Past Year Paper: 2015/16 T1
Quizzes 1 & 2 (combined)
Answers and Marking Scheme

Slides prepared by Senior TAs: Son Nguyen and Tuan Le

Question 1a

There are 10 men and 10 women.

1a. How many ways are there to arrange these people in a row? (2 marks)

Answer: $20!$ ——— $\boxed{\text{2 marks}}$

Some of you:

$10! \times 10! \times 2!$ ——— $\boxed{\text{0.5 mark}}$

Question 1b

There are 10 men and 10 women.

1b. How many ways are there to seat these 20 people, if there are two different circular tables? One table is blue, and the other table is red. Each table has 5 men and 5 women alternating. Two seatings for a table are considered the same when everyone on that table has the same two neighbors without regard to whether they are right or left neighbors.(3 marks)

Question 1b (Answer)

+ Divide 20 people into two groups. Each group has 5 men, 5 women:

$$\frac{C_5^{10} \times C_5^{10} \times C_5^5 \times C_5^5}{2!}$$ — 0.5 mark

+ Assign tables to these two groups: 2! — 0.5 mark

+ Arrange 10 people in **blue** table: $\frac{4! \times 5!}{2}$ — 0.5 mark

+ Arrange 10 people in **red** table: $\frac{4! \times 5!}{2}$ — 0.5 mark

+ Final answer: $C_5^{10} \times C_5^{10} \times C_5^5 \times C_5^5 \times \frac{4! \times 5!}{2} \times \frac{4! \times 5!}{2}$ — 1 mark

Question 2a

- It is given that `submethod` is a function that takes an integer n as input, and has a complexity of O(n).

2a. What is the Big O complexity of `method_2a`? Clearly show your working. (2 marks)

```
def method_2a(n):
    for i in range(1, n):
        for j in range(1, i):
            submethod(n)
```

Question 2a (Answer)

```
def method_2a(n):
    for i in range(1, n):
        for j in range(1, i):
            submethod(n)
```

0.5 mark

For each *i*, **submethod** is called *i* times

The total times **submethod** is called is:

$$\sum_{i=1}^{n} i = 1+2+3+\ldots+(n-1)+n = \frac{n(n+1)}{2} \approx O(n^2)$$

0.5 mark

Since **submethod has O(n)** complexity
The complexity of **method_2a** is

$$O(n^3)$$

1 mark

Question 2b

- It is given that `submethod` is a function that takes an integer n as input, and has a complexity of O(n).

2b. What is the Big O complexity of `method_2b`? Clearly show your working. (1.5 marks)

```
def method_2b(n):
    for i in range(1, n):
        for j in range(1,(i*i)):
            submethod(n)
```

Question 2b (Answer)

```
def method_2b(n):
    for i in range(1, n):
        for j in range(1, (i*i)):
            submethod(n)
```

0.5 mark

For each *i*, **submethod** is called *i*i* times

The total times **submethod** is called is:

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2 = \frac{n(n+1)(2n+1)}{6} \approx O(n^3)$$

0.5 mark

Since **submethod has O(n)** complexity
The complexity of **method_2b** is

$$O(n^4)$$

0.5 mark

Question 2c

- It is given that `submethod` is a function that takes an integer n as input, and has a complexity of O(n).

2c. What is the Big O complexity of `method_2c`? Clearly show your working. (1.5 marks)

```python
def method_2c(n):
    for i in range(1, n):
        for j in range(1, n):
            if j % i == 0:
                for k in range(1, n):
                    submethod(n)
```

Question 2c (Answer)

```
def method_2c(n):
    for i in range(1, n):
        for j in range(1, n):
            if j % i == 0:
                for k in range(1, n):
                    submethod(n)
```

$$\left.\begin{array}{l}\texttt{for k in range(1, n):}\\ \quad\texttt{submethod(n)}\end{array}\right\} O(n^2)$$

0.5 mark

For each i, this condition is true n/i times

Total number of times the condition is true:

$$n\sum_{i=1}^{n}\frac{1}{i} = n\left(1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n-1}+\frac{1}{n}\right) \approx O(n\log n)$$

0.5 mark

The complexity of **method_2c** is:

$$O(n^3 \log n)$$

0.5 mark

Question 3a

3a. Suppose that you run binary search on an array A (the algorithm is given below). What are the final values of lower and upper when you search for number 21? Clearly show how you got the results. Note that index starts from 0, i.e., A[0] = 3. (2 marks)

A = [3, 7, 10, 15, 21, 36, 44, 52, 58, 65, 79, 85, 87, 90, 93]

Binary search algorithm:

```
def bSearch(array, target):
    lower = -1
    upper = len(array)
    while not (lower + 1 == upper):      #fail if region empty
        mid = (lower + upper)//2         #find middle of region
        if target == array[mid]:         #succeed if k is at mid
            return mid
        elif target < array[mid]:
            upper = mid                  #search lower region
        else:
            lower = mid                  #search upper region
    return -1                            #not found
```

Question 3a (Answer)

Lower = 3; Upper = 5

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	3	7	10	15	**21**	36	44	52	58	65	79	85	87	90	93

	lower	*upper*	*mid*
0.5 mark	-1	15	7
0.5 mark	-1	7	3
0.5 mark	3	7	5
0.5 mark	3	5	4

Question 3b

3b. You are given as input an unsorted array B, containing n distinct integers (no duplicates) in the range from 1 to (n+1), whereby exactly one integer in this range is missing from the array. The objective is to determine which integer is missing.

For example, for the array B = [2, 3, 5, 6, 1, 7], we have n = 6, and the range is from 1 to 7. The missing integer is 4.

Describe an algorithm (in pseudo/Python code) to find the missing integer in an input array B. The lower the complexity of your algorithm is, the higher will the marks be.

(Maximum 3 marks)

Question 3b (Answer)

$O(n^2)$: 1 mark

Solution 1:

- For each value i from 1 to (n+1)
  - Search for i in B
    - If cannot find i in B
      - Return i as the missing value

Question 3b (Answer)

O(n^2): 1 mark

Solution 2:

- Sort B using insertion sort (**O(n^2)**)
- Search for the missing value (linear search/binary search)

Question 3b (Answer)

O(nlogn): 2 marks

Solution:

- Sort B using merge sort (**O(nlogn)**)
- Search for the missing value (linear search/binary search)

Question 3b (Answer)

O(n): 3 marks

Solution 1:

```
• Create an array C containing (n+1) elements 1 (O(n))
• for each value i in B:      (O(n))
    • set C[i] = 0
• for i in range(1, (n+1)):      (O(n))
    • if C[i] == 1
        • return i as the missing value
```

Question 3b (Answer)

O(n): 3 marks

Solution 2:

- Compute u = 1 + 2 + … + n + (n+1)

  u = (n+2)*(n+1)/2
- t = 0
- for each value i in **B:** (O(n))
  - t += i
- return (u - t) as the missing value

Question 3b (Answer)

Other cases:

- Say: sort the array, then search …: do not specify which sort algorithm you use

