

# CSE12 - Lecture 11 - C00

Monday, October 17, 2022 11:00 AM

PA3 due tomorrow

Exam 1 on Friday

## Measuring Runtime

Count how many times each line executes, then say which  $\Theta()$  statement(s) is(are) true.

```
int maxDifference(int[] arr){
    max = 0;
    for (int i=0; i<arr.length; i++) {
        for (int j=0; j<arr.length; j++) {
            if (arr[i] - arr[j] > max)
                max = arr[i] - arr[j];
        }
    }
    return max;
}
```

$n^2 + 2$   
 $\Theta(n^2)$   
↑

Assume  $n = \text{arr.length}$

- A.  $f(n) = \Theta(2^n)$   
 B.  $f(n) = \Theta(n^2)$   
 C.  $f(n) = \Theta(n)$   
 D.  $f(n) = \Theta(n^3)$   
 E. Other/none/more

$1 + (n+1) + n$   
 $n + [1 + (n+1) + n]$   
 $0$  best case  
 $N$  worst case  
 $4 + 2n + n(2 + 4n)$   
 $4n^2 + 4n + 4$   
 $4n^2 + 4n^2 + 4 \rightarrow 8n^2 + 4$   
 $C = 8$   
 $N_0 = 4$   
 $g(n) = n^2$   
 $\Theta(n^2)$

Big  $\mathcal{O}$  Upper bound

$f(n) = \mathcal{O}(g(n))$ ,  $f(n) \leq c * g(n)$   
 for all  $n \geq n_0$

Big  $\Omega$  omega Lower bound

$f(n) = \Omega(g(n))$ ,  $f(n) \geq c * g(n)$   
 for all  $n \geq n_0$

Big  $\Theta$  Tight bound

$f(n) = \Theta(g(n))$ ,  $f(n) = c * g(n)$   
 for all  $n \geq n_0$

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

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

Count how many times each line executes, then say which  $\Theta()$  statement(s) is(are) true.

```
int sumTheMiddle(int[] arr){
    int range = 100;
    int start = arr.length/2 - range/2;
    int sum = 0;
    for (int i=start; i<start+range; i++)
        sum += arr[i];
    return sum;
}
```

$6 + 300$

$306$

Assume  $n = \text{arr.length}$

- A.  $f(n) = \Theta(2^n)$   
 B.  $f(n) = \Theta(n^2)$   
 C.  $f(n) = \Theta(n)$   
 D.  $f(n) = \Theta(1)$   
 E. None of these

$f(n) = 306$

$C = 306$   
 $N_0 = 0$

$g(n) = 1$

$\Theta(1)$

constant time

range = 100

start =  $\frac{n}{2} - 50$

start + range =  $\frac{n}{2} + 50$

$n = 100$

start =  $\frac{100}{2} - 50 = 0$

start + range =  $\frac{100}{2} + 50 = 100$

$n = 10000$

start =  $\frac{10000}{2} - 50 = 4950$

start + range =  $\frac{10000}{2} + 50 = 5050$

$$\begin{aligned} & \text{for (int i = 0; i < size; i++)} \{ \quad 1 + (n+1) + n \quad 3n + 2 \\ & \quad \text{printf("%d\n", arr[i]);} \quad \quad \quad \downarrow \\ & \} \end{aligned}$$

$$\begin{aligned} & \text{for (int i = 0; i < size; i++)} \{ \quad 1 + (n+1) + n \quad 3n + 2 \\ & \quad \text{printf("%d\n", arr[i]);} \quad \quad \quad \hline & \} \quad \quad \quad 6n + 4 \end{aligned}$$

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

$$M=4 \quad Q(N)$$

```
{
    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");
    }
}
```

$$\frac{N}{2} \left[ \begin{array}{c} f \\ \vdots \\ \end{array} \right]$$
$$100 \left[ \begin{array}{c} f \\ \end{array} \right]$$
$$h(n)$$
$$\begin{array}{r}
 1 \\
 1 + (\frac{n}{2} + 1) + \frac{n}{2} \\
 1 + (100 + 1) + 100 \\
 100
 \end{array}
 \qquad
 \begin{array}{r}
 1 + \\
 \frac{3n}{2} + 2 \\
 + \\
 2 + 300 \\
 \hline
 \frac{3n}{2} + 305 \\
 \boxed{\Theta(n)}
 \end{array}$$

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

$$\sim \left[ \sim \left[ \right] \right]$$

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

$$\begin{array}{r} 3n+2 \\ + \\ 2n+2+ \\ n+(3n+2) \end{array}$$
$$N^2 + N$$
$$3\underline{n^2} + 7\underline{n} + 4_-$$

$\sim (n^2)$

## Selection Sort

```
import java.util.Arrays;
public class Sort {
    public static void sortA(int[] arr) {
        for(int i = 0; i < arr.length; i += 1) {
            System.out.print(Arrays.toString(arr) + " -> ");
            int minIndex = i;
            for(int j = i; j < arr.length; j += 1) {
                if(arr[minIndex] > arr[j]) { minIndex = j; }
            }
            int temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
            System.out.println(Arrays.toString(arr));
        }
    }
}
```

*N*  $\left[ \begin{array}{l} \frac{N}{2} \\ \text{swap} \end{array} \right]$

Selection Sort – what does it print out?

Sort.sortA(new int[] { 53, 83, 15, 45, 49 });

[53, 83, 15, 45, 49] ->  $N=5$

15	83	53	45	49	5
15	45	53	83	49	4
15	45	49	83	53	3
15	45	49	53	83	2
15	45	49	53	83	1

Worst case: reverse sorted array  
83 53 49 45 15

best case: sorted array  
15 45 49 53 83

What is the runtime? Consider the shape of the input array.

Worse case:

$\Theta(N^2)$

Best case:

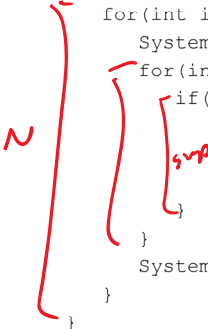
$\Theta(N^2)$

is Sorted ( )

$\Theta(N)$

## Insertion Sort

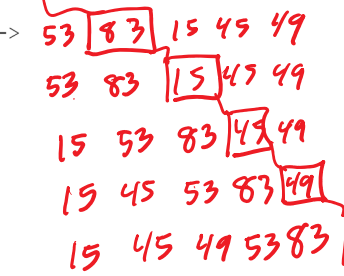
```
import java.util.Arrays;
public class Sort {
    public static void sortB(int[] arr) {
        for(int i = 0; i < arr.length; i += 1) {
            System.out.print(Arrays.toString(arr) + " -> ");
            for(int j = i; j > 0; j -= 1) {
                if(arr[j] < arr[j-1]) {
                    int temp = arr[j-1];
                    arr[j-1] = arr[j];
                    arr[j] = temp;
                }
            }
            System.out.println(Arrays.toString(arr));
        }
    }
}
```



Insertion Sort – what does it print out?

```
Sort.sortB(new int[]{ 53, 83, 15, 45, 49 });
```

[53, 83, 15, 45, 49] -> 53 83 15 45 49  
53 83 15 45 49  
15 53 83 45 49  
15 45 53 83 49  
15 45 49 53 83



What is the runtime? Consider the shape of the input array.

Worse case:

Best case: