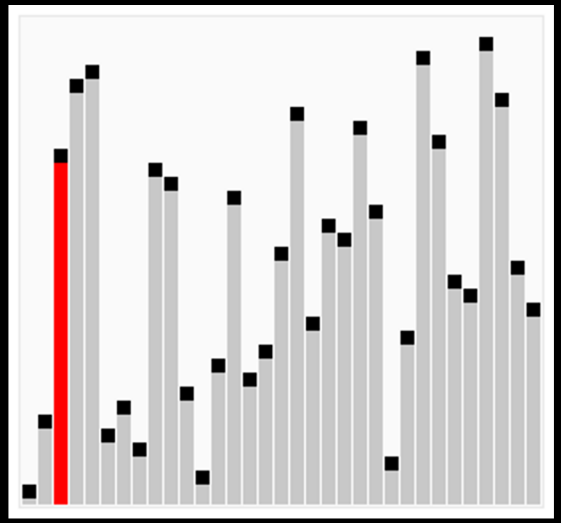
Which sorting algorithm is this?



Bubbles

In a file on your IDE, create a ”person” struct with attributes name and age. Then, in int main(void), create an array of 2 persons. Iterate through the array and print each name and age.

```
typedef struct
{
    string name;
    int age;
}
person;
```

```
int main (void)
{
    person people[2]; → person array named people wl len 2
    people[0].name = "Phyllis";
    people[0].age = 19;
    people[1] //repeat
    for (int i=0; i<2; i++)
    {
        printf("%s\n", people[i].name);
    }
}
```

```
int arr[5];
    ↳ integer array named arr wl len 5
```

CS50 Section 3

- Searching and Running Times
- Structs
- Sorting
- Recursion

Runtime

- Best Case? Ω

  $\Omega(1)$: constant

  $\Omega(n)$: linear

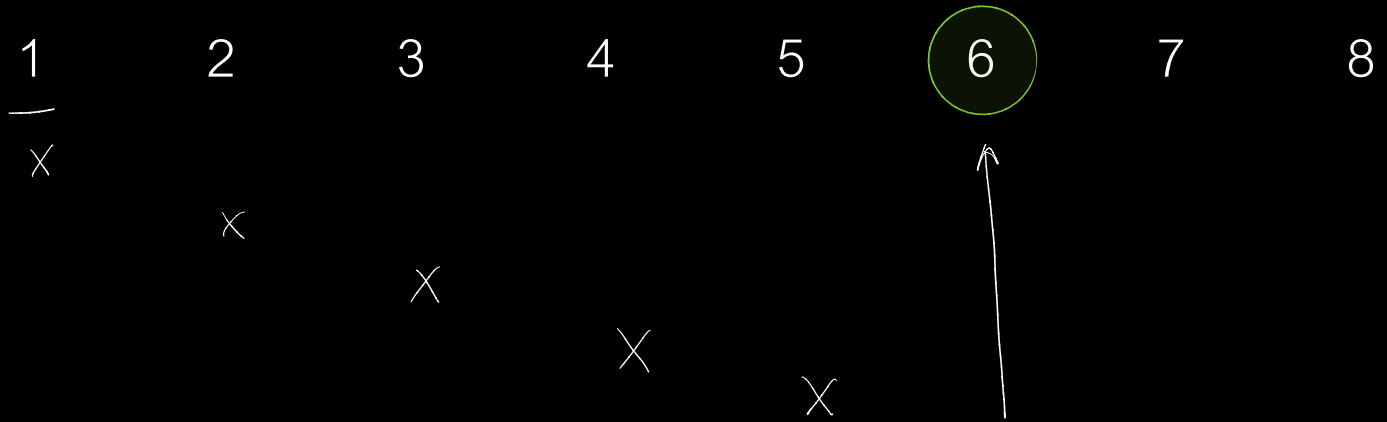
- Worst Case? O

  $O(n)$: constant

Searching

- Linear Search

go through array from beginning
↳ return once you find the element
- some sort of output; fxn returns output
- function stops



