① จงอธิบายเกี่ยวกับโครงสร้างข้อมูลและ Algorithm พร้อมยกตัวอย่าง (อย่างน้อย 5 โครงสร้างข้อมูล)

- Array เป็นโครงสร้างพื้นฐานที่อยู่กับแบบต่อเนื่องในหน่วยความจำที่ไม่มีความซับซ้อน โดยข้อมูลในโครงสร้างต้องมีชนิดเดียวกัน สามารถเข้าถึงข้อมูลภายในได้ด้วย Index และ Array นำไปต่อยอดเป็นโครงสร้างอื่นได้ เช่น STACK, QUEUE

  ตัวอย่าง : int Array[] = {11,12,13,14,15} →

11	12	13	14	15
0	1	2	3	4

  Array[0] = 11 (← index)

- Stack เป็นโครงสร้างข้อมูลแบบต่อเนื่องที่มีลักษณะพิเศษ โดยนำเข้าข้อมูลและออกทางเดียว ใครเข้าก่อนออกทีหลัง FILO (First In-Last Out) เช่น กระปุกมันฝรั่งทอด (ที่ชิ้นสุดท้ายของกระปุก คือชิ้นแรกที่เข้ากระปุกไป)

  ตัวอย่าง :

ช่อง	ข้อมูล
4	← MAXSTK = 4
3	111 ← TOP = 3, STACK[TOP] = 111
2	000
1	101

  หาก TOP = MAXSTK แปลว่าไม่สามารถนำเข้าข้อมูลได้อีก เนื่องจาก MAXSTK คือค่าขนาดความจุของ STACK

- Queue เป็นโครงสร้างข้อมูลแบบต่อเนื่องที่มีคุณสมบัติ เข้าก่อนออกก่อน โดยมีทางเข้าและออกทางเดียวเหมือน STACK ข้อมูลที่เข้าไปก่อนจะถูกประมวลผลก่อน เช่น การต่อคิวจ่ายเงินที่แคชเชียร์ ใครมาก่อนได้จ่ายก่อนตามหลัก FIFO (First In-First Out)

  ตัวอย่าง : ให้มีข้อมูลใน Queue เป็น 10, 20, 30, 40

  ออก ←

10	20	30	40	
1	2	3	4	5

  ← เข้า

  Queue[FRONT] = 10

  ขนาดเท่ากับ 5   FRONT = 1   REAR = 4

- Linked List (รายการโยง) เป็นโครงสร้างข้อมูลที่มีความซับซ้อน ข้อมูลแต่ละรายการเรียกว่า node ประกอบด้วยข้อมูลและส่วนเชื่อมโยงไป node อื่นๆ (address) โครงสร้างชนิดนี้ใช้ตัวชี้ (Pointer) เพื่อเชื่อมแต่ละ node เช่น รถไฟที่แต่ละโบกี้จะเชื่อมกันจนถึงท้ายขบวน

  ตัวอย่าง : [หัวรถไฟ |] → [โบกี้ |] → [ท้ายรถ | X]   เส้นที่เชื่อม เป็นตัวเชื่อมระหว่างรถไฟ (pointer)

- Tree เป็นโครงสร้างข้อมูลแบบไม่ต่อเนื่องที่ใช้ลักษณะมุมมองจากบนลงล่างเชื่อมโยงกันใน node แม่และโหนดลูก เชื่อมโยงกันโดยใช้ตัวชี้ไปยัง node ซ้ายและขวาเป็นหลัก หรือ เชื่อมโยงไปมากกว่า 2 node

- Graph เป็นโครงสร้างข้อมูลแบบไม่ต่อเนื่องที่ซับซ้อนที่สุด มีการสร้างเป็น node คล้าย Tree แต่ต่างกันที่ node หนึ่งมีเส้นเชื่อมโยงเข้าออกได้มากกว่าหนึ่งเส้นทาง ถูกนำไปประยุกต์ใช้ในโลกจริงเรียกว่า Real World data Structure เช่น GPS

