Data Structure & Algorithm Review by P.Time

Basic Structure

Primitive class ---> Default class of language

Example int,float,char,double

Secondary class (class) ---> ทุกอย่างนอกจาก primitive

Example list,vector และ class ที่สร้างเอง

Structure

---> Data type (int, float, object ต่างๆ)

---> Collections (array, vector, list, object ที่เก็บ objects)

functions

Return type pattern(รูปแบบ)

Object function(){

return object;

}

- Object คือชื่อ class เช่น int, float, double, Fraction, List, Vector, ...

Big-O(Run time Complexity)

- โดยปกติถ้าพูดถึง Big-O ก็จะพูดถึง worst case
- Average case --> Big theta
- Best case --> Big omega
- Run time Complexity หรือสั้นๆ Big-O คือเวลาที่ใช้ในการทำงานของโปรแกรม
- แบ่งเป็น O(1) - Constant Time, O(logn) Logarithmic Time, O(n) - Linear Time
- O(nlogn) Linearithmic Time, O(n^2) - Quadratic Time, O(n^3) Cubic Time
- O(2^n) Exponential Time, O(n!) Factorial Time เรียงจากน้อยสุดไปมากสุด

Big-O Amortized

- Worst case โอกาสเกิดน้อยกว่า Best case
- เช่น การ insert ของ vector

1	2	3	4	5	6

Insert 6

Big-O เป็น O(1)

- การ insert ใน vector ที่ยังมีช่องว่าง Big-O เป็น O(1) เสมอ

1	2	3	4	5	6

Insert 7

- เนื่องจาก vector เต็มจึงเกิดการสร้าง vector ใหม่โดย copy ข้อมูลของ vector เดิมไป O(n)

1	2	3	4	5	6	7			

Big-O เป็น O(n)

Pointers

int *p; การประกาศตัวแปร Pointer ต้องมี * เสมอ

Cout << p; หมายถึง cout << ที่อยู่ของสิ่งที่ pointer p ชี้อยู่

Cout << *p; หมายถึง cout << ค่าของสิ่งที่ P ชี้อยู่

Cout << &p; หมายถึง cout << ที่อยู่ของ pointer p

* เรียกว่า dereferencing operator

& เรียกว่า referencing operator

Linked Lists

- Singly Linked list
- Doubly Linked list
- Skip list
- Circular list

Singly & Doubly Comparison

Singly linked list (ข้อดี)

- เข้าใจง่าย ง่ายต่อการเขียน
- (Simple)
- ใช้พื้นที่น้อย
- (Require less space)

Doubly linked list (ข้อดี)

- เข้าถึง list ได้ 2 ทาง
- (Two ways to traverse through the list)
- การ delete ใช้เวลาน้อยกว่า singly
- (Require less time for node deletion)

Singly & Doubly Comparison

Singly linked list (ข้อเสีย)

- ใช้เวลา delete นาน
- (take more time to delete node in the list)
- เข้าถึง List ได้แค่ทางเดียว
- (one way to traverse through the list)

Doubly linked list (ข้อเสีย)

- ใช้พื้นที่มาก
- (take more memory space)
- เข้าใจได้ยาก ต้องจัดการ pointer หลายตัวมากขึ้น
- (Harder to manage(maintain) the list since it contains one more pointer)

Recursion

- Run time stack complexity
- Direct recursion (Tail recursion , Non-tail Recursion)

Tail recursion -> มีการเรียก recursion ที่การ return ของ function นั้นๆ

เช่น fibonacci function, factorial function

- Indirect recursion ไม่ได้เรียกตัวเองแต่เรียก function อื่น แล้ว function นั้นมาเรียกต่อ

เช่น work() -> bed() -> work() -> bed()

Work เรียก bed และ bed เรียก work ต่อเป็นต้น

- Easy to read and understand code โค้ดเข้าใจได้ง่าย และสวยงาม

Sorting

- Bubble Sort -> Best case: O(n), Worst case: O(n^2)
- Cocktail Sort -> Best case: O(n), Worst case: O(n^2)
- Selection Sort -> Best case: O(n), Worst case: O(n^2)
- Merge Sort -> Best case: O(n), Worst case: O(nlogn)
- Quick Sort -> Best case: O(n), Worst case: O(nlogn)
- Shell Sort -> Best case O(n), Worst case: O(nlogn)

Key questions

- What is the difference between Array, Vector, Linked list
- Explain step by step quick sort of {2, 5, 10, 3, 6, 8, 4, 9, 7, 1}
- Difference between singly and doubly linked list
- Why recursion? When to use it? Why it is bad?