CSE12 - Lecture 10

Monday, April 24, 2023 8:00 AM

PA3 due tomorrow
Exam 1 → Friday

Categorizing Runtimes

Let f(n) = 100

Which of the following is **NOT** a correct bound?

A. f(n) is O(2^n)
B. f(n) is O(n^2)
C. f(n) is O(n)
D. f(n) is O(n^{100})
E. None of these

Let f(n) = $3n^3 + 2n + 7$

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n^2)
C. f(n) is O(n)
D. f(n) is O(n^3)
E. None of these

$3n^3 + 2n^3 + 7n^3 = 12n^3$

$c = 12$, $N_0 = 0$, $g(n) = n^3$

$3n^3 + 2n^3 + 7 \rightarrow 5n^3 + 7$ $c = 5$, $N_0 = 7$, $g(n) = n^3$

```
void printAllElementOfArray(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        printf("%d\n", arr[i]);
    }
}
```

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n^2)
C. f(n) is O(n)
D. f(n) is O(n^3)
E. None of these

worst case

$1 + (N+1) + N$

$3N + 2$

$3N + 2n = 5N$

$c = 5$, $N_0 = 0$, $g(n) = N$

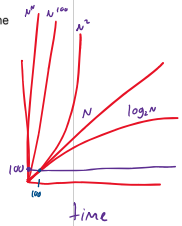
$3N + 2$ $c = 3$, $N_0 = 2$, $g(n) = N$

Big-O

f(n) = O(g(n)), f(n) $\leq$ c * g(n)
for all n ≥ n0

For each function in the list below, it is related to the function below it by O, and the reverse is not true. That is, n is O(n^2) but n^2 is not O(n).

- f(n) = $1/(n^2)$
- f(n) = 1/n
- f(n) = 1
- f(n) = log(n)
- f(n) = sqrt(n)
- f(n) = n
- f(n) = n^2
- f(n) = n^3
- f(n) = n^4
- … and so on for constant polynomials …
- f(n) = 2^n
- f(n) = n!
- f(n) = n^n



Name: _____ PID: _____ Code: 8353

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is $O(n^2)$
C. f(n) is O(n)
D. f(n) is $O(n^3)$
E. None of these

```
void printAllPossibleOrderedPairs(int arr[]) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr.length; j++) {
            printf("%d = %d\n", arr[i], arr[j]);
        }
    }
}
```

Had multiple errors from going to fast!

missed 1++ → N

$1 + (N+1) + 1$ $\quad N+3 +$

$1 + (N+1) + 1$ $\quad N(N+4)$

$= N^2 + 4N + N + 3$

$f(N) = N^2 + 5N + 3$

$N^2 + 5N^2 + 3$

$6N^2 + 3$

$C = 6$ $\quad g(N) = N^2$

$N_0 = 3$

$1 + (N+1) + N + N(1 + (N+1) + N + N) = 2N + 2 + N(2N+3) = 2N^2 + 5N + 2$

$3N + 2$

$3N^2 + 4N + 2$

$f(N) \Rightarrow$

$3N^2 + 4N^2 + 2$

$7N^2 + 2$

$C = 7$ $\quad g(N) = N^2$

$N_0 = 2$

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is $O(n^2)$
C. f(n) is O(n)
D. f(n) is $O(n^3)$
E. None of these

$f(N) \leq C + g(N)$

```
int fibonacci(int num) {
    if (num <= 1) return num;
    return fibonacci(num - 2) + fibonacci(num - 1);
}
```

1 to →

Which of the following is a correct bound?

A. f(n) is $O(2^n)$
B. f(n) is $O(n^2)$
C. f(n) is O(n)
D. f(n) is $O(n^3)$
E. None of these

N=5

f(5)

f(3) f(4)

f(1) f(2) f(2) f(3) f(3) f(4) f(i)

f(1) f(1) f(2) f(2) f(2) f(3) f(3) f(i)

f(2) f(i)

N=5 15 times

N=4 9 times

N=3 5 times

$2^N \rightarrow 2^5 \rightarrow 32$

$2^4 \rightarrow 16$

$2^3 \rightarrow 8$

sorted

```
// A recursive binary search function. It returns location
// of x in given array arr[l..r] is present, otherwise -1
int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        // If the element is present at the middle
        // itself
        if (arr[mid] == x)
            return mid;

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, r, x);
    }

    // We reach here when element is not
    // present in array
    return -1;
}
```

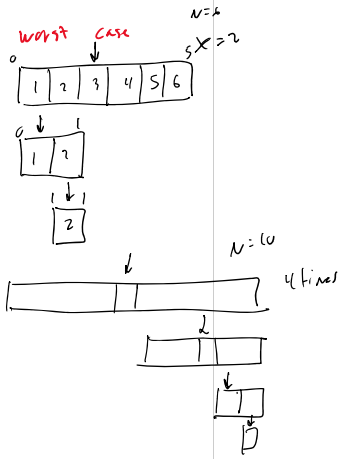
left right

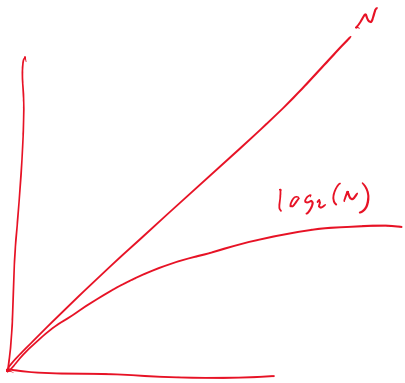
1

1
0 or 1

1
0 or ?

or?





$\log_2(N)$ → # of times you divide by 2 to get 0 (or 1)

What are some correct bounds for binarySearch? What is the smallest correct bound?

O(N), O(N²), O(N³) O(log₂(N))

```
boolean isPrimeAll(int num) {
    // Check for divisors of num
    for (int i = 0; i < num; i += 1) {
        if (num % i == 0) {
            // Any divisor other than 1 or num means num is not prime
            return false;
        }
    }
    // No other divisors found means num is prime
    return true;
}
```

What is the smallest correct bound? O(N)

```
boolean isPrimeHalf(int num) {
    // Check for divisors of num
    for (int i = 0; i < num / 2; i++) {
        if (num % i == 0) {
            // Any divisor other than 1 or num means num is not prime
            return false;
        }
    }
    // No other divisors found means num is prime
    return true;
}
```

What is the smallest correct bound? O(N)

```
void printAllItemsTwice(int arr[], int size)
{
    for (int i = 0; i < size; i++) {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < size; i++) {
        printf("%d\n", arr[i]);
    }
}
```

What is the smallest correct bound?

```
void printFirstItemThenFirstHalfThenSayHi100Times(int arr[], int size)
{
    printf("First element of array = %d\n",arr[0]);

    for (int i = 0; i < size/2; i++) {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < 100; i++) {
        printf("Hi\n");
    }
}
```

What is the smallest correct bound?

```
void printAllNumbersThenAllPairSums(int arr[], int size)
{
    for (int i = 0; i < size; i++) {
        printf("%d\n", arr[i]);
    }

    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d\n", arr[i] + arr[j]);
        }
    }
}
```

What is the smallest correct bound?