DSA Exam

[redacted]

(Time complexity)

1.For a given function g(n), we denote by O(g(n)) the set of functions

A. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= g(n) <= c*f(n) for all n>= n0}
B. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= g(n) <= c*f(n) for all n<= n0}
C. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= f(n) <= c*g(n) for all n>= n0}
D. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= f(n) <= c*g(n) for all n<= n0}

2.For a given function g(n), we denote by Omega(g(n)) the set of functions

A. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c+g(n) <= f(n) for all n>= n0}
B. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c+g(n) <= f(n) for all n<= n0}
C. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c*g(n) <= f(n) for all n>= n0}
D. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c*g(n) <= f(n) for all n<= n0}

3.For a given function g(n), we denote by Theta(g(n)) the set of functions

A. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c1+g(n) <= f(n) <= c2+g(n) for all n>= n0}
B. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c1+g(n) <= f(n) <= c2+g(n) for all n<= n0}
C. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c1*g(n) <= f(n)<= c2*g(n) for all n>= n0}
D. O(g(n)) = {f(n): there exists positive constants c and n0 such that 0<= c1*g(n) <= f(n)<= c2*g(n) for all n<= n0}

(Properties of data structures)

1. The 2 dimensional array(Matrix) is

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

2.The 1 dimensional array(vector) is

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

3.The stack is

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

4.The queue is

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

5.The linked list is

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

6.The table is:

   A. static and sequential data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.
   D. dynamic and associative data structure.

7.The tree is:

   A. dynamic and hiearchical data structure.
   B. dynamic and sequential data structure.
   C. static and associative data structure.

D. dynamic and associative data structure.

8.The set is:

A. static and sequential data structure.
B. dynamic and sequential data structure.
C. static and associative data structure.
D. dynamic and without structure data structure.

9.The multi-set is:

A. static and sequential data structure.
B. dynamic and sequential data structure.
C. static and associative data structure.
D. dynamic and without structure data structure.

10.The sparce tree is:

A. static and sequential data structure.
B. dynamic and sequential data structure.
C. static and associative data structure.
D. dynamic and associative data structure.

11. The matrix properties are?

A. static and sequential
B. dynamic and sequential
C. static and associative
D. dynamic and associative

12. The vector properties are?

A. static and sequential
B. dynamic and sequential
C. static and associative
D. dynamic and associative

(Tree, table and linked lists)

1.For self ordering table we can use:

A. linked representation and serial search
B. linked representation and binary search
C. continuous representation and serial search
D. continuous representation and serial search

2.We cannot construct original binary tree from:

A. preorder,inorder and post order traversal
B. preorder and post order traversal
C. inorder traversal
D. None of them

3.We can construct expression tree from:

A. prefix,Infix and postfix expression
B. prefix and postfix expression
C. infix expression
D. None of them

4.The binary-search-tree property allows us to print out all the keys in a binary search tree in sorted order by a simple recursive algorithm, called:

A. preorder tree walk
B. inorder tree walk
C. postorder tree walk

5. The average-case time of searching in a hash-table is

A. O(log n)
B. O(n)
C. O(n*n)
D. O(1)

6. The average-case time of searching in a linked list is:

A. O(log n)
B. O(n)
C. O(n2)
D. O(1)

7. We cannot reconstruct the original expression tree from?

- Infix

8. We can use the following algorithm and data structure to find the shortest path between two vertices of a directed graph.

- Breadth first search using a queue

(Red-Black tree)

1.A red-black tree with n internal nodes has height at most:

A. log2(n)
B. 2∗log2(n)

C. log2(n+1)
D. 2*log2(n+1)

2.For each node of a red-black tree, all simple paths from the node to descendant leaves:

A. contain the same number of black nodes.
B. contain the same number of red nodes.
C. contain different number of black nodes.
D. contain different number of red nodes.

3. Find the FALSE property of the following description of red-black tree:

A. Every node is either red or black.
B. The root is black.
C. Every leaf (NIL) is black.
D. If a node is red, then both its children are black.
E. For each node, all simple paths from the node to descendant leaves contain the same number of red nodes.

4. Find the FALSE property of the following description of red-black tree:

A. Every node is either red or black.
B. The root is black.
C. Every leaf (NIL) is black.
D. If a node is black, then both its children are red.
E. For each node, all simple paths from the node to descendant
F. leaves contain the same number of black nodes.

5. The average case time of searching in a red black tree is?

- O(log2n)

(B-tree)

1.Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer $t \geq 2$ called the minimum degree of the B-tree. Choose the correct B-tree property:

A. Every node other than the root must have at least 2*t−1 keys.
B. Every node other than the root may contain at most 2*t−1 keys.
C. Every node may contain at most 2*t−1 keys.
D. Every node must have at least 2*t−1 keys.

2.The minimum value of degree of B-tree is:

A. 0
B. 1

C. 2
D. 3

3. Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer t ≥ 2 called the minimum degree of the B-tree. Choose the correct B-tree property:

A. Every node other than the root must have at least t − 1 keys.
B. Every node may contain at most t − 1 keys.
C. Every node other than the root may contain at most t−1 keys.
D. Every node must have at least t − 1 keys.

4. Let us denote the minimum degree of the b tree by t. Find the true sentence.

- Every internal node of the B-tree may have at most 2*t pointers

(Parallel algorithm)

1. We have four algorithms solving the same problem. First solution runs in polynomial time, second solution runs in exponential time, third solution runs in logarithmic time and the last solution runs in linear time. Which is the fastest?

A. polynomial
B. exponential
C. logarithmic
D. linear

2.In the multithreaded matrix multiplication function P-SQUARE-MATRIX-MULTIPLY(A,B) function the first 5 rows are:

1.n= A.rows

2.let C be a new n*n Matrix

3.parallel for i=1to n

4.parallel for j=1 to n

5.Cij = 0

continue…

A. 6.for k = 1 to n
   7. Cij = aik . bkj
   8.return c
B. 6. for k = 1 to n
   7.Cij = Cij + aik . bkj
   8. return C
C. 6.parallel for k = 1 to n
   7. Cij = aik . bkj

8.return c

D. 6.parallel for k = 1 to n
   7.Cij = Cij + aik . bkj
   8. return C

3. In the LIST-DELETE(L,x) function the first 3 rows are:

1. if x.prev != NIL

2. x.prev.next =x.next

3. else L.head=x.next

continue...

A. 4. if x.next=NIL
   5. x.next.prev=x.prev

B. 4. if x.next!=NIL
   5. x.next.prev=x.prev

C. 4. if x.next=NIL
   5. x.next.prev=x.next

D. 4. if x.next!=NIL
   5. x.next.prev=x.next

(P,NP and NP complete)

1. A problem is NP complete if:

A. it is in P and it is in NP.
B. it can be verified in polynomial time.
C. it is in NP and it can be verified in polynomial time.
D. it is in NP and it is NP-hard.

2.A problem is NP-complete if:

A. it can be solved in polynomial time.
B. it can be verified in polynomial time.
C. it can be solved in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be solved in polynomial time.
D. it can be verified in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be verified in polynomial time.

3.A problem is NP if:

A. it can be solved in polynomial time.
B. it can be verified in polynomial time.
C. it can be solved in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be solved in polynomial time.
D. it can be verified in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be verified in polynomial time.

4.A problem is P if:

A. it can be solved in polynomial time.
B. it can be verified in polynomial time.
C. it can be solved in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be solved in polynomial time.
D. it can be verified in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be verified in polynomial time.

5.The radix sort(A,d) function is the following:

A. 1.for i 1 to d
   2. use a sort to sort Array A on digit i;

B. 1.for i 1 to d
   2. use a sort to sort Array A on digit d;

C. 1.for i 1 to d
   2. use a stable sort to sort Array A on digit i;

D. 1.for i 1 to d
   2. use a stable sort to sort Array A on digit d;

1. The 2 dimensional array is:
A. static and sequential data structure.
B. dynamic and sequential data structure.
C. static and associative data structure.
D. dynamic and associative data structure.

2. The average-case time of searching in a hash-table is:
A. O(log n)
B. O(n)
C. O(n*n)
D. O(1)

3. For each node of a red-black tree, all simple paths from the node to descendant leaves:

A. contain the same number of black nodes.
B. contain the same number of red nodes.
C. contain different number of black nodes.
D. contain different number of red nodes.

4. Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer $t \geq 2$ called the minimum degree of the B-tree. Choose the correct B-tree property:

A. Every node other than the root must have at least $2*t-1$ keys.
B. Every node other than the root may contain at most $2*t-1$ keys.
C. Every node may contain at most $2*t-1$ keys.
D. Every node must have at least $2*t-1$ keys.

5. We have four algorithms solving the same problem. First solution runs in polynomial time, second solution runs in exponential time, third solution runs in logarithmic time and the last solution runs in linear time. Which is the fastest?

A. polynomial
B. exponential
C. logarithmic
D. linear

6. A problem is NP complete if

A. it is in P and it is in NP.
B. it can be verified in polynomial time.
C. it is in NP and it can be verified in polynomial time.
D. it is in NP and it is NP-hard.

1.) The average-case time of searching in a linked list is
A. $\mathcal{O}(1)$.
B. $\mathcal{O}(n)$.
C. $\mathcal{O}(n^2)$.
D. $\mathcal{O}(log_2 n)$.

2.) The vector is
A.) dynamic and sequential data structure.
B.) static and associative data structure.
C.) dynamic and associative data structure.
D.) static and sequential data structure.

3.) We can not reconstruct the original expression tree from
A.) postfix expression.
B.) prefix expression.
C.) infix expression.

4.) A red-black tree with n internal nodes has height at most
A.) $log_2(n)$.
B.) $2 * log_2(n)$.
C.) $log_2(n+1)$.
D.) $2 * log_2(n+1)$.

5.) Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer t called the minimum degree of the B-tree.
Choose the correct B-tree property.
A.) Every node other than the root may contain at most $2 * t - 1$ keys.
B.) Every node other than the root must have at least $2 * t - 1$ keys.
C.) Every node may contain at most $2 * t - 1$ keys.
D.) Every node must have at least $2 * t - 1$ keys.

6.) The problem A is in NP if
A.) it can be solved in polynomial time and for every problem B in NP, there is a polynomial-time reduction from B to A.
B.) it can be verified in polynomial time and for every problem B in NP, there is a polynomial-time reduction from B to A.
C.) it can be verified in polynomial time.
D.) it can be solved in polynomial time.

Name: Bilal Arshad

Neptun code: PS79FI

1.) For a given function $g(n)$, we denote by $\mathcal{O}(g(n))$ the set of functions

A.) $\mathcal{O}(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq g(n) \leq c * f(n)$ for all $n \geq n_0\}$.

B.) $\mathcal{O}(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq g(n) \leq c * f(n)$ for all $n \leq n_0\}$.

C.) $\mathcal{O}(g(n)) = \{f(n)$: there exst positive constants c and n_0 such hat $0 \leq f(n) \leq c * g(n)$ for all $\geq n_0\}$.

D.) $\mathcal{O}(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq c * g(n)$ for all $n \leq n_0\}$.

