2021 数学オリンピック予選問題 12 その解の１９枚のコイン配置手順

梅山新一郎 2022 年 9 月

問題文

12. 7×7 のマス目があり, 上から 1 行目, 左から 4 列目のマスに 1 枚のコインが置かれている. マス Y がマス X の**左下のマス**であるとは, ある正の整数 k について, Y が X の k マス左, k マス下にあることをいう. 同様に, マス Y がマス X の**右下のマス**であるとは, ある正の整数 k について, Y が X の k マス右, k マス下にあることをいう. 1 番下の行以外のマス X について, X にコインが置かれているとき次の 4 つの操作のうちいずれかを行うことができる：

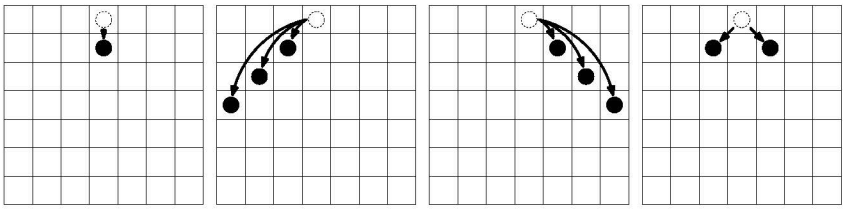
(a) X からコインを取り除き, X の 1 つ下のマスにコインを 1 枚置く.

(b) X からコインを取り除き, X の左下のマスそれぞれにコインを 1 枚ずつ置く.

(c) X からコインを取り除き, X の右下のマスそれぞれにコインを 1 枚ずつ置く.

(d) X からコインを取り除き, X の 1 マス左, 1 マス下にあるマスと X の 1 マス右, 1 マス下にあるマスにコインを 1 枚ずつ置く. ただし, そのようなマスが 1 つしかない場合はそのマスのみにコインを 1 枚置く.

ただし, コインを置こうとする場所にすでにコインがある場合, その場所にはコインを置かない.
操作を何回か行ったとき, マス目に置かれているコインの枚数としてありうる最大の値を求めよ.



(a) (b) (c) (d)

「コインの枚数は１９枚以下である」ことの証明

情けない話ですが、ここは分かりません。「大学への数学 2021-3」やネット上の解説を読んでください。分からないのは、それらの解説にある「マスに数を書き込む」というところですね。解説の後半は、19 枚のコインが実際に置ける手順が示されています。これで最大枚数が 19 枚と解説されていました。ここに示された手順もなかなか「思いつく」ものではないのす。そこで、分からない前半は目をつぶって、後半の 19 枚のコインを置く手順を探索してみました。

最大枚数とその手順を見つける「全数探索」

x 列 y 行にあるコインを (x, y) と書き、マス目上の n 個のコインのリストを $[(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)]$ と表すことにします。このリストの中のコイン $(x_i, y_i) \quad (i = 1, 2, \cdots, n)$ 全てに対して次の処理をする

- コインが最下行のときはこのリストを破棄する。
- コインが最下行以外のとき、
  - そのコイン対して４つの操作を行う
  - そのコインはリストから取り除く
  - 操作によって新たに置かれたコインがあればそれをリストに付け加える

新たなリストが作られたとき、そのそれぞれのリストに対して同様の処理を行います。ただし、この新たなリストのコインの組み合わせがこれまでの処理にあるときは、この新たなリストは処理対象から外します。コインのリスト $[(4, 1)]$ から始めて、コインのリストがなくなったとき探索を終了します。

このコインの操作手順は上の行にあるコインから処理していくものだと思いいくつか試してみた時には、16 個が最大値でしたが探索の結果は解説にあった通りの個数になりました。19 個のコインの組み合わせは 16 通り、その操作手順は合計 86 通りありました。以下に、実行結果の一部とプログラムリストを載せておきます。

searched **all**
elapsed time 1:12:18
max: 19 coins ——— 16 conins examples, 86 paths

