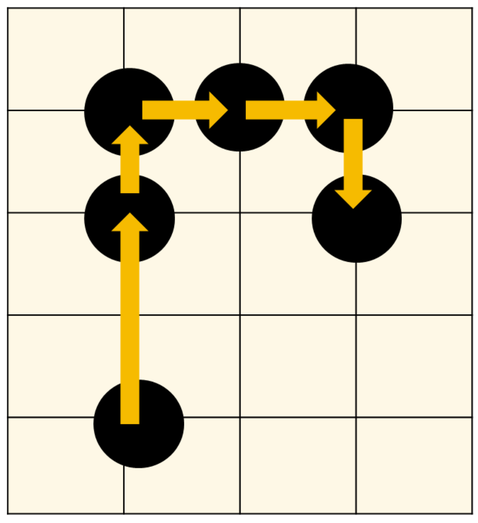
「碁石拾い」の解法数を算出できるプログラムの作成

兵庫県立宝塚北高等学校GS科　　大庭颯来　永谷洸人

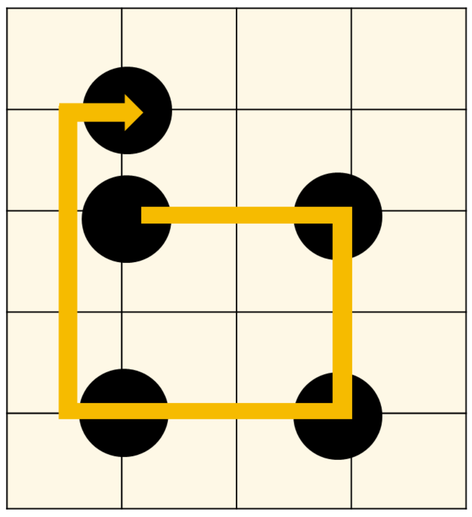
① 碁石拾い(ひろいもの)とは

・江戸時代の日本発祥
・碁盤上の石を全て拾うパズルゲーム

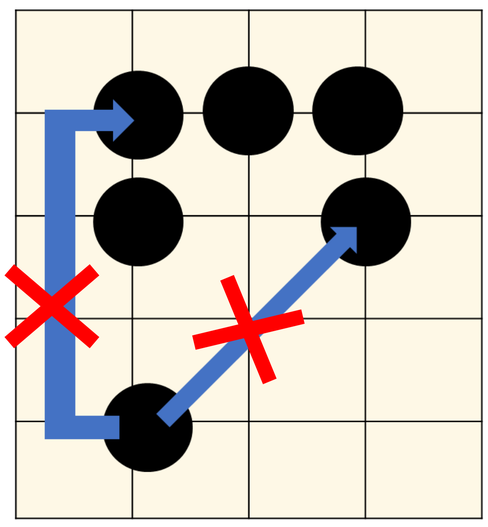
② 碁石拾いのルール



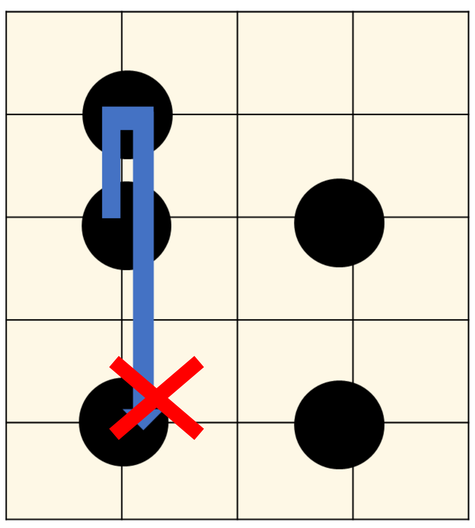
・**縦横**に移動
・石同士の間の**隙間**は
飛び越えてよい



・**拾った後**の石は
無いものとして捉える



・**斜め**の移動は不可
・**石を飛び越える**ことは不可



・**180°** の方向転換は不可

③研究の動機・目的

主体性や**創造性**の育成のため、
学校の授業に導入する試みがある

問題点

・碁石拾いの問題や解法をまとめたサイト・書籍が**少ない**
・経路が複雑で、**解法が複数**考えられる
→作問も**難しい**

碁石拾いを簡単に広めたい・活かしたい

・**解法数**を算出したい(④)
・**作問**を簡単にしたい(⑤)

人の手では難しい

プログラム(Python)の活用

④解法数を算出するプログラム

Python3.7を用いて、プログラムを作成した
(一部省略して掲載)

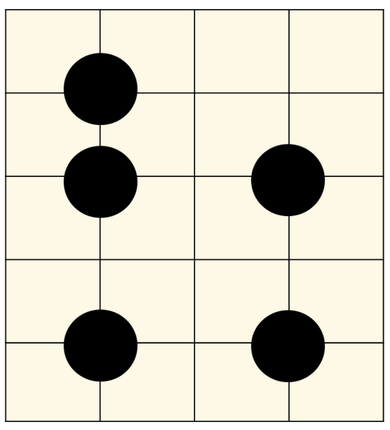
1. n (石の数), h (縦座標), w (横座標)を入力

```
n = int(input())
for i in range(n):
    w[i],h[i] = map(int,input().split())
```

