Durham University

International Study Centre

SUMMATIVE ASSIGNMENT FRONT COVER SHEET

Student ID	23456789
CIS Username	abcd12
Programme	IFY Computer Science
Module	Computer Science with Extended Research
Teaching Group	JFSCS_CSER
Tutor	Chris Roberts

Assignment Title	F_CSER_J_A2:
Assignment Deadline	30/07/2021

It is very important that any work you present as yours must in fact be your own work, and not taken from another place or done by another person. Cheating, collusion (working together with another person on an assessment which is not intended to be collaborative) and copying from unacknowledged sources (plagiarism) are all serious offences and must be avoided.

DEFINITION OF ACADEMIC IMPROPRIETY

Academic impropriety is a term that covers cheating, attempts to cheat, plagiarism, collusion and any other attempts to gain an unfair advantage in assessments. Assessments include all forms of written work, presentations and all forms of examination. Academic impropriety, in any form, is a serious offence and the penalties imposed would reflect this.

DECLARATION

By entering my Student ID below I confirm that this piece of work is a result of my own work except where it forms an assessment based on group project work. In the case of a group project, the work has been prepared in collaboration with other members of the group. Material from the work of others not involved in the project has been acknowledged and quotations and paraphrases suitably indicated. Furthermore, I confirm that I understand the definition of Academic Impropriety that is used by Durham University International Study Centre.

Student ID: 23456789	Date: 30/07/2021

IFY Computer Science

F_CSER_J_A2:
Project: Minimum Connector Problem

23456789

30/07/2021

Contents

1 Describe the System

The minimum specification for the program is as follows:

- console/text-based
- basic menu system to allow the user to:
  - import a new graph via csv file
  - output the adjacency list as a table
  - find MST using algorithm 1
  - find MST using algorithm 2
  - compare algorithm running times against number of vertices / number of edges / minimum weight
  - (optionally) save data to file in CSV format
  - exit the program in a controlled fashion
- run in a Linux operating system using the approved modules list

2 Create a Solution

- system parameters: none
- Flowcharts:

