Shriharsha A Vaidhyam

Project 2

Problem 1:

```
def LargestSquare(mat):
    if not mat or not len(mat):
        return 0

    T = [[0 for x in range(len(mat[0]))] for y in range(len(mat))]
    max = 0
    alist = []
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            T[i][j] = mat[i][j]
            if i > 0 and j > 0 and mat[i][j] == 1:
                T[i][j] = min(T[i][j - 1], T[i - 1][j], T[i - 1][j - 1]) + 1
            if max < T[i][j]:
                max = T[i][j]

    result = [(i1 - max + 2, i2 - max + 2) for (i1, v1) in enumerate(T) for (i2, v2)
in enumerate(v1) if v2 == max]

    return [result, "square size: ", max]



mat1 = [
        [1, 0, 1, 0, 0],
        [1, 0, 1, 1, 1],
        [1, 1, 1, 1, 0],
        [1, 1, 0, 1, 0],
    ]

mat2 = [
        [1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 0, 0],
        [1, 1, 1, 1, 1, 1],
        [1, 1, 1, 0, 0, 0],
        [1, 0, 1, 0, 1, 1],
        [0, 0, 1, 1, 1, 1],
        [0, 0, 1, 1, 1, 1]
    ]

print("First test case:", LargestSquare(mat1))
print("Second test case:", LargestSquare(mat2))
```

Output: (both test case)

harshavaidhyam@Harshas-MacBook-Pro project 2 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 63176 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2/prob1.py

First test case: [[(2, 3), (3, 1)], 'square size: ', 2]
Second test case: [[(1, 1), (1, 2), (2, 1)], 'square size: ', 3]

Problem 2:
Part A

```
class newNode:
    def __init__(self, data):
        self.data = data
        self.left = self.right = None


def mirror(node):
    if (node == None):
        return
    else:

        temp = node

        mirror(node.left)
        mirror(node.right)

        temp = node.left
        node.left = node.right
        node.right = temp


""" Helper function to print Level order traversal."""


def printLevelOrder(root):
    h = height(root)
    for i in range(1, h + 1):
        printCurrentLevel(root, i)


def printCurrentLevel(root, level):
    if root is None:
```

```
        return
    if level == 1:
        print(root.data, end=" ")
    elif level > 1:
        printCurrentLevel(root.left, level - 1)
        printCurrentLevel(root.right, level - 1)


def height(node):
    if node is None:
        return 0
    else:

        lheight = height(node.left)
        rheight = height(node.right)

        if lheight > rheight:
            return lheight + 1
        else:
            return rheight + 1


# Driver code
if __name__ == "__main__":
    root1 = newNode(4)
    root1.left = newNode(2)
    root1.right = newNode(7)
    root1.left.left = newNode(1)
    root1.left.right = newNode(3)
    root1.right.left = newNode(6)
    root1.right.right = newNode(9)


    mirror(root1)

    print("\n",
          "the mirror tree(Test Case 1) : ")
    printLevelOrder(root1)

    root2 = newNode(34)
    root2.left = newNode(24)
    root2.right = newNode(96)
    root2.left.left = newNode(10)
    root2.left.right = newNode(None)
    root2.right.left = newNode(None)
    root2.right.right = newNode(None)

    mirror(root2)
```

```
    print("\n",
          "the mirror tree(Test Case 2) :")
    printLevelOrder(root2)
```

Output:

harshavaidhyam@Harshas-MacBook-Pro project 2 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 63210 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2/prob2_partA.py

the mirror tree(Test Case 1) :
4 7 2 9 6 3 1
the mirror tree(Test Case 2) :
34 96 24 None None None 10

Part B:

```
from contextlib import nullcontext


class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


def areMirror(a, b):


    if a is None and b is None:
        return True


    if a is None or b is None:
        return False


    return (a.data == b.data and
            areMirror(a.left, b.right) and
            areMirror(a.right , b.left))
```

```
#test case 1
root1 = Node(1)
root2 = Node(1)

root1.left = Node(2)
root1.right = Node(3)
root1.left.left = Node(4)
root1.left.right = Node(5)
root1.right.left = Node(6)
root1.right.right = Node(7)

#root2
root2.left = Node(3)
root2.right = Node(2)
root2.left.left = Node(7)
root2.left.right = Node(6)
root2.right.left = Node(5)
root2.right.right = Node(4)


#test case 2
root1 = Node(1)
root2 = Node(1)

root1.left = Node(2)
root1.right = Node(2)
root1.left.left = Node(3)
root1.right.left = Node(3)

#root2
root2.left = Node(2)
root2.right = Node(2)
root2.left.right = Node(3)
root2.right.right = Node(3)


if areMirror(root1, root2):
    print ("Mirror Image")
else:
    print ("Not a Mirror Image")
```

Output: (both test cases)

harshavaidhyam@Harshas-MacBook-Pro project 2 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2 ; /usr/bin/env /usr/local/bin/python3 /Users/harshavaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 63220 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2/prob2_partB.py

