Web 付録　第 4 章　Python プログラム

プログラム 1　1 対 1 マッチングにおける受入保留方式

以下のプログラムは，この受入保留方式に従ってマッチングを決定するプログラムです。最初に女性と男性の数を設定します。

次に，女性と男性の希望順位をそれぞれ 2 次元リスト w_pref, m_pref として入力します。ここでは，**例 4.5** の場合にしています。リストの上の行から順に女性w_1, w_2, w_3の希望順位となっています。例えば，1 行目の[1,2]は女性w_1,にとってm_1が 1 位，m_2が 2 位という意味です。次に，確認のために，女性と男性の希望順位を画面表示します。

それから，マッチング結果を求める処理の便宜上，男性の希望順位を左から順に女性w_1, w_2, w_3に対する順位になるように並べ替え，それを 2 次元リスト m_order に格納します。リストの上の行から順に女性m_1, m_2の希望順位となります。

この後，受入保留方式に従ってマッチングを決めていきます。while 文を使って，仮マッチした女性の数 num_match が W になるまで繰り返し処理をしています。なお，この num_match という変数は，女性が誰かと仮マッチするたびに 1 ずつ増え，また，希望順位にある男性すべてにプロポーズしてしまった場合も 1 増えるようになっています。

仮マッチを決める際は，まだ誰とも仮マッチしていない女性は，その希望順位の上位の男性でまだプロポーズしていない男性にプロポーズし，(1) その男性がまだ誰とも仮マッチしていなければ仮マッチ，(2) もし誰かとマッチしていて，その女性の方がすでに仮マッチしている女性より順位が高い場合には仮マッチ相手を入れ替え，(3) それ以外の場合は仮マッチせず，このステップを終了します。男女それぞれの仮マッチ状況はリスト w_filled, m_filled に記録します。このリストの要素が 0 ならばまだ誰ともマッチしておらず，1 ならば誰かと仮マッチしています。男女それぞれが仮マッチした相手はリスト w_matched, m_matched に格納されます。また，女性がすでに何位の男性にまでプロポーズしていたかはリスト position に記録されます。プログラムではその都度，誰が誰にプロポーズし，その結果がどうだったのかを順に画面出力しています。最後に，マッチング結果をまとめて画面出力しています。

なお，プログラムの画面表示では本文の例に合わせて男女の番号を 1, 2, 3,……としていますが，Python では基本的に変数の値は 0, 1, 2,……と 0 から始まりますので，番号を表す変数である i, j に適宜 1 を足したり引いたりしていることに注意してください。

```
# DA mechanism
print('受入保留方式によるマッチング')
print()
```

```
# 女性の数
W=3

# 男性の数
M=2

# 女性の選好
w_pref=[
        [1,2],
        [2,1],
        [1,2]
]
for i in range(W):
    print('女性',i+1,'の希望順位：',end='')
    for j in range(M):
        print(' m{0}'.format(w_pref[i][j]),end='')
    print()
print()

# 男性の選好
m_pref=[
        [2,1,3],
        [1,2,3],
]
for i in range(M):
    print('男性',i+1,'の希望順位：',end='')
    for j in range(W):
        print(' w{0}'.format(m_pref[i][j]),end='')
    print()
print()

# 男性の希望順位を左から w1 の順位，w2 の順位，……に変換する
m_order = [[0]*W for i in range(M)]
for i in range(M):
    for j in range(W):
        k = m_pref[i][j]
```

```
            m_order[i][k-1] = j+1

# マッチした相手を格納するリスト
w_matched=[0]*W
m_matched=[0]*M

# 受入保留方式でマッチングを決める

# 仮マッチしている女性の数
num_match = 0

# 仮マッチした相手がいれば 1，そうでなければ 0
w_filled=[0]*W
m_filled=[0]*M

# 第何希望まですでにプロポーズしたか
position = [0]*W

# ステップ数
t=1
while num_match < W:
    print('ステップ {}'.format(t))
    for i in range(W):
        # 女性がまだ誰ともマッチしていないなら
        if w_filled[i]==0:
            # 女性がプロポーズする相手
            j = w_pref[i][position[i]]-1
            # プロポーズした男性の現在の仮マッチ
            k = m_matched[j]
            print('w{0}が m{1}にプロポーズ'.format(i+1,j+1))
            # 男性がまだ誰ともマッチしない場合
            if m_filled[j]==0:
                # i と j がマッチ
                m_matched[j]=i
                w_matched[i]=j
                print('  w{0}と m{1}が仮マッチ'.format(i+1,j+1))
```

```
                w_filled[i]=1
                m_filled[j]=1
                num_match +=1
            # 男性がすでに誰かとマッチしている場合
            elif m_order[j][i] < m_order[j][k]:
                w_filled[k]=0
                order[k]+=1
                print('  m{0}が w{1}をリジェクト'.format(j+1,k+1))
                m_matched[j]=i
                w_matched[i]=j
                print('  w{0}と m{1}が仮マッチ'.format(i+1,j+1))
                w_filled[i]=1
                m_filled[j]=1
            else:
                print('  m{0}が w{1}をリジェクト'.format(j+1,i+1))
                position[i]+=1
                # すべての男性にプロポーズしたらアンマッチ
                if position[i]==M:
                    w_matched[i]=-1
                    w_filled[i]=1
                    num_match +=1
            print()
    t+=1
    print()

# マッチング結果の印字
print('マッチング結果')
for i in range(W):
    if w_matched[i]==-1:
        print('w{0}:'.format(i+1))
    else:
        print('w{0}: m{1}'.format(i+1,w_matched[i]+1))

for j in range(M):
    if m_filled[j]==0:
        print('   : m{0}'.format(j+1))
```