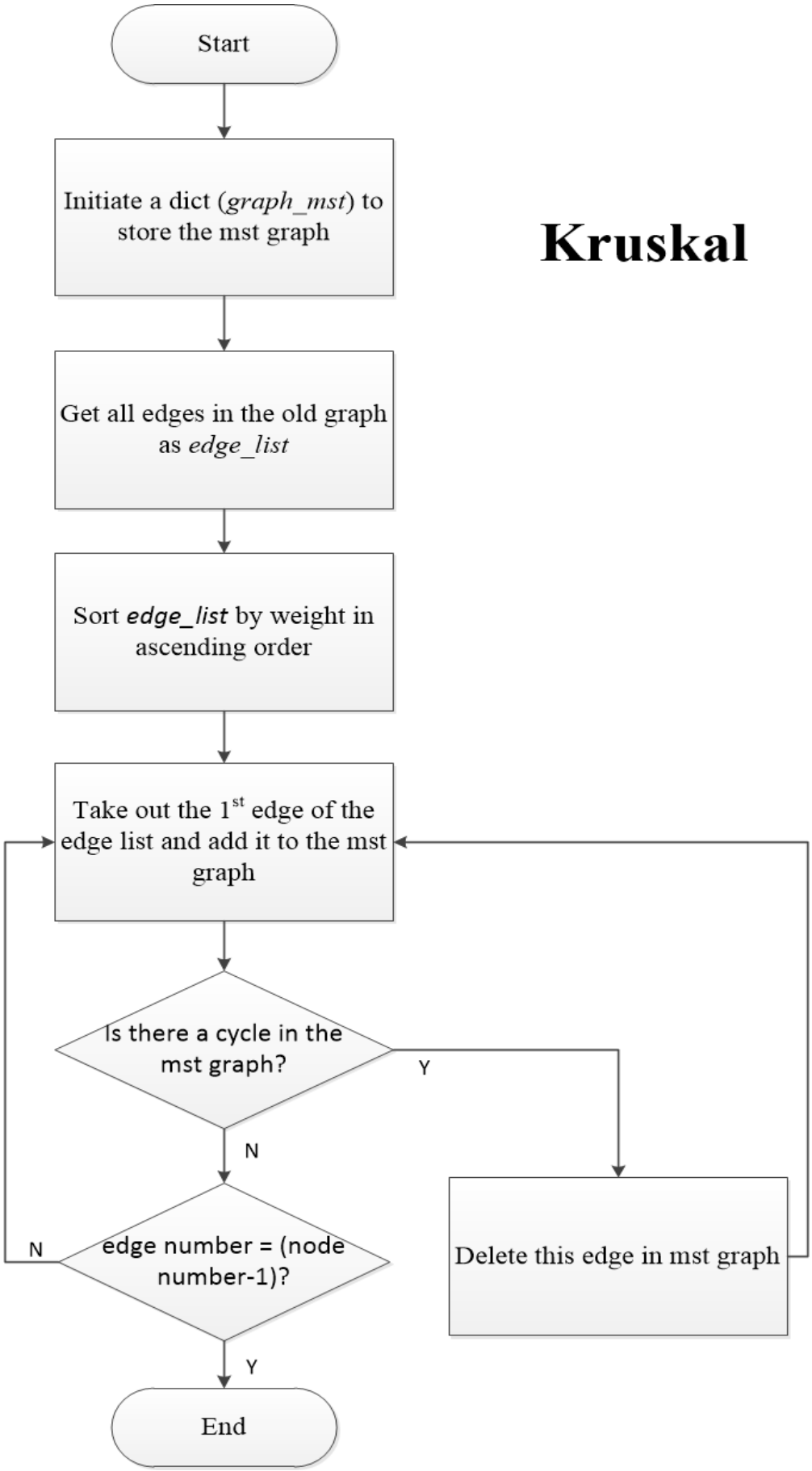
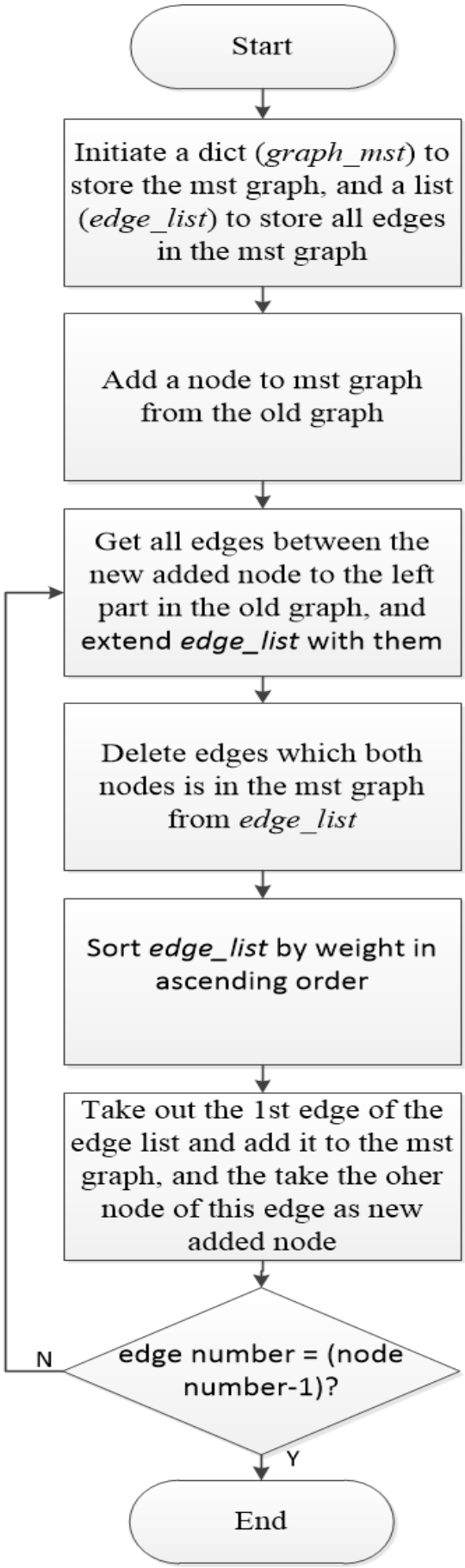