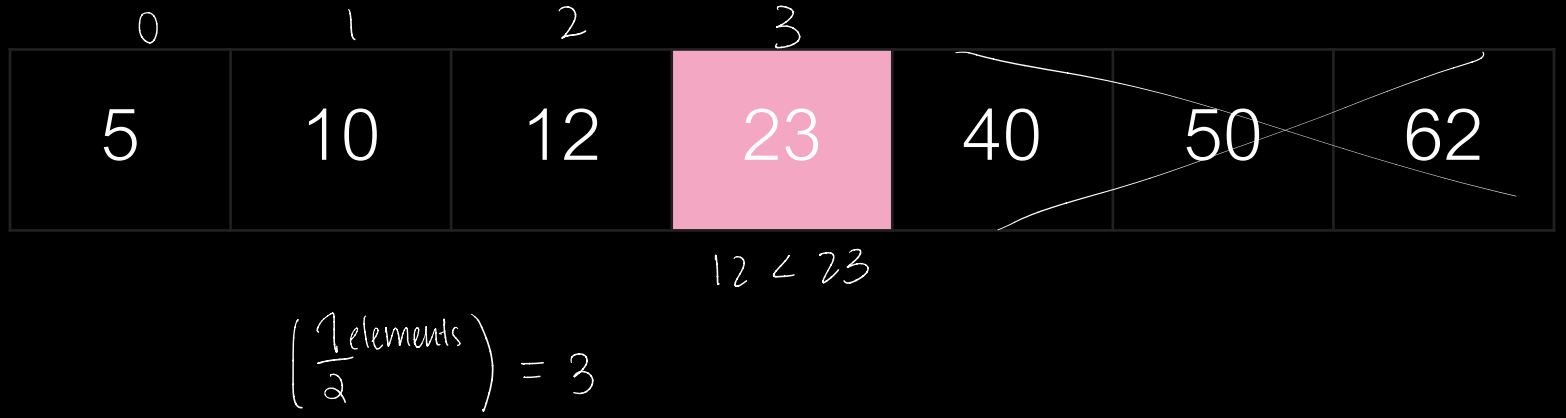
Searching and Run Times

- Linear Search

- Best Case? $\Omega(1)$
- Worst Case? $O(n)$

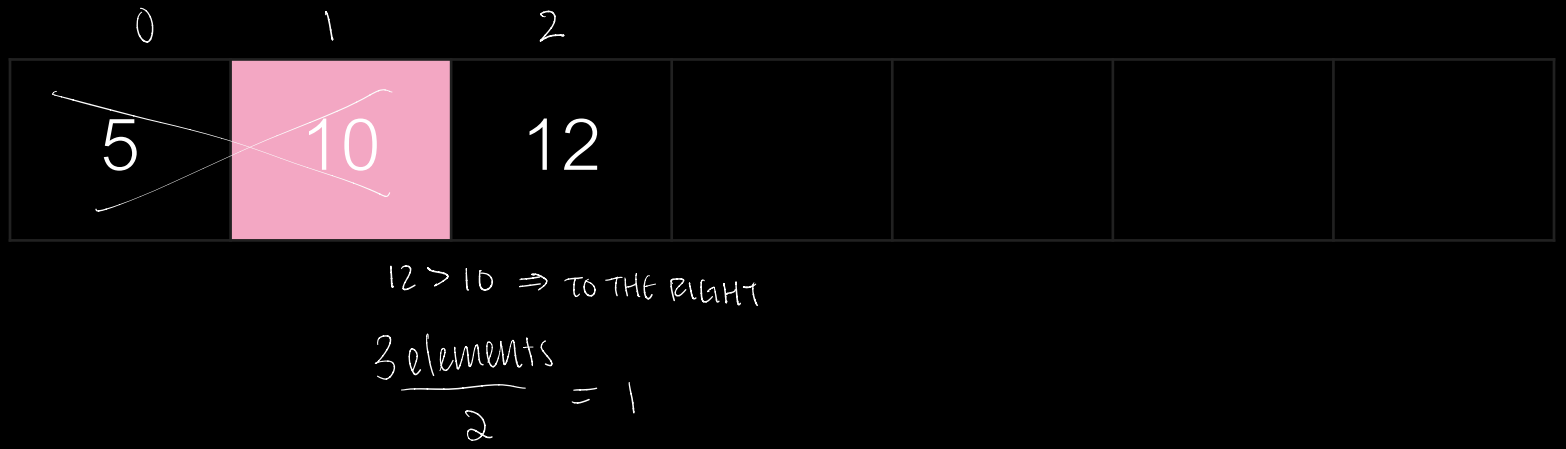
Searching and Run Times

- Binary Search – Let's look for the number 12.



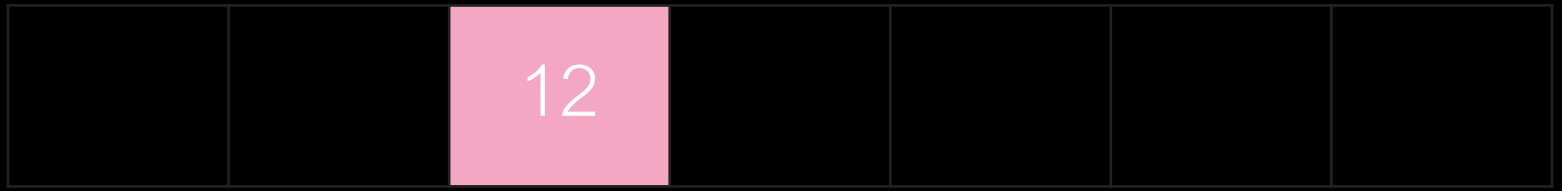
Searching and Run Times

- Binary Search – Let's look for the number 12.