The 2 dimensional array is

A.) static and sequential data structure.
B.) dynamic and sequential data structure.
C.) static and associative data structure.
D.) dynamic and associative data structure.

or a self-ordering table we use

) linked representation and serial search.
) linked representation and binary search.
) continuous representation and serial search.
continuous representation and binary search.

the expression tree from

Name:

Neptun code:

1.) For a given function $g(n)$, we denote by $\mathcal{O}(g(n))$ the set of functions

A.) $\mathcal{O}(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq g(n) \leq c * f(n)$ for all $n \geq n_0\}$.

B.) $\mathcal{O}(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq g(n) \leq c * f(n)$ for all $n \leq n_0\}$.

C.) $\mathcal{O}(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq c * g(n)$ for all $n \geq n_0\}$.

D.) $\mathcal{O}(g(n)) = \{f(n) :$ there exist positive constants c and n_0 such that $0 \leq f(n) \leq c * g(n)$ for all $n \leq n_0\}$.

2.) The 2 dimensional array is

A.) static and sequential data structure.
B.) dynamic and sequential data structure.
C.) static and associative data structure.
D.) dynamic and associative data structure.

3.) For a self-ordering table we use

A.) linked representation and serial search.
B.) linked representation and binary search.
C.) continuous representation and serial search.
D.) continuous representation and binary search.

4.) We can reconstruct the expression tree from

A.) prefix expression, infix expression and postfix expression.
B.) prefix expression and postfix expression.
C.) infix expression.
D.) none of them.

5.)A red-black tree with n internal nodes has height at most

A.) $log_2(n+1)$.

B.) $2 * log_2(n+1)$.

C.) $log_2(n)$.

D.) $2 * log_2(n)$.

6.) Nodes have lower and upper bounds on the number of keys they can contain. We express these bounds in terms of a fixed integer $t \geq 2$ called the minimum degree of the B-tree. Choose the correct B-tree property:

A. Every node other than the root must have at least $2 * t - 1$ keys.
B. Every node other than the root may contain at most $2 * t - 1$ keys.
C. Every node may contain at most $2 * t - 1$ keys.
D. Every node must have at least $2 * t - 1$ keys.

Name:

Neptun code:

1.) For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions

A.) $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq c + g(n) \leq f(n)$ for all $n \leq n_0\}$.

B.) $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq c + g(n) \leq f(n)$ for all $n \geq n_0\}$.

C.) $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq c * g(n) \leq f(n)$ for all $n \leq n_0\}$.

D.) $\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq c * g(n) \leq f(n)$ for all $n \geq n_0\}$.

2.) The stack is

A.) static and sequential data structure.
B.) dynamic and sequential data structure.
C.) static and associative data structure.
D.) dynamic and associative data structure.

3.) We can reconstruct the original binary tree from

A.) preorder, inorder and postorder traversal.
B.) preorder and postorder traversal.
C.) inorder traversal.
D.) none of them.

4.) Find the FALSE property of the following description of red-black tree:

1. Every node is either red or black.
2. The root is black.
3. Every leaf (NIL) is black.
4. If a node is black, then both its children are red.
5. For each node, all simple paths from the node to descendant leaves contain the same number of black nodes.

5.) The minimum value of the minimum degree of the B-tree is:

A.) 0.

B.) 1.

C.) 2.

D.) 3.

6.) In the multithreaded matrix multiplication function P-SQUARE-MATRIX-MULTIPLY(A,B) function the first 5 rows are:

1. n=A.rows
2. let C be a new $n \times n$ matrix
3. parallel for i=1 to n
4. parallel for j=1 to n
5. $c_{ij} = 0$

continue...

A.)
6. for k=1 to n
7. $c_{ij} = a_{ik} \cdot b_{kj}$
8. return C

B.)
6. for k=1 to n
7. $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$
8. return C

C.)
6. parallel for k=1 to n
7. $c_{ij} = a_{ik} \cdot b_{kj}$
8. return C

D.)
6. parallel for k=1 to n
7. $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$
8. return C

7.) A problem is in P if

A.) it can be solved in polynomial time.
B.) it can be verified in polynomial time.
C.) it can be solved in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be solved in polynomial time.
D.) it can be verified in polynomial time and there is a polynomial time reduction algorithm for each other problems, which can be verified in polynomial time.

8.) The RADIX-SORT(A,d) function is the following:

A.)
1. for i=1 to d
2. use a sort to sort array A on digit i

B.)
1. for i=1 to d
2. use a sort to sort array A on digit d

C.)
1. for i=1 to d
2. use a stable sort to sort array A on digit i

D.)
1. for i=1 to d
2. use a stable sort to sort array A on digit d

Each good answer is 6 points, each wrong answer is -2 points.

25–30 points: 2
31–36 points: 3
37–42 points: 4
43–48 points: 5

① function first(n)
i = 2
m = n * n
while i < m
i = i * i
end function

running time is
① $\Theta(n)$
✓ ② $\Theta(\log_2(n))$
③ $\Theta(n^2)$
④ $\Theta(2^n)$

n	n^2	step
1	1	0
2	4	1 ✓
3	9	
4	16	
5	25	
6	36	
7	49	
8	64	
9	81	
10	100	

② A = {1, 1, 3, 5, 5, 6, 6}
B = {2, 2, 3, 6, 6, 6, 6}

A - B = ? A \ B = ?