ทุกโครงสร้างที่กล่าวไปเป็นเพียงวิธีจัดเก็บข้อมูลลง Computer และ Algorithm ก็เป็นเพียงวิธีการดำเนินการกับข้อมูลในขั้นตอนวิธีในการแก้ไขปัญหา เช่น Algorithm insert, Delete ดังนั้นความสัมพันธ์ของ Algorithm และโครงสร้างข้อมูล คือการที่ Algorithm จะเปลี่ยนไปตามโครงสร้างข้อมูลที่เราเลือกใช้ในการจัดเก็บข้อมูล เช่น Algorithm ในการ insert Array กับ Algorithm ในการ insert Queue ก็จะแตกต่างกัน

☆ จำให้หมด

1-5 = 55 คะแนน
6-9 = 45 คะแนน
ตัด A ที่ 80 คะแนน
อย่างมากข้อละ 15 นาที

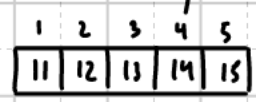
1. จงอธิบายพร้อมยกตัวอย่างว่าโครงสร้างข้อมูล array, stack, queue, linked lists คืออะไร มีความสำคัญอย่างไร รวมถึงมีความสัมพันธ์กับอัลกอริทึมอย่างไร **(10 คะแนน)**

- Array คือ โครงสร้างพื้นฐานที่อยู่แบบต่อเนื่องในหน่วยความจำที่ไม่มีความซับซ้อน ข้อมูลภายในโครงสร้างต้องมีชนิดข้อมูลเดียวกันทั้งหมด เราสามารถเข้าถึงข้อมูลภายในได้ โดย index และ วงวน เช่น คะแนนสอบ

ข้อดี สามารถเข้าถึงข้อมูลได้อย่างรวดเร็วง่ายดาย

ข้อเสีย ขนาดคงที่ ไม่สามารถเพิ่มหรือลดได้ง่ายๆ

ตย. int Array [5]

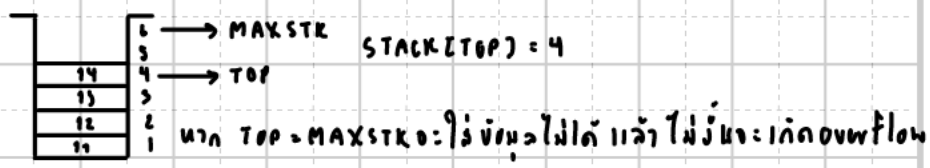


- Stack คือ โครงสร้างแบบต่อเนื่องที่มีลักษณะพิเศษ โดยมีทางเข้าและออกของข้อมูลเพียงทางเดียว ใครเข้าก่อนออกทีหลัง ตามหลัก FILO (First In-Last Out) เช่น กระปุกขนมปรุงทอด (ชิ้นแรกที่เข้าเป็นชิ้นสุดท้ายที่ออก)

ข้อดี จัดการลำดับข้อมูลในการเข้าออกได้ดี

ข้อเสีย เปลี่ยนลำดับไม่ได้ ต้องปฏิบัติตามกฎเพียงแบบ

ตย.

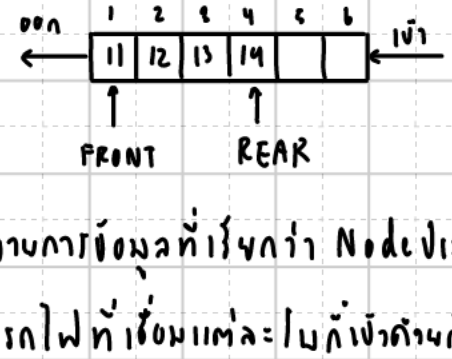


- Queue คือ โครงสร้างแบบต่อเนื่องที่มีคุณสมบัติพิเศษโดยมีทางเข้าและออกของข้อมูลเพียงทางเดียวเหมือน Stack เข้าก่อนออกก่อน ใครเข้าก่อนจะถูกนำออกมาก่อน ตามหลัก FIFO (First In-First Out) เช่น ต่อแถวจ่ายเงิน (มาก่อนจ่ายก่อน)

ข้อดี จัดการลำดับข้อมูลในการเข้าออกได้ดี

ข้อเสีย เปลี่ยนลำดับไม่ได้ ต้องปฏิบัติตามกฎเพียงแบบ

ตย.

