Lecture 5 - Array Insetion & deletion

Insertion

In last | Based on Index

In last

Arr =

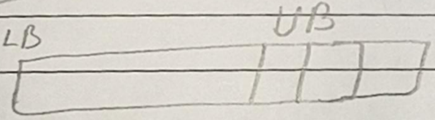
0	1	2	3	4	5	6	7	8	9 ← maxlimit
'a'	'b'	'c'	'd'	'e'					

LB = 0
UB = 4
n = 5

```
Insesnt (arr; item) {
    if (UB = maxlimit) {
        overflow & return
    }
    A[UB+1] = item
    UB++
    n++
}
```

Time complexity = $\theta(1)$ ⇒ Constant

Based on Index



i = index at where new elimant to be insreted

insetion(A[], item, i) { $LB \le i \le UB+1$

Case I $i = 5$

0	1	2	3	4	5	6	7
'a'	'b'	'c'	'd'	'e'			

Case 2 : i = 2 , item = 'x'

0	1	2	3	4	5	6	7	8
a	b	~~c~~	~~d~~	~~e~~	e			
		x	c	d				

In this case we need to shift every element from i=2 to the & right side.

only & only if i = UB+1 , then no shifting will be required

Algo :

```
insert(A[], item, i) {
    if (UB == maxlimit)              ] — Constant
        overflow & return;           ]
    for(K = UB ; K >= i ; k--) {     ] dependent
        A[K+1] = A[K]                ] — on the
    }                                ] shifting, maximum
                                       shifting = n
    A[i] = item
    UB++;            } → Constant
    n++;
}
```

Time Complexity = O(n) - Big Oh of n

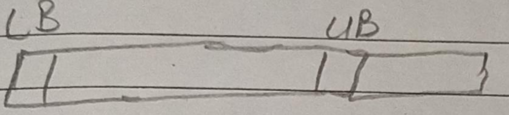
Deletion

- Based of index
- from last

0	1	2	3	4	5	6	7
a	b	c	d	~~e~~			

```
UB-
n--        if(n==0){
               undeflow
           }
```

* Based on Index



i= index from, where the element to be deleted

$LB \leq i \leq UB$

Run time Complexity

```
delete(A[], i){
    if(n==0){
        undeflow & return
    }
    for(k=i; k<UB; k++){
        A[k]= A[k+1]
    }
    UB--
    n--;
}
```

Time Complexity = O(n) - Big oh of n

Quiz Number of shifting of the element to insert a new element at starting index in an array of 100 elements is?

Ans $n = 100$

(2) No. of shifting required to delete an element from start

A $N-1$

(3) Consider an array A of n elements $(n>1)$. In this array A, n insertion and n deletion are performed in arbitrary order. What should be the best and worst case complexity of all operation on array?

Ans

n isertion
n deletion
2n operation – O(n)

best case
↓

worst case
each operation O(n)
for 2n operations $O(n^2)$

Lecture 6 - Searching

- Linear Search
- Binary Search

Prerequiste for BS : array should be sorted

* Linar Search

LB ... UB

LB									UB
1	7	12	3	8	4,	6	2	19	11

```
for(i=LB; i<=UB; i++) {
    if(A[i] == item) {
        return i;
  }
return LB-1
```

Time Complexity = O(n) - Big oh of n

* Binary Search

* array should be sorted to perform this

0	1	2	3	4	5	6	7	8	9
1	4	9	11	15	21	23	29	34	50

Search = 15

$$mid = \frac{LB + UB}{2} = 4$$

$$\frac{0+9}{2} = 4$$

```
if(A[mid = item) {
        return mid;
}
```

- Search 23

  mid = 4

  A[mid]

  if (A[mid] = ~~2~~23) {

  ↓

  ~~no~~ return

```
if (item > A[mid]) {
        LB = mid
} else if( item < A[mid]) {
        UB = mid - 1
}
```

when (LB > UB)

stop the loop

- Case 3: Search 10

L 0	1	2	3	4	5	6	7	8	9 H
1	4	9	11	15	21	23	29	24	50

mid = $\frac{0+9}{2}$ = ~~4~~ ~~1~~ ~~2~~ 3

L = ~~0~~ ~~2~~ 3

H = ~~9~~ ~~3~~ 2

] L > H ⇒ unsucessfull search

Time complexity = $O(\log_2 n)$

Quiz Consider a sorted array of size n with duplicate elements. you have been given an element k, what is the time complexity to find the element k is appeared atleast n/2 times in the array or not?

a) O(1)
b) O(logn) ✓ Ans
c) O(n)
d None

$n = 8$
$k = 2$

0	1	2	3	4	5	6	7
1,	2,	2,	2,	2,	2,	3,	9

log n
- apply modified binary search & find index i at which k appears first time in array from start

O(1)

-
  ```
  if (A[i+n/2]==k){
      return true;
  }
  ```

Q Consider a sorted array of size n with duplicate elements. What is the time Complexity to find that any element is appeared at least n/2 times in the array or not?

a) O(1)
b) O(log n) ✓
c) O(n)
d) None

$$2, 5, 5, \overset{\text{mid}}{\downarrow}\!\!\!5, 5, 5, 6, 10$$

what if

1, 2, 3, 4, 5, 5, 5, 5