Searching and Run Times

- Binary Search – Let's look for the number 12.



Searching and Run Times

- Binary Search – Let's look for the number 12.

- Best Case? $\Omega(1)$
- Worst Case? $O(\log n)$

$n = 7$

$\log_2 n \sim 3$

- Searching and Running Times
- Structs
- Sorting
- Recursion

Structs

```
typedef struct
{
    string name;
    int age;
}
person;
```

```
typedef struct
{
    type var1;
    type var2;
    ...
}
var3; //name of structure
```

Structs

```
typedef struct
{
    string name;
    int age;
}
person;
```

int i; → int → person

```
person p;
p.name = “Phyllis”;
p.age = 19;
```

Why a struct?

group together properties of an item

Exercise 1: Struct Practice

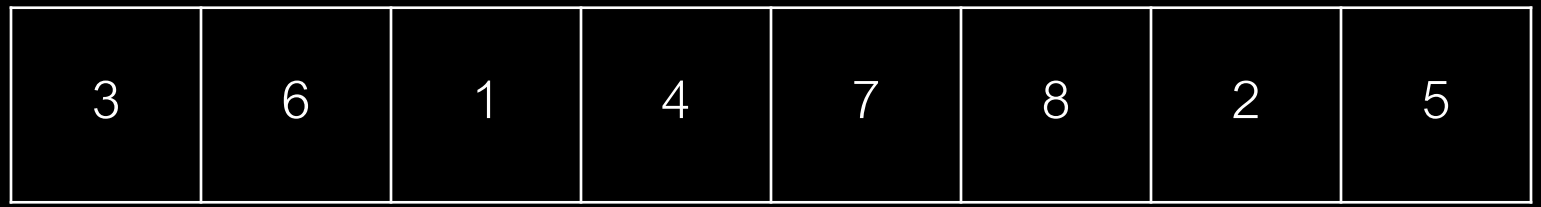
```
typedef struct
{
    string name;
    int age;
}
person;
```

Construct a struct called “birthday” that contains a variable *name* as a *string*, the variable *month* as an *int*, and the variable *day* as an *int*.

Create an array storing birthdays, initialize your contents, and print the contents of your array.

- Searching and Running Times
- Structs
- Sorting
- Recursion

Bubble Sort



ITERATION 1

3 6 1 4 7 8 2 5

3 1 6 4 7 8 2 5

3 1 4 6 7 8 2 5

3 1 4 6 7 8 2 5

3 1 4 6 7 2 8 5

3 1 4 6 7 2 5 8

IT2 3 1 4 6 7 2 5 8

1 3 4 6 7 2 5 8

1 3 4 6 7 2 5 8

1 3 4 6 7 2 5 8

1 3 4 6 7 2 5 8

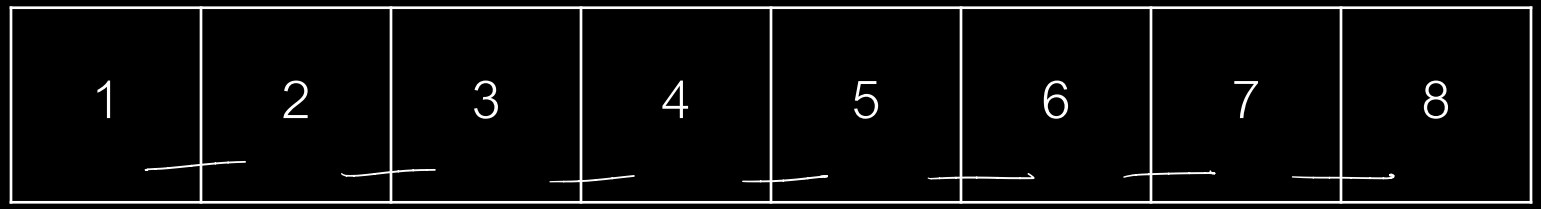
1 3 4 6 2 7 5 8

1 3 4 6 2 5 7 8

1 3 4 6 2 5 7 8

KEEP ITERATING UNTIL NO SWAPS

Bubble Sort



already sorted: bubble sort makes 1 pass

1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
...

