International University - VNUHCM
School of Computer Science and Engineering

Final Examination

Date: 16/06/2021; Duration: 120 minutes

Online, Open-book

SUBJECT: Algorithms & Data Structures (IT013IU)	
Approval by The SCSE Signature Full name: Dr. Nguyen Van Sinh	Lecturer: Signature Full name: Trần Thanh Tùng
Proctor 1 Signature Full name:	Proctor 2 Signature Full name:
STUDENT INFO	
Student name: Võ Anh Việt **Student ID: ITITIU19243**	

INSTRUCTIONS: the total point is 100 (equivalent to 40% of the course)

1. *Purpose:*
   - Test your knowledge on data structures and algorithms in the following topics: Binary Tree, Hash Table, Graphs, Advanced graph algorithms
   - Examine your skill in analysis and design algorithms
2. *Requirement:*
   - Read carefully each question and answer it following the requirements
   - Write the answers and draw models CLEAN and TIDY directly in a **WORD** file
   - You can draw your trees, graphs by hand or by any tool (like draw.io)
   - Include the **SETTING** session below in your answer file.

Note: For all calculations in this subject, the following **rounding convention** is used: **7/2 = 4**

HCMC National University

International University

Student Name:...............

Student ID:...............

----

0. **Setting – TO INSERT TO YOUR ANSWER FILE**

   a. Write the last 2 digits of your student ID (called is x):_43_____________ (TO FILL IN)

   b. Compute your **OFFSET** = x % 5 = _3_____________ (TO FILL IN)

   c. Using the table below to compute your **Starting node**: _D_____________ (TO FILL IN)

OFFSET	0	1	2	3	4
STARTING NODE	A	B	C	D	E

<u>Your list iteration procedure</u>:

For all lists of items in the following sessions, take items from left to right starting from your **OFFSET.** If the end of the list is reached before all items are taken, continue from index 0 (wrap around).

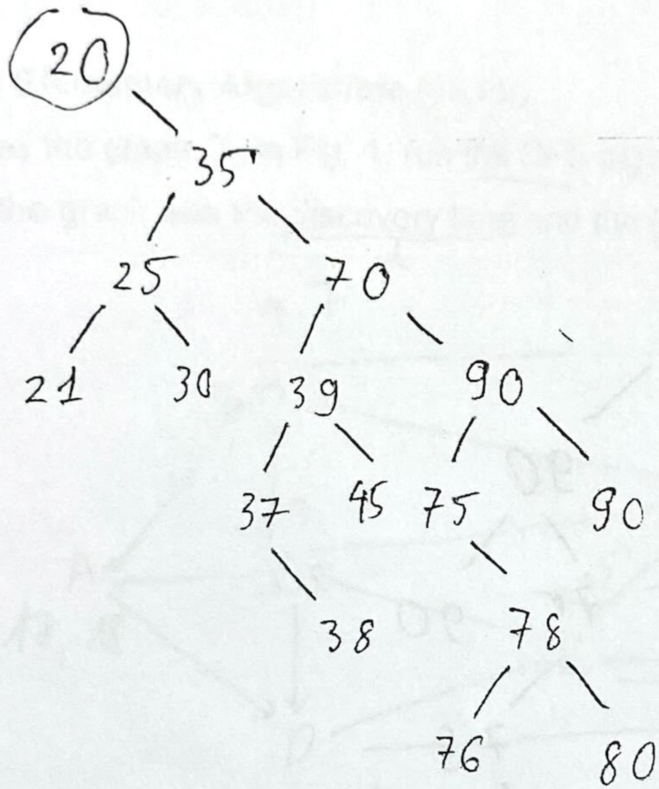
1. **Binary search tree (25pts)**

Given a list of items, take items one by one using **<u>your list iteration procedure</u>**.