→ 1.5 mark

- The idea is correct but return the missing value wrongly, e.g., return B[i] instead of return i → mark – 0.5
- The proposed algorithm cannot find the missing value → 0.5 mark

Question 4a

4a. Suppose a sorted array C = [5, 8, 12, 19, 22, 36, 40, 52, 58, 63] is the output you get after applying insertion sort algorithm on another input array C'. **What is the input array C'** that would require:

i. the fewest number of comparisons. **What is that number of comparisons**? (1 mark)

ii. the most number of comparisons. **What is that number of comparisons**? (1 mark)

Question 4a (Answer)

i. the fewest number of comparisons (n = C.length = 10)

C’ is already sorted:

C’ = [5, 8, 12, 19, 22, 36, 40, 52, 58, 63]

0.5 mark

Or C’ = [8, 5, 12, 19, 22, 36, 40, 52, 58, 63]

Number of comparisons:

# = 0 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = **9**

0.5 mark

ii. the most number of comparisons

C’ is sorted in a reverse way:

C’ = [63, 58, 52, 40, 36, 22, 19, 12, 8, 5]

0.5 mark

Number of comparisons:

# = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = **45**

0.5 mark

Question 4b

You are given K ascending sorted arrays {A1, A2, …, AK}, where each array Ai has n integers. For example, the input may be K = 4 sorted arrays of n = 3 integers each, i.e., A1 = [1, 3, 5], A2 = [4, 9, 10], A3 = [2, 6, 7], and A4 = [8, 12, 15]. The objective is to combine these K input arrays into a single sorted array. The expected output is the combined sorted array, i.e., [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15].

i. Provide an algorithm (in pseudo code or Python code) to perform this task. (2 marks)

ii. What is the Big O complexity of your proposed algorithm in terms of K and n? (1 mark)

Question 4b (Answer)

Solution 1:

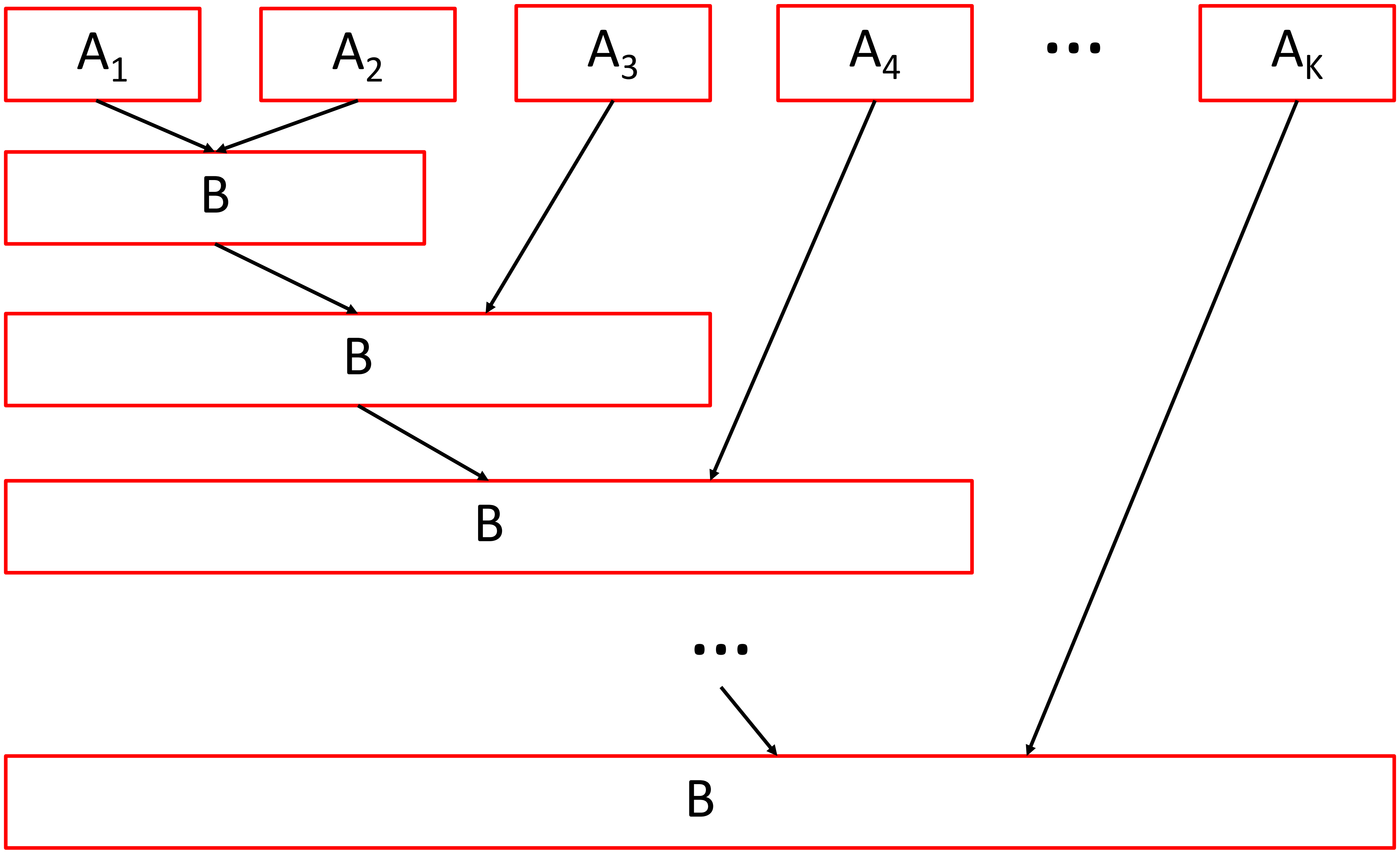
- Combine all the K arrays into 1 array B (**B.length = nK**)
- Sort B

Complexity:

- Sort B by **insertion** sort: **$O(n^2K^2)$**
- Sort B by **merge** sort: **$O(nK \log(nK))$**

Question 4b (Answer)

Solution 2:

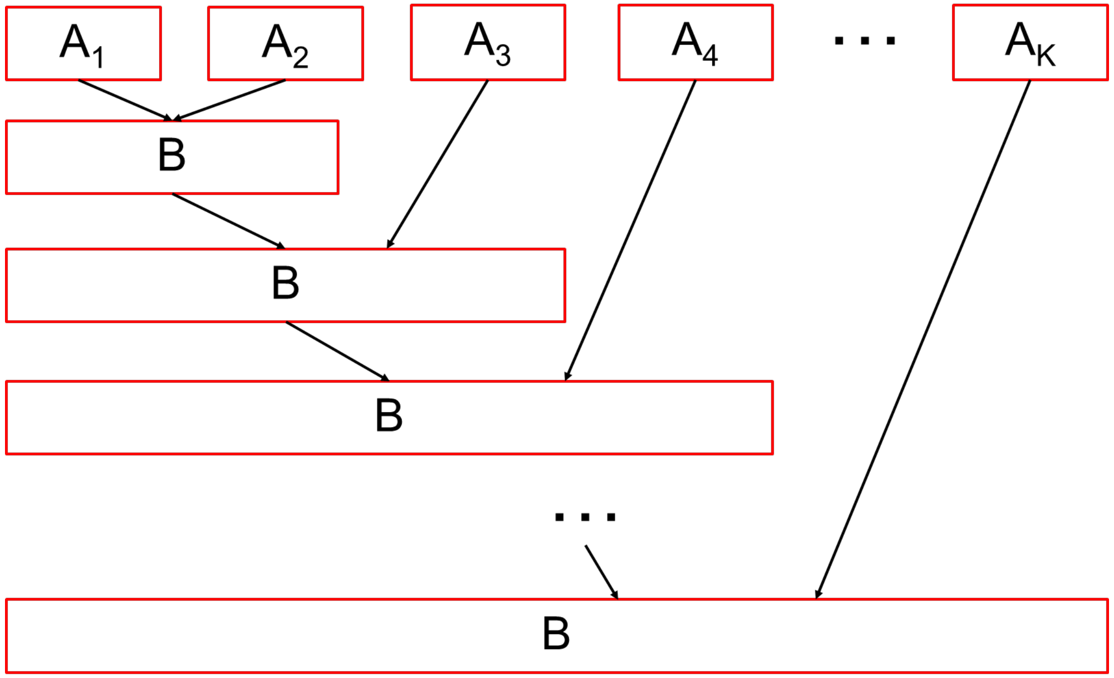


Question 4b (Answer)

Solution 2:

Complexity:

i	#comparisons
1	0
2	2n
3	3n
…	…
K	Kn

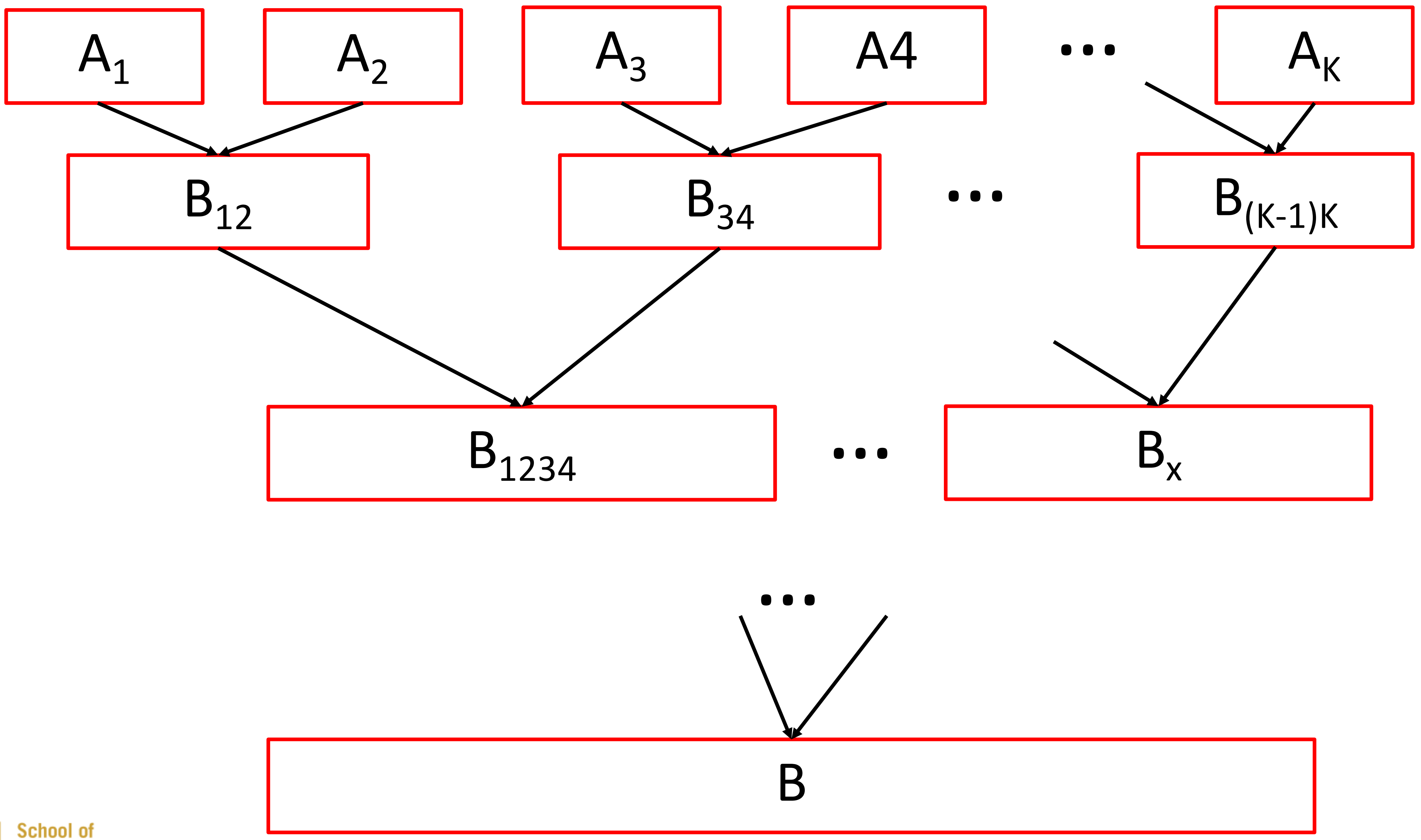


Complexity = 0 + 2n + 3n + … + Kn = n(K+2)(K-1)/2

→ $O(nK^2)$

Question 4b (Answer)

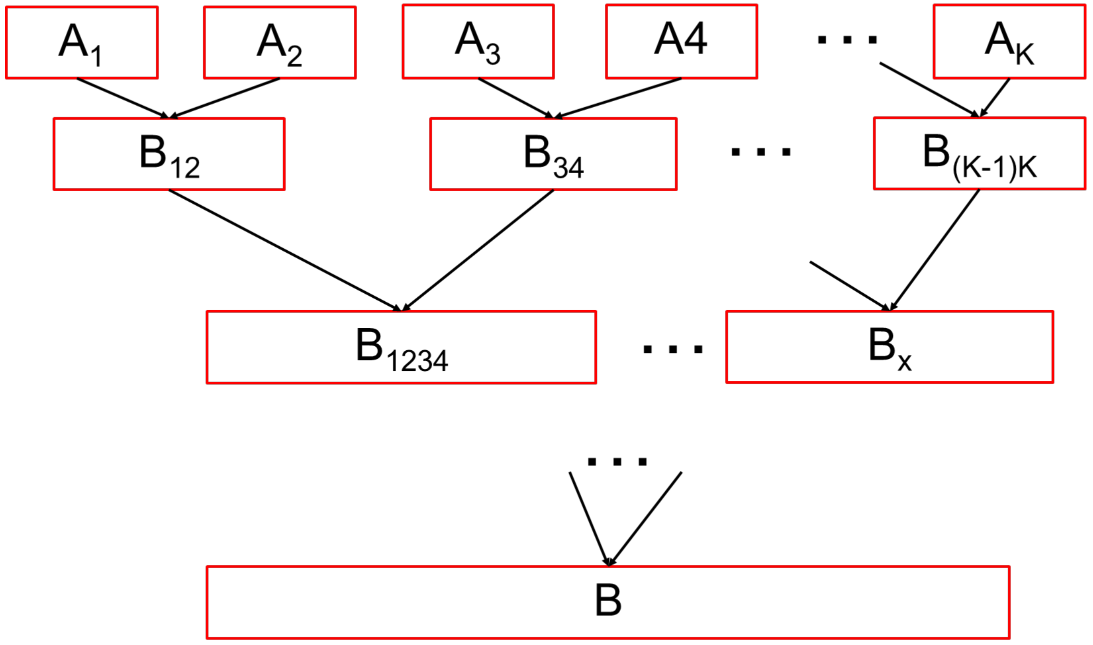
Solution 3:



Question 4b (Answer)

Solution 3:

Complexity:



- At each loop, the number of comparisons is **n*K**
- We have **logK** levels (i.e., loop)

➔ O(nK logK)

Question 5a

5a. What is the sequence of values printed out by the following operations? (1.5 marks)

Question 5a

```
s1= Stack.new
s2= Stack.new
s1.push(7)
s1.push(12)
s2.push(s1.pop)
s2.push(s1.pop)
p s2.pop  #print out the value popped from s2
s1.push(8)
s1.push(28)
p s2.pop #print out the value popped from s2
s2.push(s1.pop)
s2.push(s1.pop)
p s2.pop  #print out the value popped from s2
p s2.pop  #print out the value popped from s2
```

The sequence:

7 — $\boxed{\text{0.5 mark}}$

12 — $\boxed{\text{0.5 mark}}$

$\left.\begin{array}{l}8\\28\end{array}\right\}$ — $\boxed{\text{0.5 mark}}$

Question 5b

5b. What is the sequence of values printed out by the following operations? (1.5 marks)

Question 5b

```
q1= Queue.new
q2 = Queue.new
q1.enqueue(7)
q1.enqueue(26)
q1.enqueue(8)
q1.enqueue(28)
q2.enqueue(q1.dequeue)
q2.enqueue(q1.dequeue)
q2.enqueue(q1.dequeue)
p q1.dequeue
q1 = q2
q2 = Queue.new
q2.enqueue(q1.dequeue)
q2.enqueue(q1.dequeue)
p q1.dequeue
q1 = q2
q2 = Queue.new
q2.enqueue(q1.dequeue)
p q1.dequeue
q1 = q2
q2 = Queue.new
p q1.dequeue
```

The sequence:

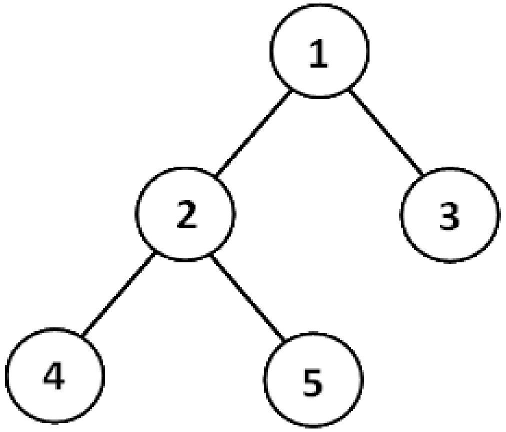
28 — $\boxed{\text{0.5 mark}}$

8 — $\boxed{\text{0.5 mark}}$

$\left.\begin{array}{l}\textbf{26}\\ \textbf{7}\end{array}\right\}$ — $\boxed{\text{0.5 mark}}$

Question 6a

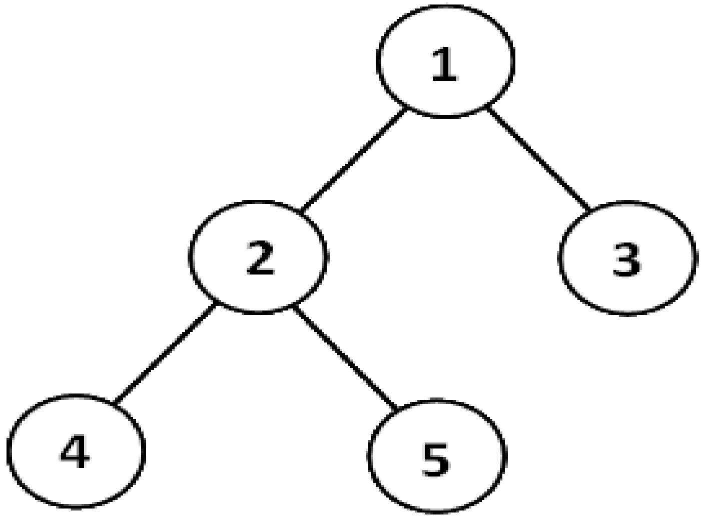
6a. What is the sequence in which the values of the nodes are printed (**line 6**) when running **mirrorIterativeQueue** with the following input binary tree A? (1 mark)



Binary Tree A

```
01:   def mirrorIterativeQueue(root):
02:       nodeQ = Queue()
03:       nodeQ.enqueue(root)
04:       while nodeQ.count() > 0:
05:           node = nodeQ.dequeue
06:           print(node.value) #print the value of the node
07:
08:           if(node.left != None):
09:               nodeQ.enqueue(node.left)
10:           if(node.right != None)
11:               nodeQ.enqueue(node.right)
12:
13:           temp = node.left
14:           node.setLeft(node.right) #set the left child of node
15:           node.setRight(temp) #set the right child of node
```

Question 6a (Answer)



Binary Tree A

The sequence:

1
2 } 0.5 mark

3
4
5 } 0.5 mark

Question 6b

6b. Suppose that the complexity of each **enqueue** or **dequeue** operation is **O(1)**. What is the overall Big O complexity of **mirrorIterativeQueue**? (1 mark)

Each node is enqueued and dequeued one time

O(*n*)
n is the number of nodes

O(2^k)
k is the height of the binary tree

1 mark

Question 6c

6c. The function **mirrorIterativeQueue** above is implemented using a queue (**line 2**). Rewrite it into another function **mirrorIterativeStack** using a stack instead of a queue. The template is given below, and you may fill in the blank lines. (2 marks)

Question 6c (Answer)

```
def mirrorIterativeStack(root):
    nodeS = Stack()
    nodeS.push(root)
    while nodeS.count() > 0:
        node = nodeS.pop()
        print(node.value)

        if node.left != None:
            nodeS.push(node.left)
        if node.right != None:
            nodeS.push(node.right)

        # swap left & right subtrees
        temp = node.left
        node.setLeft(node.right)
        node.setRight(temp)
```

0.5 mark

0.5 mark

0.5 mark

0.5 mark

Question 6d

6d. Design an algorithm to derive the mirror of a binary tree using recursion.