Mirror Image

Problem 3:

```
import numpy as np


def min_zero_row(zero_mat, mark_zero):



    min_row = [99999, -1]

    for row_num in range(zero_mat.shape[0]):
        if np.sum(zero_mat[row_num] == True) > 0 and min_row[0] >
np.sum(zero_mat[row_num] == True):
            min_row = [np.sum(zero_mat[row_num] == True), row_num]


    zero_index = np.where(zero_mat[min_row[1]] == True)[0][0]
    mark_zero.append((min_row[1], zero_index))
    zero_mat[min_row[1], :] = False
    zero_mat[:, zero_index] = False


def mark_matrix(mat):


    cur_mat = mat
    zero_bool_mat = (cur_mat == 0)
    zero_bool_mat_copy = zero_bool_mat.copy()

    marked_zero = []
    while (True in zero_bool_mat_copy):
        min_zero_row(zero_bool_mat_copy, marked_zero)

    marked_zero_row = []
    marked_zero_col = []
```

```
    for i in range(len(marked_zero)):
        marked_zero_row.append(marked_zero[i][0])
        marked_zero_col.append(marked_zero[i][1])

    non_marked_row = list(set(range(cur_mat.shape[0])) - set(marked_zero_row))

    marked_cols = []
    check_switch = True
    while check_switch:
        check_switch = False
        for i in range(len(non_marked_row)):
            row_array = zero_bool_mat[non_marked_row[i], :]
            for j in range(row_array.shape[0]):

                if row_array[j] == True and j not in marked_cols:

                    marked_cols.append(j)
                    check_switch = True

        for row_num, col_num in marked_zero:

            if row_num not in non_marked_row and col_num in marked_cols:

                non_marked_row.append(row_num)
                check_switch = True

    marked_rows = list(set(range(mat.shape[0])) - set(non_marked_row))

    return (marked_zero, marked_rows, marked_cols)


def adjust_matrix(mat, cover_rows, cover_cols):
    cur_mat = mat
    non_zero_element = []


    for row in range(len(cur_mat)):
        if row not in cover_rows:
            for i in range(len(cur_mat[row])):
                if i not in cover_cols:
                    non_zero_element.append(cur_mat[row][i])
    min_num = min(non_zero_element)


    for row in range(len(cur_mat)):
        if row not in cover_rows:
            for i in range(len(cur_mat[row])):
                if i not in cover_cols:
```

```
                    cur_mat[row, i] = cur_mat[row, i] - min_num

    for row in range(len(cover_rows)):
        for col in range(len(cover_cols)):
            cur_mat[cover_rows[row], cover_cols[col]] = cur_mat[cover_rows[row],
cover_cols[col]] + min_num
    return cur_mat


def hungarian_algorithm(mat):
    dim = mat.shape[0]
    cur_mat = mat


    for row_num in range(mat.shape[0]):
        cur_mat[row_num] = cur_mat[row_num] - np.min(cur_mat[row_num])

    for col_num in range(mat.shape[1]):
        cur_mat[:, col_num] = cur_mat[:, col_num] - np.min(cur_mat[:, col_num])
    zero_count = 0
    while zero_count < dim:

        ans_pos, marked_rows, marked_cols = mark_matrix(cur_mat)
        zero_count = len(marked_rows) + len(marked_cols)

        if zero_count < dim:
            cur_mat = adjust_matrix(cur_mat, marked_rows, marked_cols)

    return ans_pos


def ans_calculation(mat, pos):
    total = 0
    ans_mat = np.zeros((mat.shape[0], mat.shape[1]))
    for i in range(len(pos)):
        total += mat[pos[i][0], pos[i][1]]
        ans_mat[pos[i][0], pos[i][1]] = mat[pos[i][0], pos[i][1]]
    return total, ans_mat


def main():
    cost_matrix = np.array([[22, 14, 120, 21, 4, 51],
                            [19, 12, 172, 21, 28, 43],
                            [161, 122, 2, 50, 128, 39],
                            [19, 22, 90, 11, 28, 4],
                            [1, 30, 113, 14, 28, 86],
                            [60, 70, 170, 28, 68, 104]])
    ans_pos = hungarian_algorithm(cost_matrix.copy())
```

```
    ans, ans_mat = ans_calculation(cost_matrix, ans_pos)

    print(f"Minimized cost: {ans:.0f}")

    cost_matrix1 = np.array([[108, 125, 149],
                             [150, 135, 175],
                             [122, 148, 250]])
    ans_pos1 = hungarian_algorithm(cost_matrix1.copy())
    ans1, ans_mat1 = ans_calculation(cost_matrix1,
                                     ans_pos1)

    print(f"Minimized cost: {ans1:.0f}")


if __name__ == '__main__':
    main()
```

Output:

harshavaidhyam@Harshas-MacBook-Pro project 2 % cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2 ; /usr/bin/env /usr/local/bin/python3 /Users/harsh avaidhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 63262 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/project\ 2/prob3.py

Minimized cost: 51
Minimized cost: 406

Time complexity:
$O(n^3)$

Sources for Hungarian algorithm:

https://iq.opengenus.org/hungarian-maximum-matching-algorithm/
https://arshren.medium.com/hungarian-algorithm-6cde8c4065a3