専門教育入門セミナー

http://www.cs.miyazaki-u.ac.jp/~date/lectures/2020fre

伊達 章

宮崎大学 工学部 情報システム工学科

2020 年 11 月 6 日

専門教育入門セミナー

- 担当教員： 伊達 章，A-333
  date@cs.miyazaki-u...

  date333cs → 検索

- 成績の評価方法： 発表，レポート

わけがわからない事，多すぎ！
かもしれないが，とにかく，さわって理解する

講義のスケジュール（案）

10/31 プログラミング環境の整備・インストール Python

11/6 xxx

11/xx

11/xx

11/21 発表会（Lifegame の改変，考察など）

Python 入門

ss001.py

```
1 T = 30;
2 for i in range(T):
3     print("iter␣=␣", i)
```

- 3 行目，tab で挿入したスペースは重要
- range とは？
  ↓ コマンドプロンプト から ipython

```
1 >>> range(10)
2 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3
4 >>> help(range)
```

おまじない

- 日本語文字の取り扱い
  - ファイルの先頭におまじないをつける
    # -*- coding: utf-8 -*-
- # で始まる行はコメント（例外あり）

変数の取り扱い

- 代入

```
a = 3
b = b + 1
c += 1  # c=c+1 と同じ
```

- 配列（のようなもの．リスト型という）

```
a = [3,5,6,1]
a.append(4)   # a=[3,5,6,1,4]
  # ipython で a. の後，tab で候補を表示
print(7 in a)  # False ← 7 はあるか？
print(a)

b = [1.0]*3      # b=[1.0, 1.0, 1.0] と同じ
b[2] = 5.0       # b=[1.0, 1.0, 5.0] となる
```

制御構文

- ループ（繰り返し）

```
for i in range(10):
    print(i)
    # 0 から 9 までが出力される
    # インデント（空白）が大事
```

- 条件分岐： if then else

```
if a == 0 :    # 条件の時は = が 2 つ必要!!
    print(a)
elif a == 3:
    print(a*10)
else:
    print(5)
```

ライフゲームの制作

ライフゲームの制作

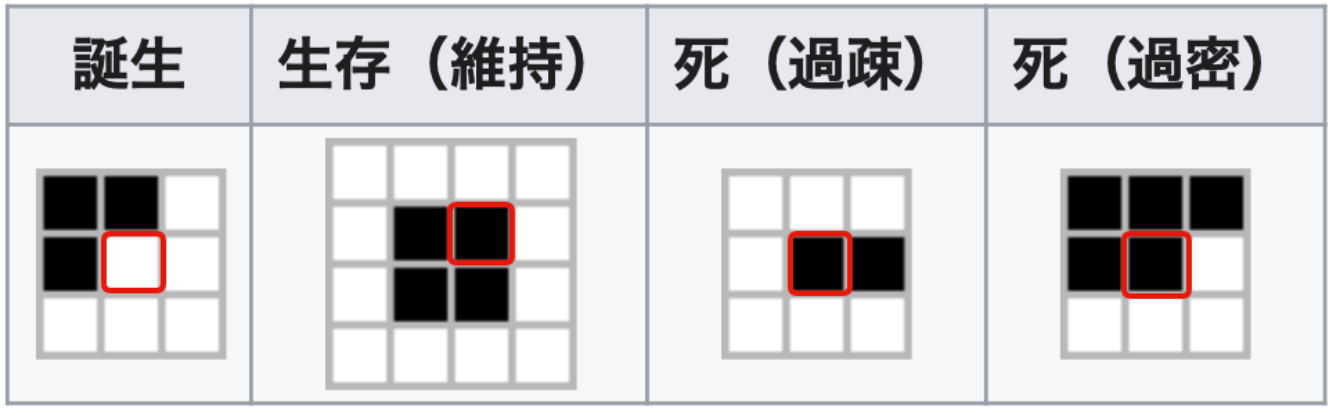
- ライフゲームとは
- ルール
- 実装
- 考察：ライフゲームの面白さ

ライフゲームとは

- Conway's Game of Life
- 生物シミュレーションのプログラム
  （生命の誕生，進化，淘汰）
- 単純なルール．しかし，複雑な動きをする．
- プログラムの行数は短い
- 歴史
  - John Horton Conway（イギリスの数学者）
  - Mathematical games: The fantastic combinations of John Conway's new solitaire game "life", Martin Gardner, Scientific American vol. 223 (October 1970), 120-123.

