aa-10. 遷移関数、探索、総当たり

（人工知能）

URL: https://www.kkaneko.jp/ai/mi/index.html

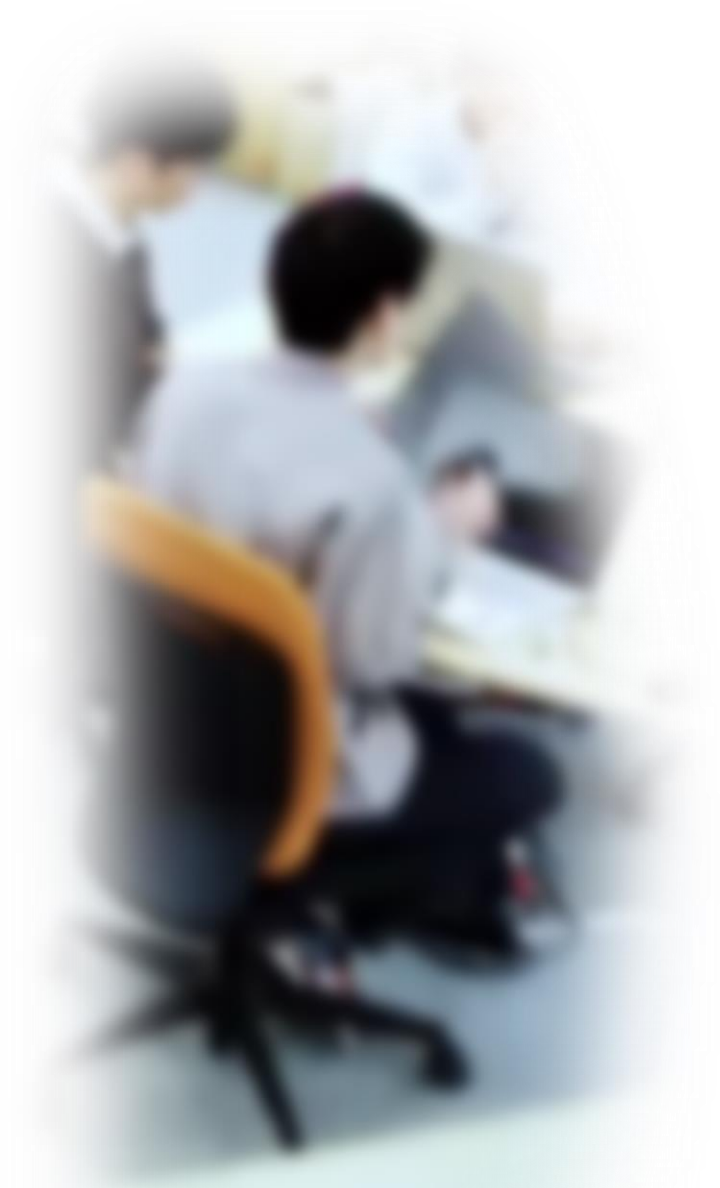
金子邦彦

① 遷移関数、探索、総当たりというAIの概念

② 遷移関数によるAIの行動予測と問題解決のプロセス

③ 総当たり探索の利点と、パスの概念

AIを活用した問題解決の基本的なアプローチの１つを紹介．体系的な知識とスキルを提供.



アウトライン

1. 遷移関数と探索
2. 総当たり
3. 総当たりのパス
4. 総当たりの例
5. 総当たりを行う人工知能
6. ２つの水差し

10-1 遷移関数と探索

エレベーターの遷移関数

- **状態**

  エレベーターの現在の階数

- **行動**

  エレベーターを**上に移動させる（行動１）**か、
  下に移動させる（行動２）か、
  その場に留まる（行動３）

- **遷移関数**

  行動１：　もし、現在が最上階でなければ
  階数 = 階数 + 1　・・・階数が１増える
  行動２：　もし、現在が最下階でなければ
  階数 = 階数 – 1　・・・階数が１減る
  行動３：　階数 = 階数　・・・階数は変わらない

21 ゲームの遷移関数

- **状態**

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21　　※ 最初は 0 である

- **行動**

  １を足す（行動１）、２を足す（行動２）、３を足す（行動３）

- **遷移関数**

  行動１：　（数＜２１　のときのみ可能）

  数 = 数 + １　・・・数が１増える

  行動２：　（数＜２１　のときのみ可能）

  数 = 数 + ２　・・・数が２増える

  行動３：　（数＜２１　のときのみ可能）

  数 = 数 + ３　・・・数が３増える

遷移関数まとめ

- **遷移関数**は、**特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則**である。

- **状態**は、**具体的な問題や状況**を表す。

遷移関数と状態は、AIの動作原理を理解し、問題解決能力を向上させるための重要な要素

探索、選択

遷移関数を用いることで、AI は可能な行動とその結果を予測し、その中から**最適な行動を選択することが可能**となる

そのための２つの段階「探索」と「選択」

- **探索**

AIは**遷移関数**を使用して、**可能な行動**とその行動が引き起こす**新しい状態**を調べる。

- **選択**

AIは、探索により得られた**新しい状態**を評価し、その中から最も目標達成に寄与する**行動を選択**する。

ゲームのAIの場合

①　遷移関数を使用して、現在の状態から可能な行動と、その結果導かれる新しい状態を調べる（探索）

②　探索の結果得られた新しい状態を評価し、**ゲームに勝つという目標を達成する最善の行動を選択（選択）**

チェス、囲碁、将棋などのゲーム

自動運転車のAIの場合

①　遷移関数を使用して、現在の状態から可能な行動と、その結果導かれる新しい状態を調べる（探索）

現在の状態：位置、速度、周囲の状況

可能な行動：加速、減速、左折、右折

②　探索の結果得られた新しい状態を評価し、安全に目的地に到着するという目標を達成する最善の行動を選択（選択）

他の例：工場の製造工程のスケジューリング

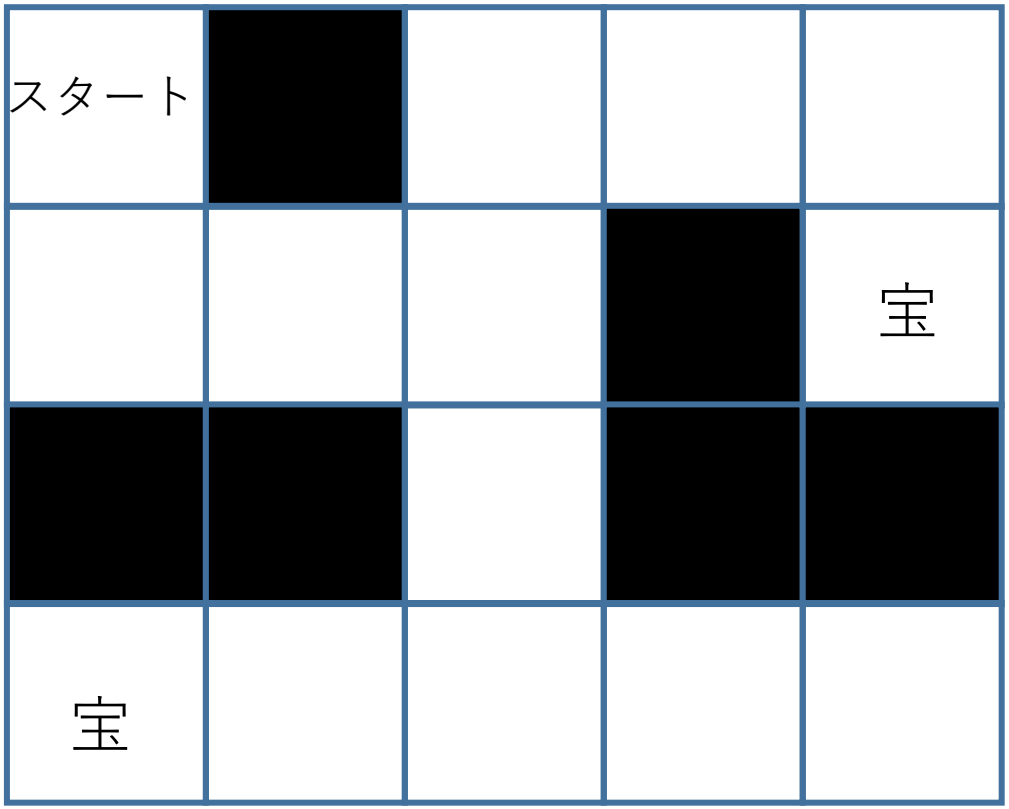
各工程の順序，時間配分を最適化．遷移関数によって可能なスケジュールを生成し，探索を通じて最も効率的なスケジュールを決定

10-2 総当たり

総当たり

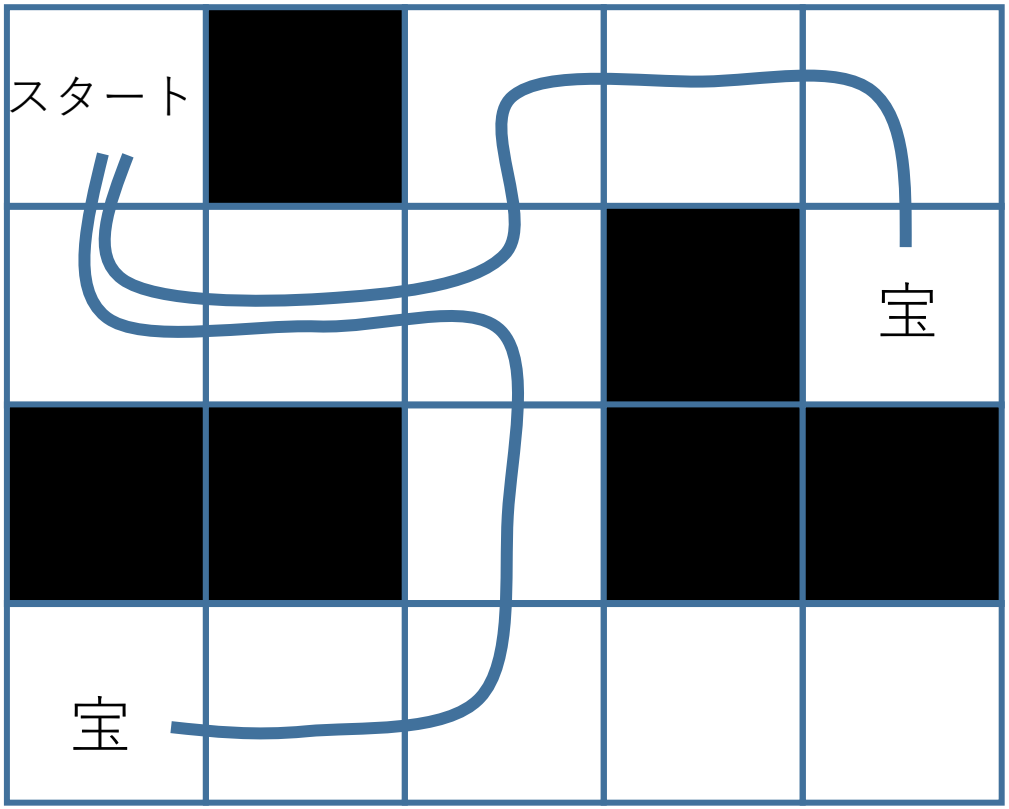
- **総当たり**は，**探索**の一種
- **可能なすべての行動を順番に試す**ことで、問題解決を行う
- 一連の行動のことを「**経路**（**パス**）」と呼ぶことがある
- **総当たり**のメリットは、**全ての可能性を試しつくす**ことで、最善の行動を選択すること。