このプログラムを実行すると，以下のような結果が出力されます。**例 4.5** の結果と一致していることを確認してみてください。

```
受入保留方式によるマッチング

女性 1 の希望順位： m1 m2
女性 2 の希望順位： m2 m1
女性 3 の希望順位： m1 m2

男性 1 の希望順位： w2 w1 w3
男性 2 の希望順位： w1 w2 w3

ステップ 1
w1 が m1 にプロポーズ
  w1 と m1 が仮マッチ

w2 が m2 にプロポーズ
  w2 と m2 が仮マッチ

w3 が m1 にプロポーズ
  m1 が w3 をリジェクト


ステップ 2
w3 が m2 にプロポーズ
  m2 が w3 をリジェクト


マッチング結果
w1: m1
w2: m2
w3:
```

プログラム2　TTC **方式**

以下のプログラムは，TTC 方式に従ってマッチングを決めるものです。最初に学生と研究室の数を設定します。

次に，学生と研究室の希望順位をそれぞれ 2 次元リスト s_pref, c_pref として入力します。ここでは，**例 4.8** の場合にしています。リストの上の行から順に学生s_1, s_2, s_3, s_4の希望順位となっています。次に，確認のために，学生と研究室の希望順位を画面表示します。各研究室の定員はリスト capacity に設定します。左から順位に研究室c_1, c_2, c_3の定員となります。

この後，TTC 方式に従ってマッチングを決めていきます。生徒全員の配属が決まるまで while 文を使って，以下のような処理を繰り返します。まず，サイクルができたかどうかを確認するために，指差しをした生徒と学校を交互にリスト cycle に記録していきます。具体的には，まず生徒 x は第 1 希望の学校 k を指差し，リスト cycle に生徒 x を記録します。指差された学校 k の定員に空きがなければ，生徒 x はこの時点ではサイクルに入れなかったので，次の生徒が指差しをします。指差された学校 k の定員に空きがあれば，学校 k はまだ誰ともマッチしていない生徒の中で優先順位が 1 番高い生徒 y を指差し，リスト cycle に学校 k を記録します。ここで，(1) 学校 k が指差した生徒 y がすでにマッチしていれば，生徒 x はこの時点ではサイクルに入れなかったので，次の生徒が指差しをすることになり，ここでループを抜けます。(2) 学校 k が指差した生徒 y が実は生徒 x だった場合，ここでサイクルができたことになり，ここでループを抜けます。各生徒は指差した学校にマッチすることになります。(3) 学校 k が指差した生徒 y がまだ誰ともマッチしていない場合，今度は生徒 y が指差しをする番になります。リスト cycle に生徒 y を記録し，生徒 y を改めて生徒 x として while 文の先頭に戻ります。すべての生徒を検討し終えたら，次はアンマッチの学生について同様に検討していきます。

学生と研究室それぞれのマッチ状況はリスト s_filled, c_filled に記録します。このリストの要素が 0 ならばまだマッチしておらず，1 以上ならばマッチしています。生徒と研究室それぞれがマッチした相手はリスト s_matched, c_matched に格納されます。また，生徒がすでに何位の学校にまで指差していたかはリスト position に記録されます。なお，生徒がある時点でサイクルに入れなかったとしても，指差した学校の定員に空きがあれば，次の時点で再び指差しできるので，そのような場合には position を 1 つ戻すことにしています。プログラムではその都度，誰がどの学校を指差し，その結果がどのようなサイクルができたのかを順に画面出力しています。最後に，マッチング結果をまとめて画面出力しています。