- Provide a pseudocode for mirrorRecursive using recursion. (2 marks)
- What is the Big O complexity of this recursive function? (1 mark)

O(*n*) — 1 mark

n is the number of nodes

O(2^k)

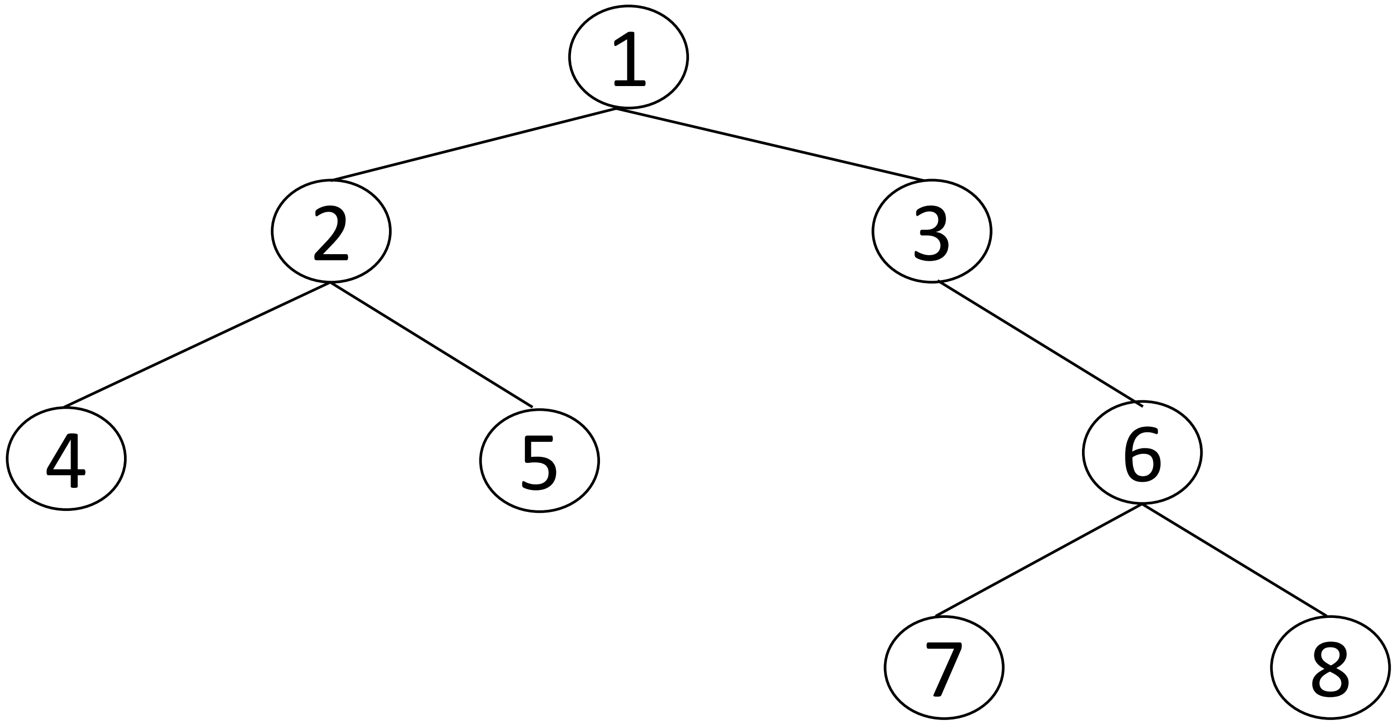
k is the height of the binary tree

```
def mirrofRecursive(root):
    if root != None:                 — 0.5 mark
        temp = root.left
        root.setLeft(root.right)     — 0.5 mark
        root.setRight(temp)

        mirrorRecursive(root.left)   — 0.5 mark
        mirrorRecursive(root.right)  — 0.5 mark
```

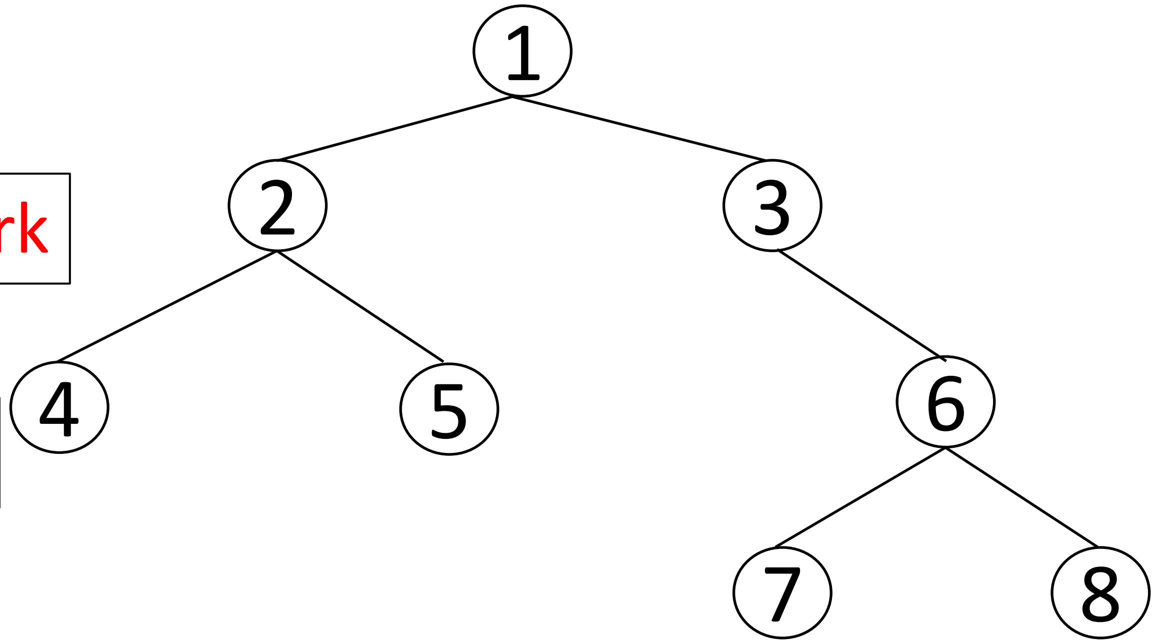
Question 7a

7a. Given a binary tree as follows:



Which traversal method(s), out of: preorder, inorder, postorder, will visit 7 before 3 on the tree above? (1 mark)

Question 7a (Answer)