探索と経路（パス）



- スタートから出発
- 迷路をたどり，宝に至る**経路**（**パス**）を探す

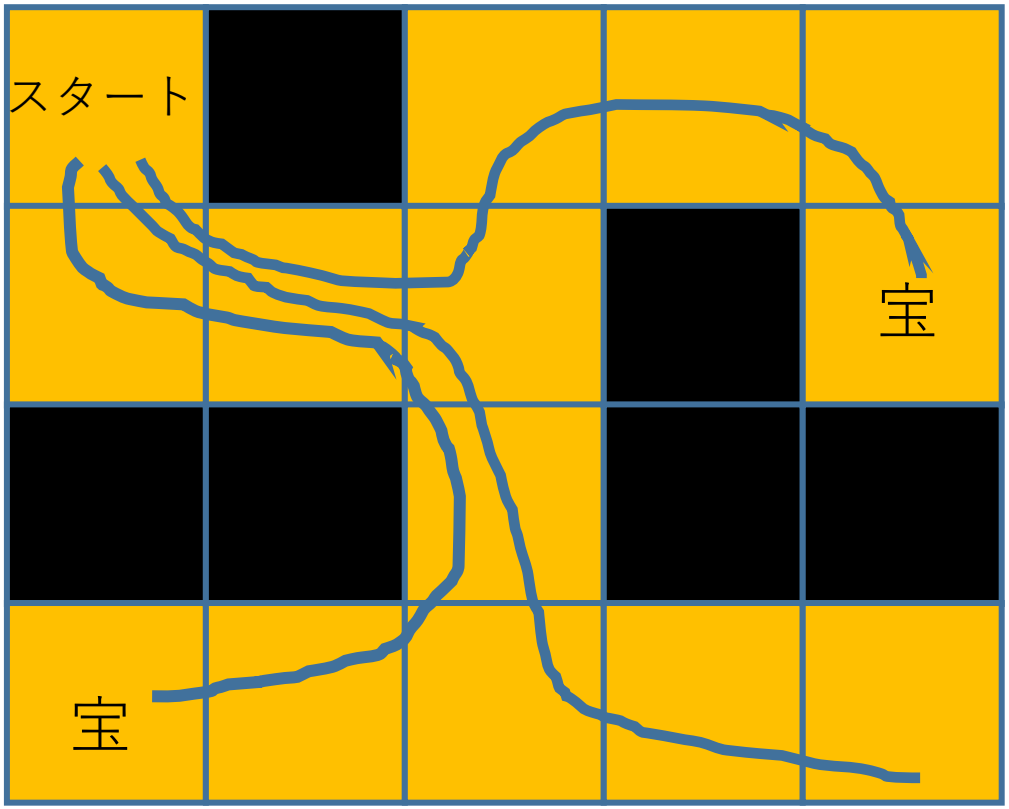
探索と経路（パス）



- スタートから出発
- 迷路をたどり，宝に至る**経路**（**パス**）を探す

※ 同じ経路（パス）を２回試さないことは当たり前

総当たり



総当たりは，
可能なすべての経路
（パス）を試す

10-3 総当たりのパス

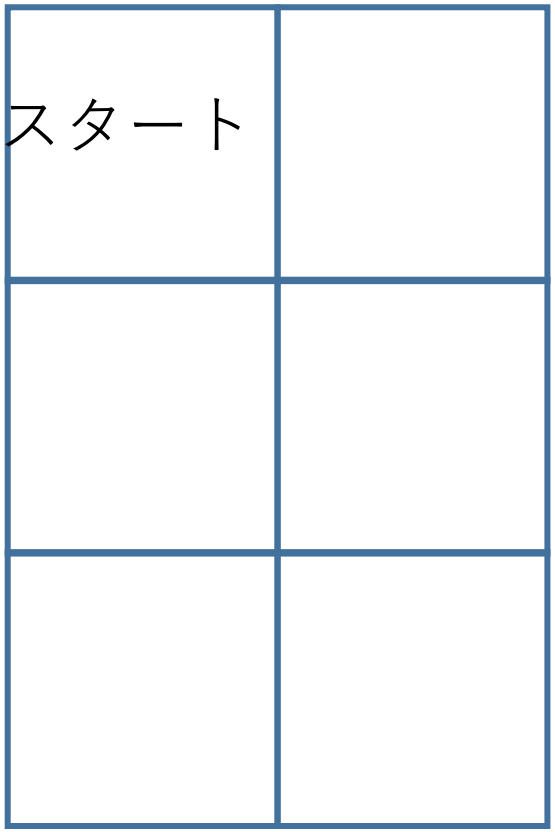
総当たりの特徴

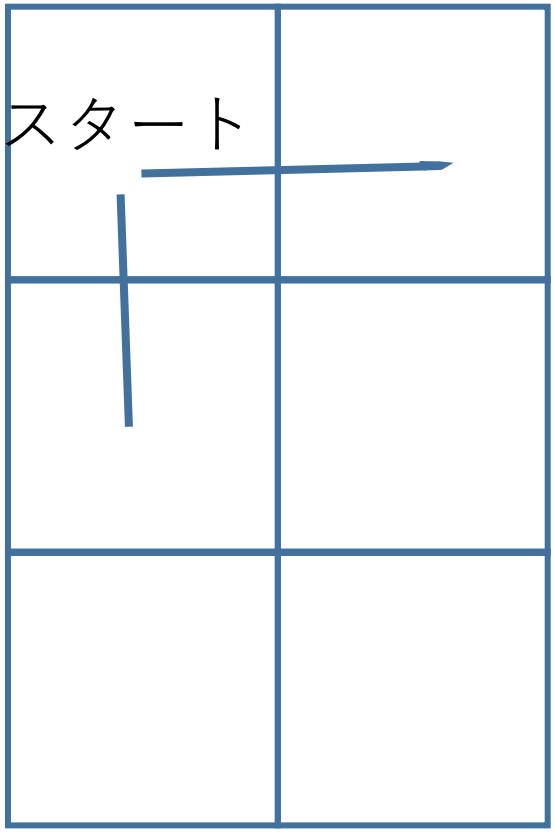
- **総当たりは、可能な全ての経路（パス）を試す**

- それぞれの経路（パス）は、一度だけ試される。二度試すことはない。

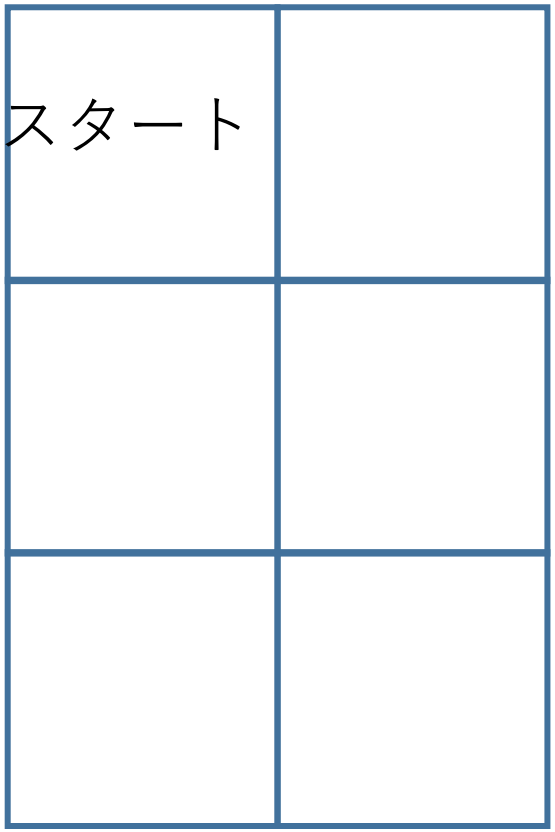
- **異なる経路**を通じて**同じ「場所」、同じ「状態」に至る**ことはあり得る。従って、同じ場所や状態を何度も訪れることはありえる。