Figure 1: Flowchart-Kruskal



Prime

Figure 2: Flowchart-Prime

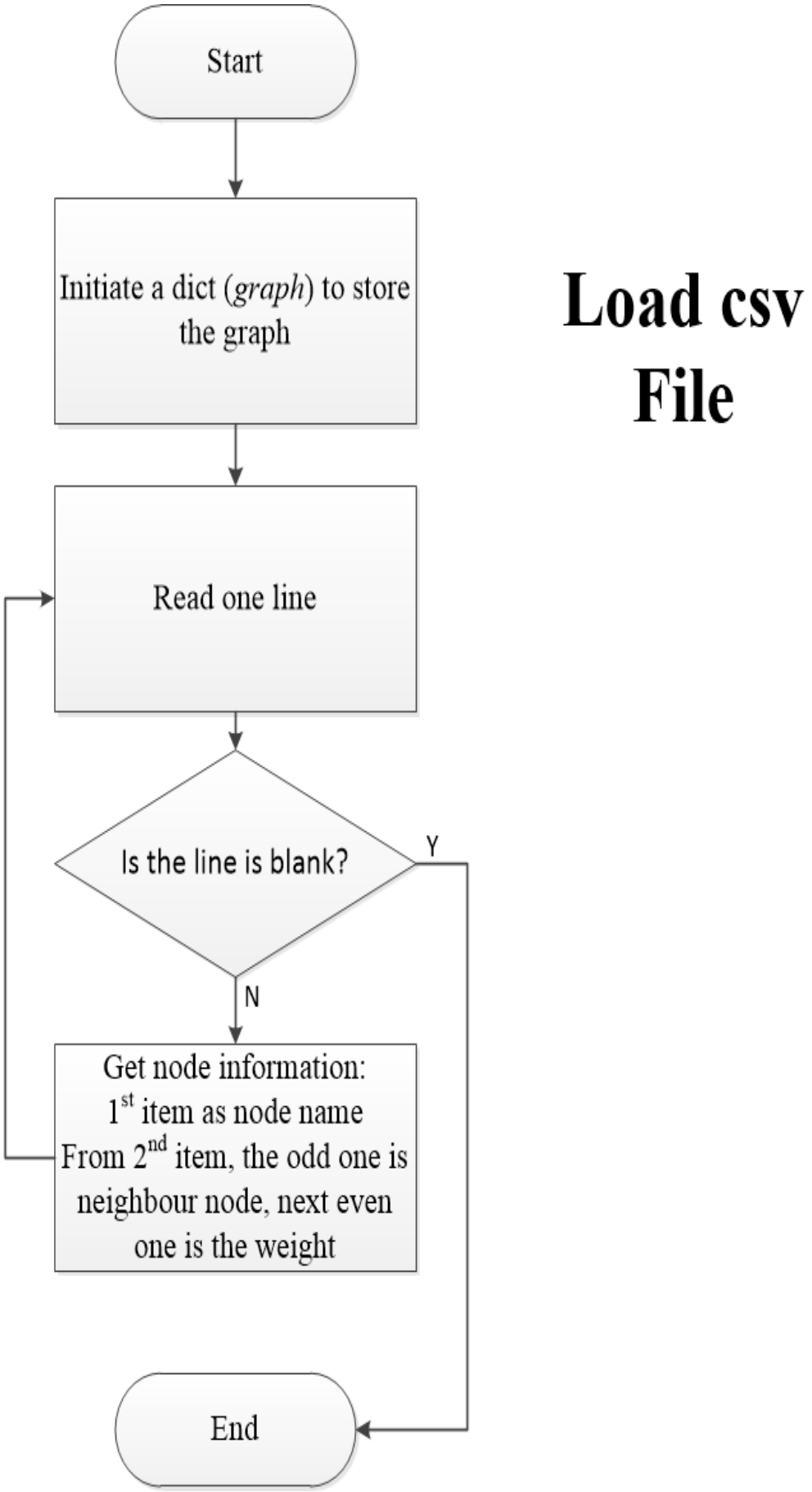
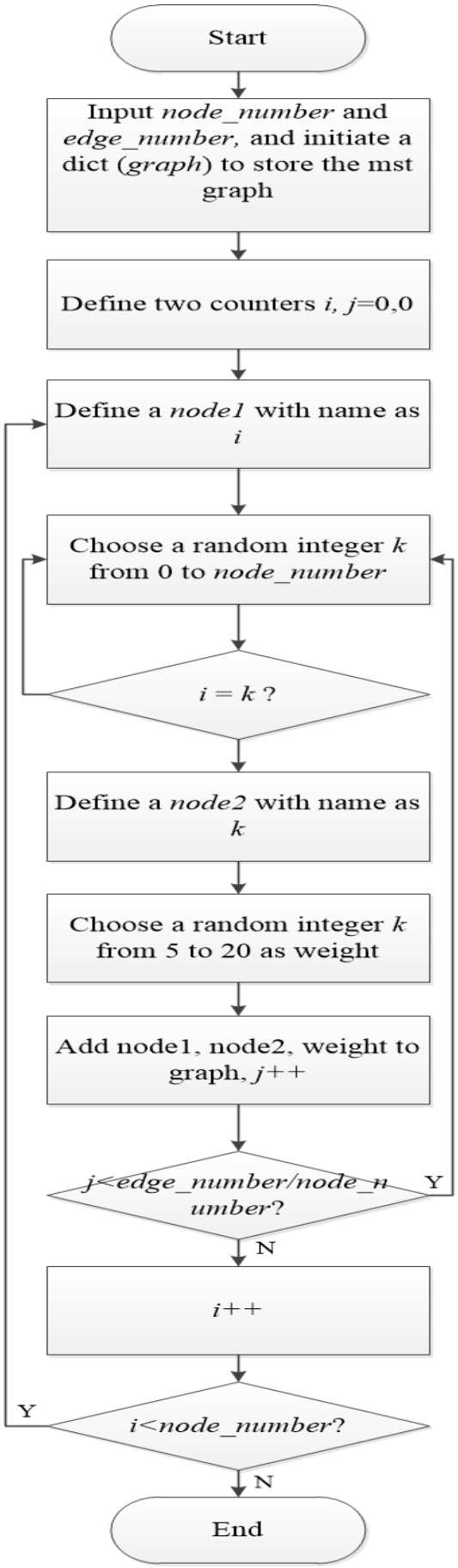