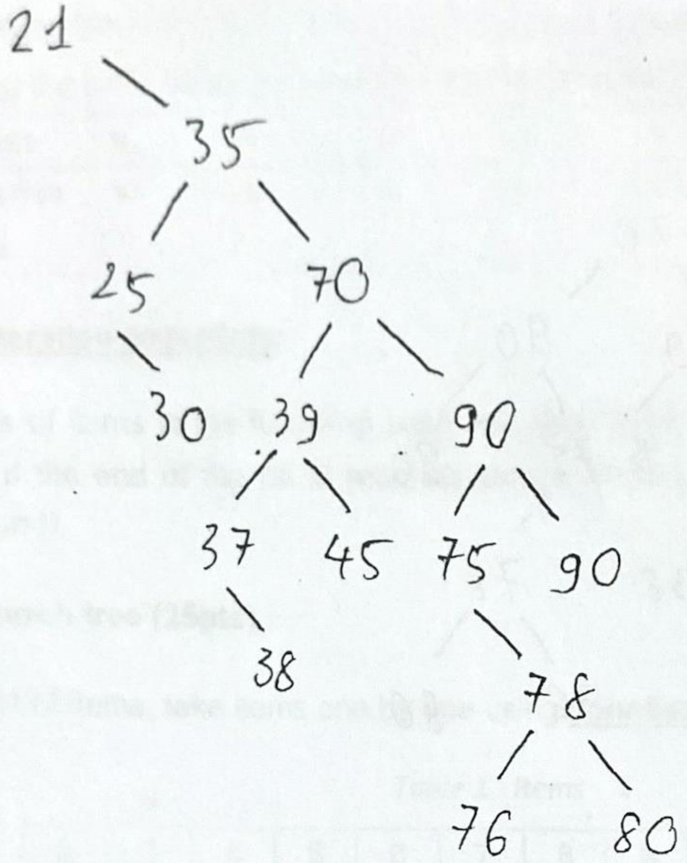
Table 1 - Items

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
45	30	80	20	35	70	90	25	21	39	37	38	75	78	76	90

1.a. Insert all items into a binary search tree and draw the tree (15pts)



1.b. Delete the root node and redraw the tree (10pts)



2. **Hash table (20pts)**

Given a list of items in table 1, take items using **<u>your list iteration procedure</u>**.

2.a. Insert all items into the hash table of size **27** by using the linear probing algorithm to solve collisions (10pts).

We use hash function: x % 27:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	2
			30					35	90	37	38	39	90			70		45		20	21	75	76	78	25	8

2.b. Change the hash table's size to **31**, redraw it (10pts)

We must rehashing with hash function: x % 31

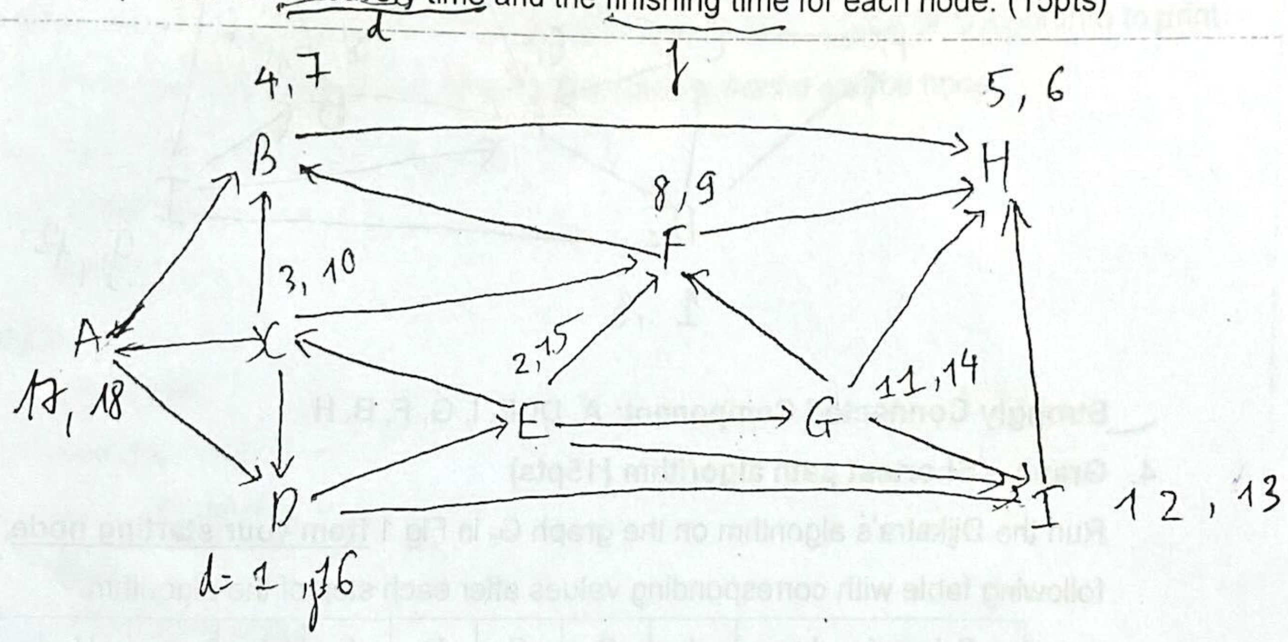
Student Name:.......................................