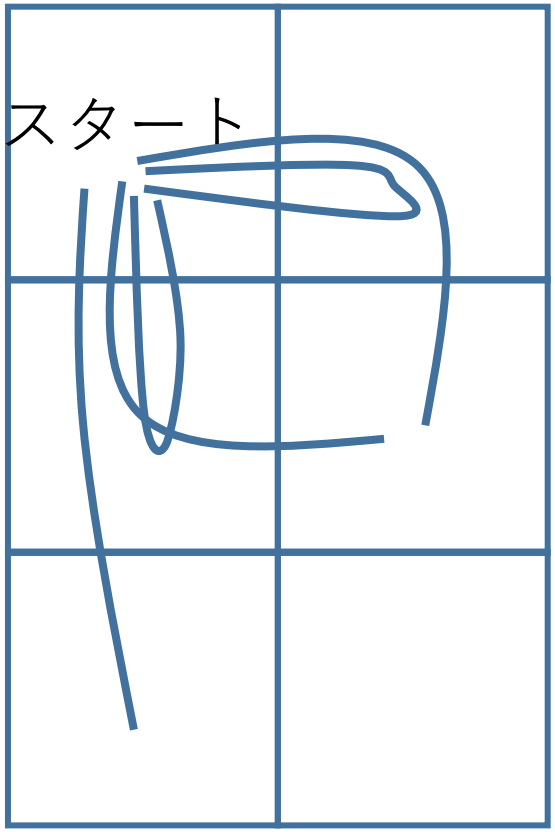
総当たりの例　パス長１



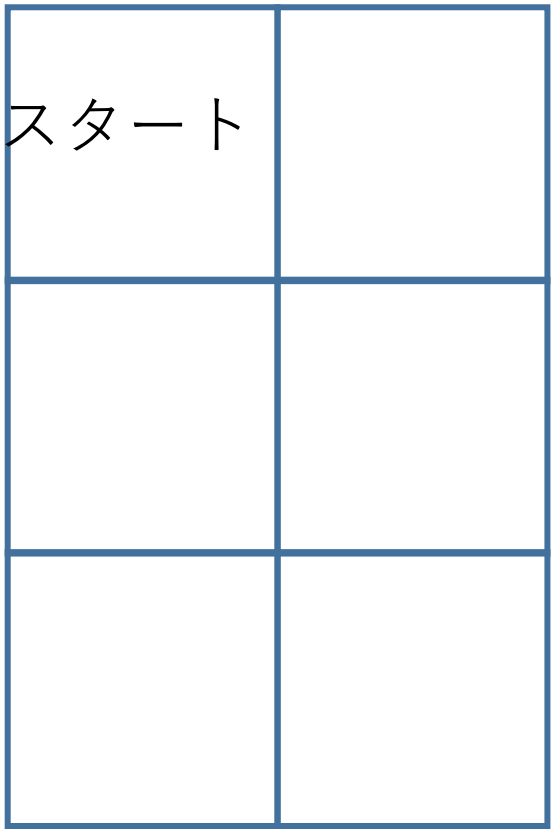


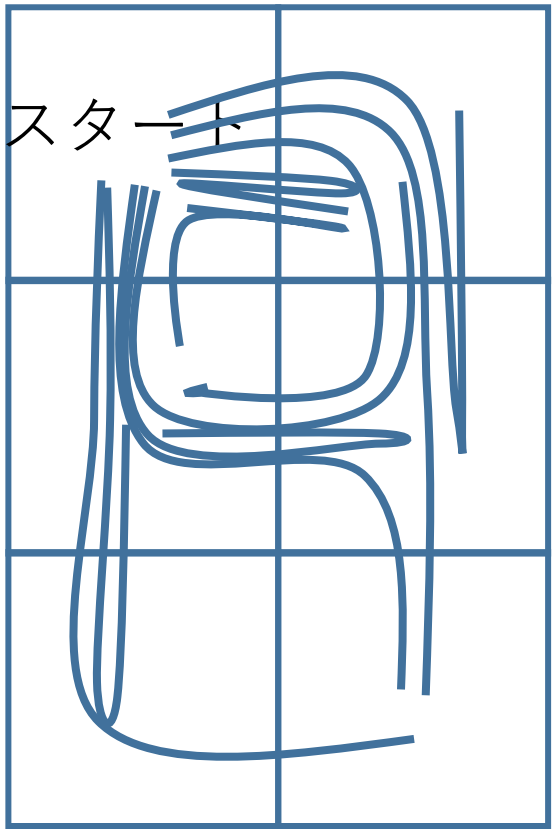
総当たりの例　パス長２





総当たりの例　パス長３





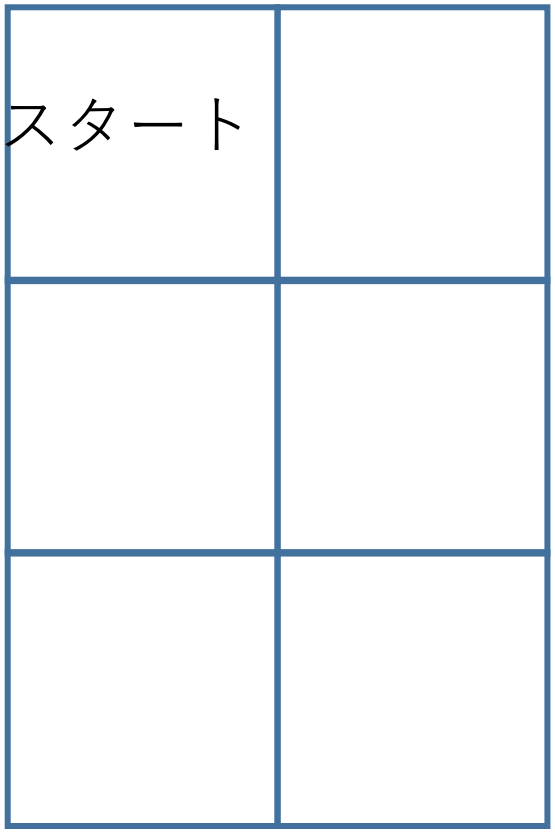
10-4 総当たりの例

総当たり

- **総当たりは、可能な全ての経路（パス）を試す**

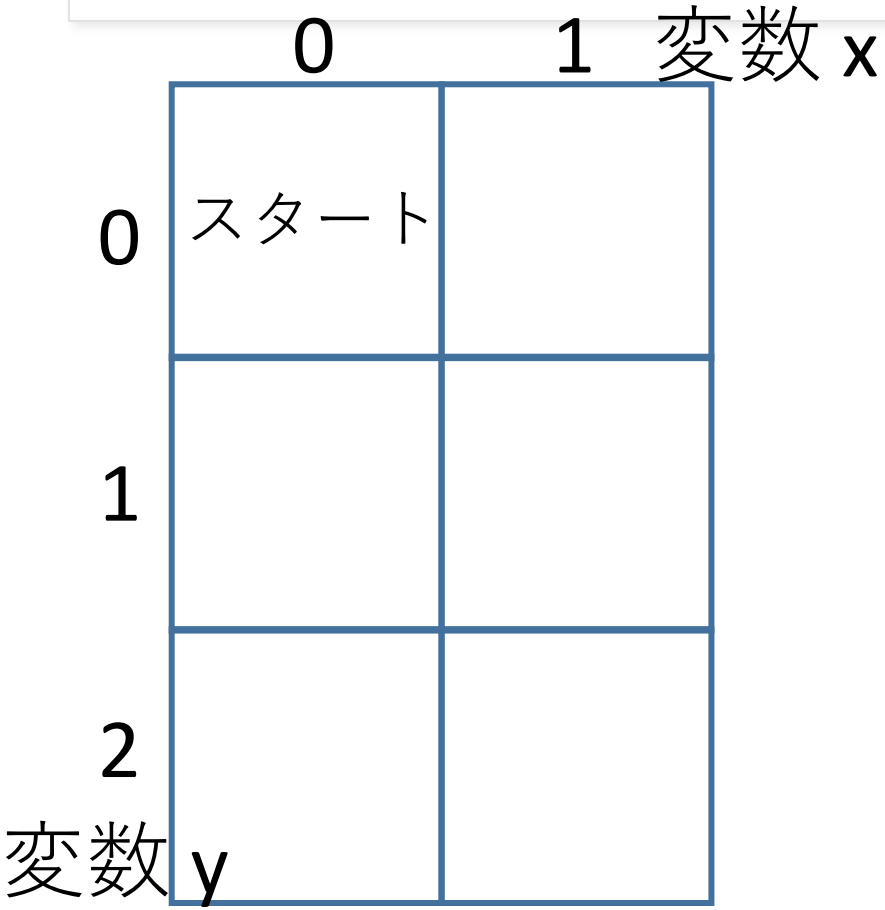
- 状態 (x, y) の変化

探索の対象



縦　3，横　2

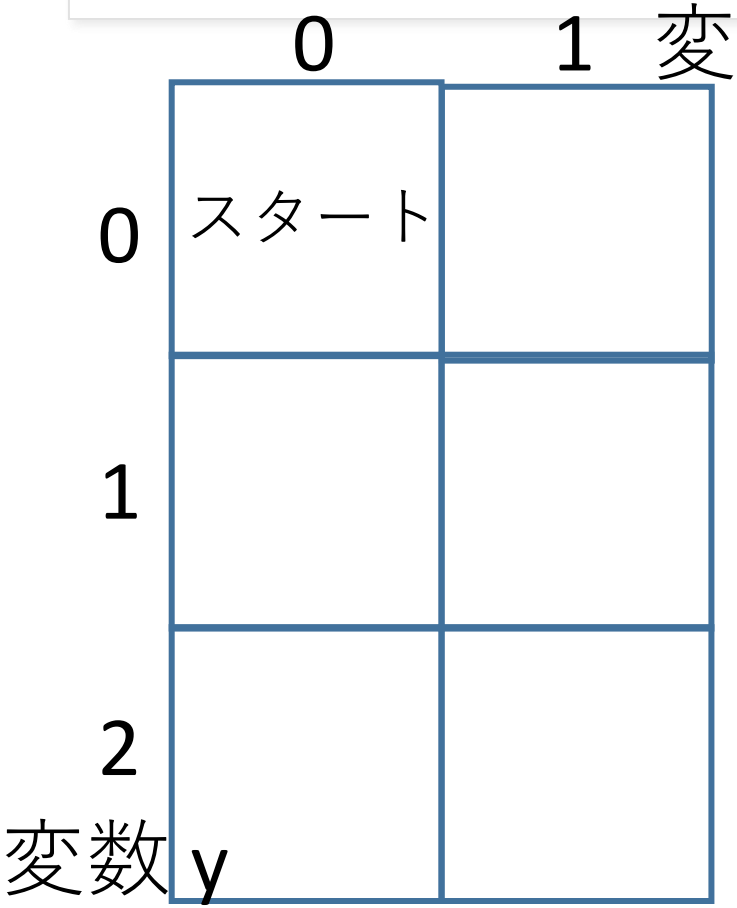
状態を変数で表現



現在地は状態である。
状態は**変数**で表現する。

変数 x, y

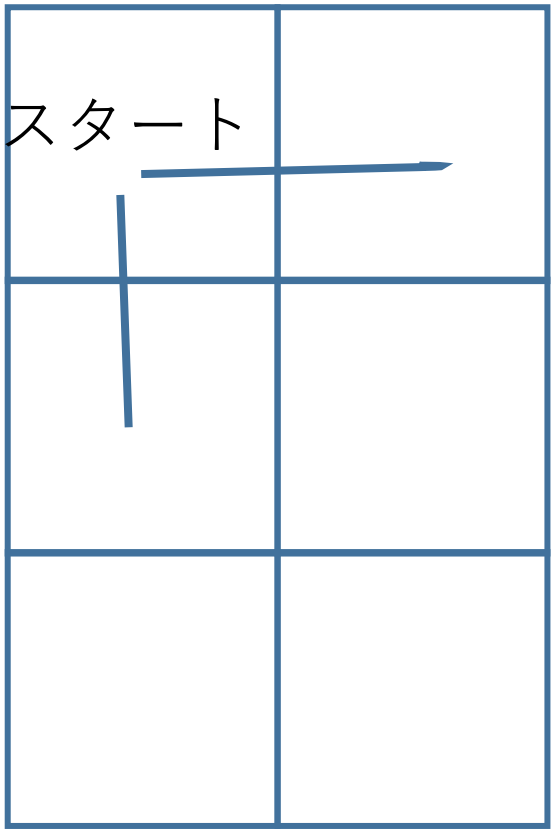
遷移関数と行動番号



行動番号

1. 右へ (**x < 1 のときのみ可能**)
   x = x + 1, y = y
2. 左へ (**x > 0 のときのみ可能**)
   x = x − 1, y = y
3. 下へ (**y < 2 のときのみ可能**)
   x = x, y = y + 1
4. 上へ (**y > 0 のときのみ可能**)
   x = x, y = y - 1