    https://www.ibiblio.org/lifepatterns/october1970.html

ライフゲームのルール
(Conways genetic laws)

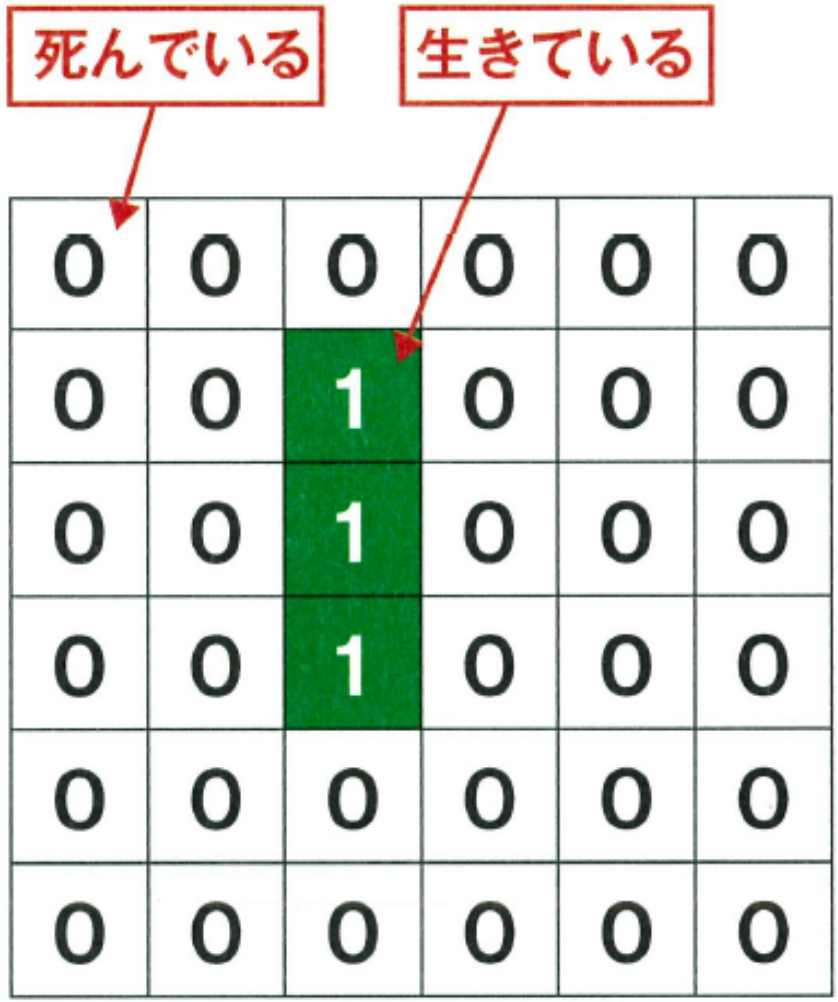


https://ja.wikipedia.org/wiki/ライフゲーム

- Births. Each empty cell adjacent to exactly three neighbors–no more, no fewer–is a birth cell. A counter is placed on it at the next move.
- Survivals. Every counter with two or three neighboring counters survives for the next generation.
- Deaths. Each counter with four or more neighbors dies (is removed) from overpopulation. Every counter with one neighbor or none dies from isolation.