} look through all n elements

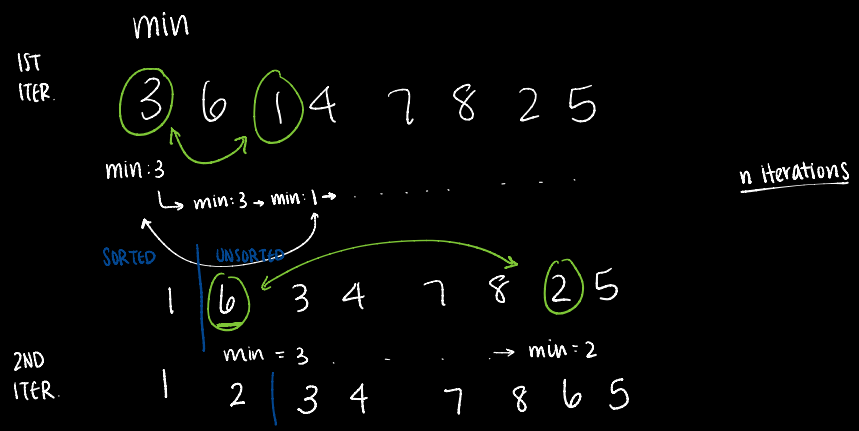
Bubble Sort

3	6	1	4	7	8	2	5

- Best Case? $\Omega(n)$
- Worst Case? n iterations, each iteration: n } $O(n^2)$

