COP 3502 Computer Science I

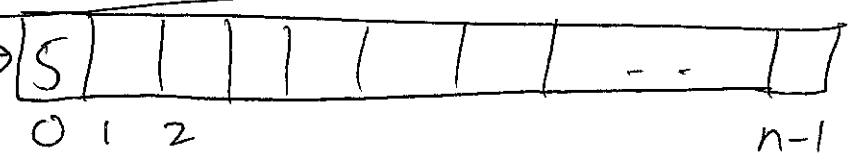
ARUP GUHA

Birthday Hunt

① Go to Webcourses
② Find assgn
③ Upload .txt file with info by Friday.
→ Counts as "attendance" grade for financial aid.

~~Dynami~~ Dynamic Memory Allocation for arrays

```
int i,n;
{
Scanf("%d", &n);
int* values = malloc(n * sizeof(int));
```

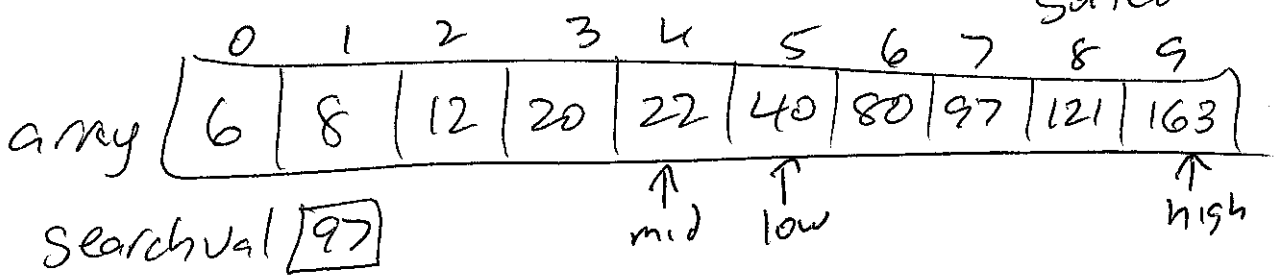



```
values[0] = 5;
for (i=0; i<n; i++)
    values[i] = 0;
free(values);
```

"every single malloc has an equal and opposite free"

Searching for a value in an array

sorted




```
int search(int* array, int length, int searchval) {
    int i;
    for (i = 0; i<length; i++)
        if (array[i] == searchval)
            return 1;
    return 0;
}
```

O(n)

Binary Search on a sorted array:
keep a low + high index and for each "guess", guess 1/2 way inbetween the two indexes

```
int search(int * array, int length, int searchval) {
    int low=0, high=length-1;
    while (low <= high) {
        int mid = (low+high)/2;
        if (searchval > array[mid])
            low = mid+1;
        else if (searchval < array[mid])
            high = mid-1;
        else return 1;
    }
    return 0;
}
```

O(lg n)

n	
n/2	
n/4	
n/8	
⋮	
$\frac{n}{2^k} = 1$	
$n = 2^k$	
$k = \log_2 n$	

$\log_3 81 = 4 \iff 3^4 = 81$

Reasons to use dynamically allocated memory

---

1. I need lots of memory
2. I'd like the memory I allocate to still be allocated when the function within which I allocated the memory finishes.

Sorted List Matching Problem

Input: 2 sorted lists of unique numbers

Output: sorted list of all #s that appear in both lists

n list1: 2, 6, 12, 13, 14, 19, 27, 42

m list2: 1, 4, 5, 6, 8, 13, 18, 19, 22, 25, 42, 88

```
for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
        if (list1[i] == list2[j]) {
            printf("%d ", list1[i]);
            break;
        }
    }
}
```

$O(n \cdot m)$

```
for (i=0; i<n; i++) {
    if (binsearch(list2, m, list1[i])
        printf("%d ", list1[i]);
}
```