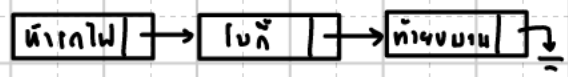


- linked List หรือรายการโยงคือ โครงสร้างแบบต่อเนื่องที่มีรายการข้อมูลที่เรียกว่า Node ประกอบด้วย ข้อมูลและเส้นเชื่อมโยง การจะเชื่อมแต่ละ node ต้องใช้เส้นเชื่อมหรือตัวชี้ (Pointer) เช่น รถไฟที่เชื่อมแต่ละโบกี้เข้าด้วยกันยังไง ตั้งท้ายขบวน

ข้อดี มีโครงสร้างที่ยืดหยุ่น สามารถใส่หรือลดข้อมูลตรงไหนก็ได้

ข้อเสีย การจะเข้าถึงข้อมูลต้องเข้าจากจุดเริ่มต้น

ตย.



ทุกโครงสร้างที่ได้กล่าวถึงไปเป็นวิธีการเก็บข้อมูลลงใน Computer และ Algorithm เป็นขั้นตอนวิธีในการแก้ไขปัญหา เช่น Algorithm Insert , Delete ก็เป็นวิธีการแทรกและลบข้อมูลภายในโครงสร้าง ดังนั้นโครงสร้างข้อมูลใช้ Algorithm ในการดำเนินการข้อมูล เช่น Algorithm Insert ใน Array และ Queue ก็ต่างกัน นั้นหมายความว่าเราใช้ Algorithm ในการแก้ไขปัญหาในโครงสร้างตามที่เราต้องการ

Graph (กราฟ)

- **ข้อดี: เหมาะสำหรับแสดงความสัมพันธ์ที่ซับซ้อน**

- **ข้อเสีย: การคำนวณบางประเภทอาจซับซ้อน**

Array

• Search (10)

```
Algorithm : Linear_Search
Input : LA , N, ITEM
Output : LOC
LOC := 0 , I := 1                    -2
While  I<=N  DO                      - n
    IF  LA[i] = ITEM  Then           - 1
        LOC := i                     - 1
        return LOC                   - 1
    Else
        I := I+1                     - 1
    End of IF
End of While
return LOC                           - 1
```

f(n) = 2 + 3n + 1

= O(n)

• Insert (7)

```
Algorithm : Insert
Input  : LA, N, K, ITEM
Output : item is inserted
         into LA at K
I := N                               - 1
While  i >= K  DO                    - n
    LA[i+1] := LA[i]                 - 1
    i := i-1                         - 1
End While
LA[K] := item                        - 1
N := N+1                             - 1
```

f(n) = 1 + 2n + 2

= 2n + 3

= O(n)

• Delete (5)

```
Algorithm : Delete
Input : LA, N, K, ITEM
Output : Data at LA[K] is deleted
         from LA
ITEM := LA[K]                        - 1
for i := K to N-1  Do                - n
    LA[i] := LA[i+1]                 - 1
End for
N := N-1                             - 1
```

f(n) = 1 + n + 1

= n + 2

= O(n)

Stack

• Push (6)

f(n) = 3

= O(1)

```
Algorithm : Push
input : STACK , TOP, MAXSTK, ITEM
output : item is pushed into stack
If   TOP = MAXSTK  Then              - 1
     Print "Stack Overflow"         - 1
Else                                 } 3
     TOP := TOP + 1                  - 1
     STACK [TOP] := ITEM             - 1
End If
```

• Pop (6)

```
Algorithm : Pop
input : STACK , TOP, MAXSTK, ITEM
output : STACK[TOP] is removed from STACK, and stored into ITEM
IF  TOP = 0  Then
    Print "Stack Underflow"
Else
    ITEM := STACK[TOP]
    TOP := TOP - 1
End If
```

```
public static void POP(){ //เมธอดการป๊อบ
    if(TOP==-1)   //เทียบว่ากองซ้อนว่างหรือไม่
        System.out.println("UNDERFLOW"); //ถ้าว่างแสดงว่าไม่มีข้อมูล
    else
    {
        ITEM=STACK[TOP];   //หากกองซ้อนไม่ว่าง ป๊อบข้อมูลเก็บใน ITEM
        TOP--;             //ลดค่าตัวชี้บนสุดหนึ่งลำดับ
    }
    System.out.println("Pop Data: "+ITEM); //แสดงข้อมูลที่ถูกป๊อบ
}
```