A) 2, 2, 0, 0, 2, 2
B) 2, 0, 0, 0, 2, 2
✓ C) 2, 0, 0, 0, 2, 0
D) 2, 2, 0, 0, 2, 0

Standard Representation

2	0	1	0	2	2

0	2

③

0	3	0
0	0	0
0	0	1
2	0	0

4x3

One column representation

Row	1
Colum	2
element	3

(a) 3 2 1
(b) 3 1 2
(c) 2 1 3
✓ (d) 3 2 1

cyclic

(4) | * | 2 | 3 | _ | _ |

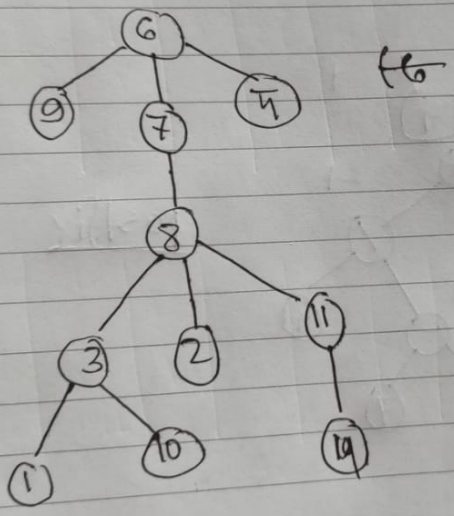
enque (5, 4)
deq (5)
enq (5, 5)
enq (5, 6) – overflow
deq (5)
deq (5)

a) | 4 | 5 | 6 | - | - |

b) | 6 | - | - | 4 | 5 |

c) overflow
d) underflow

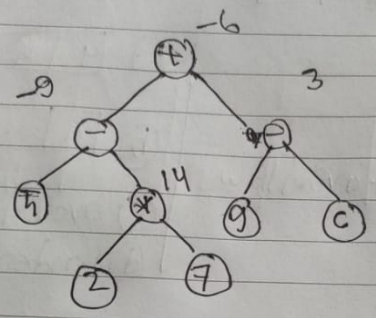
(5)



Hierical represe

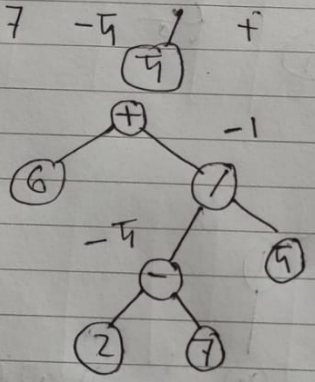
(6 (9) (7 (8 (3 (1) (10)) (2) (11 (19)) (5))

(6) + - 5 x 2,7 -9,6 ⟶ -6

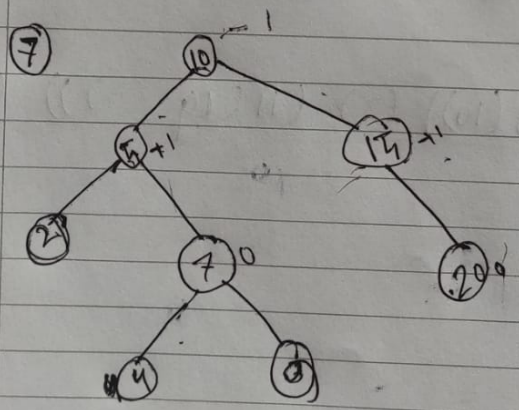


Prefix

(6) 6, 2, 7 -5 * +

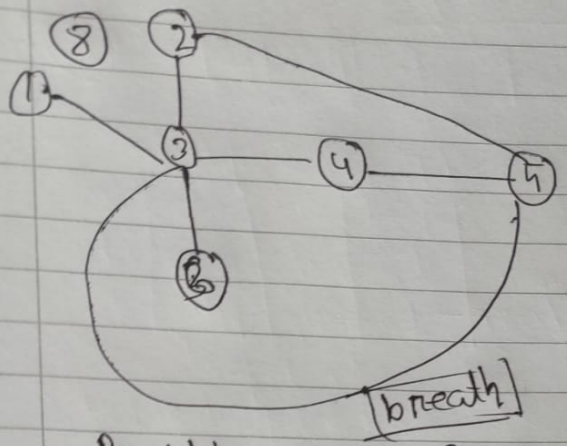


postfix



Balanced	search
✓	X

(7) Balaced / not search



Possible depth first traversal,

in (a) 1, 3, 5, 6, 2, 4

(b) 5, 6, 3, 4, 2, 1 (a)

(c) 6, 4, 5, 3, 2, 1

(d) 6, 1, 2, 3, 4, 5

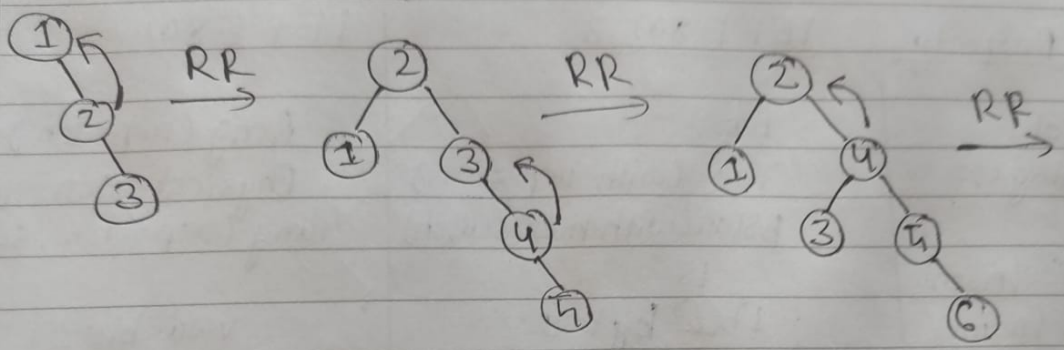
possible breath first traversal.