謝辞：本プログラム作成に際しては，大阪大学の安田洋祐先生にも検討に加わっていただき，アドバイスをいただきました。

```
# TTC 方式
print('TTC 方式によるマッチング')
print()

# 学生の数
S=3

# 研究室の数
C=3

# 生徒の選好，左から c1 の順位，c2 の順位，……
s_pref=[
        [2,1,3],
        [1,2,3],
        [1,2,3]
]
for i in range(S):
    print('生徒',i+1,'の希望順位：',end='')
    for j in range(C):
        print(' c{0}'.format(s_pref[i][j]),end='')
    print()
print()

# 学校の優先順位，左から s1 の順位，s2 の順位，……
c_pref=[
        [1,3,2],
        [2,1,3],
        [2,1,3]
]
for i in range(C):
    print('学校',i+1,'の優先順位：',end='')
    for j in range(S):
        print(' s{0}'.format(c_pref[i][j]),end='')
    print()
print()
```

```
# 研究室の定員
capacity=[1,1,1]
for i in range(C):
    print('学校',i+1,'の定員：',end='')
    print(capacity[i])
print()

# マッチした相手
s_matched=[-1]*S # アンマッチなら-1
c_matched=[[0]*S for i in range(C)]

# 学生の選好を第 1 希望から順に並べ替える
s_propose=[[0]*C for i in range(S)]
for i in range(S):
    for j in range(C):
        k = s_pref[i][j]
        s_propose[i][k-1]=j

# マッチしている学生の数
num_match = 0

# 学生にマッチした相手がいれば 1，そうでなければ 0
s_filled=[0]*S

# すでに学校に割り当てられている人数
c_filled=[0]*C

# 生徒が第何希望まですでに指差したか
position = [0]*(C+1)

# TTC 方式でマッチングを決める

# すべての生徒がマッチするまで繰り返す
t=1
while num_match < S:
    print('ステップ{0}'.format(t))
```

```
# サイクル候補
cycle = [0]*(S+C)

# 生徒が順に指差していく
for i in range(S):
    # 生徒がまだ誰ともマッチしていないなら
    if s_filled[i]==0:

        # この生徒を含むサイクルが形成されるかのフラグ
        flag = 0

        # サイクルの長さ
        num = 0

        # 最初に指差す学生
        x = i

        # ループ
        while flag==0:
            # 学生自身をサイクル候補に入れる
            cycle[num]=x

            # 学生がプロポーズする学校
            k = s_propose[x][position[x]]
            print('s{0}が c{1}を指差す'.format(x+1,k+1))

            # 学校 k の定員がすでに埋まっていたらループを抜ける
            if c_filled[k]==capacity[k]:
                print('c{0}の定員が埋まっている'.format(k+1))
                flag = 1
            else:
                # 学校 k をサイクル候補に入れる
                num += 1
                cycle[num]=k+S
```

```
                    # まだ配属されていない学生の中から一番順位の高い生徒 y を選ぶ
                    m = 0
                    y = c_pref[k][m]-1
                    while s_filled[y]==1:
                        m += 1
                        y = c_pref[k][m]-1

                    # もし学校が生徒 x を指差していたらサイクル形成
                    if y in cycle:
                        print('c{0}が s{1}を指差す'.format(k+1,y+1))
                        print()
                        print('サイクル形成')
                        flag = 2
                    else:
                        print('c{0}が s{1}を指差す'.format(k+1,y+1),end='')
                        # 学校 k が指差した生徒 y を x にしてループに戻る
                        x = y
                        num += 1

                # サイクルができた場合
                if flag == 2:
                    # サイクルの最後尾の学校が指差した生徒の cycle 内のインデックス
                    z = cycle.index(y)
                    for l in range(z,num+1,2):
                        print('s{0}  →  c{1}  →  '.format(cycle[l]+1,cycle[l+1]-
(S-1)),end='')
                        s_matched[cycle[l]]=cycle[l+1]-S
                        s_filled[cycle[l]]=1
                        num_match += 1
                        c_filled[cycle[l+1]-S]+=1
                    print('s{}'.format(cycle[z]+1))
                print()

    # マッチできなかった生徒は次の希望順位に応募
    for i in range(S):
        if s_filled[i]==0:
```

```
                position[i] += 1
            # すべての学校にプロポーズしたらアンマッチ
            if position[i]==C:
                s_matched[i]=-1
                s_filled[i]=1
                num_match +=1

    t += 1
    print()

# マッチング結果の印字
print('マッチング結果')
for i in range(S):
    if s_matched[i]==-1:
        print('s{0}:'.format(i+1))
    else:
        print('s{0}: c{1}'.format(i+1,s_matched[i]+1))

for j in range(C):
    if c_filled[j]==0:
        print('   : c{0}'.format(j+1))
```

このプログラムを実行すると，以下のような結果が出力されます。**例 4.8** の結果と一致していることを確認してみてください。

```
TTC 方式によるマッチング

生徒 1 の希望順位： c2 c1 c3
生徒 2 の希望順位： c1 c2 c3
生徒 3 の希望順位： c1 c2 c3

学校 1 の優先順位： s1 s3 s2
学校 2 の優先順位： s2 s1 s3
学校 3 の優先順位： s2 s1 s3

学校 1 の定員：1
```

学校 2 の定員：1
学校 3 の定員：1

ステップ 1
s1 が c2 を指差す
c2 が s2 を指差す
s2 が c1 を指差す
c1 が s1 を指差す

サイクル形成
s1 → c2 → s2 → c1 → s1

s3 が c1 を指差す
c1 の定員が埋まっている

ステップ 2
s3 が c2 を指差す
c2 の定員が埋まっている

ステップ 3
s3 が c3 を指差す
c3 が s3 を指差す

サイクル形成
s3 → c3 → s3

マッチング結果
s1: c2
s2: c1
s3: c3