Shriharsha A Vaidhyam

**Project 4**

Problem 1:

Source code:

```
def cc_sort(arr):

  table = []
  count = []
  for a in arr:
    c1 = 0
    c2 = 0
    for i in range(len(arr)):
      if arr[i] < a:
        c2 += 1
    while(c2 in table):
      c2 += 1
    table.append(c2)
  print("Table for the counts:", table)
  new = [0 for x in range(len(arr))]
  for i in range(len(arr)):
    new[table[i]] = arr[i]
  print("Sorted array: ", new)


#test case 1:
print(" ")
print("test case 1:")
cc_sort([1, 4, 1, 2, 7, 5, 2])

#test case 2:
print(" ")
print("test case 2:")
cc_sort([5, 2, 9, 5, 2, 3, 5])
```

Output:

harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd /Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env /usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 64796 -- /Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem1.py

**test case 1:**
**Table for the counts: [0, 4, 1, 2, 6, 5, 3]**
**Sorted array:  [1, 1, 2, 2, 4, 5, 7]**

**test case 2:**
**Table for the counts: [3, 0, 6, 4, 1, 2, 5]**
**Sorted array:  [2, 2, 3, 5, 5, 5, 9]**

Problem 2:

Source code:

```
from collections import defaultdict
import time
import random
import matplotlib.pyplot as plt


start_time=time.time()
def generate(n):
  graph = dict()
  for i in range(1,n+1):
    vi = random.sample(range(1,n+1), random.randint(0,n))
    if i in vi:
      vi.remove(i)
    vi.sort()
    graph[i] = vi
  return graph

def graphSets(graph):


    if(len(graph) == 0):
        return []


    if(len(graph) == 1):
        return [list(graph.keys())[0]]


    currnt = list(graph.keys())[0]


    graph2 = dict(graph)
```

```
    del graph2[currnt]


    res1 = graphSets(graph2)




    for v in graph[currnt]:


        if(v in graph2):
            del graph2[v]


    res2 = [currnt] + graphSets(graph2)


    if(len(res1) > len(res2)):
        return res1
    return res2

def locate_clique(graph):


    cl =[]
    vertices = list(graph.keys())
    for ve in vertices:
      clique = []
      clique.append(ve)
      for v in vertices:
          if v in clique:
              continue
          isNext = True
          for u in clique:
              if u in graph[v]:
                  continue
              else:
                  isNext = False
                  break
          if isNext:
              clique.append(v)

      cl.append((len(clique),sorted(clique)))
    (cin, clique) = max(cl)
    return clique
```

```
# Test case 1:
graph = dict([])

graph[1] = []
graph[2] = [3,5]
graph[3] = [2,6]
graph[4] = [5]
graph[5] = [4,6]
graph[6] = [5]
graph[7] = [8,9]
graph[8] = [7,9]
graph[9] = [7,8]

maximalIndependentSet = graphSets(graph)
clique = locate_clique(graph)
# Prints the Result
print("largest independent set: ")
for i in maximalIndependentSet:
    print( i, end =" ")
print(" ")
print("Max clique is: ", clique)

# Test case 2:
# graph = dict([])

# graph[1] = [4]
# graph[2] = [3,5,6]
# graph[3] = [2,5,6]
# graph[4] = [7,8]
# graph[5] = [2,3,6]
# graph[6] = [2,3,5]
# graph[7] = [4,8]
# graph[8] = [4,7,9]
# graph[9] = [8]

# maximalIndependentSet = graphSets(graph)
# clique = locate_clique(graph)
# # Prints the Result
# print("largest independent set: ")
# for i in maximalIndependentSet:
#   print( i, end =" ")
# print(" ")
# print("Max clique is: ", clique)
```

```
# #plotting of time complexity

# x_coordinate = []
# y_coordinate = []


# for i in maximalIndependentSet:
#       print( i, end =" ")
#     x_coordinate.append(i)
#     y_coordinate.append(round(time.time() - start_time, 6))

# plt.plot(x_coordinate, y_coordinate, marker="o")
# plt.xlabel("Size")
# plt.ylabel("Time")
# plt.show()
```