https://www.ibiblio.org/lifepatterns/october1970.html

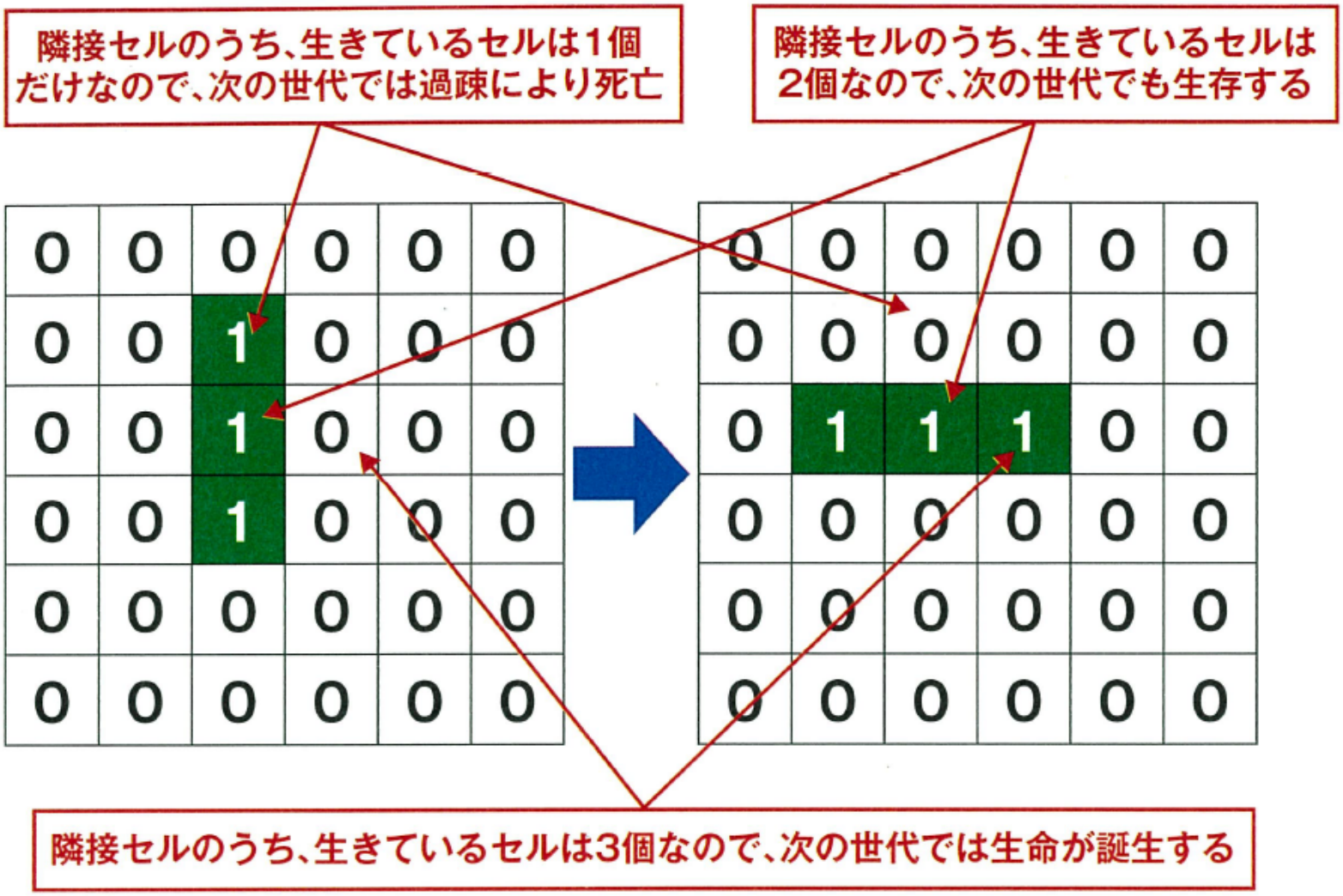
現在の状態から次の状態が決まる



ルールを適用し，次の状態を予測せよ

武部健一，Python でコスパ最高プログラミング，日経ソフトウエア（2019）

現在の状態から次の状態が決まる



武部健一，Python でコスパ最高プログラミング，日経ソフトウエア（2019）

ライフゲームの実装

- データ構造（セルの表現）
  - 現在の状態
  - 次の状態
  - 隣接関係
- アルゴリズム
  1. 初期状態の設定
  2. ルールを適用し，状態を更新.
  3. 2. に戻る

いろんな実装の仕方がある！

ライフゲームの実装（武部）

- データ構造（セルの表現）
  - 現在の状態： `cells[y][x]`
  - 次の状態： `tcells[y][x]`
  - 隣接関係： `dir`
- アルゴリズム
  1. 初期状態の設定
  2. ルールを適用し，状態を更新.
  3. 2. に戻る

2次元配列の表現

- 基本形

```
a = [0]*3          # a = [0, 0, 0]
a = [[0]*3]*2      # a = [[0, 0, 0], [0, 0, 0]]
```

- 2次元リストの生成方法

```
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

- 少しわかりにくい書き方（リスト内包表記）

```
a = [ [0]*5 for i in range(3) ]
```

- わかりやすい書き方

```
a = []
for i in range(3):
    a.append([0]*5)
```

- Python では [0,0] などのデータ構造を「リスト」とよんでいる

2次元配列の表現（タプル）

- 基本形

```
d = ( (-1,1), (-2,2) )
d*2   #  ((-1, 1), (-2, 2), (-1, 1), (-2, 2))
```

- こんな使い方をする