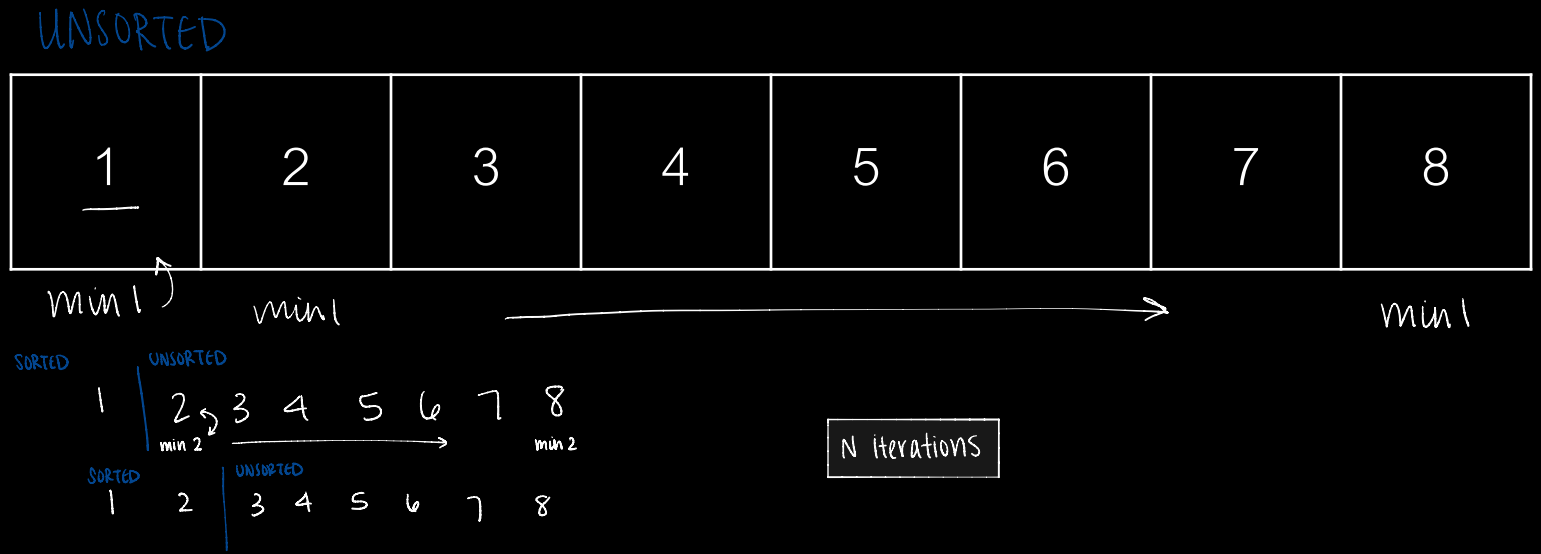
Selection Sort

3	6	1	4	7	8	2	5

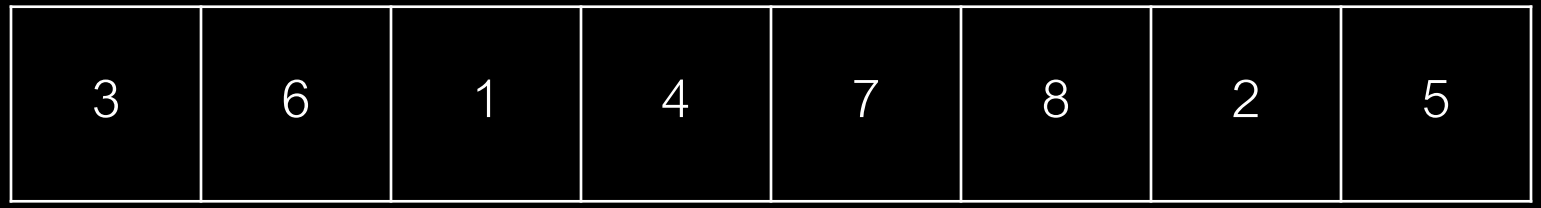


n iterations

Selection Sort



Selection Sort



- Best Case? $\Omega(n^2)$
- Worst Case? $O(n^2)$

Selection Sort

3 5 2 6 1 7

SORTED | UNSORTED

1 | 5 2 6 3 7

i=1

j=1

arr = [3, 5, 2, 6, 1, 7]

```
for (int i=0; i<6 ; i++)
{
    min= arr[i];
    idx = i;
    for (int j=i; j<6 ; j++)
    {
        if (min > arr[j])
        {
            min = arr[j];
            idx = j;
        }
    }
    temp= arr[i];
    arr[i]= min;
    arr[idx]= temp;
}
```

Go through n iterations

FOUND MIN OF UNSORTED ARRAY!

[1, 2, 3]

int temp = arr[0] ⇒ temp=1

arr[0] = arr[2] ⇒ [3, 2, 3]

arr[2] = temp ⇒ [3, 2, 1]

0 1 2 3 4 5

3, 5, 2, 6, 1, 7

i=0

min = 3

idx = 0

j = 0 .. 2 ... 4 ... 5

if (3 > arr[2])

2

min = 2

idx = 2

if 2 > 1

min = 1

idx = 4

min = 1

idx = 4

arr[0] = arr[2]; ⇒ arr[0] = 3

arr[2] = arr[0];

arr[2] = 3

[3,2,3]

temp = 3

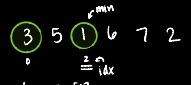
[3,2,3]

