การเรียงข้อมูล (Sorting)

* Insertion
* Selection
* Bubble
* Shell
* Heap
* Merge
* Quick
* Radix
* BST sort

**Basic OP ของการเรียงข้อมูล**

1. Comparison
2. Moving
3. Recursion
4. Bucketting

**ปัจจัยที่มีผลต่ออัลกอริทึมการเรียงข้อมูล**

**Running Time ขึ้นกับ**

Number of - comparision

- moving

**ปัญหามีสองประเภท**

* Online / incremental -> รับมาแล้วทำไปเลย เช่น insertion sort
* Batch -> รับข้อมูลมาทั้งหมดก่อน แล้วประมวลผล (อย่างน้อย O(n))

ต้องพิจารณาว่าอัลกอเหมาะแก่การทำ online ไหม หรือว่าได้แค่ batch อย่างเดียว

**Memory ที่ใช้ในการ Sort**

* O(n) -> main memory (ข้อมูลที่จะ sort)
* เมมโมรี่เสริม (AUX)
* Stack for recursion

**Best / Worst / Avg Cases**

* แต่ละอัลกอมีลักษณะข้อมูล ที่เหมาะสมกับมันต่างกัน

**Implement/Debug time**

**Stability**

* ความสามารถในการรักษาลำดับตั้งต้นของข้อมูลที่เข้ามา ในกรณีที่มี Key เดียวกัน

Tree

**Binary Tree**

* สามารถอ่านข้อมูลแบบเรียงลำดับใน Tree ได้โดยใช้การท่องแบบ Inorder
* ถ้าอยากให้ข้อมูลที่ใส่เข้ามามี Stability ควรจะให้ x<K อยู่ซ้าย และ x>=K อยู่ขวา (ตัวที่เท่ากับค่าของโหนด ให้ไปฝั่งขวา)
* Delete โดยการนำ มากสุดของฝั่งซ้าย หรือ น้อยสุดของฝั่งขวา มาแทนที่
* การหา average case ของการ search (Big O รวม หารด้วย จำนวนเคส)
* 20 + 21 + 22 + … + 2k = n
* 2k+1 -1 = n
* k = log(n-1)-1

sum = 1\*20 + 2\*21 + 3\*22 + 4\*23 + … +(k+1)2k

= 20 + 21 + 22 + 23 + … +2k

+ 21 + 22 + 23 + … + 2k

+ 22 + 23 + … + 2k

**…**

+ 2k

Sum = k2k+1 +1

Avg =

* Median of medians 🡨 การบ้าน

# AVL Tree

* Self-balancing tree
* แต่ละ node จะเก็บ hl,hr ไว้ หมายถึง ความสูง Sub Tree ด้านซ้าย และ ด้านขวา
* |hl – hr | < 2 เสมอ (มากกว่าจะต้องหมุนต้นไม้)
* เป็น good balance แต่ไม่ perfect balance

# Splay tree

* เป็น Balanced tree เหมือน AVL
* จะสมดุลขึ้น เมื่อมีการกระทำกับ tree เช่น insert,search
* แต่ละครั้งที่ถูก insert หรือ search โหนดนั้นจะถูก “Splay” คือการหมุนโหนดนั้นขึ้นมาจนถึงด้านบนสุด
* การหมุนเหมือนกับ AVL Tree เลย
* เมื่อพิสูจน์แล้วพบว่าเร็วมาก O(log n)

# สรุป

* AVL ทำให้สมดุลด้วยการ insert/delete
* Splay ทำให้สมดุลด้วยการค้นหาข้อมูลซ้ำๆบ่อยๆ