Figure 3: Flowchart-Load csv file



Generate Random Graphs

Figure 4: Flowchart-Generate Random Graphs

- Descriptions:
  - Kruskal:
    1. Get all edges from the old graph
    2. Sort edge list by weight in ascending order
    3. Take out the 1st edge of the edges and add it to the mst graph
    4. Judge if there are cycles in mst graph, if so, delete it
    5. judge if the number of edges is enough (node number -1), if so, we already get the result, else, back to step3
  - Prime:
    1. Add a node to mst graph from the old graph
    2. Get all edges between the new added node to the left part in the old graph, and extend edge list with them
    3. Delete edges which both nodes is in the mst graph from edge list
    4. Sort edge list by weight in ascending order
    5. Take out the 1st edge of the edge list and add it to the mst graph, and the take the oher node of this edge as new added node
    6. judge if the number of edges is enough (node number -1), if so, we already get the result, else, back to step2

3 Test and Validate your Code

4 Summarise and Evaluate

- Conclusion:
  1. When node number is a constant, the time cost of kruskal increases with the increase of edge number, while the time cost of prime is basically static
  2. When edge number is a constant, the time cost of kruskal waves greatly with the increase of node number, while the time cost of prime is basically static
- System Limitations:
  1. The interaction to the user is not convenient enough
  2. The show of the result is not visual
- Future Improvements
  1. Build gui interface to make it esaier for user to load csv file and get the result

2. Generate gif pictures to show the solving process of the algorithm

5 References

Appendices

A Python code: full listing

Listing 1: Automatic Accountant Solution

```
# reads the given file and returns the available slots and input coins as lists
def read_file(file_path):

  # open the file in read mode
  with open(file_path,"r") as my_file:

    # read the number of slots from the first line
    no_of_slots = int(my_file.readline())
    slots = [] # used to store the available slots

    for i in range(no_of_slots):
      # read the next line and split the values seperated by spaces to a list
      # containing the slot thickness/trigger mass
      slot = my_file.readline().split()
      slots.append(slot) # add slot list to the list of all slots

    # read the number of coins from the next line
    no_of_coins = int(my_file.readline())
    coins = [] # used to store all input coins

    for i in range(no_of_coins):
      # read the next line and splot the values seperated by spaces to a list
      # containing the coin thickness/mass
      coin = my_file.readline().split()
      coins.append(coin) # add coin list to the list of all coins

    return slots, coins

slots = [] # used to store list of slots
coins = [] # used to store list of coins

# set file path to read input
read_file_path = "aa_test_02.txt"

distance = 0 # total distance travelled by the coins

#read the input and return lists of slots/coins
slots, coins = read_file(read_file_path)

# loop through list of coins
for coin in coins:
```

```
42   # loop through list of slots
43   for slot in slots:
44     # if coin thickness <= slot thickness and coin mass >= trigger mass
45     if int(coin[0]) <= int(slot[0]) and int(coin[1]) >= int(slot[1]):
46       distance += slots.index(slot) + 1 # add to total distance
47       break # jump out of slots loop as found coin
48
49 # print total distance
50 print(distance)
51
```

B Csv files

Listing 2: test_graph.csv

```
A,B,7,C,9
B,A,7,C,6,D,19,F,14
C,A,9,B,6,D,11,E,14
D,B,19,C,11,E,10,F,13,G,27,I,23
E,C,14,D,10,I,15
F,B,14,D,13,G,25,H,16
G,D,27,F,25,H,20,I,28
H,F,16,G,20,I,17
I,D,23,E,15,G,28,H,17
```