Output:

Test case 1:

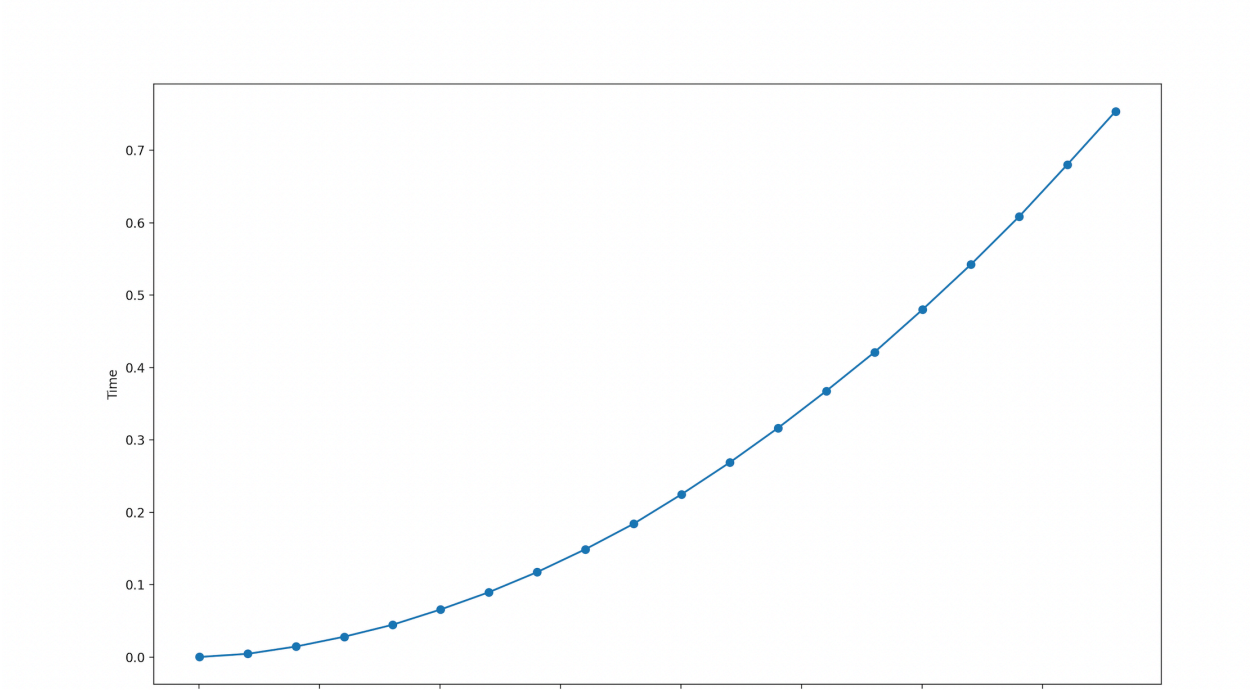
harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd
/Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env
/usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-
2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 64968 --
/Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem2.py

**largest independent set:**
**1 2 4 6 7**
**Max clique is:  [7, 8, 9]**

Test case 2:

**largest independent set:**
**1 2 7 9**
**Max clique is:  [2, 3, 5, 6]**

Time complexity:
O(2^n)




Problem 3 (a):