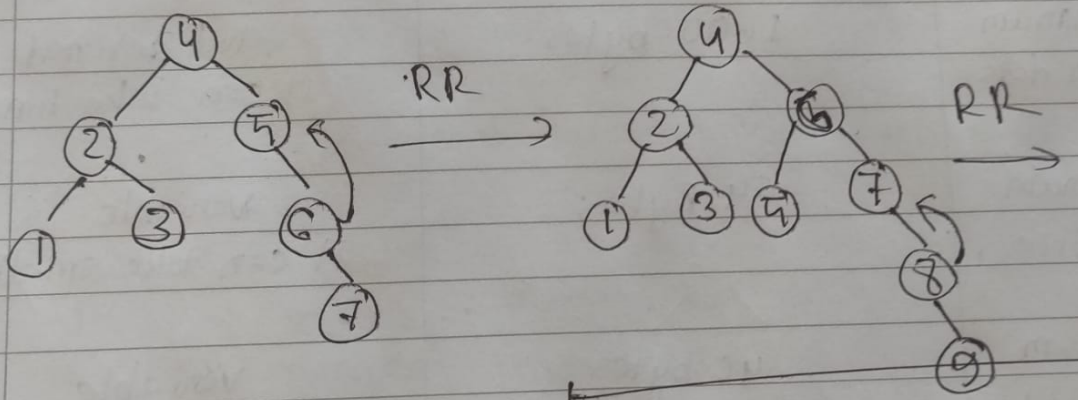
Q = (a) 2, 5, 3, 6, 4, 1 (d) + (b)

