# COP 3502

8/22/23



✓ (0) Intro

✓ (1) webpage - https://www.cs.ucf.edu/courses/cop3502/fall2023

mysite - https://www.cs.ucf.edu/~dmarino

ARUP GUHA
DMARINO@UCF.EDU

✓ (2) course success

✓ (3) Diff btw COP 3502 and COP3223

(4) SLM
(5) Malloc
} example w/ diff run times

(6) Prog Team (last 5)

COP 3223 ⇒ syntax
language use
efficiency & internals

COP 3502 ⇒ care about how fast/efficient solutions are
Data Structures
cleanliness of code

# Sorted List Matching



list 1: 2, 4, 6, 10, 21, 26, 32, 41 — n #s

list 2: 3, 5, 6, 8, 9, 10, 11, 31, 36, 38, 41, 47 — m #s

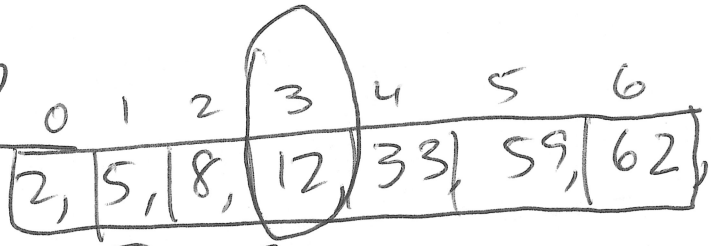
**Alg 1**

```
res = 0
for each item x in list 1:
    for each item y in list 2:
        if x == y:          *
            res = res + 1
```

O(n*m)

Issue: we're not utilizing the fact that the lists are sorted

**Binary Search**




Input sorted list:

value: 5, 71

goal: determine if value is in the list

looking for 5 — index 0 low, index 6 high — mid = (0+6)/2 = 3

since list[3] is too big set high = 2 = mid-1

mid = (low + high)/2 = (0+2)/2 = 1

list[1] = 5 ✓ value is found

# Dynamically Allocated Memory



Most of Intro to C only Statically allocated memory was used:

Comes from program stack

```
int value;
int freq[26];
char name[10];
```

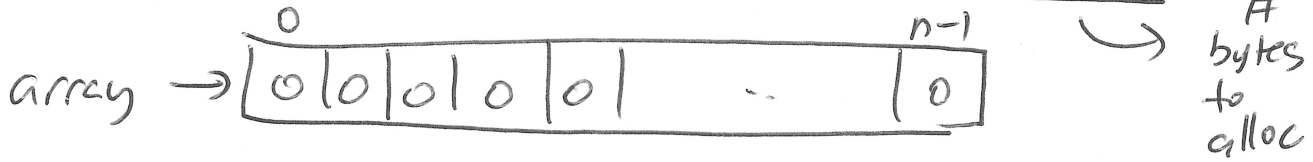
Amt of space known at compile time

---

If memory requirement is unknown at compile time, it has to allocated dynam. cally.

Core heap

```
scanf("%d", &n);
int* array = malloc( sizeof(int) * n );
```




```
for (int i = 0; i<n; i++)
    array[i] = 0;
```

DONE

```
free(array)
```

## REASONS TO USE

1) unknown amount needed at compile
2) lots of mem is needed (100,000 or more)
3) memory to persist after the function in which it was created is done.

binary search run time

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$k = \log_2 n$ * # of steps on binary search on n elements takes

## Alg 2 SLMP

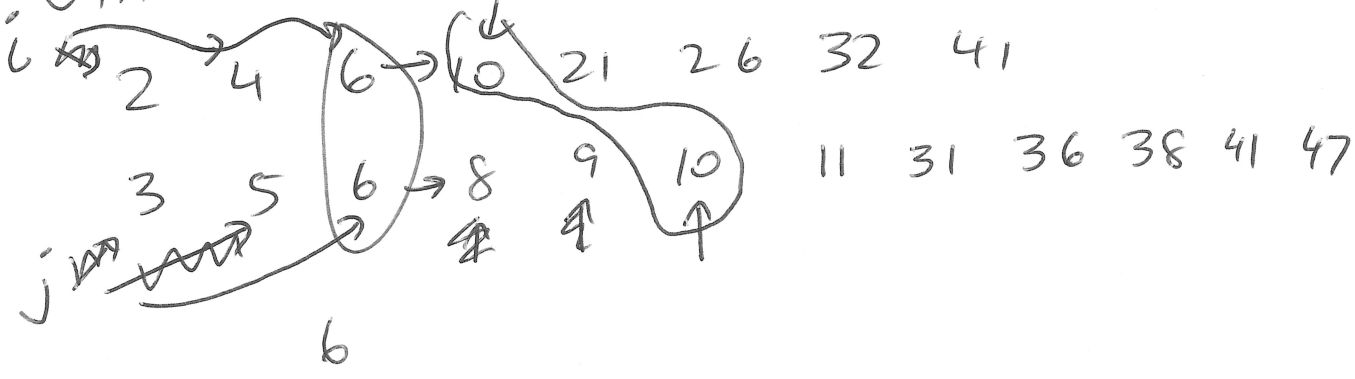
```
res = 0
for each x in list1:
    if binarysearch(x, list2):
        res = res + 1
```

n lg m

new runtime $O(n * \lg m)$

## Alg 3

Utilize fact both lists sorted



```
void simp(int* arr1, int len1, int* arr2, int len2) {
    int i=0, j=0;
    while (i < len1 && j < len2) {
        if (arr1[i] < arr2[j])
            i++;
        else if (arr2[j] < arr1[i])
            j++;
        else {
            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
}
```

O(n+m)