ณภัทร วัฒนาสุทธิวงศ์ 62130500023

Assignment 1

1. ให้วิเคราะห์เปรียบเทียบ Complexity of Algorithm ของโครงสร้างข้อมูลแบบ List ตาม operation ที่กำหนด โดยมี implementation ที่แตกต่างกันหลายรูปแบบดังนี้
   a. List แบบ A ใช้วิธีเก็บข้อมูลแต่ละตัว ลงในแต่ละช่องของ array ขนาด N, list แบบนี้ จะเก็บข้อมูลเรียงตามลำดับที่เพิ่มข้อมูลเข้ามาใน list และหากมีการลบข้อมูลออกจาก array จะปล่อยให้ช่องถูกลบไปกลายเป็นช่องว่างโดยไม่ขยับข้อมูลอื่นมาแทนที่ และเมื่อที่การเพิ่มข้อมูลเข้าไปอีกใน array ก็จะใส่ข้อมูลไปต่อท้ายข้อมูลตัวสุดท้ายที่มีอยู่ใน array เสมอ, และด้วยเหตุที่ไม่มีการขยับข้อมูลไปแทนที่ช่องว่างที่เกิดขึ้น ดังนั้น การเข้าข้อมูลโดยระบุลำดับที่ใน list จึงอาจจะไม่สอดคล้องกับตำแหน่ง index ใน array (เช่น ข้อมูลลำดับที่ 5 (นับลำดับแรกเป็นลำดับที่ 0) ใน list อาจไม่ได้อยู่ใน array ช่องที่ 5), list แบบนี้ไม่มี variable สำหรับจำว่ามีข้อมูลอยู่ใน array นี้กี่ตัวแล้ว หรือข้อมูลตัวสุดท้ายอยู่ตำแหน่งไหนใน array
   b. List แบบ B เหมือน list แบบ A เพียงแต่ว่า list แบบ B มี variable สำหรับจำว่ามีข้อมูลอยู่ใน array นี้กี่ตัวแล้ว, เมื่อมีการเพิ่มข้อมูลเข้าไปใน array หรือลบข้อมูลออกจาก array, variable นี้ ก็จะถูก update ให้สอดคล้อง
   c. List แบบ C เหมือน list แบบ A เพียงแต่ว่า list แบบ C มี variable สำหรับจำว่าข้อมูลตัวสุดท้ายที่อยู่ใน array อยู่ที่ตำแหน่งไหนของ array, เมื่อมีการเพิ่มข้อมูลเข้าไปใน array หรือลบข้อมูลออกจาก array, variable นี้ ก็จะถูก update ให้สอดคล้อง

   จงวิเคราะห์ worst-case complexity of operations ดังต่อไปนี้ โดยให้ตอบในรูปแบบ Big Oh ลงในตาราง

ประเภท List → / Operation ↓	List แบบ A	List แบบ B	List แบบ C
เพิ่มข้อมูล 1 ตัวเข้าไปใน list	(1A) O(n)	(1B) O(n)	(1C) O(1)
เข้าถึงข้อมูล 1 ตัว โดยระบุลำดับที่ของข้อมูลใน list ที่ต้องการ	(2A) O(n)	(2B) O(n)	(2C) O(n)
สอบถามจำนวนข้อมูลที่มีอยู่ใน list	(3A) O(n)	(3B) O(1)	(3C) O(n)

   ให้อธิบายเหตุผลประกอบแต่ละคำตอบ

   (1A) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันโดยไม่มีตัวแปรมาระบุตำแหน่งทำให้ต้องวนลูปหาตำแหน่งสุดท้ายที่จะเก็บข้อมูล

   (1B) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันโดยไม่มีตัวแปรมาระบุตำแหน่งทำให้ต้องวนลูปหาตำแหน่งสุดท้ายที่จะเก็บข้อมูล

   (1C) O(1) เพราะ List C มี variable เก็บข้อมูลของ index สุดท้ายที่มีข้อมูล ทำให้สามารถ assign ข้อมูลให้ index ต่อไปได้เลย

(2A) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันทำให้ index ของลำดับข้อมูลจริง ๆ กับ index ของข้อมูลใน Array ไม่ตรงกันจึงต้องวนลูปหาตำแหน่งของข้อมูลที่ต้องการจริง ๆ

(2B) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันทำให้ index ของลำดับข้อมูลจริง ๆ กับ index ของข้อมูลใน Array ไม่ตรงกันจึงต้องวนลูปหาตำแหน่งของข้อมูลที่ต้องการจริง ๆ

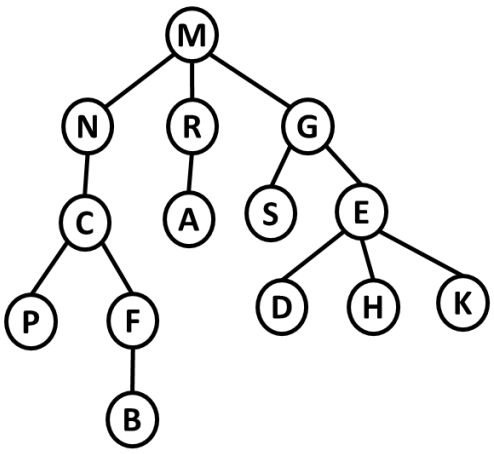
(2C) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันทำให้ index ของลำดับข้อมูลจริง ๆ กับ index ของข้อมูลใน Array ไม่ตรงกันจึงต้องวนลูปหาตำแหน่งของข้อมูลที่ต้องการจริง ๆ

(3A) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันโดยไม่มีตัวแปรมาระบุตำแหน่งทำให้หาจำนวนข้อมูลทันทีไม่ได้และต้องวนลูปเก็บค่านับไปตามช่องเรื่อย ๆ เพื่อหาช่องที่มีข้อมูลอยู่

