Zagadnienia przedzialu - algorytm wegierski

Dominik Matracki
Patryk Lyczko
Michal Rola

In [3]:
```
import numpy as np
```

In [4]:
```
mat = np.array([[0,np.nan,1,np.nan,6],
                [1,6,2,0,3],
                [0,1,0,4,0],
                [3,9,0,4,np.nan],
                [0,0,1,3,np.nan]])

mat_gl = np.array([[5,2,3,2,7],
                   [6,8,4,2,5],
                   [6,4,3,7,2],
                   [6,9,0,4,0],
                   [4,1,2,4,0]])
```

In [5]:
```
def reduce_matrix(M, inplace=False):
    """Reduces rows and columns"""
    if inplace:
        result = M
    else:
        result = M.copy()

    for row in result:
        row -= min(row)

    result -= np.min(result, axis=0)
    if not inplace: return result
```

In [6]:
```
# Redukcja macierzy
mat_n = reduce_matrix(mat_gl, inplace=False)
mat_n
```

Out[6]:
```
array([[0, 0, 1, 0, 5],
       [1, 6, 2, 0, 3],
       [1, 2, 1, 5, 0],
       [3, 9, 0, 4, 0],
       [1, 1, 2, 4, 0]])
```

In [7]:
```
def independent_zeros(matrix):
    """Choose independent zeros and fill in dependent with -1"""
    a, b = matrix.shape
    row_free = [i for i in range(a)]
    col_free = [i for i in range(b)]
    result = matrix.copy()
    pos = []

    for i in range(a):
        for j in range(b):
            if result[i][j] == 0:
                if i in row_free and j in col_free:
                    pos.append((i,j))
                    row_free.remove(i)
                    col_free.remove(j)
                else:
```

```
                    result[i][j] = -1
    return result, pos
```

In [8]:
```
mat, pos = independent_zeros(mat_n)
mat, pos
```

Out[8]:
```
(array([[ 0, -1,  1, -1,  5],
        [ 1,  6,  2,  0,  3],
        [ 1,  2,  1,  5,  0],
        [ 3,  9,  0,  4, -1],
        [ 1,  1,  2,  4, -1]]),
 [(0, 0), (1, 3), (2, 4), (3, 2)])
```

In [9]:
```
def find_best_pos(n, pos):
    """Szukamy najlepszej kombinacji zer niezaleznych"""
    pos_new = []

    while len(pos) != 0:
        dic_pos_x = {}

        # Wpisujesz wszystkie wiersze
        for i in range(n):
            dic_pos_x[i] = 0

        # Zaznaczasz te ktore maja zera
        for i,j in pos:
            dic_pos_x[i] += 1

        unique_row = []

        for key,val in dic_pos_x.items():
            if val == 1:
                unique_row.append(key)

        ## add to return this rows
        col_to_remove = []
        for i,j in pos:
            if i in unique_row:
                pos_new.append((i,j))
                col_to_remove.append(j)

        ## remove from pos this col
        acc = 0
        while True:
            i,j = pos[acc]
            if j in col_to_remove:
                pos.remove((i,j))
            else:
                acc += 1
            if acc == len(pos):
                break
        # print(pos)
    return pos_new
```

In [10]:
```
### Rodzaj danych
## lista tupli

arr = [(1,1), (2,4), (3,5),(4,3)]


def cross_zeros(matrix, lis_of_tuple):
    """Funkcja wykreslajaca"""
    a,b = matrix.shape
    rows = [i for i,j in lis_of_tuple]
    cols = [j for i,j in lis_of_tuple]
```

```
    row_0free = []
    for i in range(a):
        if i not in rows:
            row_0free.append(i)

    col_0free = []
    res_col = np.where(mat == -1)
    for i in res_col:
        col_0free.append(i[1])

    col_0plus = []
    res_col = np.where(mat == 0 )
    for i in res_col:
        col_0plus.append(i[1])

    matx = mat.copy()
    z = 0
    for i in row_0free:
        matx = np.delete(matx, i - z, 1)
        z += 1

    # print(row_0free)
    z = 0
    row = []
    for i in col_0free:
        row.append(i)

    for i in col_0plus:
        row.append(i)

    row.sort()
    for i in range(len(row) - 1):
        if row[i] == row[i + 1]:
            row.pop(i)
            i -= 1
    # print(row)
    z = 0
    for i in row:
        matx = np.delete(matx, i - z, 0)
        z += 1

    # print(row)
    # print(row_0free)
    if len(row) + len(row_0free) < mat.shape[0]:
        dic_row = {}
        dic_col = {}
        acc1 = 0
        acc2 = 0
        for i in range(a):
            if i not in row:
                dic_row[acc1] = i
                acc1 += 1
            if i not in row_0free:
                dic_col[acc2] = i
                acc2 += 1

        # print(dic_col)
        # print(dic_row)
        mat_del = reduce_matrix(matx)
        # print(mat_del)
        new_pos, new_zeros = independent_zeros(mat_del)
        for val1,val2 in new_zeros:
            # print(val1)
            lis_of_tuple.append((dic_row[val1], dic_col[val2]))
```

```
    lis_pos = [i for i in range(mat.shape[0])]
    # print(lis_pos)

    ## make unique
    list_of_tuple_enchanted = []
    for i,j in lis_of_tuple:
        if (i,j) not in list_of_tuple_enchanted:
            list_of_tuple_enchanted.append((i,j))
    # print(list_of_tuple_enchanted)

    ## dict mozliwych pozycji na jednym miejscu
    ## funckja do twego najlepiej
    new_pos_zero = find_best_pos(mat.shape[0], list_of_tuple_enchanted)

    mat_out = np.zeros((mat.shape))
    for i,j in new_pos_zero:
        mat_out[i,j] = 1


    return mat_out
```

In [12]:
```
mat_cros = cross_zeros(mat_gl, pos)
print(mat_cros)
```

```
[[1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0.]]
```

Zadanie 3

1. Wykonując redukcję najpierw po wierszach, a następnie po kolumnach uzyskamy inną macierz zredukowaną, niż przy redukcji kolumny-wiersze (zera będą miały te same lokalizacje).

2. Minimalna ilość zer niezależnych dla macierzy zredukowanej NxN wynosi 1, gdy zera wszystkich wierszów znajdują się w pierwszej kolumnie ORAZ zera wszystkich kolumn znajdują się w pierwszym wierszu. Maksymalna ilość zer niezależnych dla macierzy zredukowanej NxN wynosi N i występuje gdy dla każdego wiersza/kolumny przynajmniej jedno z zer jest zerem niezależnym.

3. Nie bedzie ono prawidlowe, poniewaz moze ono spowodowac ze uzyskamy mniejsza ilosc zer niezaleznych.

4. Minimalna liczba linii wykreślających zera występuje dla minimalnej liczby zer niezależnych w macierzy zredukowanej, a maksymalna liczba zer niezależnych spowoduje wystąpienie maksymalnej ilości linii wykreślających zera.

5. Procedura zwiekrzania zer niezalezynch nie zawsze jest skuteczna. O ile zwiekrzona zostanie liczba zer niezaleznych zalezy liczba zer zaleznych.