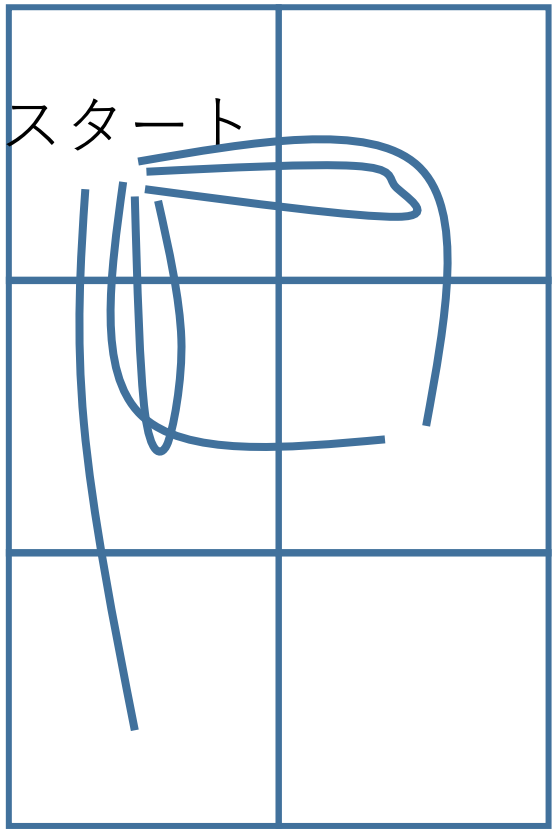
総当たりの例　パス長１



総当たりの対象
- 一連の行動：**1**
- 一連の行動：**3**

それぞれが
経路（パス）

総当たりの例　パス長２

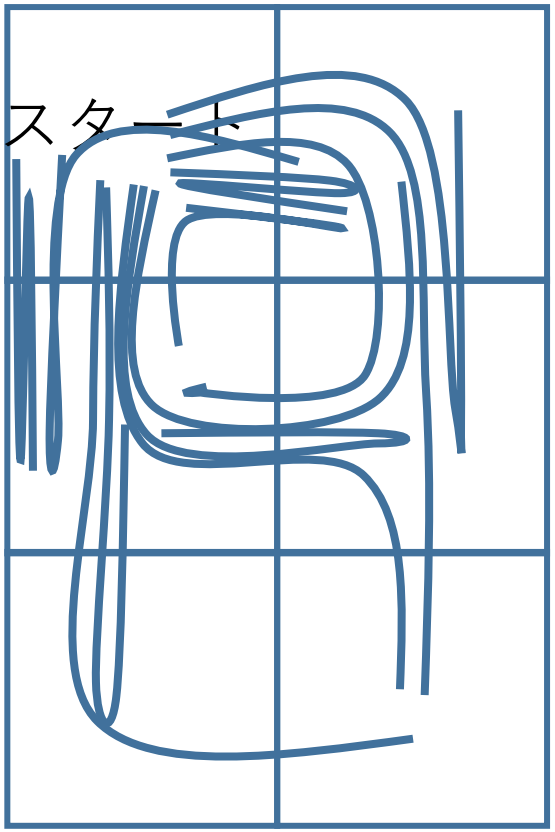


総当たりの対象

- 一連の行動：　**1, 2**
- 一連の行動：　**1, 3**
- 一連の行動：　**3, 1**
- 一連の行動：　**3, 3**
- 一連の行動：　**3, 4**

それぞれが経路（パス）

探索の例　パス長３



総当たりの対象

- 一連の行動： **1, 2, 1**
- 一連の行動： **1, 2, 3**
- 一連の行動： **1, 3, 2**
- 一連の行動： **1, 3, 3**
- 一連の行動： **1, 3, 4**
- 一連の行動： **3, 1, 2**
- 一連の行動： **3, 1, 3**
- 一連の行動： **3, 1, 4**
- 一連の行動： **3, 3, 1**
- 一連の行動： **3, 3, 4**
- 一連の行動： **3, 4, 1**
- 一連の行動： **3, 4, 3**

それぞれが
経路（パス）

経路（パス）は行動番号の並びである

- 一連の行動のことを「**経路（パス）**」と呼ぶことがある

- **経路（パス）は，行動番号の並び**

  「1」、「3-3」、「3-4-3」など

まとめ

- AIでは、**遷移関数**を用いて**可能な状態の変化を予測**し、**探索**と**選択**を通じて最適な行動を決定。

- **総当たり**では**全ての可能な経路を試し**、最善の解を見つける。

- **探索**と**選択**のプロセスは、AIが**問題解決**を行う上での重要な要素。

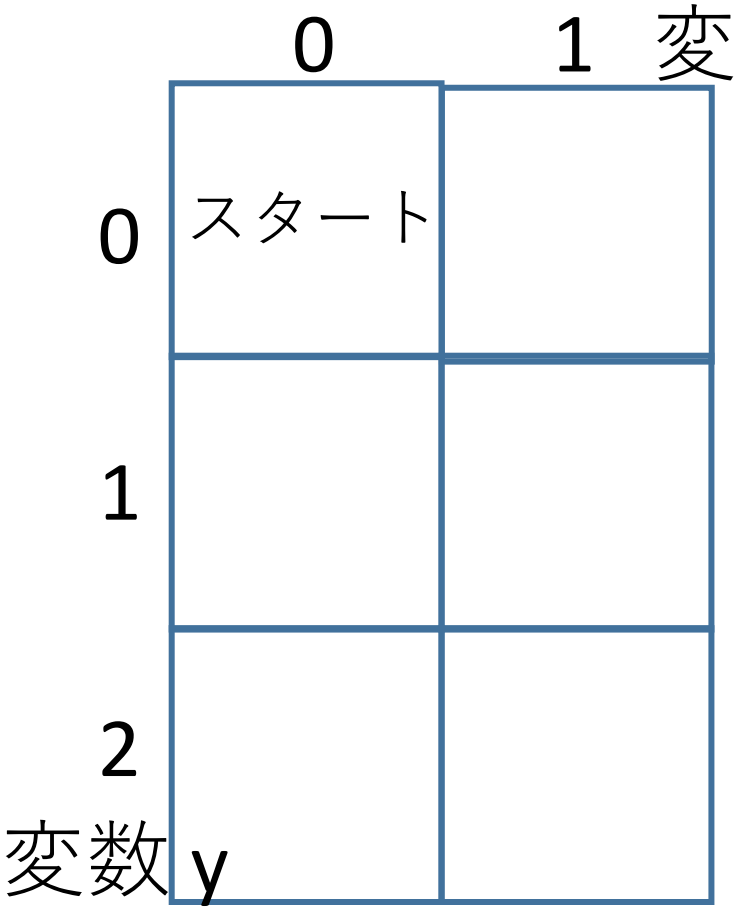
ただし、状態空間が大きい場合は、**総当たり探索は時間がかかりすぎる**。そのような場合には、A*アルゴリズムなどの、より効率的な探索手法を用いることが有効（次回授業で紹介）

10-5 総当たりを行う人工知能

はじめに

- **コンピュータ**に，**総当たり**を行わせる

遷移関数と行動番号



行動番号

1. 右へ (**x < 1 のときのみ可能**)
   x = x + 1, y = y
2. 左へ (**x > 0 のときのみ可能**)
   x = x − 1, y = y
3. 下へ (**y < 2 のときのみ可能**)
   x = x, y = y + 1
4. 上へ (**y > 0 のときのみ可能**)
   x = x, y = y - 1

選んだ行動番号を r とする

1. 右へ (**x < 1 のときのみ可能**)
x = x + 1, y = y

行動のうち１つ

```
if (r == 1) and (x < 1):
    x = x + 1
```

選んだ行動が **1** のときは,
x < 1 を調べ,
x を **x + 1** に変化
※ y は変化しないので,
y についてのプログラムは不要

プログラムの
ソースコード

演習

コンピュータに総当たりを行わせる。ページ４０まで。

【トピックス】

- **trinketでのプログラム実行**
- **遷移関数**
- **総当たり**

trinket

- **Trinket** は**オンライン**の **Python、HTML** 等の**学習サイト**
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

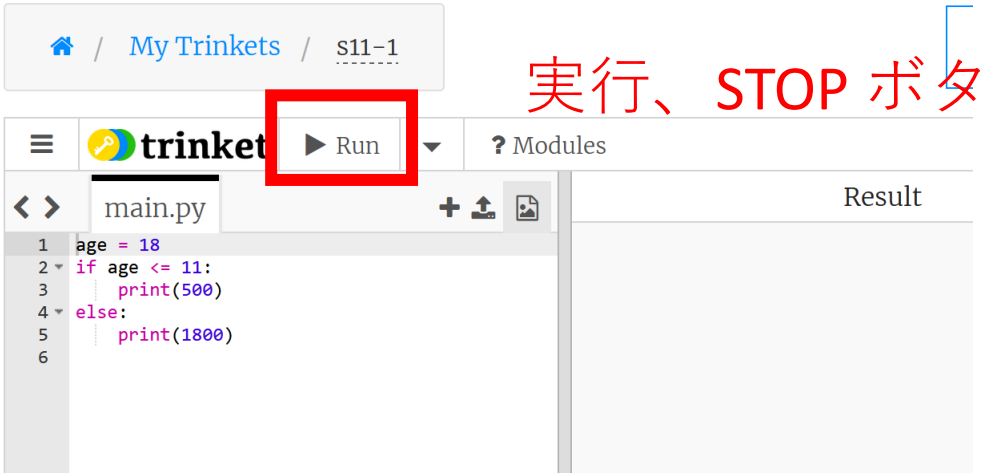
- **Python の標準機能**を登載、その他、次のパッケージがインストール済み

math, matplotlib.pyplot, numpy, operator, processing, pygal, random, re, string, time, turtle, urllib.request

