NO.

DATE April 17

~o~ Binary Search ~o~

> Types of Searching:

1.) Linear or Sequential Search

↳ Running time of O(N)

↳ Typically used in Linked Lists & arrays (mostly, direct access or random access)

2.) Hashing

↳ a.) Open Hashing

- Grouping of elements
- A running time of $O(N/G)$. G is the # of groups and N is the # of elements. N is distributed evenly in G groups.

↳ b.) Closed Hashing

- Different running times: O(1) if perfect case, O(N) if worst case, $O(N/2)$ if avg. case.

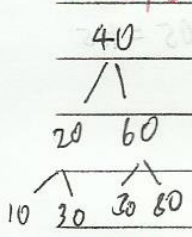
3.) Binary Search

↳ Related to BST

↳ Running time of $O(\log_2 N)$

↳ Elements must be sorted (Ascending / Descending order)

• example:



> Find elem = 15

10	20	30	40	50	60	80
0	1	2	3	4	5	6

LB ↑ mid ↑ UB ↑

LB	UB	Mid	mid elem
0	6	3	elem < 40
0	2	1	elem < 20
0	0	0	elem > 10
1	0		

• process:

- mid = (LB + UB)/2

if LB > UB, elem not found

if elem < mid elem, UB = mid - 1

if elem > mid elem, LB = mid + 1

//Note: This is not a Set & is not a Data Structure

Sterling

NO.

DATE

*Ex2:

10	20	30	40	50	60	80	90	100	105	110	120
0	1	2	3	4	5	6	7	8	9	10	11

• Find Elem = 70

	LB	UB	mid	mid Elem
• 1st iteration:	0	11	5	70 > 60
LB = 0, UB = 11, mid = 5, midelem = 60	6	11	8	70 < 100
70 < 60 → move LB	6	7	6	70 < 80
• 2nd iteration:	6	5		
LB = 6, UB = 11, mid = 8, mid elem = 100				
70 < 100 → move UB				
• 3rd iteration				
LB = 6, UB = 7, mid = 6, mid elem = 80				
70 < 80 → move UB				
• 4th iteration				
LB = 6, UB = 5 → LB > UB → Elem not found				

• Find elem = 105

	LB	UB	mid	mid elem
• 1st iteration	0	11	5	105 > 60
LB = 0, UB = 11, mid = 5, mid elem = 60	6	11	8	105 > 100
105 > 60 → move LB	9	11	10	105 < 110
• 2nd iteration	9	9	9	105 = 105
LB = 6, UB = 11, mid = 8, mid elem = 100				
105 > 100 → move LB				
• 3rd iteration				
LB = 9, UB = 11, mid = 10, mid elem = 110				
105 < 110 → move UB				
• 4th iteration:				
LB = 9, UB = 9, mid = 9, mid elem = 105. 105 = 105				

elem found!

Sterling

NO.

DATE

Exercise:

Given the sorted list, and an element x, write the code of the function findElem. The function will determine if x is in the sorted list using binary search technique & return 1; otherwise return 0;

```
#define SIZE 30
typedef struct {
int Elem[SIZE];
int count; //actual # of elements in the array
} LIST;
```

• Datastruct def:

• Code:

```
int findElem(LIST L, int x){
int LB=0, UB=L.count-1, mid;

while(LB<=UB && L.Elem[mid=(LB+UB)/2]!=x){
    (L.Elem[mid]<x)?(LB=mid+1):(UB=mid-1);
}
return(LB<=UB && L.Elem[mid]==x)?1:0;
}
```