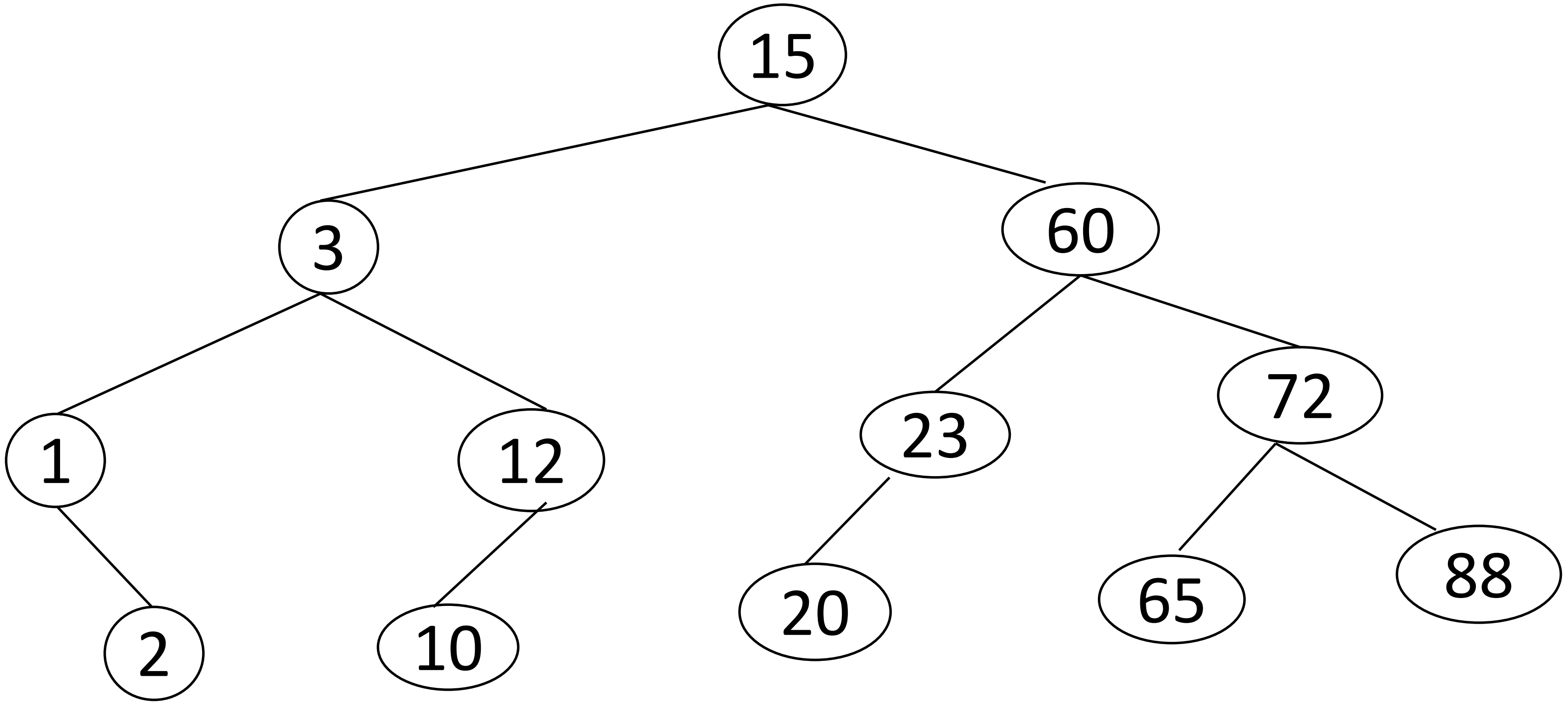
<u>Answer: Postorder</u> 1 mark

Postorder and … 0.5 mark

- Preorder (P L R): 1, 2, 4, 5, 3, 6, 7, 8
- Inorder (L P R): : 4, 2, 5, 1, 3, 7, 6, 8
- Postorder (L R P): 4, 5, 2, 7, 8, 6, 3, 1

Question 7b

7b. Given a list of integers: 15, 3, 12, 60, 72, 23, 10, 88, 20, 1, 65, 2. Draw the binary search tree (BST) obtained by inserting these integers, in the sequence as listed above, into an initially empty BST (Hint: 15 is the root node). (2 marks)



Question 7b

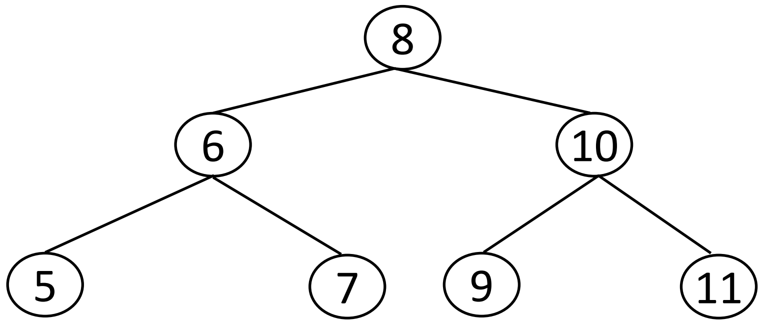
#of incorrect inserted nodes	Mark
0	2
1-3	1.5
4-6	1
7-9	0.5
10-12	0

- If the insertion is not followed the input sequence, and the tree is binary search tree: 1 mark

Question 7c

7c. Given an array of numbers, design an algorithm (in pseudocode or Python code) to check whether the array is a post-order traversal of a binary search tree or not. (2 marks)

For example: array = {5, 7, 6, 9, 11, 10, 8}, the algorithm should return true, since it is a post-order traversal of the following binary search tree:



For another example, if the input is b = [3, 1, 2], the algorithm should return false, since no BST containing these integers could possibly have b as its postorder traversal.

Your algorithm will be marked based on its complexity. Lower complexity translates to higher marks. (2 marks)

Question 7c (Answer)

Brute force: array = {a_1, a_2, …, a_n}

1. For each permutation of all the elements in the input array (n!)
2. Draw a BST from that permutation (go from the first element to the last element, inserting to the BST) (n^2)
3. Get the postorder sequence of the BST built in the previous step (n)
4. Compare the sequence with the input array, if they are the same, `return true` (n)

```
5. return false
```

Complexity: $O(n! * (n^2 + n + n)) = O(n! * n^2)$

1 mark

Question 7c (Answer)

Suppose the array is a postorder sequence of a binary search tree

- Algorithm:
  1. Draw the BST from its postorder sequence (i.e., the input array)
  2. Get the postorder sequence of the BST
  3. Compare the sequence with the input array, if they are the same, return true, otherwise, return false (as there is one and only one postorder sequence of a BST)

Question 7c (Answer)