Student ID:.......................................

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
				35		37	38	70	39				75	76	45	78		80		20	21				25			90	90	30

3. **Graph - Elementary Algorithms (30pts)**

3.a Given the graph G_S in Fig. 1, run the DFS algorithm from your **STARTING NODE** and redraw the graph with the discovery time and the finishing time for each node. (15pts)



3.b. Find all strongly connected components in G_s and draw the G_s^{-1} with the finishing time for each node(15pts)

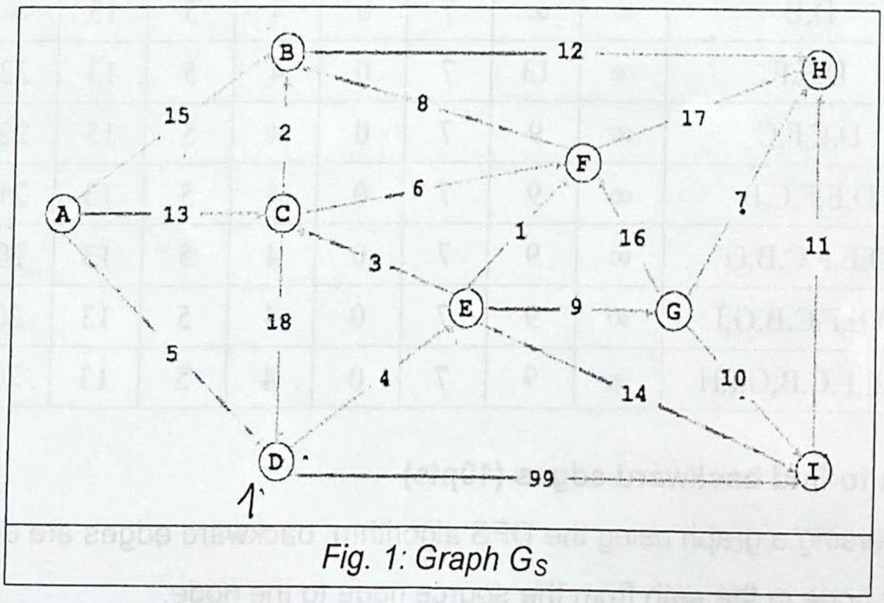
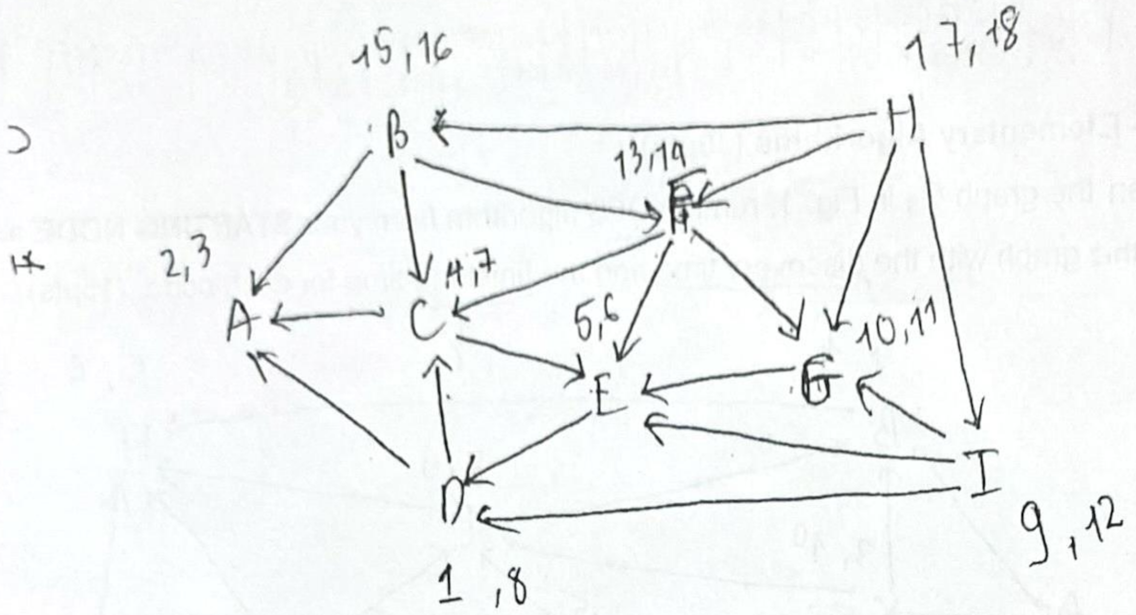


