Data Structure 2024
U2001
National Taipei University

Final Exam

1. Traverse the binary tree shown in Fig. 1a using **preorder** and **inorder** (10%)

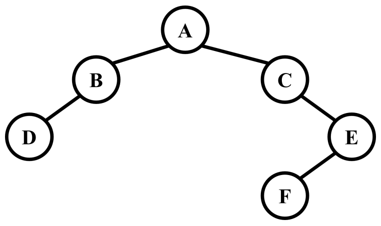


Fig.1a: tree traversal (Q1)

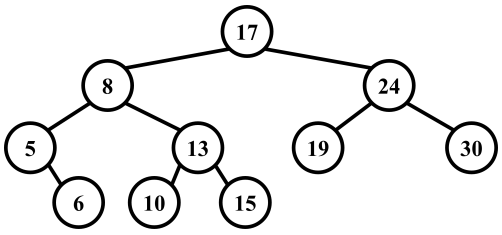


Fig.1b: binary search tree operations (Q2)

Preorder: A → B → D → C → E → F (5%)
Inorder: D → B → A → C → F → E (5%)

2. Provide algorithms with **detailed steps** for modifying the **binary search tree** shown in Fig. 1b:

   (a) Insert a node with key **23** (you need to show the process) (5%)
   We begin by searching for key 23 in the tree. Starting at the root (17), we move to the right since 23 is greater than 17. Next, we move to the left because 23 is less than 24. Finally, we move to the right as 23 is greater than 19. Upon confirming its absence, we insert it as a right child of 19.

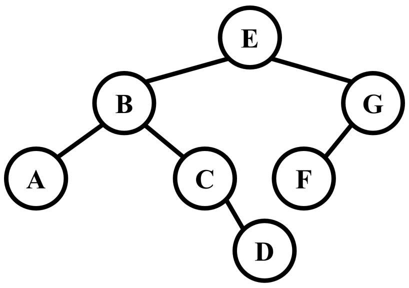
   (b) Delete the node with key **8** (5%)
   We can swap the node 8 with its successor: the minimum node, 10, of the right subtree (or the maximum node, 6, of the left subtree). After that, we can simply cut the node 8 because the successor have no children.

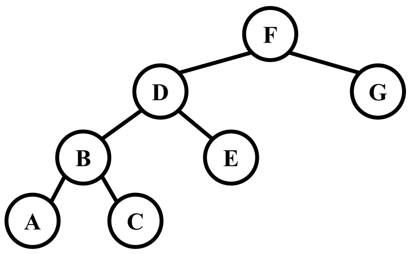
3. A **binary search tree (BST)** can be decided uniquely given its **preorder**:

   (a) Draw the BST with the **preorder** sequence: E → B → A → C → D → G → F. (4%)

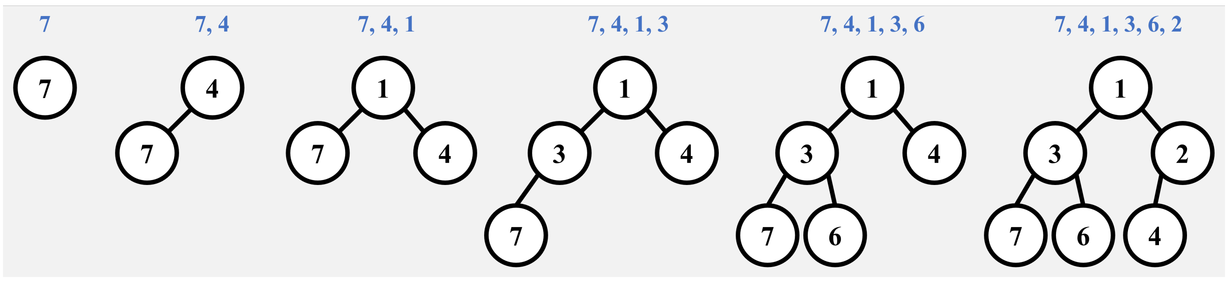
   (b) Draw the BST with the **postorder** sequence: A → C → B → E → D → G → F. (4%)

The answers are displayed below: the left side shows the preorder, while the right side shows the postorder.





