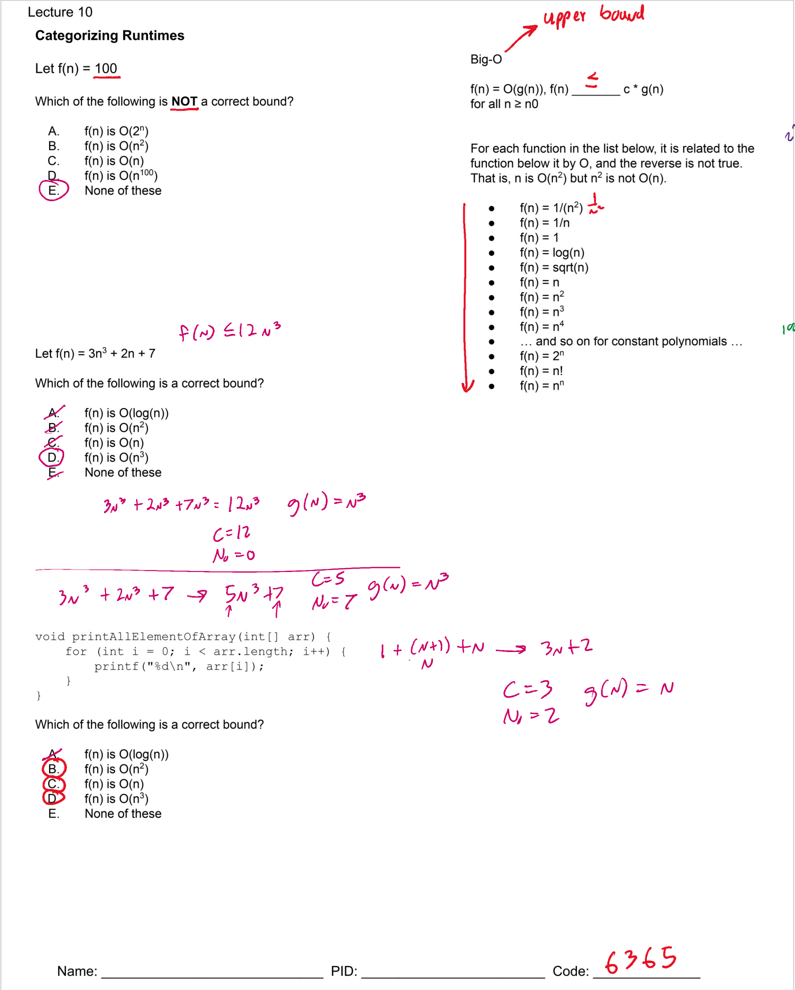
CSE12 - Lecture 10

Friday, October 20, 2023 8:00 AM

PA3 → hidden tests
feedback or wheat/chaff tests



Lecture 10

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

$f(N) \leq 12N^3$

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

$3N^3 + 2N^3 + 7N^3 = 12N^3$ $g(N) = N^3$

$C = 12$
$N_0 = 0$

$3N^3 + 2N^3 + 7 \rightarrow \underset{\uparrow}{5N^3} + \underset{\uparrow}{7}$ $C = 5$ $N_0 = 7$ $g(N) = N^3$

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

$1 + \underset{N}{(N+1)} + N \longrightarrow 3N + 2$

$C = 3$ $g(N) = N$
$N_0 = 2$

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

upper bound

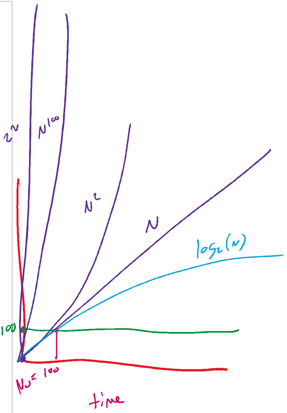
Big-O

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).

- $f(n) = 1/(n^2)$ $\frac{1}{n^2}$
- 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: 6365



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

$1 + (N+1) + N$

$\left[1 + (N+1) + N\right] * N$
N

$2N+2$ +

$(3N+2) * N$

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

$f(n) = 3N^2 + 4N + 2$

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

$7N^2 + 2$

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

$N_0 = 2$ $O(g(n))$

$O(N^2)$

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

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

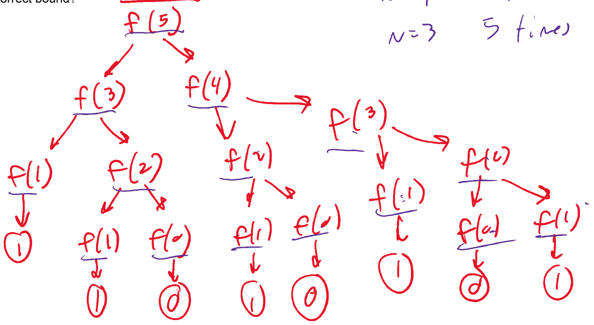
15 times

$2^N \rightarrow$ $2^5 = 32$
$2^4 \rightarrow 16$
$2^3 \rightarrow 8$

Which of the following is a correct bound?

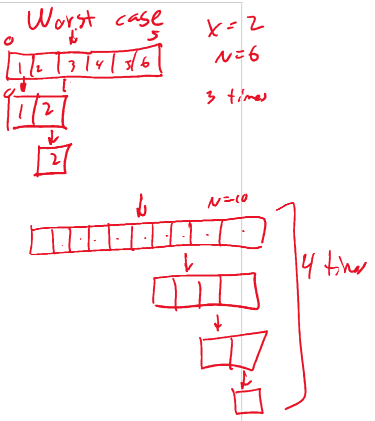
N=5

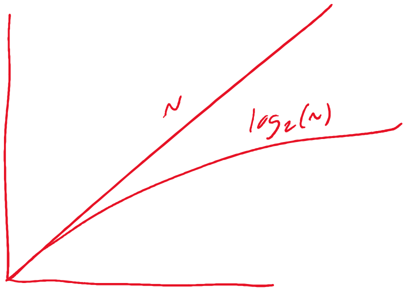
[✓ A.] f(n) is $O(2^n)$
~~B.~~ f(n) is $O(n^2)$
~~C.~~ f(n) is O(n)
~~D.~~ f(n) is $O(n^3)$
E. None of these



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)
{                                left   right
    if (r >= l) {                                         1
        int mid = l + (r - l) / 2;

        // If the element is present at the middle
        // itself
        if (arr[mid] == x)                                1
            return mid;                                   0 1

        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid] > x)                                 1 0
            return binarySearch(arr, l, mid - 1, x);      0 1 ?

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, r, x);          1 0 ?
    }

    // We reach here when element is not
    // present in array
    return -1;                                            1 0
}
```





$\log_2(N) \rightarrow$ # c- times you divide by 2 to get 0 (or -1)

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

$O(N), O(n^2), O(n^3), \ldots$ $\boxed{O(\log_2(n))}$

```
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?

$1 + (N+1) + N$
N

1

$3N + 3$

$C = 3$ $g(N) = N$ $O(N)$
$N_0 = 3$

```
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
```

What is the smallest correct bound?

$1 + (\frac{N}{2} + 1) + \frac{N}{2}$
$\frac{N}{2}$

$\frac{3N}{2} + 3$

$C = 3/2$ $g(N) = N$ $O(N)$
$N_0 = 3$

```
        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;
}
```

$\frac{N}{2}$

)

```
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?