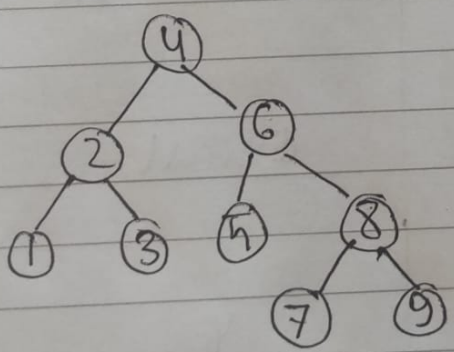
(b) 2, 3, 5, 6, 4, 6

(c) 1, 2, 3, 4, 5, 6

(d) 3, 1, 2, 4, 5, 6

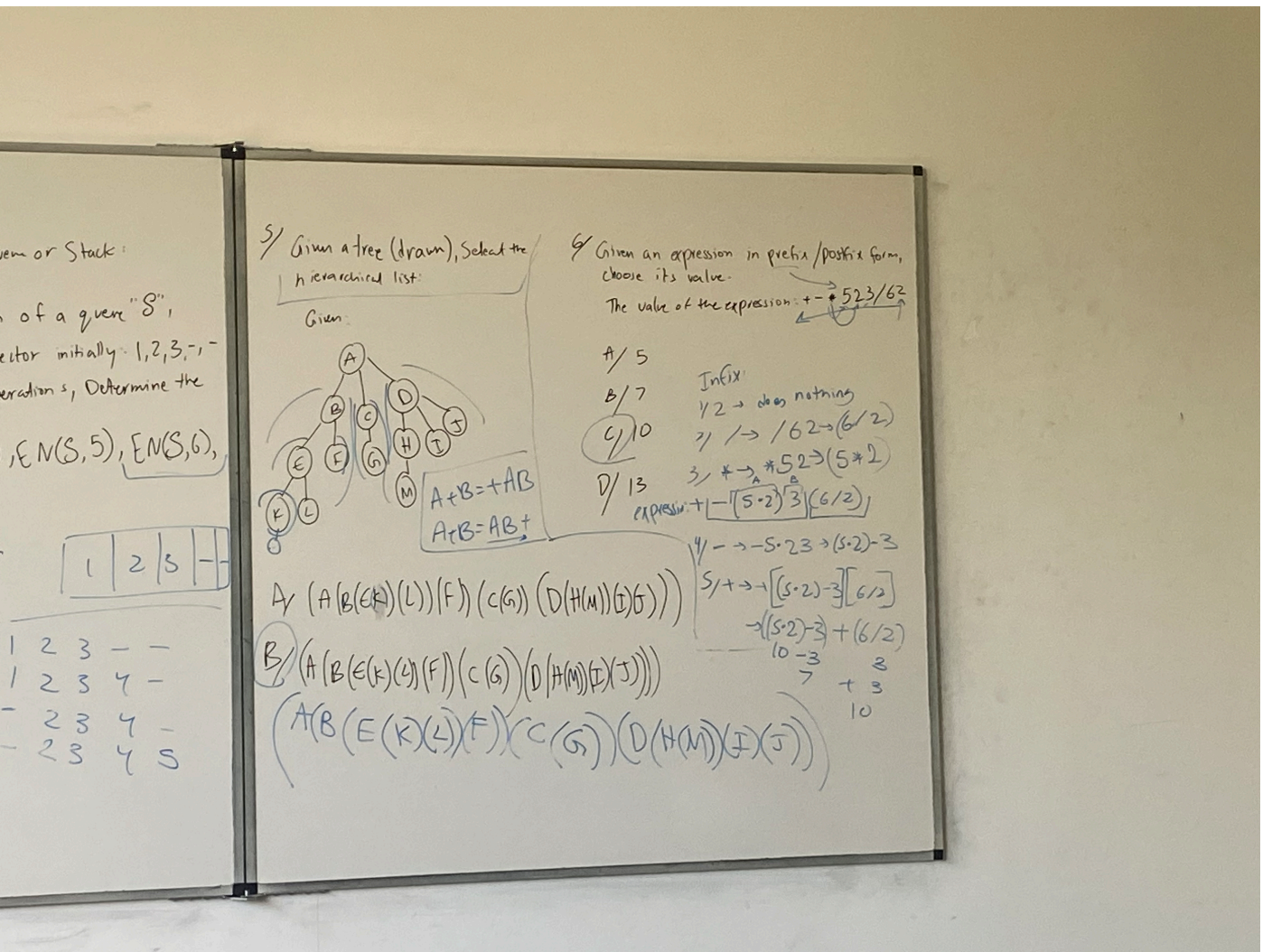


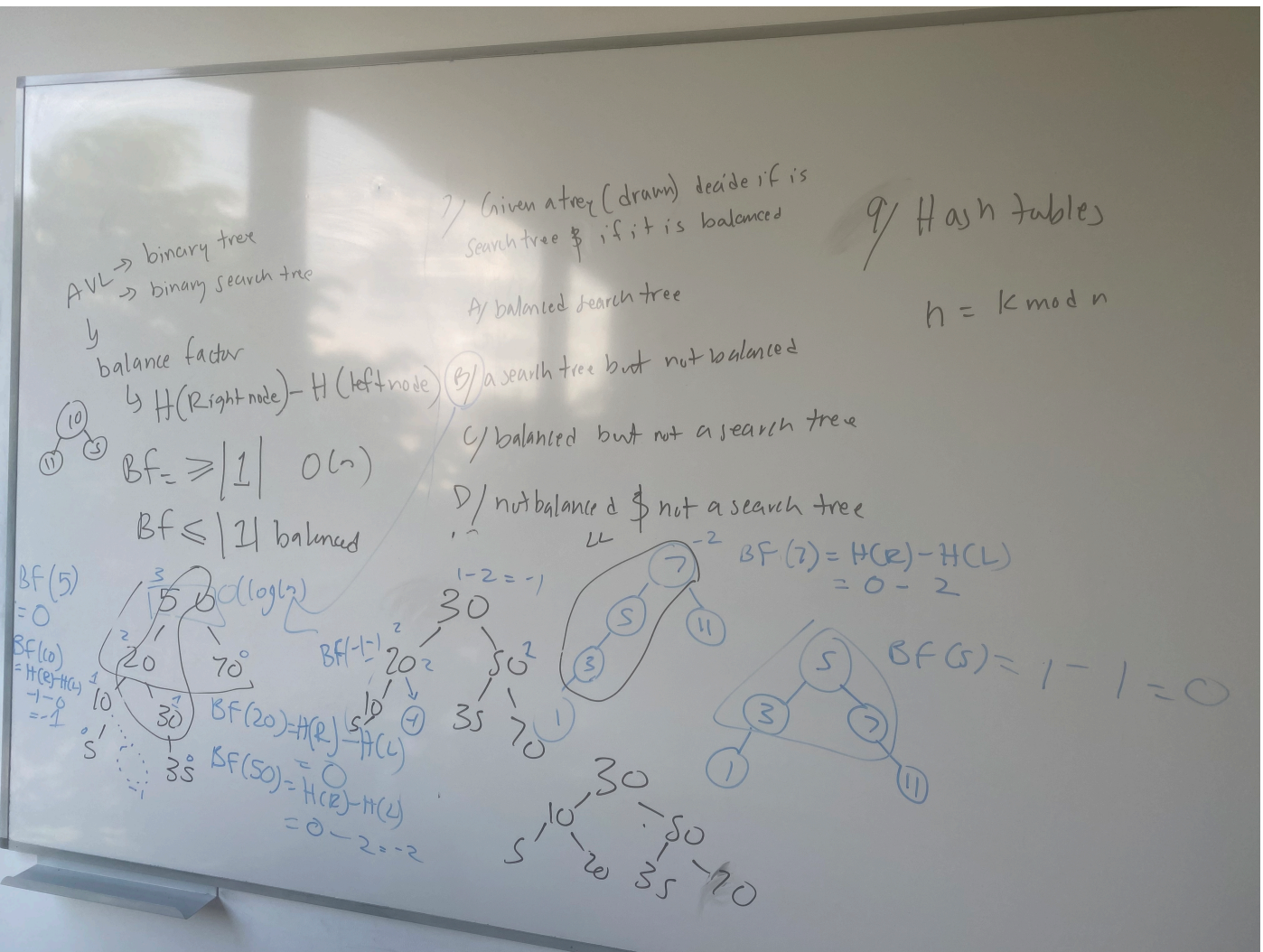


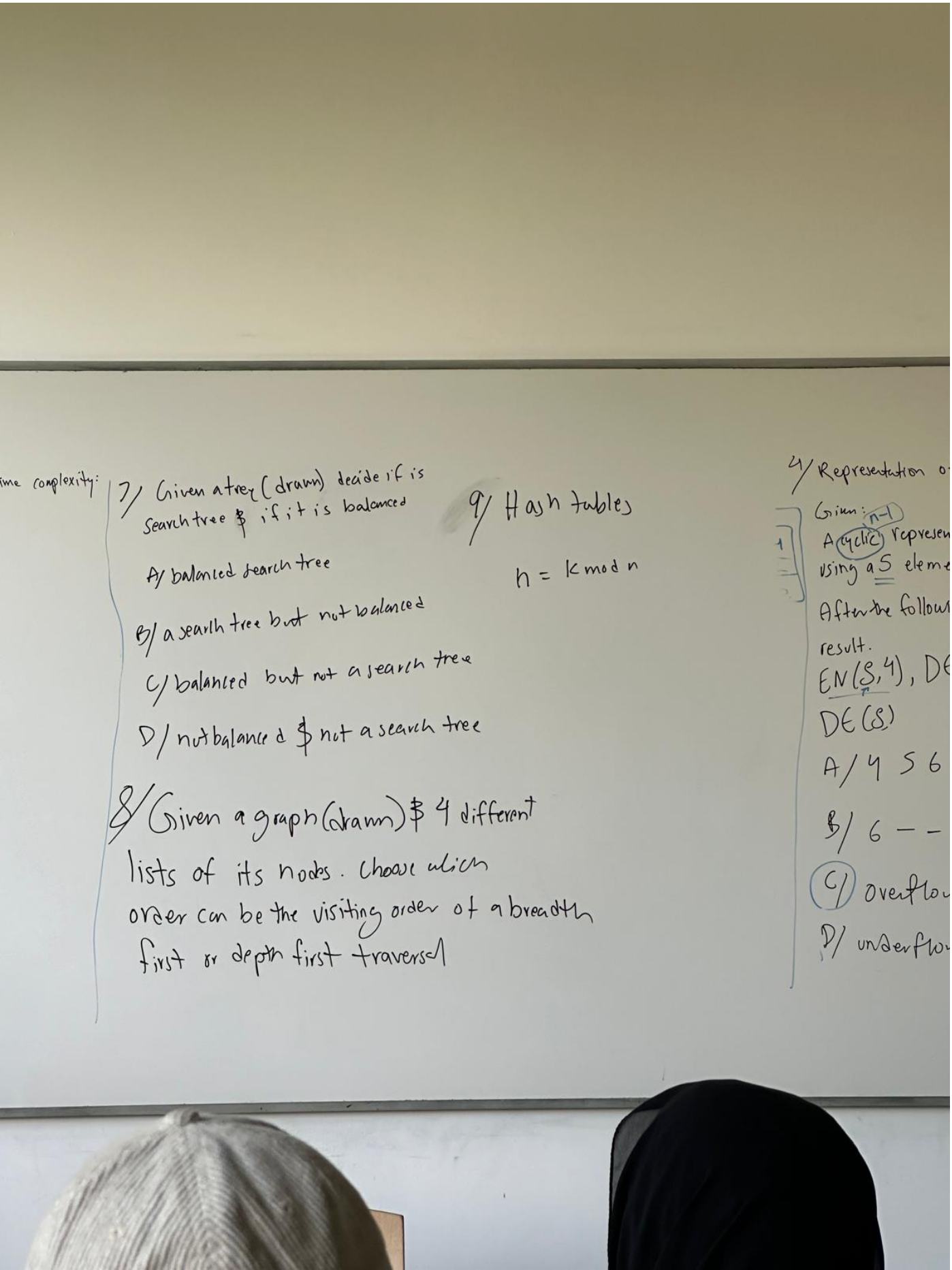


1


1/ Given a pseudocode, decide the time complexity
Given:
function first-code(n)
n=2
no. of times we entered loop
1
n=4
no... loop
2
n=6
A/ Θ(n) X
B/ Θ(log(n))
C/ Θ(n²) X
D/ Θ(2ⁿ) X
for i in range(n)
end function
2/ Given 2 multisets with elements from 1→6, Calculate A∪B, A∩B or A/B
max occurance
min occurance
Given:
A={1 1 3 5 5 6 6 7}
B={2 2 3 6 6 6 6}
The value of A/B A−B
X A/ 2 2 0 0 2 2
B/ 2 0 0 0 2 2
C/ 2 0 0 0 2 0
X D/ 2 2 0 0 2 0
A−B = -neg: 0
A∩B=
3/ Matrix representation
Given:
2 column of the 3 row representation
A/
B/
D/
row
column
value
4/ Representation of a queue or Stack:
Given
A(n-1)(n-1) representation of a queue "S", using a S element vector initially: 1,2,3,-,-
After the following operations, Determine the result
EN(S,4), DE(S), EN(S,5), EN(S,6), DE(S)
A/ 4 5 6 - -
B/ 6 - - 4 5
C/ Overflow
D/ underflow
5/ Given a tree (drawn), Select
hierarchical list
Given:
A+B=
A/
B/


em or Stack:
of a queue "S",
ector initially: 1,2,3,-,-
erations, Determine the
EN(S,5), EN(S,6),
1 2 S - -
1 2 3 - -
1 2 3 4 -
2 3 4 -
2 3 4 5
5/ Given a tree (drawn), Select the hierarchical list:
Given:
A+B=+AB
A+B=AB+
A/ (A(B(E(K)(L))(F))(C(G))(D(H(M))(I)(J)))
B/ (A(B(E(K)(L))(F))(C(G))(D(H(M))(I)(J)))