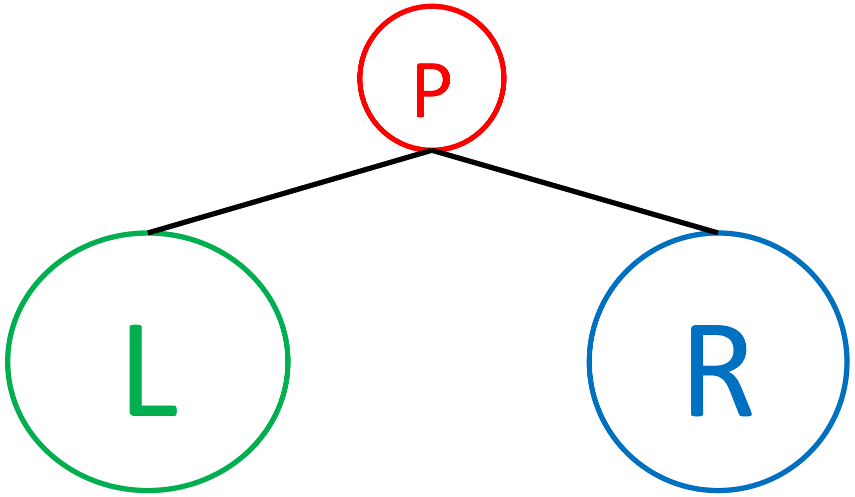
array = {5, 7, 6, 9, 11, 10, 8}

Draw the BST from its postorder sequence (i.e., the input array)

Suppose the array is a postorder sequence of a BST

→ Array = {L R P}

- Draw the L and R:
  - L: $L_1R_1P_1$
  - R: $L_2R_2P_2$



Call `draw(array, 0, array.length-1)`

```
draw(array, start, end)
1. if (start > end)
2.     return
3. insertToBST(array[end])
4. j = start-1
5. while (array[j+1] < array[end])
6.     j += 1;
7. if (j >= start)
8.     draw(array, start, j)
9. draw(array, j+1, end-1)
```

Question 7c (Answer)

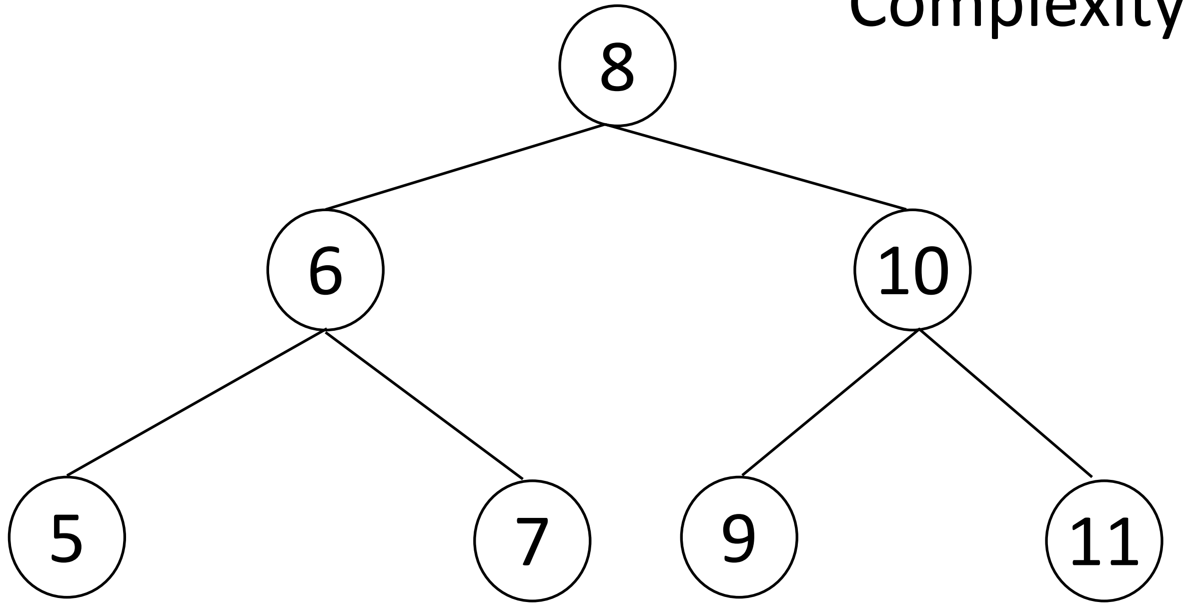
Draw the BST from its postorder sequence (i.e., the input array)

- Another approach: Draw the BST by inserting backward the elements of the input array