4. Build a **Min Heap** for the input sequence: 7, 4, 1, 3, 6, 2. You should draw the tree after **each** number is inserted into the heap. (6%)
   The answer is shown in below figures: 1% for each figure.



5. Transform the **forest** in Fig.2a into a single **binary tree** using the **left-child-right-sibling** approach. (6%)

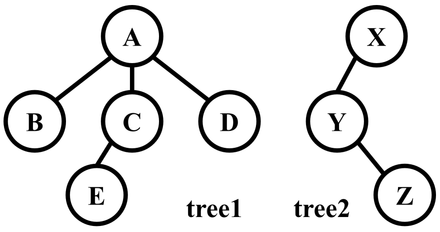
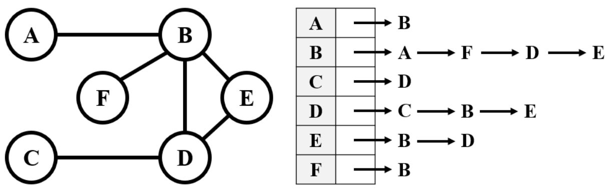


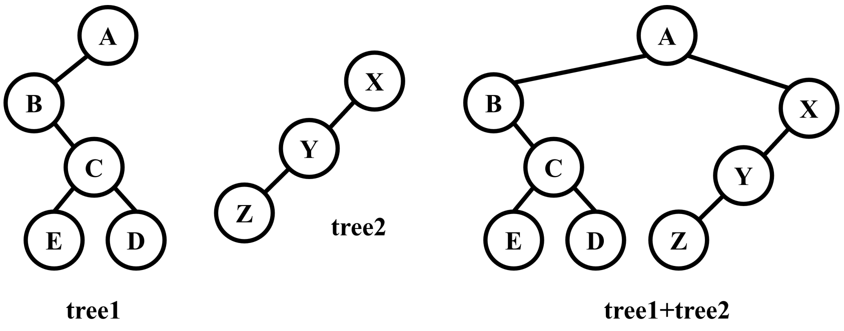
Fig.2a: forest (Q5)



A	→ B
B	→ A → F → D → E
C	→ D
D	→ C → B → E
E	→ B → D
F	→ B

Fig.2b: graph traversal (Q6)

The answers are illustrated in the figures below: drawing the correct final tree earns 6%. If the final answer is incorrect, providing a depiction of either tree1 or tree2 earns 2%.



6. Refer to the undirected graph in Fig.2b and perform a **breadth-first search (BFS)** traversal starting from node **A**, using the adjacency list provided on the right. You can just show the result. (6%)
   BFS: A $\rightarrow$ B $\rightarrow$ F $\rightarrow$ D $\rightarrow$ E $\rightarrow$ C.

7. An 8-element array is sorted increasingly using **quicksort**. It has just finished the first pass of partitioning and pivot swapping, thus transforming the original array into [18, 6, 3, 13, 25, 37, 49, 42]. Write down all possible elements that could have been the pivot in the first pass. (6%)
   25, 37

8. Describe how to use **straight-radix sort (sorting from the least significant bit)** to sort the following numbers: 180, 55, 85, 100, 812, 2, 34, 76. (6%)
First run: sort with the digits: 180 → 100 → 812 → 2 → 34 → 55 → 85 → 76.
Second run: sort with the ten digits while preserving the previous order of digits: 100 → 2 → 812 → 34 → 55 → 76 → 180 → 85.
Third run: sort with the hundred digits while preserving the previous orders of ten digits and digits: 2 → 34 → 55 → 76 → 85 → 100 → 180 → 812.

