CSE12 - Lecture 10 - C00

Friday, October 14, 2022 11:00 AM

PA3 due Tuesday

Exam1 → next Friday

↳ 2D search / BFS / DFS

Categorizing Runtimes

Let f(n) = 100

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

A. f(n) is O(2n)
B. f(n) is O(n2)
C. f(n) is O(n)
D. f(n) is O(n100)
E. None of these

Let f(n) = 3n3 + 2n + 7

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n2)
C. f(n) is O(n)
D. f(n) is O(n3)
E. None of these

$3N^3 + 2N^3 + 7N^3 \rightarrow 12N^3$ $g(N) = N^3$ $C = 12$ $N_0 = 0$

$3N^3 + 2N^3 + 7$ $5N^3 + 7$ $C = 5$ $N_0 = 7$

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

N → arr.length

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

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n2)
C. f(n) is O(n)
D. f(n) is O(n3)
E. None of these

$f(N) = 3N + 2$ $C = 3$ $N_0 = 2$ $g(N) = N$ $O(N)$

Big-O

f(n) = O(g(n)), f(n) ≤ 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(n2) but n2 is not O(n).

- f(n) = 1/(n2)
- f(n) = 1/n
- f(n) = 1
- f(n) = log(n)
- f(n) = sqrt(n)
- f(n) = n
- f(n) = n2
- f(n) = n3
- f(n) = n4
- … and so on for constant polynomials …
- f(n) = 2n
- f(n) = n!
- f(n) = nn

$f(N) \leq C * g(N)$ N_0

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n2)
C. f(n) is O(n)
D. f(n) is O(n3)
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]);
        }
    }
}
```

$1 + (N+1) + N$

$N * [1 + (N+1) + N]$

$2N + 2 + (3N + 2) * N$

$3N^2 + 4N + 2$

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

$7N^2 + 2$

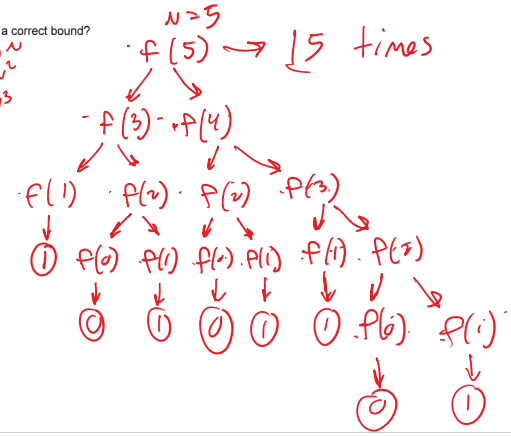
$C = 7 \quad g(N) = N^2$

$N_0 = 2$

Which of the following is a correct bound?

A. f(n) is O(log(n))
B. f(n) is O(n2) N^2
C. f(n) is O(n)
D. f(n) is O(n3) N^3
E. None of these

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



$N = 7$, 39 times

$N = 6$, 24 times

$N = 4$, 9 times

$N = 3$, 5 times

2^N → $2^6 \to 64$, $2^5 \to 32$, $2^4 \to 16$, $2^3 \to 8$

Which of the following is a correct bound?

A. f(n) is O(2n) 2^N
B. f(n) is O(n2) N^2
C. f(n) is O(n)
D. f(n) is O(n3) 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)
{
    if (r >= l) {
        int mid = l + (r - 1) / 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 0

1
0 1

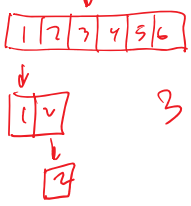
0 1
0 1

0 1

1

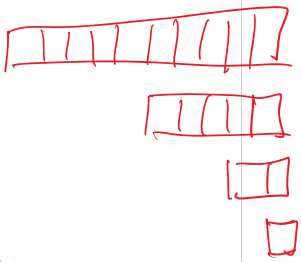
Worst case

N=6
x=2



3 times

N=10



4 times

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

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

$O(N), O(N^2), \ldots \quad O(\log_2(N))$

```
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

0

1

$3N + 3$

$O(N)$

$g(N) = N$

$C = 3$

$N_0 = 3$

```
boolean isPrimeHalf(int num) {
    // Check for divisors of num
    for (int i = 0; i < num / 2; i++) {
```

What is the smallest correct bound?

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

$3N + 3$

```
    return true;
}

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?

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

$\frac{N}{2}$

0

1

$N_0 = 3$

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

$C = 2$

$N_0 = 3$

$g(N) = N$

$O(N)$

```
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 printFirstItemThenFirstHalfThenSayHil00Times(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?