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/2955caf0c8

のように、違うプログラムには違う URL が割り当てられる



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

ソースコード

https://trinket.io/python/2955caf0c8

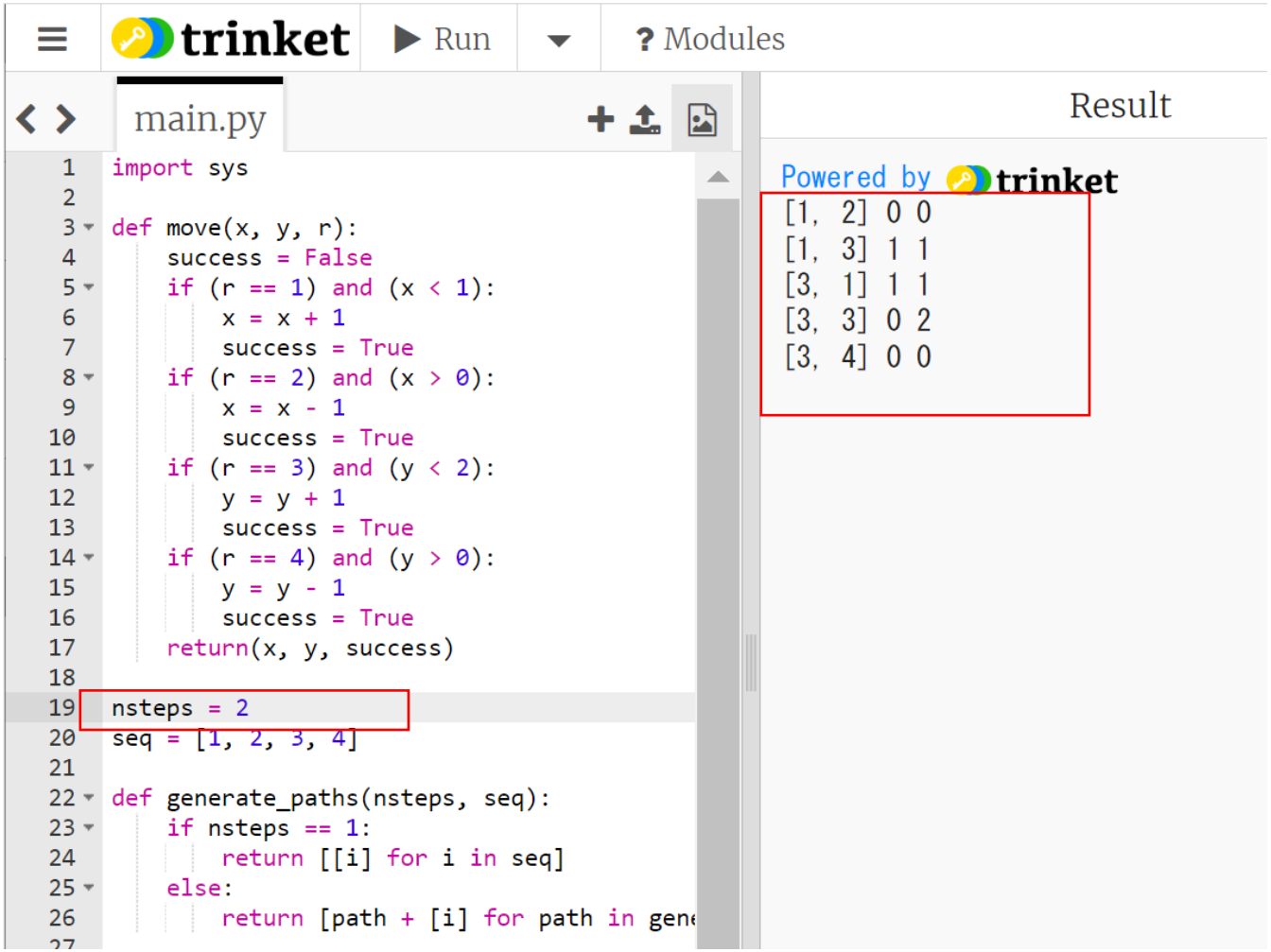
```python
import sys

def move(x, y, r):
    success = False
    if (r == 1) and (x < 1):
        x = x + 1
        success = True
    if (r == 2) and (x > 0):
        x = x - 1
        success = True
    if (r == 3) and (y < 2):
        y = y + 1
        success = True
    if (r == 4) and (y > 0):
        y = y - 1
        success = True
    return(x, y, success)

nsteps = 3
seq = [1, 2, 3, 4]

def generate_paths(nsteps, seq):
    if nsteps == 1:
        return [[i] for i in seq]
    else:
        return [path + [i] for path in generate_paths(nsteps-1, seq) for i in seq]

success = False
for j in generate_paths(nsteps, seq):
    x, y = 0, 0
    for i in j:
        x, y, success = move(x, y, i)
        if not(success):
            break
```