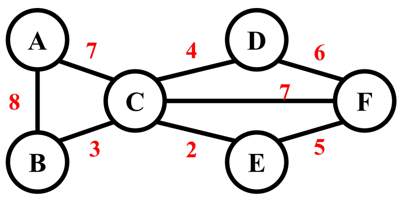
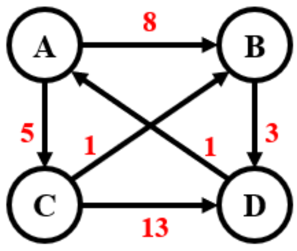
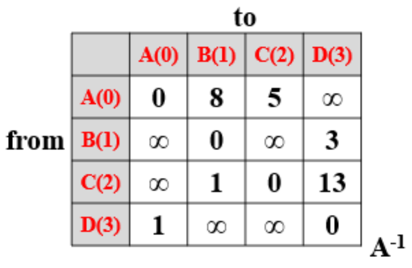


Fig.3a: minimum cost spanning tree (Q9)



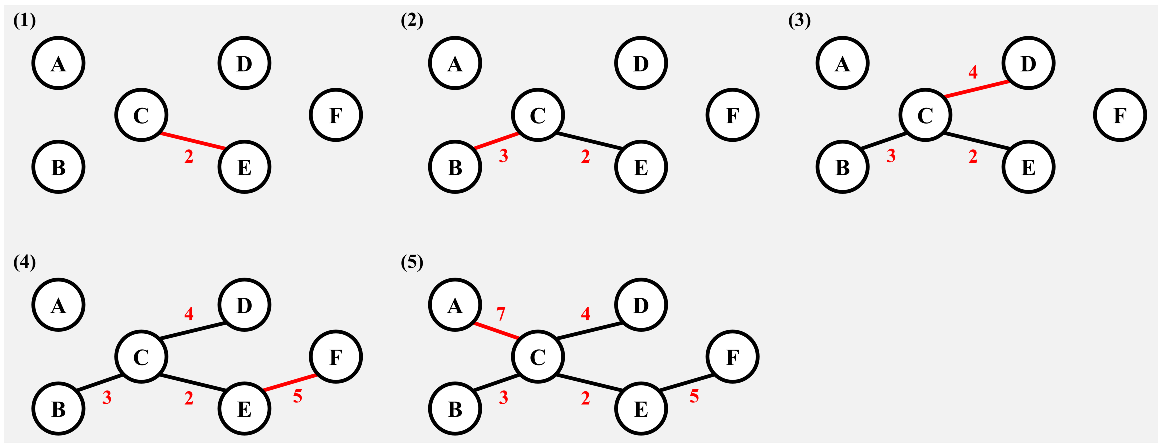


		to			
		A(0)	B(1)	C(2)	D(3)
from	A(0)	0	8	5	∞
	B(1)	∞	0	∞	3
	C(2)	∞	1	0	13
	D(3)	1	∞	∞	0

A^{-1}

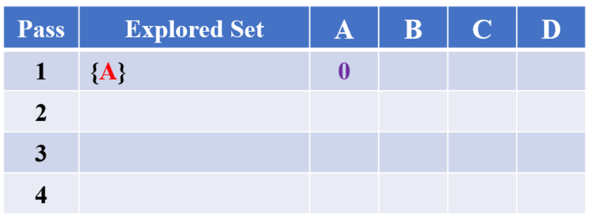
Fig.3b: shortest paths (Q10)

9. Find the minimum cost spanning tree of the graph shown in Fig.3a using **Kruskal's algorithm**. You need to show the graph in each step. (6%)
The answer is shown in the figures below:

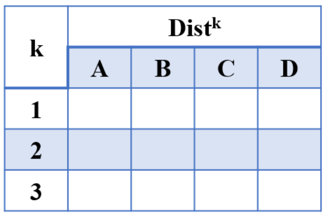


10. Answer the following questions based on the directed graph shown in Fig. 4b:

 (a) Show each step of **Dijkstra's algorithm** (using the table on the left) and **Bellman and Ford algorithm** (using the table on the right) to determine the shortest paths from vertex **A** to all other vertices. (8%)



Pass	Explored Set	A	B	C	D
1	{A}	0			
2					
3					
4					



k	$Dist^k$			
	A	B	C	D
1				
2				
3				

The answer is shown in below figures: the left side shows Dijkstra's algorithm (4%), while the right side shows the Bellman and Ford algorithm (4%).