Fig. 1: Graph G_S

C, D, E,



Strongly Connected Component: A, DCE, I, G, F, B, H

4. **Graph – Shortest path algorithm (15pts)**

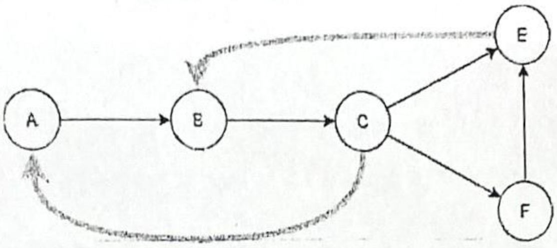
Run the Dijkstra's algorithm on the graph G_s in Fig.1 **from your starting node**, and fill the following table with corresponding values after each step of the algorithm

Selected nodes	A	B	C	D	E	F	G	H	I
	∞	∞	∞	0	∞	∞	∞	∞	∞
D	∞	∞	∞	0	4	∞	∞	∞	99
D,E	∞	∞	7	0	4	5	13	∞	18
D,E,F	∞	13	7	0	4	5	13	22	18
D,E,F,C	∞	9	7	0	4	5	13	22	18
D,E,F,C,B	∞	9	7	0	4	5	13	21	18
D,E,F,C,B,G	∞	9	7	0	4	5	13	20	18
D,E,F,C,B,G,I	∞	9	7	0	4	5	13	20	18
D,E,F,C,B,G,I,H	∞	9	7	0	4	5	13	20	18

5. **Algorithm to find backward edges (10pts)**

While traversing a graph using the DFS algorithm, backward edges are edges that link a node to another node in the path from the source node to the node.

For example, in the graph below, colored edges are backward edges.



- (10pts) Propose an algorithm (write a pseudo-code) based on the DFS algorithm to print out all the backward edges of a given graph starting from a source node.

```
DFS visit(u):
If visit[u] = true then
        Return;
Visit[u[ <- true;
For (a in G.adj[u])
        If visit[u] = true
                Print(u+ " " + v);
        Else
                DFS visit(a)
```

--- The end ---