arr[0] = 1

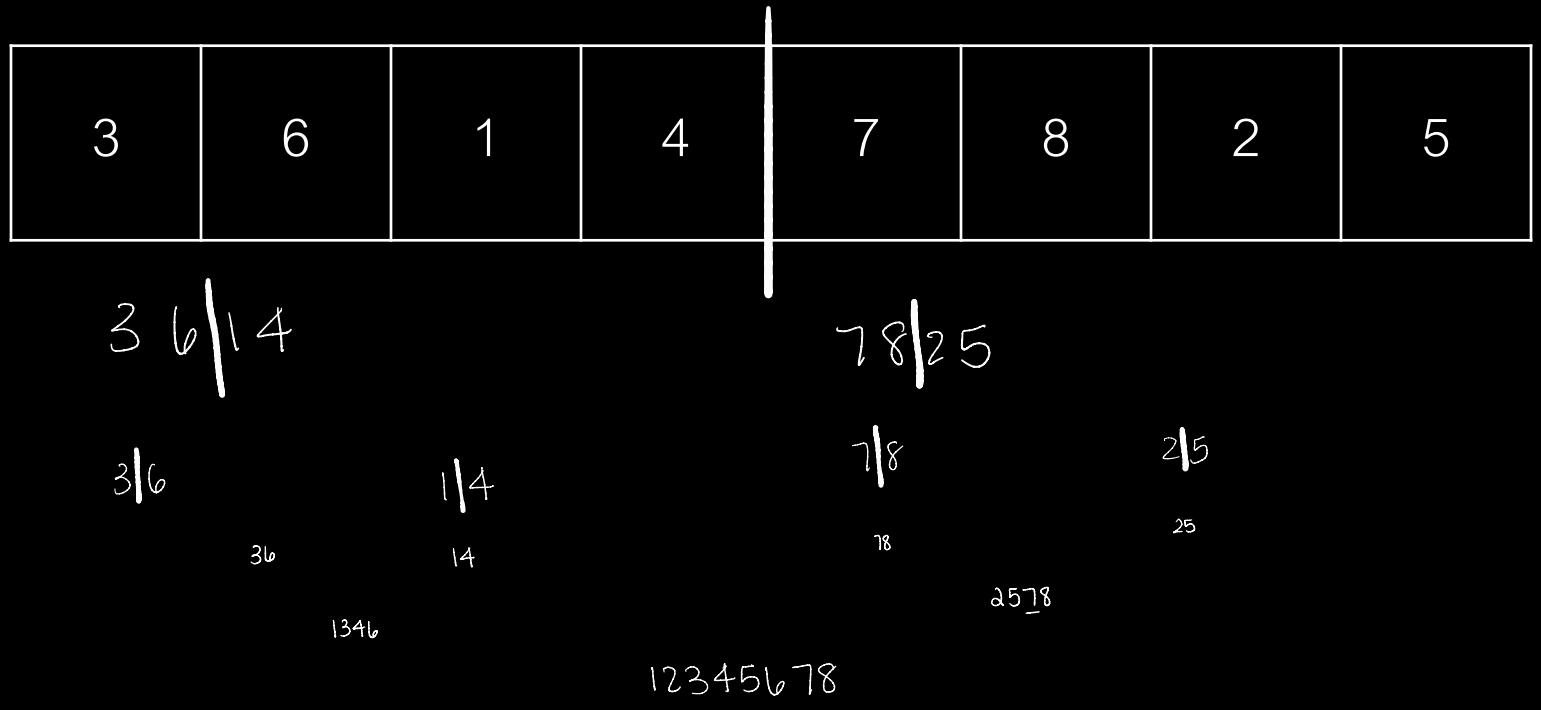
arr[4] = 3

3, 5, 2, 6, 1, 7

1, 5, 2, 6, 3, 7

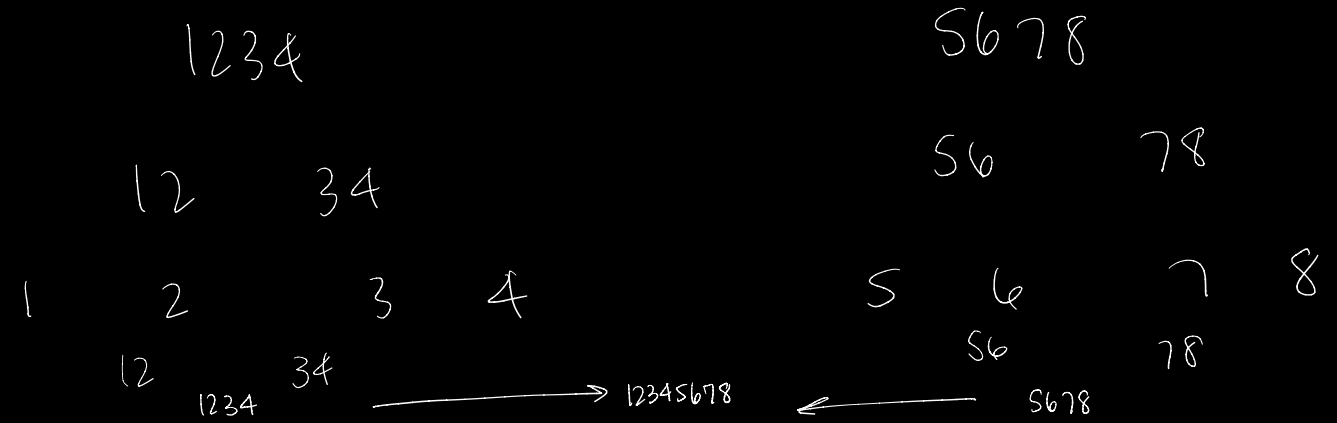


Merge Sort

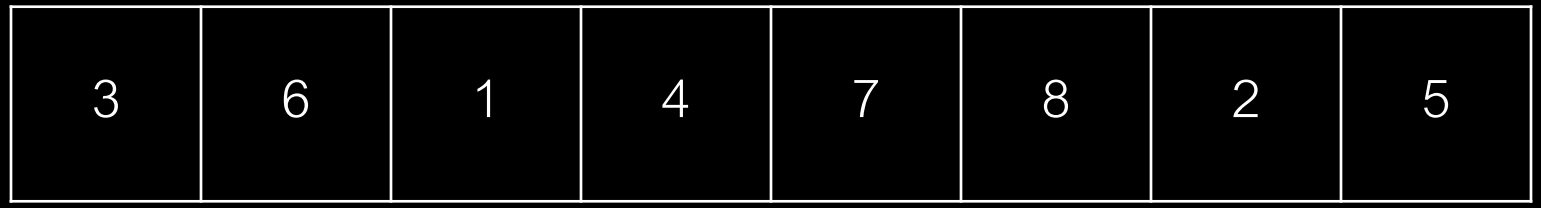


Merge Sort

1	2	3	4	5	6	7	8

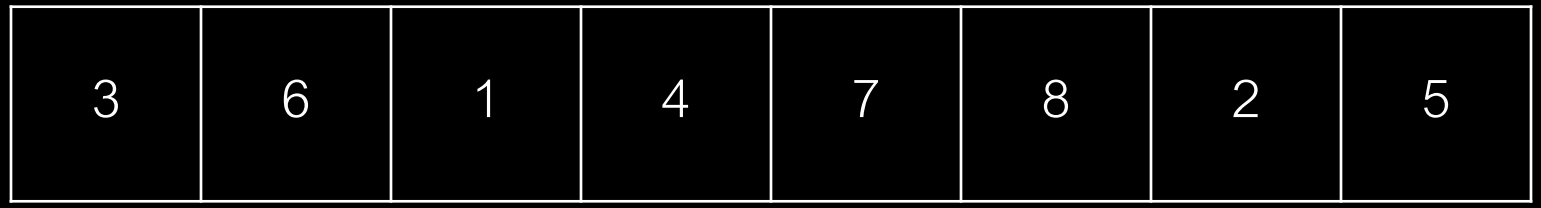


Merge Sort