Queue

- Algo : Insert (15)

```
Algorithm : CircularQueueInsert
Input : QUEUE, FRONT, REAR, ITEM
Output : item is inserted into Queue, or rejected
If  FRONT = 1 (1) and REAR = N (1) OR                    } 5
    FRONT = REAR+1 (1) Then
      Print "Queue Full"        - 1
      Return                    - 1
End If
IF  FRONT = NULL Then           - 1                      } 3
      FRONT := REAR := 1        - 1
Else
  2 { IF  REAR = N Then         - 1
        REAR := N+1             - 1
    { Else
        REAR := REAR+1          - 1
    { End If
End If
QUEUE [REAR] := ITEM            - 1
return                          - 1
```

f(n) = 5 + 3 + 1 + 1

= O(1) constant

- Algo : Delete (15)

```
Algorithm : CircularQueueDelete
Input : QUEUE, FRONT, REAR, ITEM
Output : ITEM
IF  FRONT = NULL Then           - 1                      } 3
      Print "Queue Empty"       - 1
      Return ITEM := NULL       - 1
End IF
ITEM = QUEUE [FRONT]            - 1
IF  FRONT = REAR Then           - 1                      } 3
      FRONT := REAR := NULL     - 1
Else
  2 { IF  FRONT = N Then        - 1
        FRONT := 1              - 1
    { Else
        FRONT := FRONT + 1      - 1
    { End IF
End IF
Return ITEM - 1
```

f(n) = 3 + 1 + 3 + 1

= O(1)

(6) Tree

Preorder (15)

Algorithm: Preorder

Input : ROOT, LEFT, RIGHT, STACK, TOP, INFOR

Output : All nodes in BST are traversed

TOP := 1 -1

STACK[1] := NULL -1

PTR := ROOT -1

7n { While PTR ≠ NULL DO

Process INFOR[PTR] -1

3 { If RIGHT[PTR] ≠ NULL Then -1

TOP := TOP+1 and STACK[TOP] := RIGHT[PTR] -2

End of If }

3 { If LEFT[PTR] ≠ NULL Then -1

PTR := LEFT[PTR] -1

Else

PTR := STACK[TOP] and TOP := TOP-1 -2

End of If }

End of While }

Return -1

$f(n) = 3 + 7n + 1$

$= O(n)$

(8) Sort

Selection Sort (12)

Algorithm: Selection Sort

Input: DATA, N, J, TEMP, K, LOC

Output: Sorted Data

n-1 { for K = 1, 2, ..., n-1 DO

MIN := DATA[K] and LOC := K -2

3n { for J = K+1, K+2, ..., N DO

3 { If MIN > DATA[J] Then -1

MIN := DATA[J] and LOC := J -2

End of If }

End of for }

TEMP := DATA[K] -1

DATA[K] := DATA[LOC] -1

DATA[LOC] := TEMP -1

End of for }

Return DATA -1

$f(n) = ((3n+5)(n-1)) + 1$

$= O(n^2)$

⑨ Search

Binary Search (IS)

```
Algorithm: Binary_Search
Input: sDATA, MID, FIRST, LAST, N, LOC, ITEM
Output: LOC
FIRST:=1; LAST:=N; LOC:=0   -3
while  FIRST <= LAST  DO                          } 7 log2 n
    MID:=(FIRST+LAST)/2;   -3
    If  sDATA[MID]=ITEM  Then   -1
        LOC:=MID   -1
        Return LOC   -1
    Else                                          } 4
        If sDATA[MID]>ITEM  Then   -1             } 3
            LAST:=MID-1   -2
        Else
            FIRST:=MID+1   -2
        End If
    End If
End While
Return LOC;   -1
```

$f(n) = 3 + 7\log_2 n + 1$

$= O(\log n)$