coins No1 (1, 4),(2, 5),(4, 5),(6, 5),(7, 5),(1, 6),(2, 6),(3, 6),(4, 6),
(5, 6),(6, 6),(7, 6),(1, 7),(2, 7),(3, 7),(4, 7),(5, 7),(6, 7),(7, 7)
9 paths
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d,
(1, 6)a, (3, 2)d, (4, 3)d, (5, 4)c, (3, 4)d, (2, 5)a, (2, 3)d, (3, 4)b
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d,
(1, 6)a, (3, 2)d, (4, 3)d, (5, 4)c, (3, 4)b, (2, 5)a, (2, 3)d, (3, 4)d
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d,
(1, 6)a, (3, 2)d, (4, 3)d, (3, 4)d, (2, 5)a, (2, 3)d, (3, 4)b, (5, 4)c
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d,
(3, 2)d, (4, 3)d, (5, 4)c, (3, 4)d, (2, 3)d, (3, 4)a, (3, 5)b
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (2, 5)d, (1, 6)a,
(3, 2)d, (4, 3)d, (5, 4)c, (3, 4)b, (2, 3)d, (3, 4)a, (3, 5)d
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)c, (6, 5)a, (3, 4)d, (4, 5)d, (3, 6)a, (2, 5)d, (1, 6)a,
(3, 2)d, (4, 3)d, (3, 4)d, (4, 5)a, (2, 5)a, (2, 3)d, (3, 4)b, (5, 4)d
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d,
(5, 6)a, (3, 6)b, (4, 3)d, (5, 4)c, (3, 4)d, (4, 5)d, (3, 6)a, (2, 5)d, (1, 6)a, (3, 2)d,
(4, 3)d, (5, 4)a, (3, 4)d, (2, 5)a, (2, 3)d, (3, 4)b, (5, 5)d
(4, 1)d, (5, 2)d, (6, 3)d, (7, 4)a, (5, 4)d, (6, 5)d, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (4, 3)d, (5, 4)d, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d, (1, 6)a, (3, 2)d,
(4, 3)d, (5, 4)a, (3, 4)d, (2, 5)a, (2, 3)d, (3, 4)b, (5, 5)c
(4, 1)d, (5, 2)d, (6, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d, (1, 6)a,
(3, 2)d, (4, 3)d, (5, 4)c, (3, 4)d, (2, 5)a, (2, 3)d, (3, 4)b, (7, 4)a

. . . . . .

coins No8 (5, 4),(7, 4),(2, 5),(4, 5),(6, 5),(1, 6),(2, 6),(3, 6),(4, 6),
(5, 6),(6, 6),(7, 6),(1, 7),(2, 7),(3, 7),(4, 7),(5, 7),(6, 7),(7, 7)
4 paths

(4, 1)d, (5, 2)d, (6, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (4, 5)a, (2, 5)d, (1, 6)a,
(3, 2)c, (4, 3)b, (2, 5)a, (3, 4)d
(4, 1)d, (5, 2)d, (6, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (4, 3)d, (5, 4)d, (6, 5)a, (4, 5)d, (3, 6)a, (3, 4)d, (2, 5)d, (1, 6)a, (3, 2)c,
(4, 3)b, (3, 4)a, (3, 5)d
(4, 1)d, (5, 2)d, (6, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (4, 3)d, (5, 4)a, (3, 4)d, (4, 5)d, (3, 6)a, (2, 5)d, (1, 6)a, (3, 2)c, (4, 3)b,
(2, 5)a, (3, 4)d, (5, 5)d
(4, 1)d, (5, 2)d, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)d, (4, 5)d, (5, 6)a,
(3, 6)b, (3, 4)d, (4, 5)d, (3, 6)a, (2, 5)d, (1, 6)a, (3, 2)c, (6, 5)a, (5, 4)d, (4, 5)a,
(4, 3)b, (2, 5)a, (3, 4)d, (6, 3)d

. . . . . .