Source code:

```
from typing import DefaultDict

INT_MAX = 2000000000
def findMinRoute(tsp):
    sum = 0
    counter = 0
    j = 0
    i = 0
    min = INT_MAX
    visitedRouteList = DefaultDict(int)


    visitedRouteList[0] = 1
    route = [0] * len(tsp)


    while i < len(tsp) and j < len(tsp[i]):


        if counter >= len(tsp[i]) - 1:
            break
```

```
        if j != i and (visitedRouteList[j] == 0):
            if tsp[i][j] < min:
                min = tsp[i][j]
                route[counter] = j + 1

        j += 1


        if j == len(tsp[i]):
            sum += min
            min = INT_MAX
            visitedRouteList[route[counter] - 1] = 1
            j = 0
            i = route[counter] - 1
            counter += 1


    i = route[counter - 1] - 1

    for j in range(len(tsp)):

        if (i != j) and tsp[i][j] < min:
            min = tsp[i][j]
            route[counter] = j + 1

    sum += min


    print("Minimum Cost is :", sum)


#test case 1:
# if __name__ == "__main__":

#   # Input Matrix
#   tsp = [[-1, 60, 100, 50, 90], [60, -1, 70, 40, 30], [100, 70, -1, 65,50], [50, 40,
65, -1,110]], [90, 30, 40, 110,-1]


#   findMinRoute(tsp)


#test case 2:
if __name__ == "__main__":

    # Input Matrix
    tsp = [[-1, 10, 15, 20], [10, -1, 35, 25], [15, 35, -1, 30], [20, 25, 30, -1]]
```

```
    findMinRoute(tsp)




# x_coordinate = []
# y_coordinate = []

#findMinRoute(tsp)
#x_coordinate.append(tsp)
#y_coordinate.append(round(time.time() - start_time, 6))


# plt.plot(x_coordinate, y_coordinate, marker="o")
# plt.xlabel("Size")
# plt.ylabel("Time")
# plt.show()
```

Output:

Test case 1:
harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd
/Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env
/usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-
2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 64987 --
/Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem3a.py

**Minimum Cost is : 275**

Test case 2:
harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd
/Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env
/usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-
2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 64987 --
/Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem3a.py

**Minimum Cost is : 80**

Time complexity:
O(n^2*log2N)

Problem 3(b):

Source code:

```
def tsp_heuristics(G,s):
    visited = []
    src = s
    cost = 0
    while s not in visited:

        edges = G[s]
        min_weight = -1
        min_node = -1
        visited.append(s)
        for (i, j) in enumerate(edges):

            if (min_weight == -1) and (i not in visited):
                min_weight = j
                min_node = i

            elif (min_weight != -1) and (j < min_weight) and (i not in visited):
                min_weight = j
                min_node = i
        if min_node != -1:
            s = min_node
        if min_weight != -1:
            cost += min_weight
    cost += G[s][src]
    print("The total cost is:", cost)

#test case1
G = [[-1, 60, 100, 50, 90],[60, -1, 70, 40, 30 ],[100, 70, -1, 65, 55] ,[50, 40, 65, -
1,110 ],[90, 30, 55, 110, -1]]

#test case2
# G = [[-1, 10, 15, 20], [10, -1, 35, 25], [15, 35, -1, 30], [20, 25, 30, -1]]

tsp_heuristics(G,0)




# x_coordinate = []
# y_coordinate = []
```

```
# for i in range(9):
#   g.dijkstra(i)
#     x_coordinate.append(i)
#     y_coordinate.append(round(time.time() - start_time, 6))


# plt.plot(x_coordinate, y_coordinate, marker="o")
# plt.xlabel("Size")
# plt.ylabel("Time")
# plt.show()
```

Output:

Test case 1:
harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd
/Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env
/usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-
2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 65006 --
/Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem3b.py

**The total cost is: 275**

Test case 2:
harshavaidhyam@Harshas-MacBook-Pro PROJECT4 %  cd
/Users/harshavaidhyam/Desktop/Pitt\ term-1/Algo\ Design/PROJECT4 ; /usr/bin/env
/usr/local/bin/python3 /Users/harshava
idhyam/.vscode/extensions/ms-python.python-
2022.16.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 65006 --
/Users/harshavaidhyam/Desktop/Pitt\ term-1/A
lgo\ Design/PROJECT4/problem3b.py

**The total cost is: 80**

Time complexity:
O(n^2)

Difference between time complexities between a and b:

Sample: Say we have the worst case and our data set has size 100.

- O(n^2) --> 100*100 = 10000
- O(n^2*log(n)) --> 100*100*2 = 20000

- The bigger your data set is the more it usually gets slower using an $O(n^2*\log(n))$-algorithm.

$B/A= 0.5S$