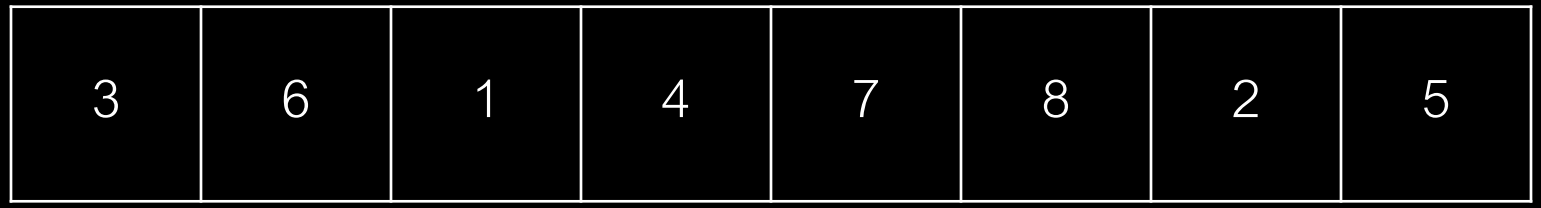
- Best Case? $\Omega(n \log n)$
- Worst Case? $O(n \log n)$

Fun Sorts :D



Stalin Sort (fake sort vibes)

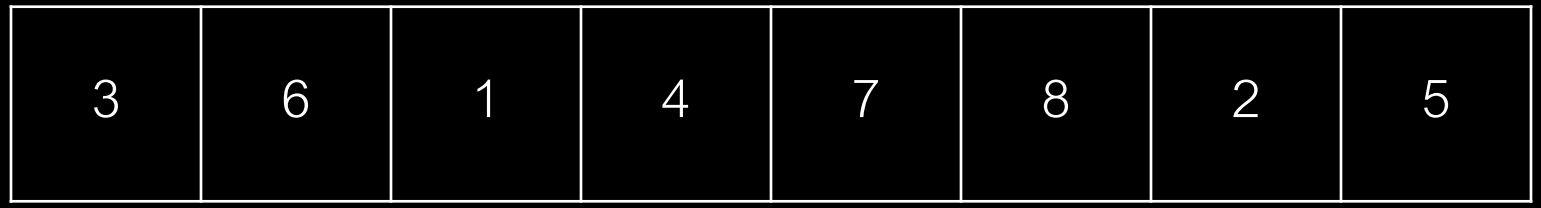
n size array



3 6 7 8

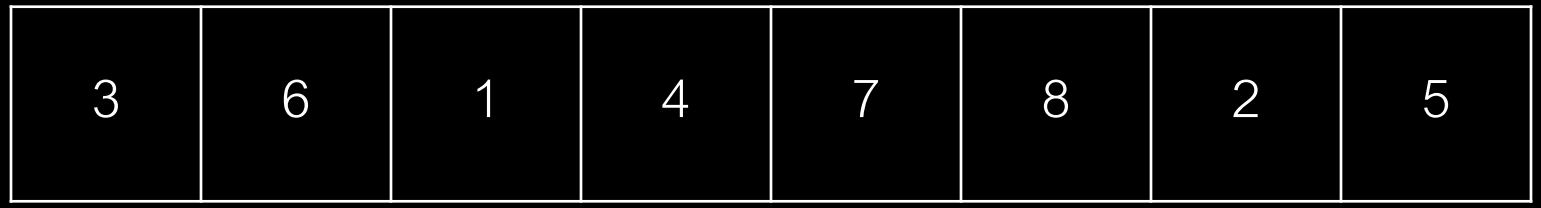
3 6 7 8

Stalin Sort (fake sort vibes)



