```

幅優先探索

```
from collections import deque

graph = []
parent = [0] * 8

def gen_graph():
    graph.append([])
    graph.append([2,3,4])
    graph.append([5,6])
    graph.append([2,4,7])
    graph.append([7])
    graph.append([6])
    graph.append([3,7])

def breadth_first_search(start, goal):
    open = deque([start])
    closed = deque([])

    while open:  # open != [] の代わりに while open: を使用
        n = open.popleft()
        if n == goal:
            return

        closed.append(n)  # ループ内でノードを closed リストに追加

        for m in graph[n]:
            if m not in open and m not in closed:
                parent[m] = n
                open.append(m)

def main():
    gen_graph()
    start, goal = 1, 7
    breadth_first_search(start, goal)

    n = goal
    print(n, end='')
    while n != start:
        n = parent[n]
        print("<-{0}".format(n), end='')

if __name__ == "__main__":
    main()
```

実行結果

```
7<-3<-1
```

水差し探索

```
from collections import deque

def successors(n):
    a, b = n
    suc = []
    if a < 4: suc.append((4,b))
    if b < 3: suc.append((a,3))
    if a > 0: suc.append((0,b))
    if b > 0: suc.append((a,0))
    if (a < 4) and (b != 0):
        if b > 4 - a:
            suc.append((4, b - (4 - a)))
        else:
            suc.append((a + b, 0))
    if( a < 3 ) and (a != 0):
        if a > 3 - b:
            suc.append((a - ( 3 - b), 3))
        else:
            suc.append((0, b + a))

    return suc


parent = {}

def breadth_first_search(start):
    open = deque([start])
    closed = deque([])

    while open != []:
        n = open.popleft()
        if(n[0] == 2) or (n[1] == 2): return n

        closed.append(n)
        for m in successors(n):
            if m not in open and m not in closed:
                parent[m] = n
                open.append(m)

def main():
    start = (0,0)
    goal = breadth_first_search(start)

    n = goal
```

```
    print(n, end='')
    while n != start:
        n = parent[n]
        print("<-{0}".format(n), end='')


if __name__ == "__main__":
    main()
```

実行結果

```
(4, 2)<-(3, 3)<-(3, 0)<-(0, 3)<-(0, 0)
```