E.g.: array = {5, 7, 6, 9, 11, 10, 8}

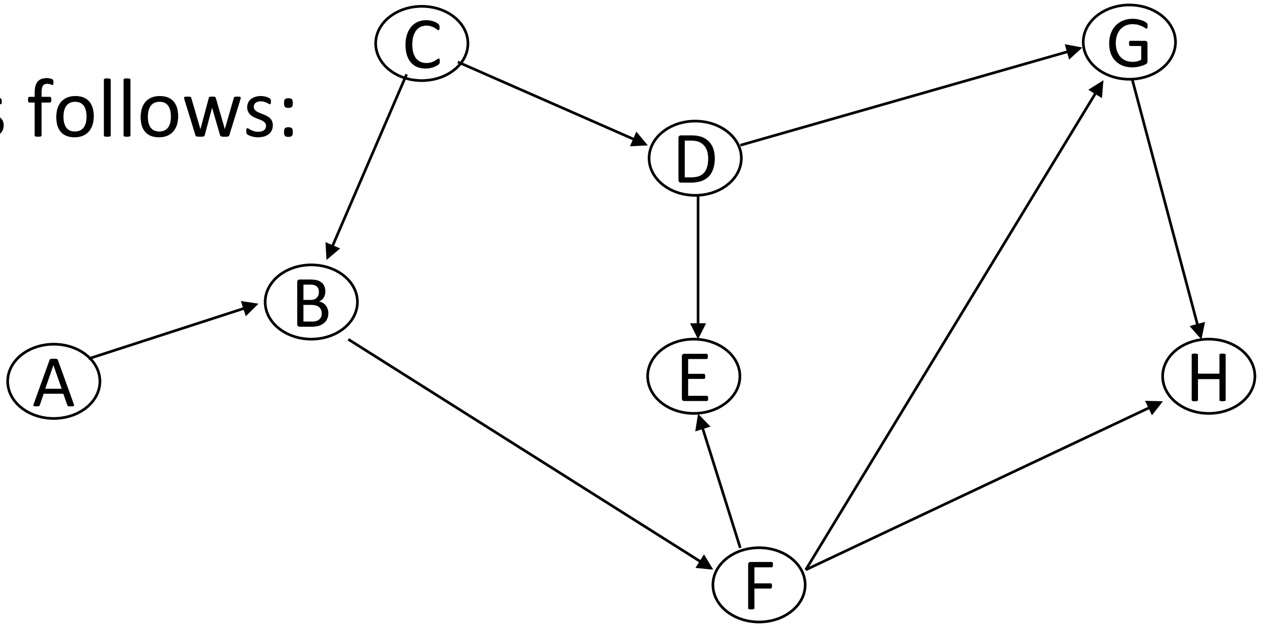
Complexity: $O(n^2)$

2 marks



Question 8a

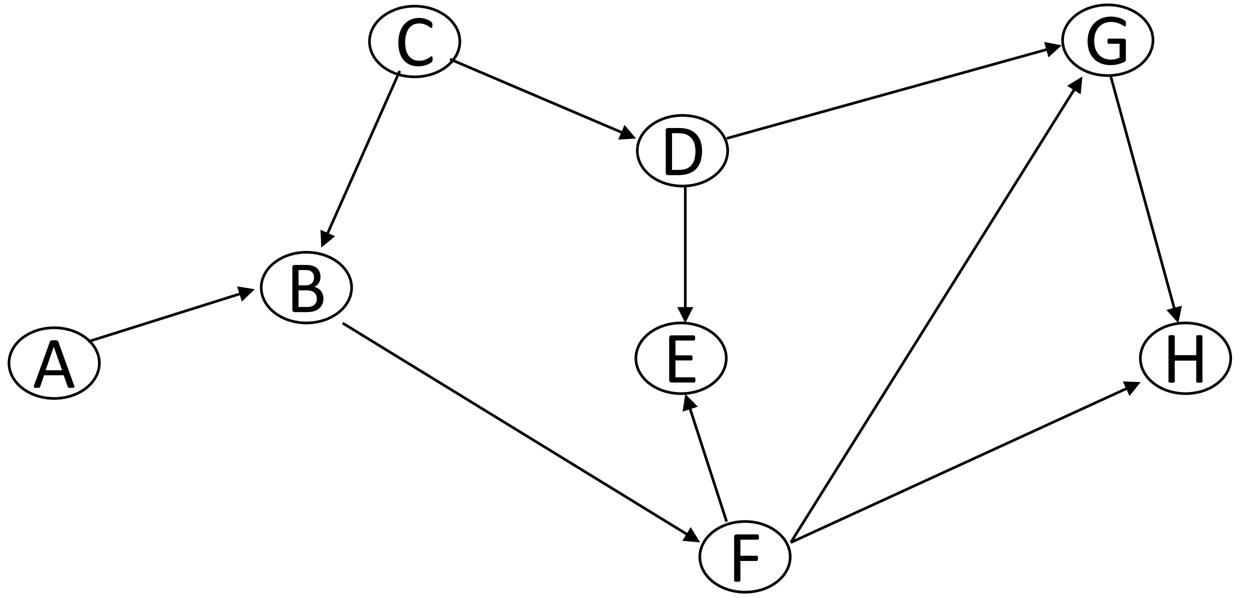
Given a graph X as follows:



8a. Write down the adjacency list of the graph X above. The neighbours of each vertex are to be listed in alphabetical order. (1 mark)

Question 8a (Answer)

A	B
B	F
C	B D
D	E G
E	-
F	E G H
G	H
H	-

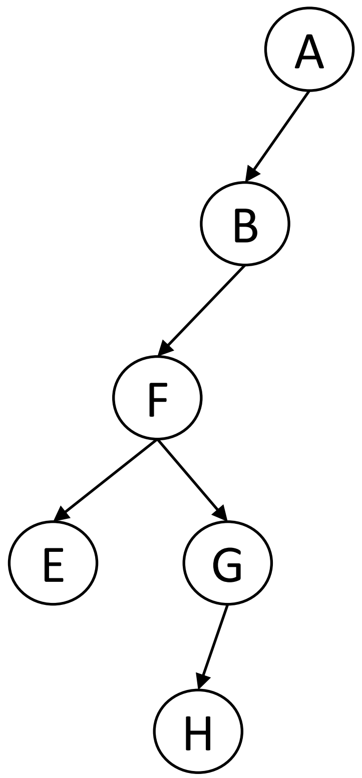


- All correct: 1 mark
- Incorrect for 1 to 5 vertices: 0.5 mark
- Incorrect for more than 5 vertices: 0 mark

Question 8b

8b. What is the sequence of nodes visited by recursive depth-first-search when traversing the graph X, starting from A, based on the adjacency list above? (1 mark)

A	B
B	F
C	B D
D	E G
E	-
F	E G H
G	H
H	-



Sequence: A B F E G H

- Correct sequence: 1 mark
- Using stack: 0.5 mark
- Correct sequence but add C, D: 0.5 mark

Question 8c

8c. What is the sequence of nodes visited by breadth-first-search on graph X, starting from **vertex C**, based on the adjacency list above? Clearly show your working. You should enqueue each vertex's neighbors according to the sequence in the adjacency list. (1 mark)

Question 8c

Answer: C B D F E G H

1 mark

A	B
B	F
C	B D
D	E G
E	-
F	E G H
G	H
H	-

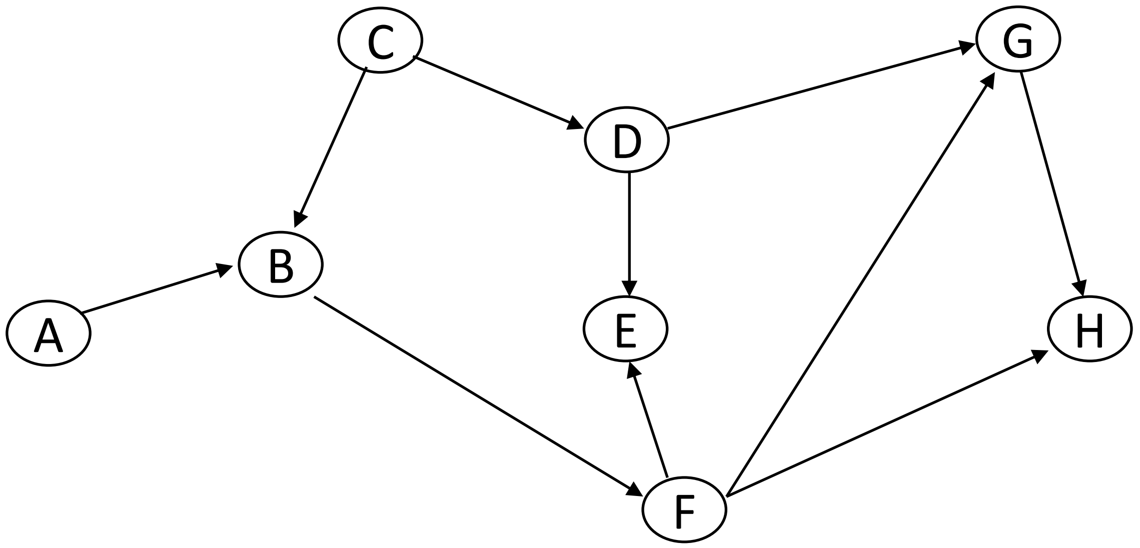
Step	Dequeued	Visited/enqueued	Queue: [H..T]
0		C	C
1	C	B, D	B,D
2	B	F	D,F
3	D	E,G	F,E,G
4	F	H	E,G,H
5	E		G,H
6	G		H
7	H		

- Visit nodes more than once: 0.5 mark
- Starting from wrong node (i.e., not C): 0 mark

Question 8d

8d. Find two topological orderings in graph X. (2 marks)

Topological Ordering: A sequence or permutation of all vertices in a graph G, such that all edges "point forward".



Topological Orderings:

1. ACBDFEGH
2. ACDBFEGH
3. CADBFEGH
4. CABDFEGH
5. …