(3B) O(1) เพราะ List B มี variable ที่เก็บค่าของจำนวนข้อมูลอยู่แล้ว ทำให้เราสามารถ return ค่านั้นออกมาได้เลยโดยใช้เวลาแค่คงที่

(3C) O(n) เพราะการเก็บข้อมูลแบบมีช่องระหว่างระหว่างกันโดยไม่มีตัวแปรมาระบุตำแหน่งทำให้หาจำนวนข้อมูลทันทีไม่ได้และต้องวนลูปเก็บค่านับไปตามช่องเรื่อย ๆ เพื่อหาช่องที่มีข้อมูลอยู่

2. จาก Rooted Tree ตามรูปนี้



2.1 จงบอก depth, height, parent node, และ child nodes ของแต่ละ node

Node	depth	height	parent	child
A	2	0	R	-
B	4	0	F	-
C	2	2	N	P, F
D	3	0	E	-
E	2	1	G	D, H, K
F	3	1	C	B
G	1	2	M	S, E
H	3	0	E	-
K	3	0	E	-

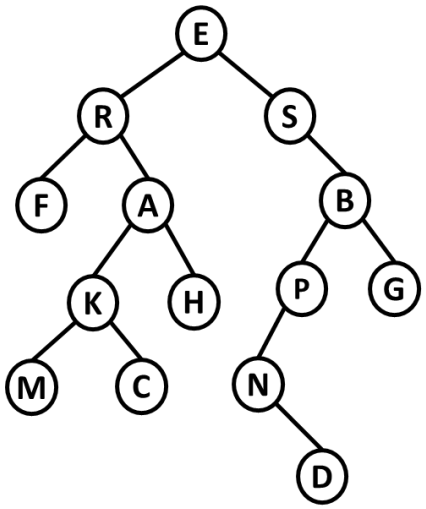
M	0	4	-	N, R, G
N	1	3	M	C
P	3	0	C	-
R	1	1	M	A
S	2	0	G	-

2.2 จงเขียน tree representation ของ subtree ที่มี node ดังต่อไปนี้ เป็น root

a) subtree ที่มี G เป็น root node: {G, {S}, {E, {D}, {H}, {K}}}

b) subtree ที่มี N เป็น root node: {N, {C, {P}, {F, {B}}}}

3. จาก Binary Tree ตามรูปนี้



3.1 จงเขียนลำดับของ node เริ่มจาก root node ได้ไล่ตามลำดับ Traversal ดังนี้

a) Pre-Order Traversal: E, R, F, A, K, M, C, H, S, B, P, N, D, G

b) Post-Order Traversal: F, M, C, K, H, A, R, D, N, P, G, B, S, E

c) In-Order Traversal: F, R, M, K, C, A, H, E, S, N, D, P, B, G

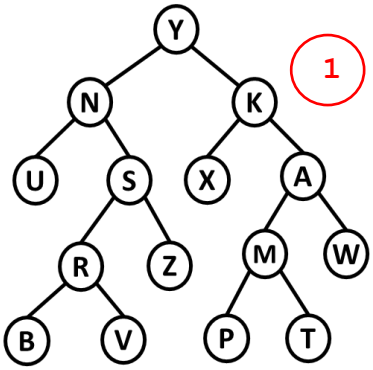
3.2 จงเขียน path จาก node หนึ่งไปยังอีก node หนึ่ง ตามที่กำหนดดังต่อไปนี้ และบอกความยาวของ Path

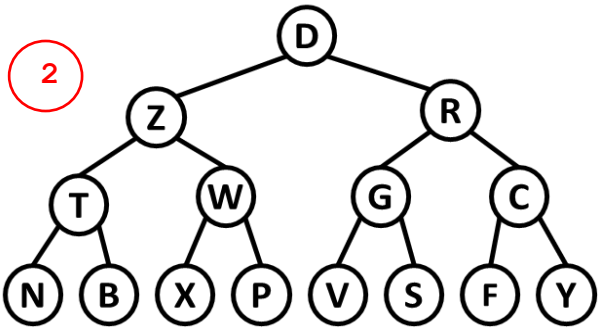
a) Path จาก node B ไปยัง node K: B, S, E, R, A, K

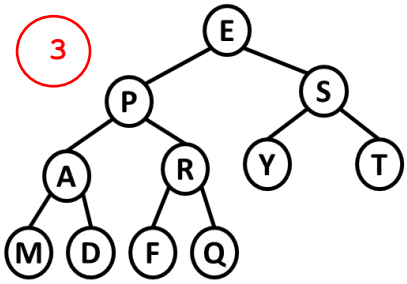
b) Path จาก node N ไปยัง node S: N, P, B, S

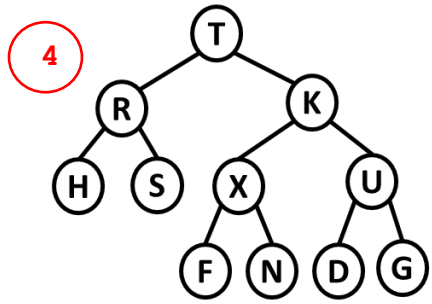
c) Path จาก node C ไปยัง node D: C, K, A, R, E, S, B, P, N, D

4. จาก Binary Tree ตามรูปนี้









Binary Tree ใด เป็น Full Binary Tree, Complete Binary Tree, และ Perfect Binary Tree บ้าง ให้ใส่ True หรือ False ลงในช่องตาราง

Binary Tree	Full Binary Tree	Complete Binary Tree	Perfect Binary Tree
1	True	False	False
2	True	True	True
3	True	True	False
4	True	False	False