- Best Case? $\Omega(n)$
- Worst Case? $O(n)$

Bogosort Sort



$O((n+1)!)$

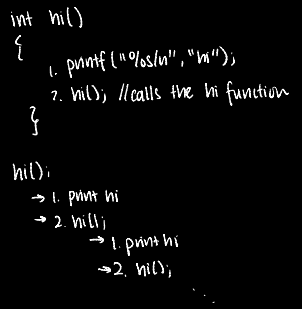
Summary

Selection Sort: BEST / WORST n^2

Bubble Sort: BEST n / WORST n^2

Merge Sort: BEST / WORST $n \log n$

- Searching and Running Times
- Structs
- Sorting
- Recursion



Recursion

- Base Case
- Recursive Call

Write a function that prints integers starting from n line by line, subtracting one for each line until zero is reached using recursion.

Recursion

```
void count(n)
{
    if (n == 0)
    {
        return;
    }
    count(n - 1);
    printf("%i\n", n);
}
```

BASE CASE

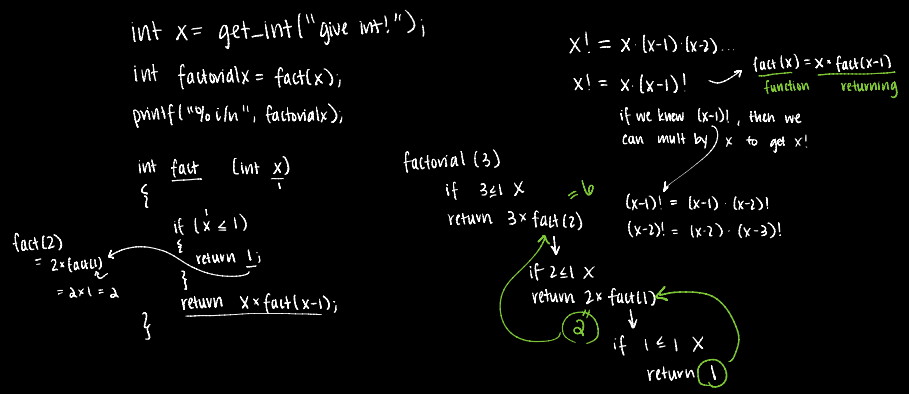
RECURSIVE CALL

```
count(3)
    if 3==0 X
    count(2);
        if 2==0 X
        count(1);
            if 1==0 X
            count(0);
            if 0==0 ✓
        print(1)
    print(2)
print(3)
```

Recursion

- Base Case
- Recursive Call

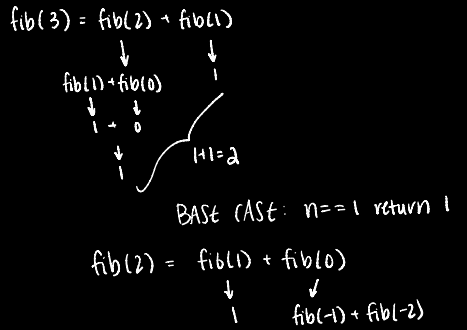
Write a function that determines the factorial of an integer n using recursion. Prompt the user for an integer and print out its factorial.

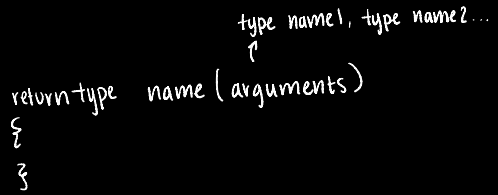


Exercise 2: Fibonacci Numbers

Write a recursive function *fib* that computes the nth Fibonacci number. The 0th Fibonacci number is 0, the 1st Fibonacci number is 1, and every subsequent Fibonacci number is the sum of the two preceding Fibonacci numbers.

fib(7) returns 13






```
int fib(int n); // tells computer fib fxn exists!
int main(void)
{
    int input= get_int("INT: ");
    printf("%i\n", fib(input));
}

int fib(int n)
{
    if (n==0)
    {
        return 0;
    }
    if (n==1)
    {
        return 1;
    }
    return fib(n-1) + fib(n-2)
}
```

Lab

	reversed [5,4,3,2,1]	sorted [1,2,3,4,5]	random [3,2,5,1,4]
Selection	n^2	n^2	n^2
Bubble	n^2	n	n^2
Merge	nlogn	nlogn	nlogn

BUBBLE
MERGE
SELECTION

time ./sort1 reversed10000.txt
USER TIME

CS50 Section 3