coins No16 (2, 5),(3, 5),(4, 5),(5, 5),(6, 5),(1, 6),(2, 6),(3, 6),(4, 6),
(5, 6),(6, 6),(7, 6),(1, 7),(2, 7),(3, 7),(4, 7),(5, 7),(6, 7),(7, 7)
5 paths
(4, 1)d, (5, 2)b, (1, 6)a, (2, 5)d, (3, 6)d, (3, 4)d, (4, 5)d, (5, 6)c, (3, 6)a,
(2, 5)a, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)a, (4, 5)d, (3, 4)d, (4, 5)a, (3, 2)c,
(6, 5)a, (5, 4)d, (4, 3)d, (5, 4)a, (3, 4)a
(4, 1)d, (5, 2)b, (1, 6)a, (2, 5)d, (3, 6)d, (3, 4)d, (4, 5)d, (5, 6)c, (3, 6)a,
(2, 5)a, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)a, (4, 5)d, (3, 4)d, (4, 5)a, (3, 2)c,
(6, 5)a, (5, 4)d, (4, 3)d, (3, 4)a, (5, 4)a
(4, 1)d, (5, 2)b, (1, 6)a, (2, 5)d, (3, 6)d, (3, 4)d, (4, 5)d, (5, 6)c, (3, 6)a,
(2, 5)a, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)a, (4, 5)d, (3, 4)d, (4, 5)a, (3, 2)c,
(6, 5)a, (5, 4)d, (4, 3)a, (4, 4)d
(4, 1)d, (5, 2)b, (1, 6)a, (2, 5)d, (3, 6)d, (3, 4)d, (4, 5)d, (5, 6)c, (3, 6)a,
(2, 5)a, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)a, (4, 5)d, (3, 4)d, (4, 5)a, (3, 2)c,
(6, 5)a, (5, 4)a, (4, 3)d, (3, 4)a, (5, 4)d
(4, 1)d, (5, 2)b, (1, 6)a, (2, 5)d, (3, 6)d, (3, 4)d, (4, 5)d, (5, 6)c, (3, 6)a,
(2, 5)a, (4, 3)d, (5, 4)d, (6, 5)d, (7, 6)a, (5, 6)a, (4, 5)d, (3, 4)a, (3, 2)c, (6, 5)a,
(5, 4)d, (4, 5)a, (4, 3)d, (5, 4)a, (3, 4)d

```
# coding: utf-8
from copy import copy
import time
class MO2021P12SearchAll():
        def __init__(self, board_size:int = 7, expected_coins_number=7*7):
                self.board_size = board_size
                self.board_with, self.board_height = self.board_size - 1, self.board_size - 1
                pos = self.board_with // 2
                self.expected_coins_number = expected_coins_number
                self.nodes = [ ([pos], []) ]
                self.coins_dic = {(pos): 1}
                self.max_coins, self.max_coins_path = -1, dict()
        def pos2xy(self, pos:int)->tuple:
                y0, x0 = divmod(pos, self.board_size)
                return (x0+1, y0+1)
        def op_abcd(self, op:str, pos:int, coins:list)->list:
                y, x = divmod(pos, self.board_size)
                ret = []
                if op == 'a':
                        pos1 = (y+1)*self.board_size + x
                        if y+1 > self.board_height or pos1 in coins:
                                return []
                        else:
                                return [pos1]
                elif op == 'b':
                        for dxy in range(1, self.board_height+1):
                                pos1 = (y + dxy) * self.board_size + (x - dxy)
                                if x-dxy < 0 or y+dxy > self.board_height:
                                        break
                                if pos1 not in coins:
                                        ret.append(pos1)
                        return ret
                elif op == 'c':
                        for dxy in range(1, self.board_height + 1):
                                pos1 = (y+dxy)*self.board_size + (x+dxy)
                                if x + dxy > self.board_with or y + dxy > self.board_height:
                                        return ret
                                if pos1 not in coins:
                                        ret.append(pos1)
                        return ret
```

```
        else:  #  op=='d'
                if y+1 > self.board_height:
                        return []
                pos1 = (y+1)*self.board_size + (x-1)
                if x - 1 >= 0 and pos1 not in coins:
                        ret.append(pos1)
                pos2 = (y+1)*self.board_size + (x+1)
                if x + 1 <= self.board_with and pos2 not in coins:
                        ret.append(pos2)
                return ret
def search_max_coins(self):
        while True:
                if len(self.nodes) == 0:
                        print('searched all')
                        return (self.max_coins, self.max_coins_path)
                if (self.max_coins == self.expected_coins_number):
                        print('reached the expected coins : ',
                         self.expected_coins_number )
                        return (self.max_coins, self.max_coins_path)
                coins, path = self.nodes.pop()
                l = len(coins)
                if l == 0:
                        continue
                for i in range(l):
                        coins1 = copy(coins)
                        pos:int = coins1[i]
                        pos_y = pos // self.board_size
                        if pos_y >= self.board_height:
                                continue
                        del coins1[i]
                        for op in ('a', 'b', 'c', 'd'):
                                coins_1: list = copy(coins1)
                                ret = self.op_abcd(op, pos, coins_1)
                                path_1:list = copy(path)
                                path_1.append((pos, op))
                                coins_1.extend(ret)
                                if len(coins_1) > self.max_coins:
                                        self.max_coins = len(coins_1)
                                        self.max_coins_path=
                                          {tuple(sorted(coins_1)): [path_1]}
```

```
                                        elif len(coins_1) == self.max_coins:
                                                t = tuple(sorted(coins_1))
                                                if t in self.max_coins_path:
                                                        self.max_coins_path[t].append(path_1)
                                                else:
                                                        self.max_coins_path[t] = [path_1]
                                        coins_t = tuple(sorted(coins_1))
                                        if (coins_t not in self.coins_dic):
                                                self.coins_dic[coins_t] = 1
                                                self.nodes.append((coins_1, path_1))
                                        else:
                                                self.coins_dic[coins_t] += 1
if __name__ == '__main__':
        m2021p12 = MO2021P12SearchAll(7, 20)
        st = time.time()
        ret =  m2021p12.search_max_coins()
        t = int(time.time() - st)
        tm, ts = divmod(t, 60)
        th, tm = divmod(tm, 60)
        print(f'elapsed time  {th}:{tm}:{ts}')
        path_total=0
        for paths in ret[1].values():
                path_total += len(paths)
        print(f'max: {ret[0]} coins ------- {len(ret[1])} conins examples, {path_total} paths')
        no = 0
        for coins, paths in ret[1].items():
                no += 1
                print(f'\ncoins No{no} ', end=' ')
                for pos in coins[:-1]:
                        print(f'{(m2021p12.pos2xy(pos))}', end=',')
                print(f'{(m2021p12.pos2xy(coins[-1]))}')
                print(f'{len(paths)} paths')
                for path in paths:
                        for pos_dir in path[:-1]:
                                print(f'{(m2021p12.pos2xy(pos_dir[0]))}{pos_dir[1]}',
                                 end=', ')
                        print(f'{(m2021p12.pos2xy(path[-1][0]))}{path[-1][1]}')
```