insert(value, index) O(n) [SEQUENCE]

lookup(index) → value O(n)

remove(index) → value O(n)

count() → O(n)/O(1)

会考O(n)和O(1)

Stack frame (implemented as linked list)

push(value) O(1)

top() → value O(1)

pop() → value O(1)

count() → n O(1)

最后输入的先处理

通常处理最后一个数据

There're push operation and pop operation

Search an element in the stack is O(1)

import collection from deque

follwed the principle of "LIFO"

[Last In, First Out]

QUEUE FIFO : [First In, First Out]

push(value) → value O(1)

top() → value O(1)

pop() → value O(1)

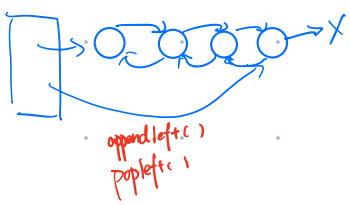
count() O(1)

先输入的先处理

通常处理第一个数据

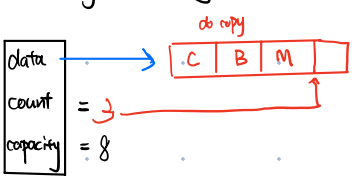
DEQUE Double-Ended QUEue

implemant as double linked list



Linked list { Vector

Vector (Dynamic array) [SEQUENCE]



function: insert(value, index) O(n)

lookup(index) → value O(1)

count() → n O(1)

remove(index) → value O(n)

Stack (implemented as a vector)

- push(value) O(n)*
- top() -> value O(1)
- pop() -> value O(1)
- count() -> n O(1)

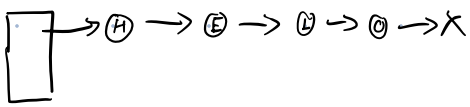
Queue (implemented as a vector)

- push(value) O(n)*
- pop() -> value O(n)
- top() -> value O(1)
- count() -> n O(1)

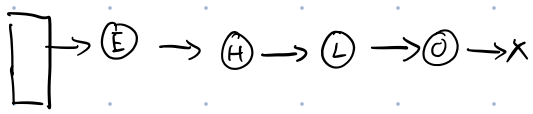
SORTED SEQUENCE (vector)

- insert(value) O(n)
- lookup(index) -> value O(1)
- remove(index) -> value O(n)
- count() -> n O(1)

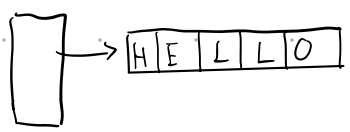
LINKED LIST



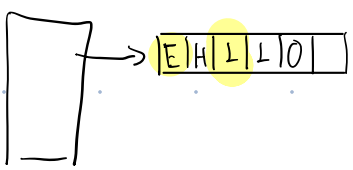
SORTED LIST



VECTOR



SORTED VECTOR



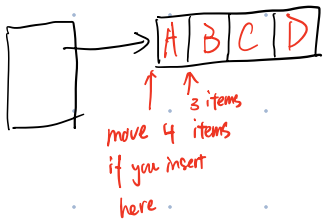
BINARY SEARCH

$n = 2^t - 1$

Average Runtime

Expected Runtime

Expected Runtime



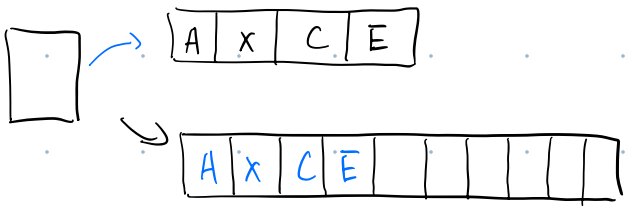
$\frac{1}{5}(4+3+2+1+0)$

$= \frac{1}{n+1}\left(\frac{n^2+n}{2}\right)$

$= \frac{1}{n+1} = \frac{(n+1)n}{2}$

$\therefore \frac{n}{2}$ $O(n)$

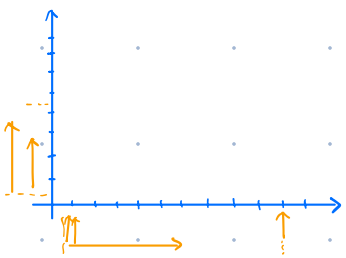
AMORTIZED Time



STACK (Vector)

push(value) $O(1)$ AMORTIZED

2. time complexity : linear time.
constant time,
quadratic time



That's linear time. when axis is increasing linearly, the vertical axis also increase linearly