6/ Given an expression in prefix/postfix form, choose its value.
The value of the expression: +-*523/62
A/ 5
B/ 7
C/ 10
D/ 13
Infix
1/2 → does nothing
2/ /→ /62→(6/2)
3/ *→ *52→(5*2)
expression: +-*523/62
4/ -→-5·23→(5·2)-3
5/ +→+[(5·2)-3][6/2]
→[(5·2)-3]+(6/2)
10 -3 7
8 + 3 10


AVL → binary tree
→ binary search tree
balance factor
H(Right node) − H(left node)
Bf ≥ |1| O(n)
Bf ≤ |1| balanced
BF(5) = 0
BF(10) = H(R) − H(L) = −1 − 0 = −1
BF(20) = H(R) − H(L) = 0
BF(50) = H(R) − H(L) = 0 − 2 = −2
O(log2)
1) Given a tree (drawn) decide if is search tree & if it is balanced
A) balanced search tree
B) a search tree but not balanced
C) balanced but not a search tree
D) not balanced & not a search tree
9) Hash tables
h = k mod n
BF(7) = H(R) − H(L) = 0 − 2
BF(5) = 1 − 1 = 0


ime complexity:
7/ Given a tree (drawn) decide if is search tree $ if it is balanced
A/ balanced search tree
B/ a search tree but not balanced
C/ balanced but not a search tree
D/ not balanced $ not a search tree
9/ Hash tables
h = k mod n
8/ Given a graph (drawn) $ 4 different lists of its nodes. Choose which order can be the visiting order of a breadth first or depth first traversal
4/ Representation o
Given: n-1
A (cyclic) represen
using a S eleme
After the follow
result.
EN(S,4), DE
DE(S)
A/ 4 5 6
B/ 6 - -
C/ overflo
D/ underflo

Graph traversal

[ ] → queue

1/ Breadth first search

↳ traverses the nodes in increasing order of distance from the source.