$10 \times \lg 10^6$

$O(n \lg m)$

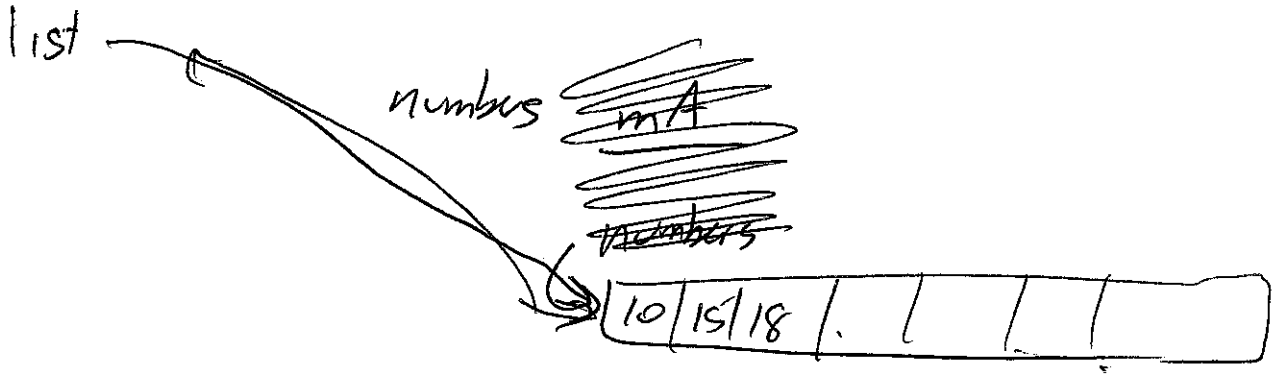
3rd algorithm with 2 "pointers" (actually integers that are used as array indexes) sweeping left to right.

$O(n+m)$

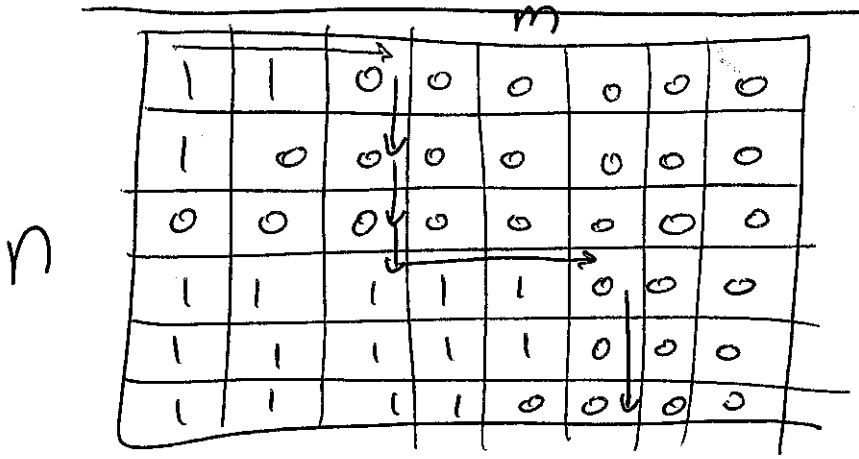
$10+10^6$

main

int* list = makeArray(1000000);



maxOnes in a Row



1st alg

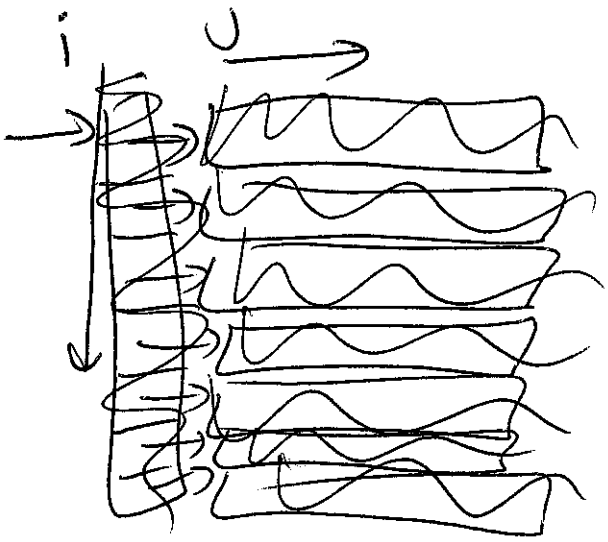
Double for loop

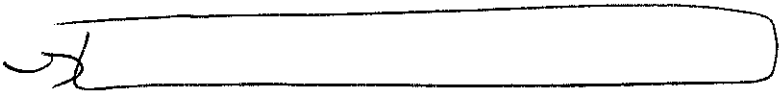
O(nm)

3rd alg

Sweep left to right until hit zero, then go down. Go right anytime you see a 1.

O(n+m)





n+m

grid(m*3+2)