遷移関数

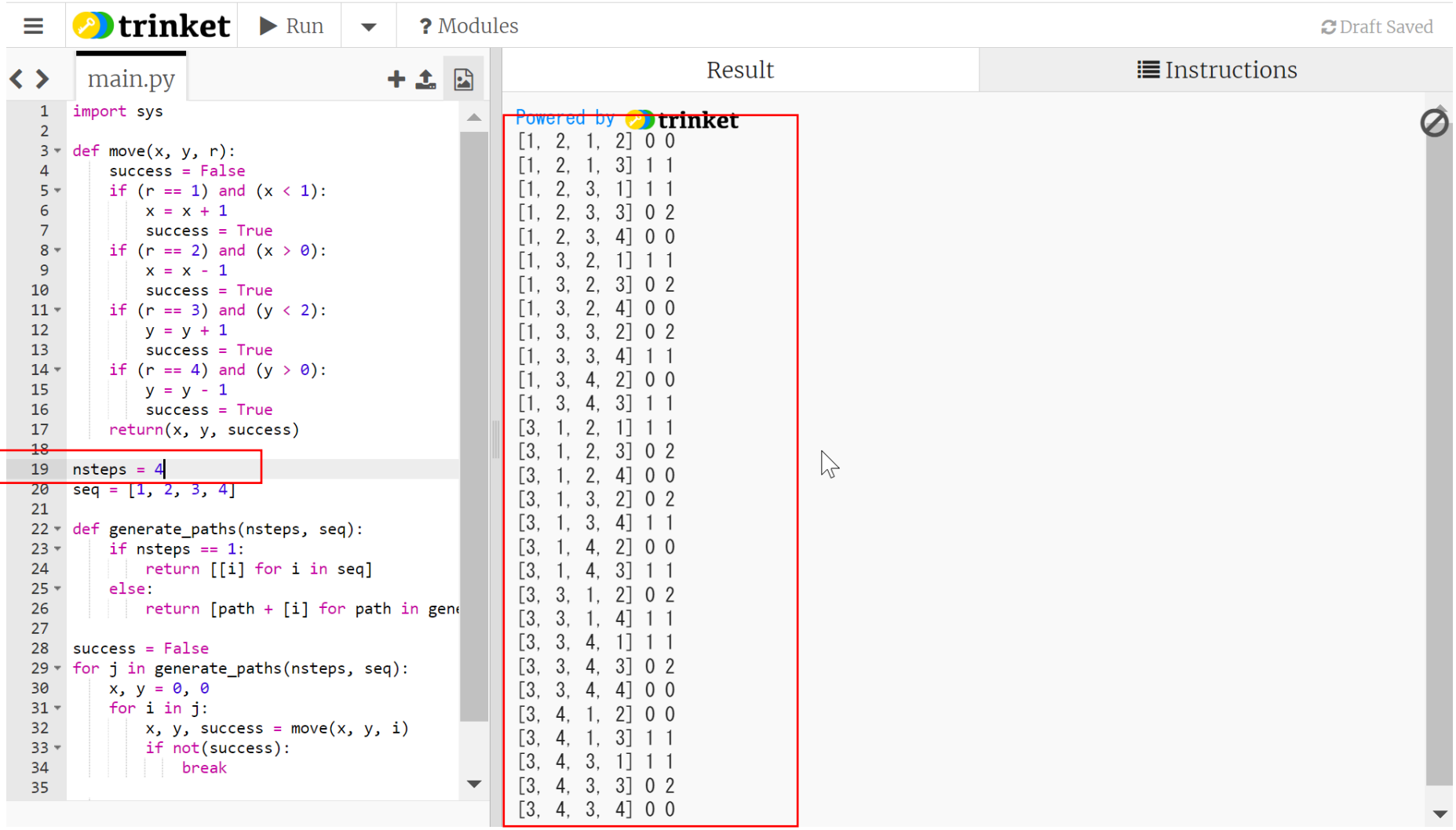
総当たりで探索するパスの
パス長は 3 に設定 nsteps = 3

最初 x, y の 0, 0
総当たり，条件に合致しない行動は総当たりの対象にしない

nsteps = 2 に書きかえて再実行



nsteps = 4 に書きかえて再実行



10-6 ２つの水差し

はじめに

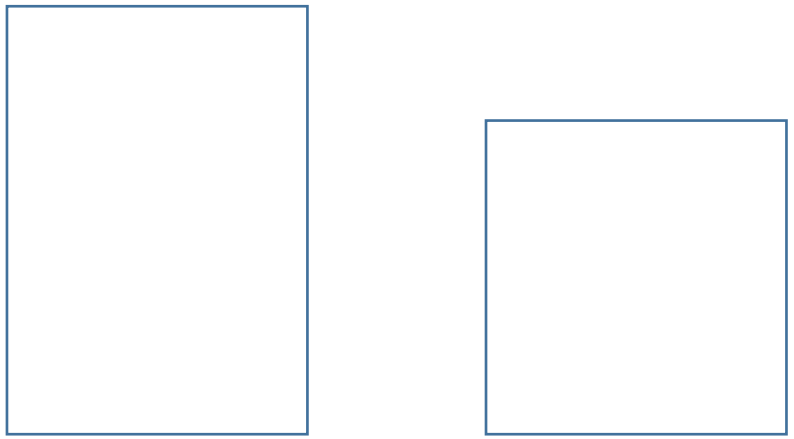
- **２つの大きさの違う「水差し」**

  大きさ**４**，大きさ**３**

- 「水の量を**２**にできるか」をコンピュータに解かせるとき，**総当たり**を行う

２つの水差し

- 水差し①　大きさ**4**
- 水差し②　大きさ**3**

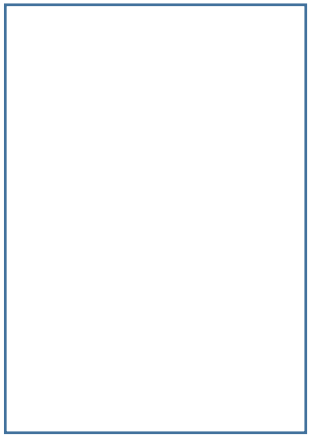


水差し①　　水差し②

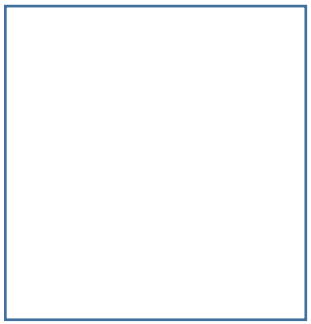
２つの変数 x, y

水差し　２つ

- 水差し①　大きさ４　・・・　水の量: **変数 x**
- 水差し②　大きさ３　・・・　水の量: **変数 y**



水差し①



水差し②

行動

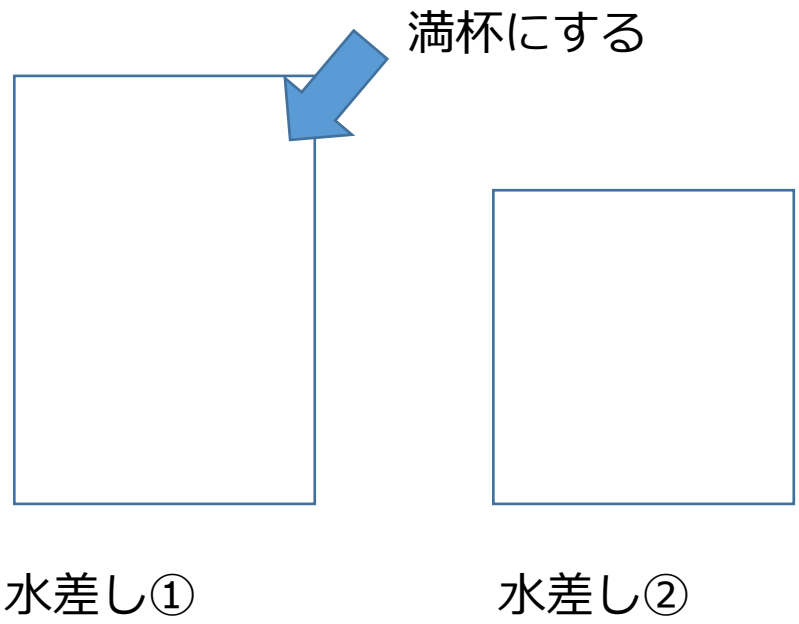
水差し①

1. 水差し①を**満杯にする**
2. 水差し①を**空にする**
3. 水差し①を**使って**，水差し②を**満杯にする**
4. 水差し①の水を**すべて**，水差し②に**入れる**

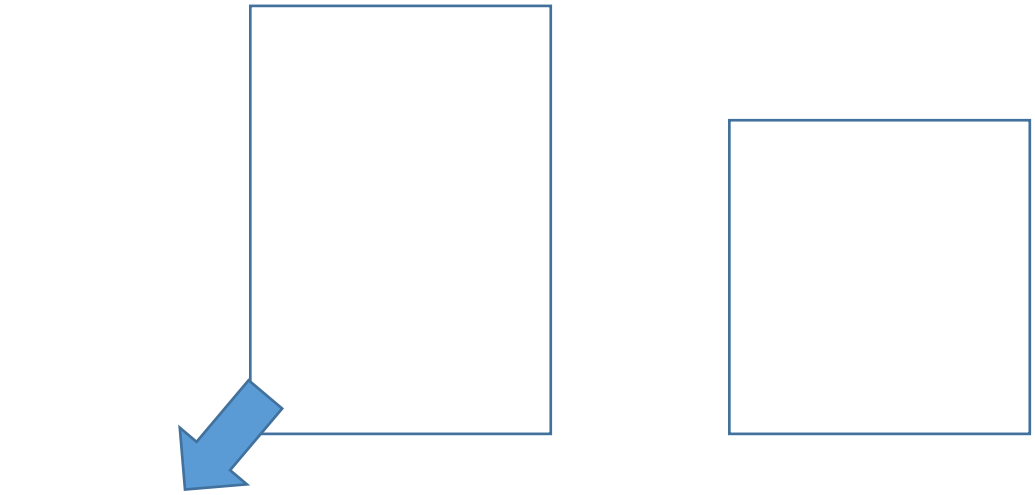
水差し②

5. 水差し②を**満杯にする**
6. 水差し②を**空にする**
7. 水差し②を**使って**，水差し①を**満杯にする**
8. 水差し②の水を**すべて**，水差し①に**入れる**