↳ uses a queue → FIFO to implement the order in which nodes are visited.

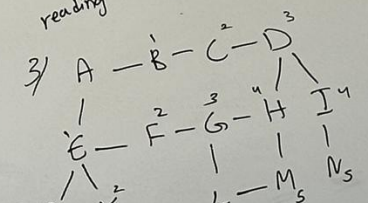
2/ Depth first search:

↳ traverses the adjacent nodes of the adjacent nodes one by one.

↳ uses a stack, so it may prioritize those further away.

reading from A:

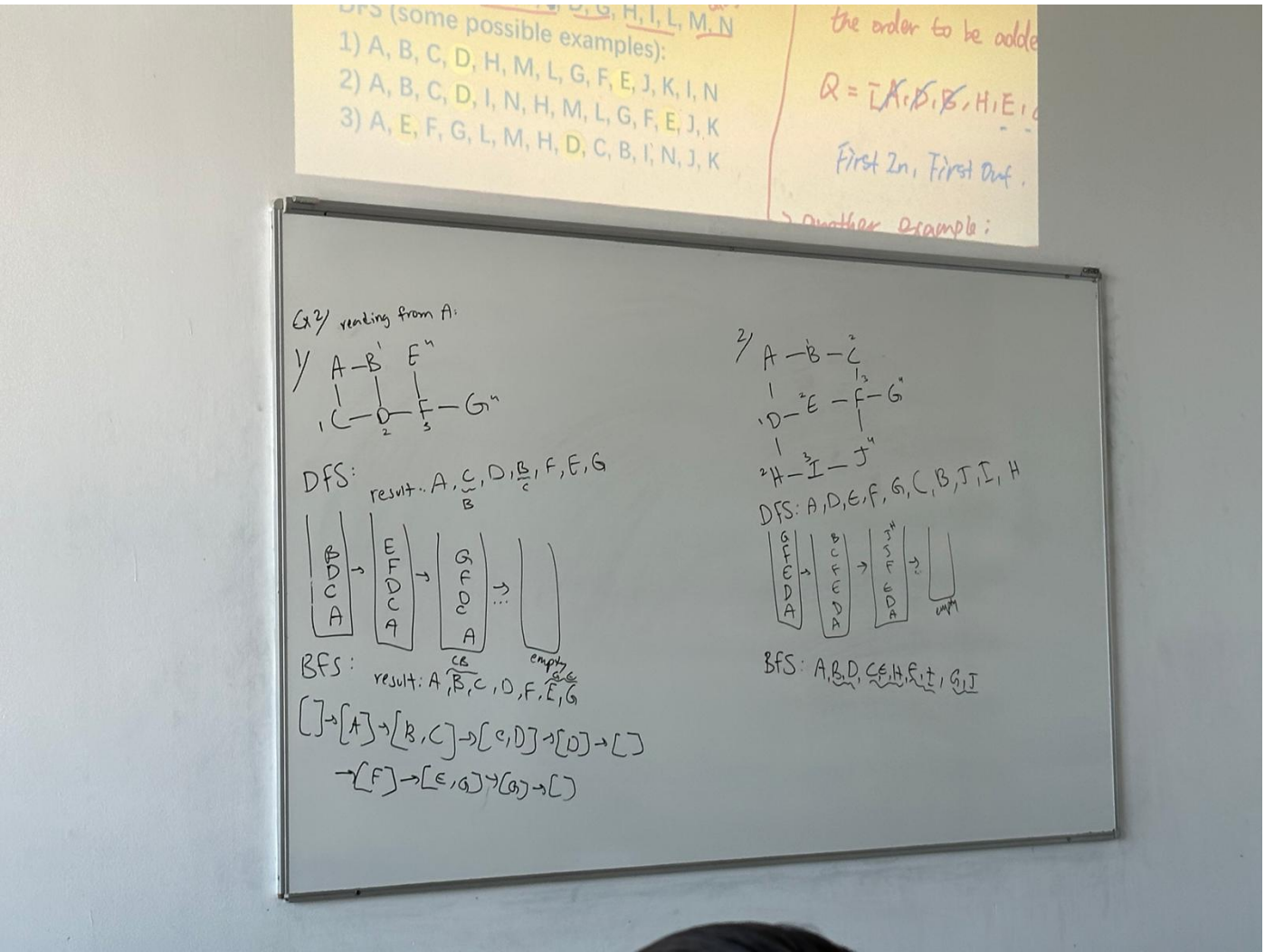
3/



DFS: A, B, C, D, I, N, H, M, L, G, F, E, J, K

DFS: A, E, J, K, F, G, H, M, L, D, I, N, C, B

BFS: A, B, E, C, F, J, K, D, G, H, L, I, M, N


DFS (some possible examples):
1) A, B, C, D, H, M, L, G, F, E, J, K, I, N
2) A, B, C, D, I, N, H, M, L, G, F, E, J, K
3) A, E, F, G, L, M, H, D, C, B, I, N, J, K
the order to be added
First In, First Out.
Another example:
Q2/ starting from A:
1/
DFS:
result: A, C, D, B, F, E, G
BFS:
result: A, B, C, D, F, E, G
[] → [A] → [B, C] → [C, D] → [D] → []
→ [F] → [E, G] → [G] → []
2/
DFS: A, D, E, F, G, C, B, I, H
BFS: A, B, D, C, E, H, F, I, G, I
empty