```
d[0]  #  (-1,1)
d[1]  #  (-2,2)
```

- 値を書き換えられない

```
d[0] = (1,3)
```

```
TypeError:  'tuple' object does not support item assignment
```

```
d = ( (-1,1), (1,3) ) # これはできる
```

リスト2●縦40×横40のマス目に緑色でお絵描きできるプログラムの「oekaki.py」

```
import pygame
from pygame.locals import *
import sys

pygame.init()
# サイズを設定して画面を生成
screen = pygame.display.set_mode((800, 800))
# 2次元リストを生成
cells = [[0] * 40 for i in range(40)]

while True: # 無限ループ
  screen.fill((255, 255, 255)) # 背景を白にする

  # 正方形を描画する2重ループ
  for y in range(0, 40):
    for x in range(0, 40):
      if cells[y][x] == 1:
        # 四角形の座標と縦横の長さを設定
        rect = Rect(x * 20, y * 20, 20, 20)
        # 緑色(0,255,0)の四角形を描く
        pygame.draw.rect(screen,(0,255,0),rect)

  # マウスからの入力を処理する
  for e in pygame.event.get():
    if e.type == MOUSEBUTTONDOWN:
      mx, my = int(e.pos[0]/20),int(e.pos[1]/20)
      cells[my][mx] = 1 - cells[my][mx]
    if e.type == QUIT:
      pygame.quit()
      sys.exit()

  pygame.display.update()
```

リスト3●ライフゲームのプログラムの「lifegame.py」

```
import pygame
from pygame.locals import *
import sys

# ルールを適用して次世代を作成する関数
def rule(cells):
  # 隣接セルにアクセスするための方向を用意
  dir = ((-1, -1), ( 0, -1), ( 1, -1),
         (-1,  0),           ( 1,  0),
         (-1,  1), ( 0,  1), ( 1,  1))

  # 次世代の状態を保持する2次元リストを生成
  tcells = [[0] * 40 for i in range(40)]

  # 周縁部を除く全セルにルールを適用する2重ループ
  for y in range(1, 40 - 1):
    for x in range(1, 40 - 1):
      c = 0
      for d in dir: # 生きている隣接セルの個数を数える
        if cells[y + d[1]][x + d[0]] == 1: c += 1
      if cells[y][x] == 0 and c == 3: # ルール1を適用
        tcells[y][x] = 1
      if cells[y][x] == 1:
        if c == 2 or c == 3: # ルール2を適用
          tcells[y][x] = 1
        else:
          tcells[y][x] = 0 # ルール3を適用

  return tcells # 次世代の2次元リストを返す

pygame.init()
# サイズを設定して画面を生成
screen = pygame.display.set_mode((800, 800))
font = pygame.font.Font(None, 28)
# セルの状態を保持する2次元リストを生成
cells = [[0] * 40 for i in range(40)]
run = 0 # 実行フラグ
gen = 1 # 世代数

while True: # シミュレーションを実行する無限ループ
  screen.fill((255, 255, 255)) # 背景を白にする

  if run == 1:
    gen += 1
    cells = rule(cells)
    pygame.time.wait(250) # 250ミリ秒間プログラムを停止

  # セルを描画する2重ループ
  for y in range(0, 40):
```

次ページに続く

リスト3の続き

```
    for x in range(0, 40):
      if cells[y][x] == 1:
        rect = Rect(x * 20, y * 20, 20, 20)
        pygame.draw.rect(screen, (0, 255, 0), rect)

# マウスからの入力を処理する
for e in pygame.event.get():
  if e.type==MOUSEBUTTONDOWN and e.button==1 and run==0:
    mx, my = int(e.pos[0] / 20), int(e.pos[1] / 20)
    if mx == 0 or mx == 40-1 or my == 0 or my == 40-1:
      cells[my][mx] = 0 # 周縁部のセルは常に0にする
    else:
      # マウスクリックしたセルが0なら1に、1なら0にする
      cells[my][mx] = 1 - cells[my][mx]
  if e.type == MOUSEBUTTONDOWN and e.button == 2:
    run = 1 - run # 実行フラグをフリップ
  if e.type==MOUSEBUTTONDOWN and e.button==3 and run==0:
    # シミュレーションをリセットする
    cells = [[0] * 40 for i in range(40)]
    gen = 1
  if e.type == QUIT:
    pygame.quit()
    sys.exit()

s1 = "running" if run == 1 else "setting"
s2 = "generation : " + str(gen)
text = font.render(s1 + "    " + s2, True, (0, 0, 0))
screen.blit(text, (1, 1)) # 画面に文字列を描く
pygame.display.update()
```

終