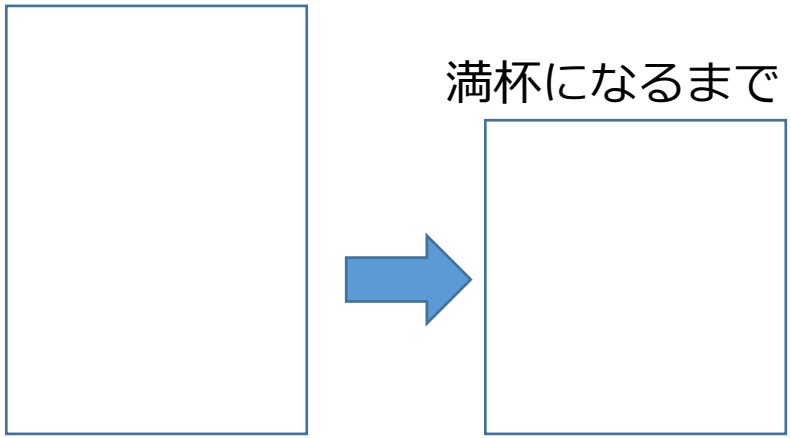
行動１　水差し①を満杯にする



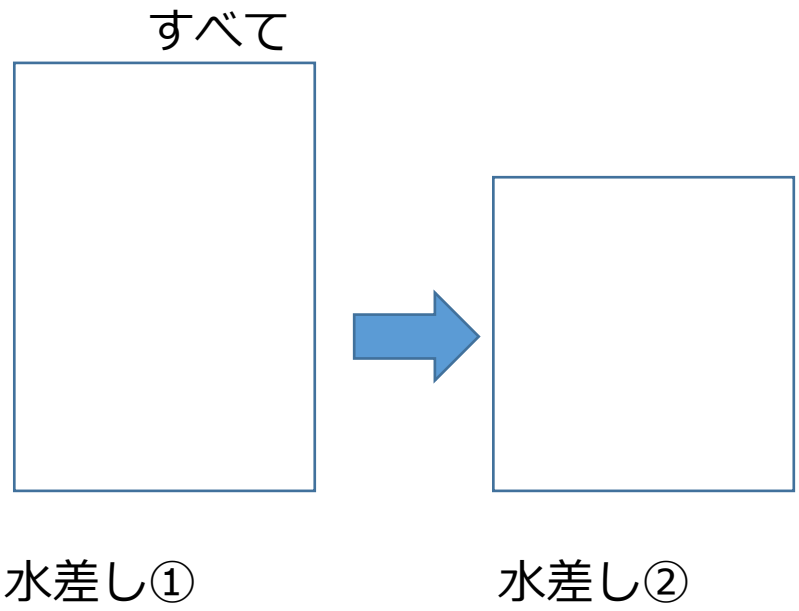
行動２　水差し①を空にする



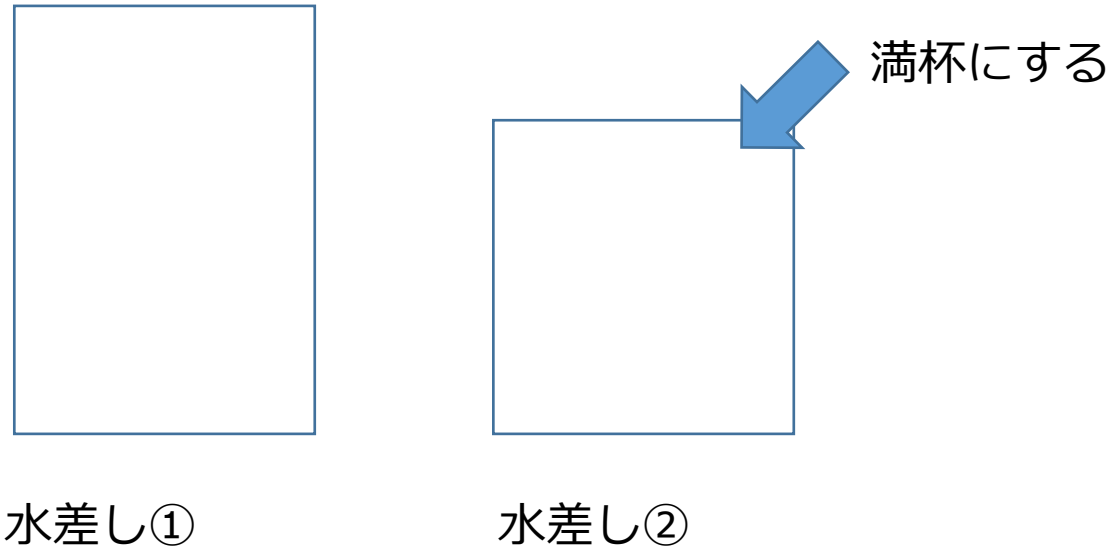
行動３　水差し①を使って，水差し②を満杯にする



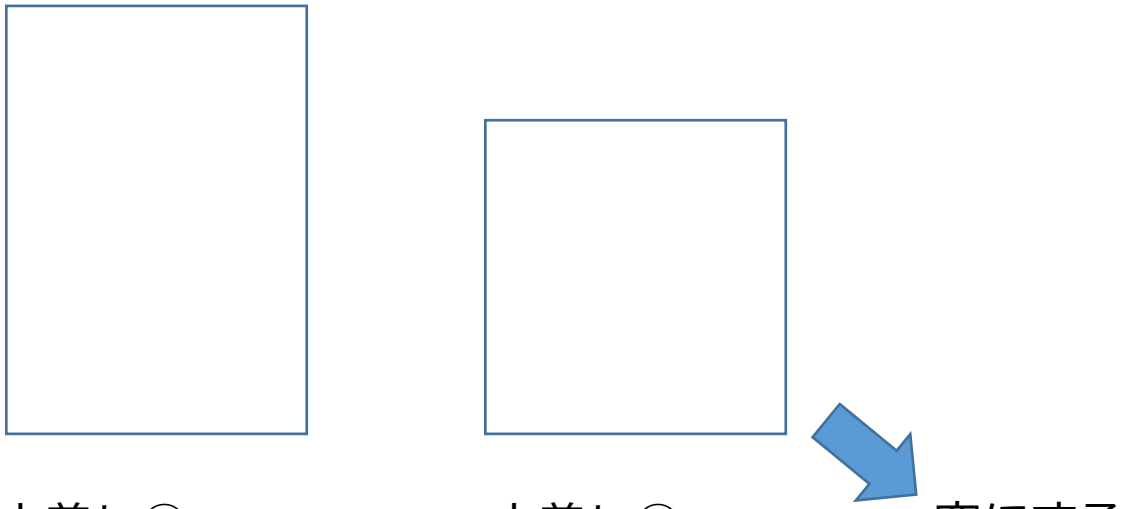
行動４　水差し①の水をすべて，水差し②に入れる



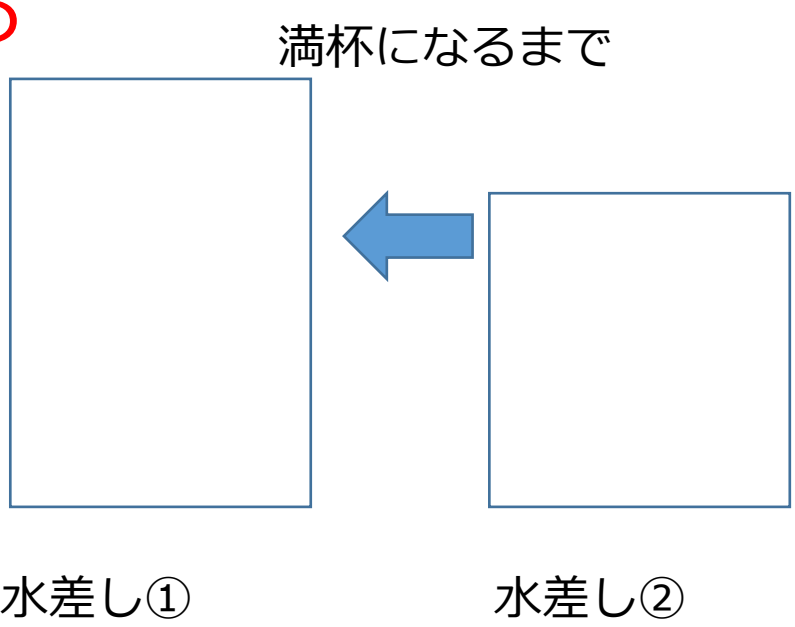
行動５　水差し②を満杯にする



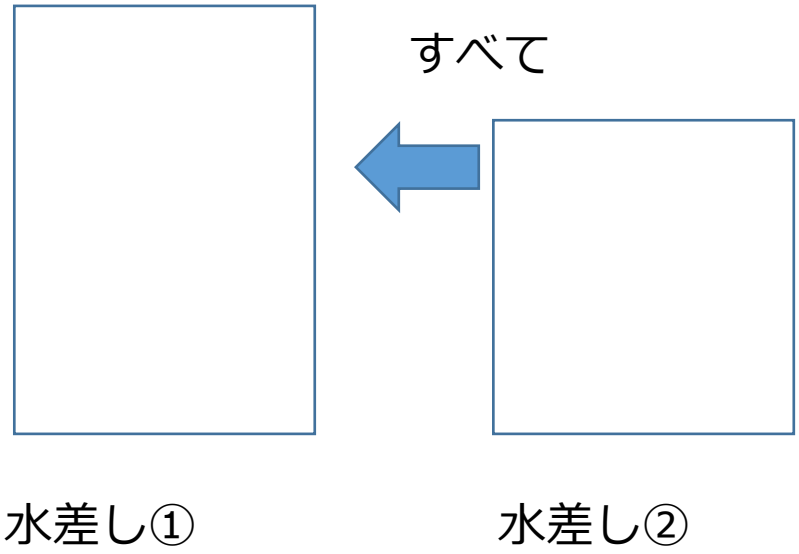
行動６　水差し②を空にする



行動７　水差し②を使って，水差し①を満杯にする



行動８　水差し②の水をすべて，水差し①に入れる



行動まとめ

水差し①

1. 水差し①を**満杯にする**
2. 水差し①を**空にする**
3. 水差し①を**使って**，水差し②を**満杯にする**
4. 水差し①の水を**すべて**，水差し②に**入れる**

水差し②

5. 水差し②を**満杯にする**
6. 水差し②を**空にする**
7. 水差し②を**使って**，水差し①を**満杯にする**
8. 水差し②の水を**すべて**，水差し①に**入れる**

行動

水差し①

1. (x, y | x < 4) → (4, y)
2. (x, y | x > 0) → (0, y)
3. (x, y | x + y >= 3 and y < 3) → (x + y - 3, 3)
4. (x, y | x + y <= 3 and x > 0) → (0, x + y)

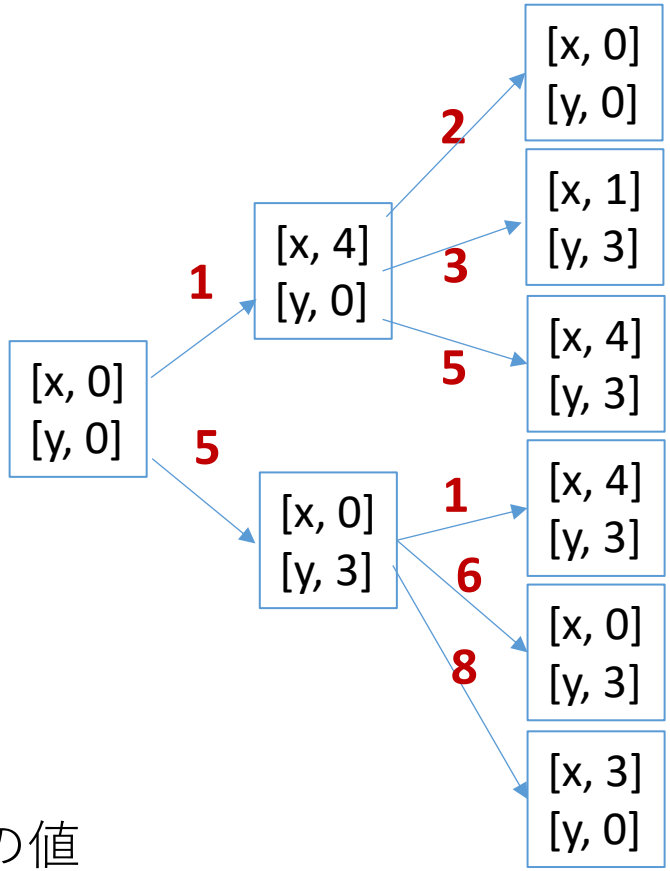
水差し②

5. (x, y | y < 3) → (x, 3)
6. (x, y | y > 0) → (x, 0)
7. (x, y | x + y >= 4 and x < 4) → (4, x + y - 4)
8. (x, y | x + y <= 4 and y > 0) → (x + y, 0)

総当たり

総当たりでは，すべての**経路**（**パス**）を試す

```
(1, 2) 0 0
(1, 3) 1 3
(1, 5) 4 3
(5, 1) 4 3
(5, 6) 0 3
(5, 8) 3 0
```



パス長２の経路（パス）をすべて試す

パス：（1，２）のように表示

右の２つ「0　0」などは x, y の最終の値

演習

コンピュータに総当たりを行わせる。ページ６２まで。

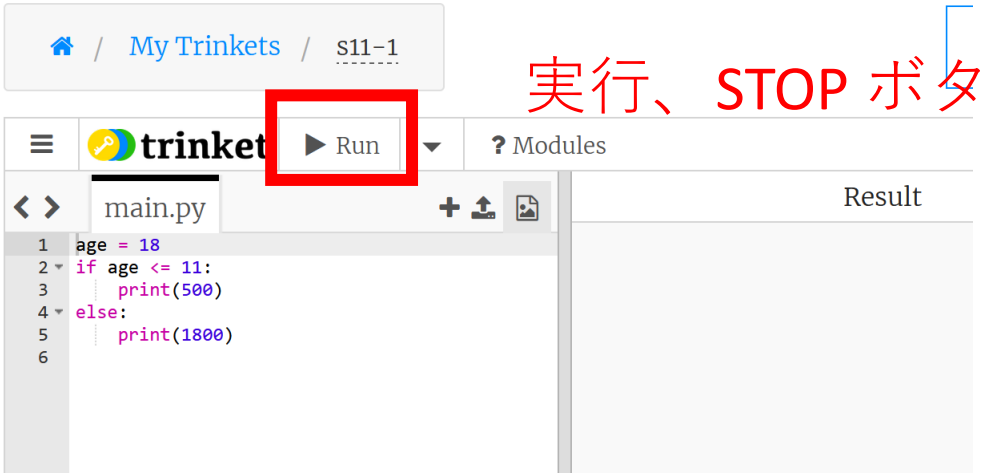
【トピックス】

- **trinketでのプログラム実行**
- **遷移関数**
- **総当たり**

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/1b2d25b99b

のように、違うプログラムには違う URL が割り当てられる



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

ソースコード

```
def move(x, y, r):
    success = False
    if (r == 1) and (x < 4):
        x = 4
        success = True
    if (r == 2) and (x > 0):
        x = 0
        success = True
    if (r == 3) and ((x + y) >= 3) and (y < 3):
        x, y = x + y - 3, 3
        success = True
    if (r == 4) and ((x + y) <= 3) and (x > 0):
        x, y = 0,  x + y
        success = True
    if (r == 5) and (y < 3):
        y = 3
        success = True
    if (r == 6) and (y > 0):
        x = 0
        success = True
    if (r == 7) and ((x + y) >= 4) and (x < 4):
        x, y = 4, x + y - 4
        success = True
    if (r == 8) and ((x + y) <= 4) and (y > 0):
        x, y = x + y, 0
        success = True

    return(x, y, success)

nsteps = 2
seq = [1, 2, 3, 4, 5, 6, 7, 8]

def generate_paths(nsteps, seq):
    if nsteps == 1:
        return [[i] for i in seq]
    else:
        return [path + [i] for path in generate_paths(nsteps-1, seq) for i in seq]

success = False
for j in generate_paths(nsteps, seq):
    x, y = 0, 0
    for i in j:
        x, y, success = move(x, y, i)
        if not(success):
            break

    if(success):
        print("%s %d %d" % (str(j), x, y))
```

