ALGO FINAL REVIEW

Note that: you must remember Pink HighLight

What is FIFO AND LIFO

- LIFO is one type of data structure known as “Stack”. it is linear data structure that can only be accessed at one end.
- New data is added to the top of stack, and data also retrieve from the top of the stack.

- FIFO is one type of data structure known as “Queue”. it also linear data structure, which both end are used.
- Data is added to one end, and retrieved from the other end.

How would you implement Stack

Linked list

- No redundant capacity
- Push and Pop are O(1)

Vector

- Capacity might be larger than size
- Push is Amortized O(n)
- Pop is O(1)

*Optional: Try coding the Stack class

Linked list

Vector

2 Ways to implement Priority Queue

Enqueue priority

Search through the list and push new element into its appropriate position

This makes it sometime O(n) or O(1)

Dequeue priority

When dequeue search through the list and pop the highest priority data

This makes it always O(n)

Recursive

What is it?

An instance of function that call another one

Or to keep it simply function that call itself

- Tail Recursion
- Non-Tail Recursion
- Direct Recursion
- Indirect Recursion

- **Tail Recursion**

When last instruction of function is the recursive call

- Non-Tail Recursion

Vice versa! (means opposite)

- Direct Recursion

Function that directly calls itself

F() => F()

- **Indirect Recursion**

Function that call another function that call itself

F() => G() => F()

Why and Why not Recursive?

Why?

- Simple
  - Sometimes it is easier to implement some algorithm by recursive
- Readability
  - Recursive is usually easier to read and understand
- No big difference

- When there are no better approaches

Why not?

- Runtime Stack
  - Data storing on runtime, risk in running out of memory
- Speed
  - Sometimes there are better approaches that consumes less time (Big-O is lower)

Sorting algorithm

O(n^2)

- Bubble Sort
- Selection Sort
- Cocktail Sort

O(nlogn)

- ****Merge Sort****
- Quick Sort
- Shell Sort

Hashing**

- Linear probing
- Key dependent probing
- Separate chaining or hashing bucket

How to do these 3-hashing step by step practice it

- Follow this link to watch video
  - shorturl.at/hEHMW

Memory Management

- **External Fragment**
  - Occurs when memory allocated, and deallocated some of it leaving gaps between used memory
- **Internal Fragment**
  - Occurs when the program request memory is larger than it needed
- **Buddy System (Binary)**
  - If needed low space, the memory can be divided into 2 fractions to make it most fit with the required space
  - Whenever possible fractions combine together to make larger one
  - Internal fragment likely to occur
- Sequential fit methods
  - First fit method, allocated first suitable block found in the memory. Fastest however external fragmentation likely to be occurred
  - Best fit method, allocated most suitable block found in the memory
  - Worst fit method, allocated the largest block found in the memory. Worst efficient

Garbage collection

- **Mark and Sweep**
  - Mark the used memory
  - Delete the unmarked memory, this can lead to gaps between memory
  - Data compaction, move memory tightly to others to remove gaps between memory
- **Incremental Garbage collection**
  - Normally mark and sweep method is likely to interrupt the running program
  - But this approach is to fix that problem