Pass	Explored Set	A	B	C	D
1	{A}	0	8	5	∞
2	{A, C}	0	6	5	18
3	{A, C, B}	0	6	5	9
4	{A, C, B, D}	0	6	5	9

k	Dist^k			
	A	B	C	D
1	0	8	5	∞
2	0	6	5	11
3	0	6	5	9

(b) The matrix A^{-1} shows the initial values to find all-pairs shortest paths using **Floyd-Warshall's algorithm**. Derive A^0, A^1, A^2, and A^3 (8%)
The answer is shown in below figures: 2% for each pass

	A(0)	B(1)	C(2)	D(3)
A(0)	0	8	5	∞
B(1)	∞	0	∞	3
C(2)	∞	1	0	13
D(3)	1	9	6	0

A^0

	A(0)	B(1)	C(2)	D(3)
A(0)	0	8	5	11
B(1)	∞	0	∞	3
C(2)	∞	1	0	4
D(3)	1	9	6	0

A^1

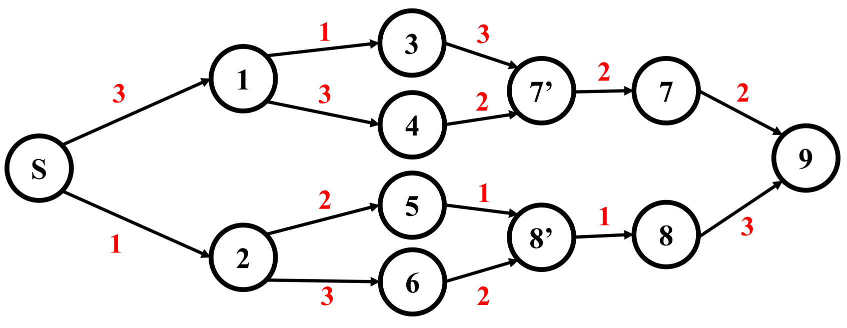
	A(0)	B(1)	C(2)	D(3)
A(0)	0	6	5	9
B(1)	∞	0	∞	3
C(2)	∞	1	0	4
D(3)	1	7	6	0

A^2

	A(0)	B(1)	C(2)	D(3)
A(0)	0	6	5	9
B(1)	4	0	9	3
C(2)	5	1	0	4
D(3)	1	7	6	0

A^3

11. Answer the following questions about the **active-on-edge (AOE)** network shown below:
   (a) Determine the earliest completion time of **node 7'** and **node 8'** (4%)
   node 7': 8 (2%); node 8': 6 (2%).
   (b) If the latest completion time (deadline) of **node 9** is 12, determine the latest completion time of **node 1** and **node 2** (4%)
   node 1: 3 (2%); node 2: 3 (2%).
   (c) Following (b), is there a critical path? If so, identify the nodes along the path. (2%)
   Yes. S → 1 → 4 → 7' → 7 → 9.



12. Select **all correct** statements about **sorting** a list of numbers (10%)
   (a) **Insertion sort** is an excellent choice when the input list is small (*e.g.*, ≤ 100).
   (b) **Quick sort** is a good option when average-case time complexity is the primary consideration.
   (c) If worst-case time complexity is a priority, **merge sort** is more suitable than **quick sort**.
   (d) To sort one million numbers with values ranging from 0 to 1000, **radix sort** is faster than **quick sort**.
   (e) All comparison-based sorting algorithms have a time complexity $\Omega(n \log n)$.

abcde