総当たりによる探索を行う
部分は、前のプログラムと同じもの

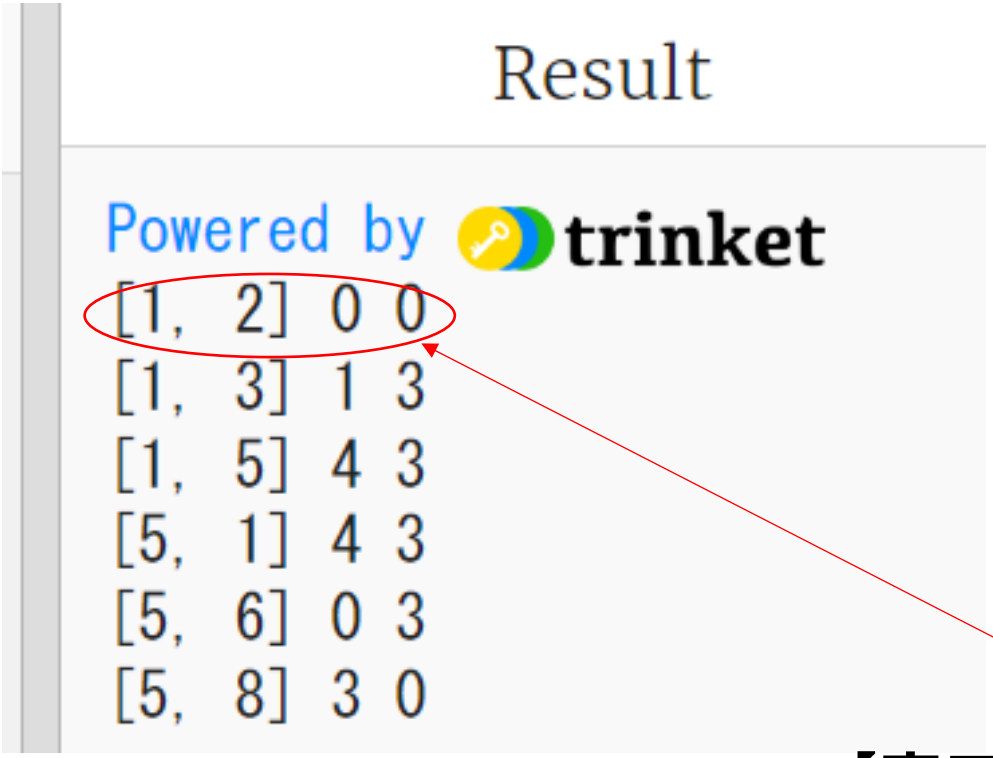
遷移関数

総当たりで探索するパスの
パス長は 3 に設定 nsteps = 3

最初 x, y の 0, 0
総当たり，条件に合致しない
行動は総当たりの対象にし
ない

実行結果から読み取ることができること

パス長 2 の**総当たり**は，6通り



実行画面

【表示の見方】
(1, 2) 0 0
行動 **1**, 行動 **2** の順で行うと
x = **0**, y = **0** になる

nsteps = 3 に書きかえて再実行

main.py

```
def move(x, y, r):
    success = False
    if (r == 1) and (x < 4):
        x = 4
        success = True
    if (r == 2) and (x > 0):
        x = 0
        success = True
    if (r == 3) and ((x + y) >= 3) and (y < 3):
        x, y = x + y - 3, 3
        success = True
    if (r == 4) and ((x + y) <= 3) and (x > 0):
        x, y = 0,  x + y
        success = True
    if (r == 5) and (y < 3):
        y = 3
        success = True
    if (r == 6) and (y > 0):
        x = 0
        success = True
    if (r == 7) and ((x + y) >= 4) and (x < 4):
        x, y = 4, x + y - 4
        success = True
    if (r == 8) and ((x + y) <= 4) and (y > 0):
        x, y = x + y, 0
        success = True

    return(x, y, success)

nsteps = 3
seq = [1, 2, 3, 4, 5, 6, 7, 8]

def generate_paths(nsteps, seq):
    if nsteps == 1:
        return [[i] for i in seq]
    else:
        return [path + [i] for path in generate_paths(nsteps-1, seq) for i in seq]
```

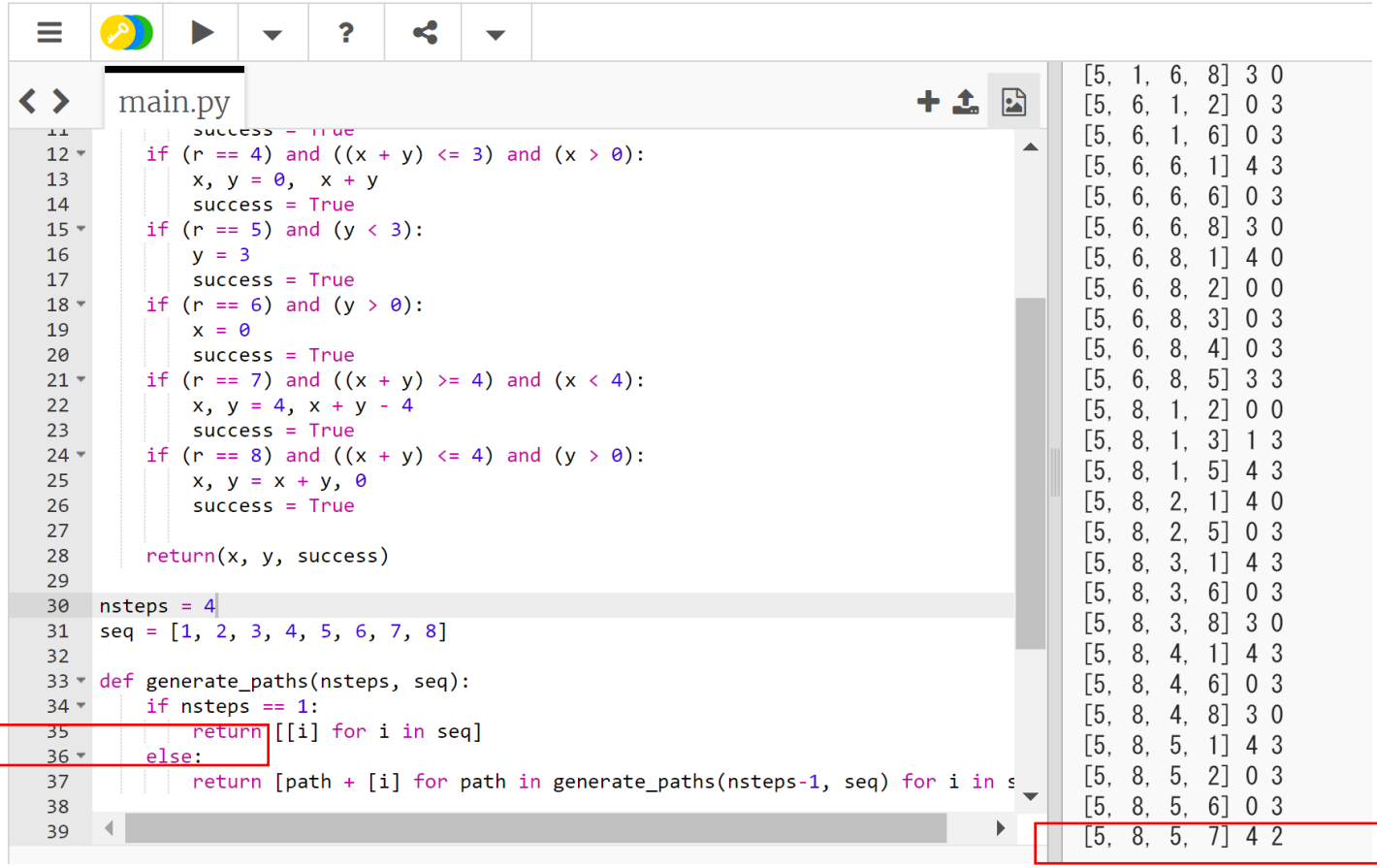
Result

Powered by trinket

```
[1, 2, 1] 4 0
[1, 2, 5] 0 3
[1, 3, 1] 4 3
[1, 3, 2] 0 3
[1, 3, 6] 0 3
[1, 3, 7] 4 0
[1, 3, 8] 4 0
[1, 5, 2] 0 3
[1, 5, 6] 0 3
[5, 1, 2] 0 3
[5, 1, 6] 0 3
[5, 6, 1] 4 3
[5, 6, 6] 0 3
[5, 6, 8] 3 0
[5, 8, 1] 4 0
[5, 8, 2] 0 0
[5, 8, 3] 0 3
[5, 8, 4] 0 3
[5, 8, 5] 3 3
```

パス長 3 の**総当たり**は，19通り

nsteps = 4 に書きかえて再実行



(5, 8, 5, 7) 4 2
行動 **5**, 行動 **8** , 行動 **5**,
行動 **7** の順で行うと
x = **4**, y = **2** になる
⇒ 水差し②は 2 になる.

まとめ

- **総当たり**では，すべての**経路**（**パス**）を試す
- 正解（例えば，「**量２の水が欲しい**」）を**発見する手段**になる
- 総当たりで正解が見つからなければ，**正解に至る経路（パス）は，存在しない**

全体まとめ

遷移関数：

- **特定の行動**を取ったときに**現在の状態**が**どのように変化するか**を定める**規則。**
- AIはこれを用いて行動の結果を**予測**し、最適な行動を**選択。**

探索と選択：

- AIは**遷移関数**を使用して可能な行動とその結果を**探索**し、その中から**最適な行動を選択する。**

総当たり：

- **全ての可能な経路**を**試す**探索手法。
- 全ての可能性を試すため、**最善の行動を見つけ出す**ことができる。

経路（パス）：

- **一連の行動の並び**。
- 問題解決の過程で用いられ、総当たりによる探索では**全ての経路（パス）が試される**。

① **遷移関数**、**探索**、**総当たり**の概念を具体例で理解。**AIの仕組みを学ぶ達成感。**

② **基礎知識の習得**で**AIエンジニアとしての能力向上。**問題解決力のアップ。

③ **身近な問題**を**AIの観点から捉える視野**が養われ、AIの可能性と限界についての理解が深まる。

④ 習得した知識は**AIの応用分野で直接活用できるもの。**