

CSE12 - Lecture 11

Wednesday, April 26, 2023 8:00 AM

PA 3 → due today @ 10pm

PA 4 → released today → due Tuesday

Exam 1 → Friday

↳ BFS/DFS and before

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;
}
```

$N \times N$ $1 + (n+1) + n$
 $N(1 + (n+1) + n)$
 0 or N
 best case worst case
 $3 + 2n + n(2 + 4n)$

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

$$P(n) = 4n^2 + 4n + 3$$

$$4n^2 + 4n + 3$$

$$8n^2 + 3$$

$$g(n) = n^2$$

$$C=8$$

$$N_0=3$$



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;
}
```

N^2 $1 + 101 + 100$
 100

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

constant time

$$P(n) = 306$$

$$C=306$$

$$N_0=0$$

$$g(n) = 1$$

$$\theta(1)$$

$$n=100$$

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

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

$$\rightarrow 100$$

$$n=10000$$

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

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

$$100$$

Big O upper bound

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

Big Ω omega lower bound

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

Big Θ 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 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 ...
- $f(n) = 2^n$
- $f(n) = n!$
- $f(n) = n^n$

$$n = 1000$$

$$n = 10000$$

$$1000000000$$

Name: _____ PID: _____ Code: 4020

```
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 tight bound?

$\Theta(N)$

$$\begin{array}{rcl}
 1 + (N+1) + N & 3N+2 \\
 N & + \\
 1 + (N+1) + N & 3N+2 \\
 N & \hline
 6N+4 & g(N) = N \\
 c=6 & \Theta(N) \\
 N_0=4 &
 \end{array}$$

```
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 tight bound?

$\Theta(N)$

$$\begin{array}{rcl}
 1 & & \\
 1 + (\frac{N}{2} + 1) + \frac{N}{2} & \frac{3N}{2} + 2 \\
 \frac{N}{2} & & \\
 1 + 101 + 100 & 302 \\
 100 & \hline
 f(N) = \frac{3N}{2} + 304 & \\
 \Theta(N) &
 \end{array}$$

```
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 tight bound?

$\Theta(N^2)$

$$\begin{array}{rcl}
 1 + (N+1) + N & 3N+2 \\
 N & & \\
 1 + (N+1) + N & 2N+2 + \\
 N(1 + (N+1) + N) & 3N^2 + 2N \\
 N & \hline
 f(N) = 3N^2 + 7N + 4 & \\
 \Theta(N^2) &
 \end{array}$$

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));
        }
    }
}
```

Selection Sort – what does it print out?

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

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

Handwritten diagram showing the selection sort process for the array [53, 83, 15, 45, 49]. The array is shown in a grid with columns representing the current state of the array. Red boxes and arrows indicate the selection of the minimum element and its swap with the first element of the current subarray. The process repeats for each subarray of size 4, 3, 2, and 1.

15	83	53	45	49
15	45	53	83	49
15	45	49	83	53
15	45	49	53	83
15	45	49	53	83

$N=5$

Handwritten diagram showing the selection sort process for the array [53, 83, 15, 45, 49]. The array is shown in a grid with columns representing the current state of the array. Red boxes and arrows indicate the selection of the minimum element and its swap with the first element of the current subarray. The process repeats for each subarray of size 4, 3, 2, and 1.

Worse 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: $\Theta(N^2) \rightarrow$ 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));
        }
    }
}
```

Handwritten notes: A large red bracket on the left of the code is labeled with a red N . Inside the code, a red bracket on the right of the inner loop is labeled with a red $\frac{N}{2}$. Another red $\frac{N}{2}$ is written next to the $j > 0$ condition.

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

Handwritten notes: The sequence of array states is shown with red boxes around the elements being compared or shifted. Red arrows indicate the movement of elements. The final state is 15 45 49 53 83.

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

Worse case: $\Theta(N^2)$

Best case: $\Theta(N^2)$