【入力例】

```
5
1 1
1 3
1 4
2 1
2 3
```

→ 石の数…5
座標…(1,1)
(1,3)
(1,4)
(2,1)
(2,3)

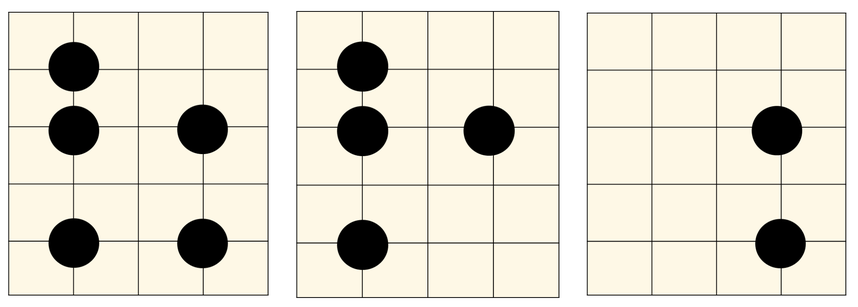


2. 石同士の位置関係を定義

```
for bit in range(1,nn):
    for i in range(n):
        for j in range(n):
```

考えられる**全ての**盤面ごとに、
それぞれの石から
どの方向にどの石があるか確認

上の入力例だと…



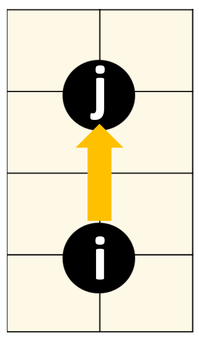
など、
2^5通りの盤面

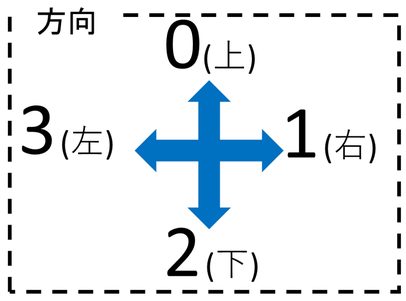
```
if(w[i] == w[j]):
        if(h[i]<h[j]):
                if(g[bit][i][0]==inf):
                        g[bit][i][0] = j
                else:
                        ch = h[g[bit][i][0]]
                        if(h[j]<ch):
                                g[bit][i][0] = j
```

・**横座標→同じ**
・**縦座標**→石 j の方が**大きい**

石 j は石 i の上側(0の方向)にある。
同方向に複数の石がある場合、
より**近い方**を参照

(他の方向についても同様)





3. 条件を満たす経路探し

```
dp = [[[0 for i in range(4)] for j in range(n)]for k in range(nn)]
for i in range(n):
    for x in range(4):
        dp[pow(2,i)][i][x] = 1
```

動的計画法(小さい計算結果から大きい計算をする手法)を用いた。
各盤面、各石から4方向について確認

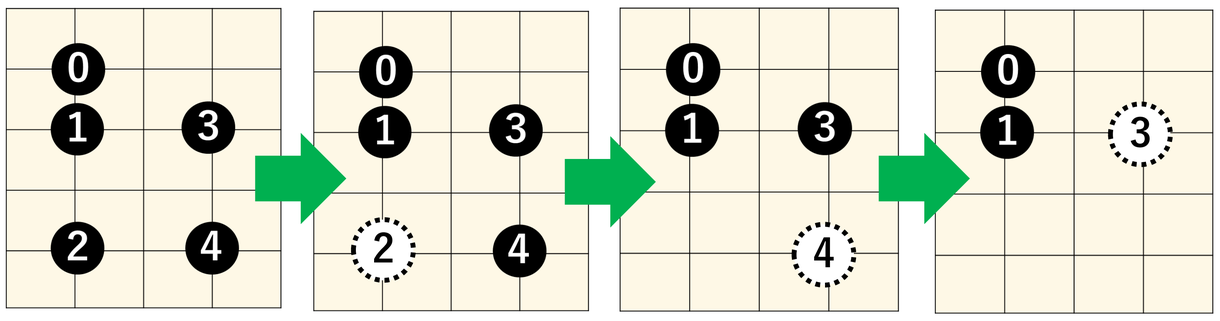
```
for bit in range(nn):
    for i in range(n):
        for x in range(4):
            for y in range(4):
                if(x == 0 and y == 2):
                    continue
```

x→**直前に**進んだ向き
y→**次に**進む向き

真逆の方向に進むと、
碁石拾いのルールに反する
(1と3でも同様)

```
j = g[bit][i][y]
    if(j == inf):
        continue
    nbit = pow(2,j)
    if((bit & nbit) == 0):
        dp[nbit+bit][j][y] += dp[bit][i][x]
```

iから y の方向に
石がない→不可



拾う石	2	4	3
bit	00000	00100	10100
nbit	00100	10000	01000
nbit+bit	00100	10100	11100

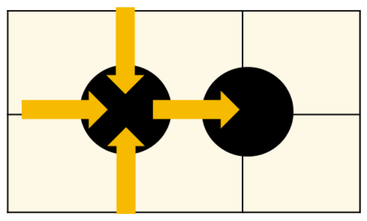
bit→今まで拾った石
nbit→今から拾う石
まだ拾っていない石なら拾える

4. 解法数の算出

```
for i in range(n):
    for x in range(4):
        res += dp[nn-1][i][x]
res /= 3
print(int(res))
```

計算過程上、始点の前の移動が
3種類存在→3で割る

解法数を表示



⑤作問ができるプログラム

④のプログラムを一部変更し、
自動で碁石拾いの問題を作成できるプログラムにした。

変更点1. 石の数と座標

(④-1の部分と入れ替え)

```
res = 0
while res <=0:
```

resの値が0以下の場合、
(すなわち**解けない**問題の場合)
石の数と座標を**変更**

```
    import random
    n = random.randint(6,10)
    for i in range(n):
        h[i] = random.randint(1,5)
        w[i] = random.randint(1,5)
```

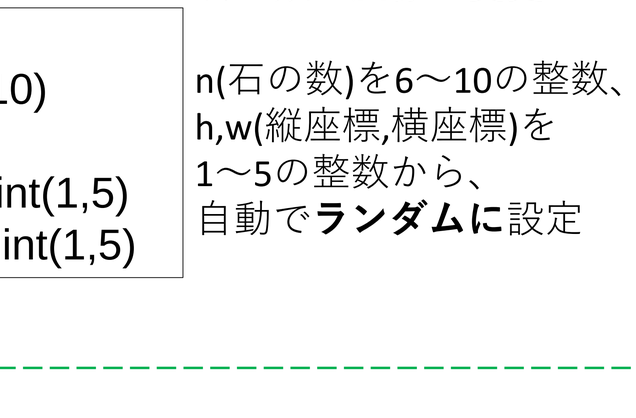
n(石の数)を6～10の整数、
h,w(縦座標,横座標)を
1～5の整数から、
自動で**ランダムに**設定

変更点2. 結果の表示

(④-4の後に追加)

```
print(h)
print(w)
print(res)
import matplotlib.pyplot as plt
for i in range(n):
    plt.plot(h[i],w[i],marker='.',markersize=50,color='blue')
```

石の配置を
数字・図の
両方で表示。



横座標
縦座標
解法数

```
[3, 1, 5, 5, 4, 3, 1]
[1, 5, 1, 3, 3, 5, 2]
2.0
```

⑥まとめ

解法数算出

石の数、座標を
手動で入力

石同士の位置関係を定義

ルールに合わない経路を除外

解法数を算出

解法数を表示

自動作問

実行

石の数、座標を
ランダムに設定

石同士の位置関係を定義

ルールに合わない経路を除外

解法数を算出

解法数は

0
(問題として不成立)

1以上

石の配置を表示

⑦考察

利点

解法数→解法の見落としを減らせる
作問→多くの問題を**短時間**で作成

改善点

解法数→**座標入力**がやや面倒
作問→**同じ座標**に複数の碁石が置かれることがある

⑧今後の展望

現在のプログラムの改善
・石の座標に被りが生じない作問プログラム作成
・より簡易な座標の入力方法や、結果の表示方法

今回作成したプログラムの応用
・解法数が0になるとき、解法数を1以上にするため
石を追加・削除するプログラム
・正しい経路を表示できるプログラム
・実際にPC上で碁石拾いを遊べるようにしたい

碁石拾いというゲーム自体の深掘り
・適切な難易度にするための石の数や、
盤面の広さについての考察

⑨参考文献

(1)中根法舳　『勘者御伽双紙』第2冊 天王寺屋市郎兵衛(1743) 国立国会図書館デジタルコレクション https://dl.ndl.go.jp/pid/3511866 (2023.06.10 閲覧)
(2)Matthew J. Costello　(1988), The GreatestPuzzles of All Time, Prentice Hall　Press
(3)片山真一,安井宏晴(2019),碁石拾いとFibonacci数,徳島科学史雑誌,NO.38 (2019),P20-26
(4)福井昌則(2017),数学的ゲーム・パズルを用いた研究・学習活動を行うための環境構築を目的にしたiOSアプリケーション開発,ゲームプログラミングワークショップ2017論文集, p 119-125
(5)Daniel Andersson,Pierluigi Crescenzi他, HIROIMONO Is NP-Complete
Fun with Algorithms. FUN 2007. Lecture Notes in Computer Science, vol 4475.