CSE12 - Lecture 11

Monday, October 23, 2023 8:00 AM

PA3 due Wed @ 8am

Exam 1 → Wed @ 8am

Lecture 11

Measuring Runtime

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

```
int maxDifference(int[] arr){
   max = 0;                                        1
   for (int i=0; i<arr.length;  i++) {            1+(N+1)+N
      for (int j=0; j<arr.length; j++) {          1+(N+1)+N
         if (arr[i] - arr[j] > max)               N
            max = arr[i] - arr[j];                0 or N
      }
   }
   return max;                                    1
}
```

N N

0 → best case
N → worst case

N^2

Assume n = arr.length

A. f(n) = θ(2^n)
[✓ B.] f(n) = θ(n^2)
C. f(n) = θ(n)
D. f(n) = θ(n^3)
E. Other/none/more

$N + N^2 + N + N^2 + N^2 + N^2 + 1 + 1 + N + 1 + N + 1$

$\boxed{4N^2} + 4N + 4$

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

$C = 8$ $g(N) = N^2$ $O(N^2)$

$N_0 = 4$

$\boxed{4N^2} + 4 + 4 = 4N^2 + 8$

$C = 4$ $\Omega(N^2)$

$N_0 = 8$

Big O upper bound

f(n) = O(g(n)), f(n) <= c * g(n) for all n ≥ n0

Big Ω omega lower bound

f(n) = Ω(g(n)), f(n) >= c * g(n) for all n ≥ n0

Big Θ theta tight bound

f(n) = Θ(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(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

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

```
int sumTheMiddle(int[] arr){
 1 int range = 100;
 1 int start = arr.length/2 - range/2;
 1 int sum = 0;
1+(100+1)+100 for (int i=start; i<start+range;  i++)
   {
100   sum += arr[i];
   }
 1 return max;
}
```

6 + 300

306

Assume n = arr.length

A. f(n) = θ(2^n)
B. f(n) = θ(n^2)
C. f(n) = θ(n)
[✓ D.] f(n) = θ(1)
E. None of these

$f(N) = 306$ $g(N) = 1$

$C = 306$

$N_0 = 0$

range = 100

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

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

N = 100

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

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

N = 10000

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

start + range = $\frac{10000}{2} + 50 = 5050$ → 100

Name: ______________ PID: ______________ Code: 8228

```
void printAllItemsTwice(int arr[], int size)
{
   for (int i = 0; i < size; i++) {           1 + (N+1) + N
      printf("%d\n", arr[i]);                  N
   }
   for (int i = 0; i < size; i++) {           1 + (N+1) + N
      printf("%d\n", arr[i]);                  N
   }
}
```

2N

N + N

2N

What is the tight bound?

→ $\boxed{\Theta(N)}$

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

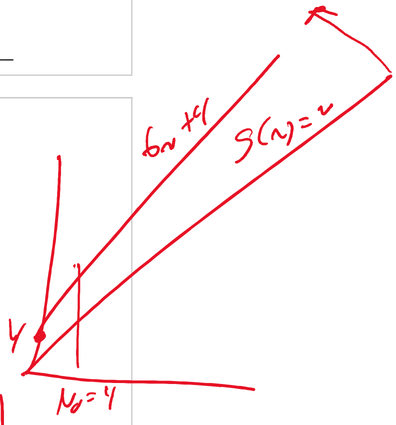
$6N + 4$

$\boxed{g(N) = N}$

$C = \cancel{6}\ 7$

$N_0 = 4$

$\boxed{\Theta(N)}$



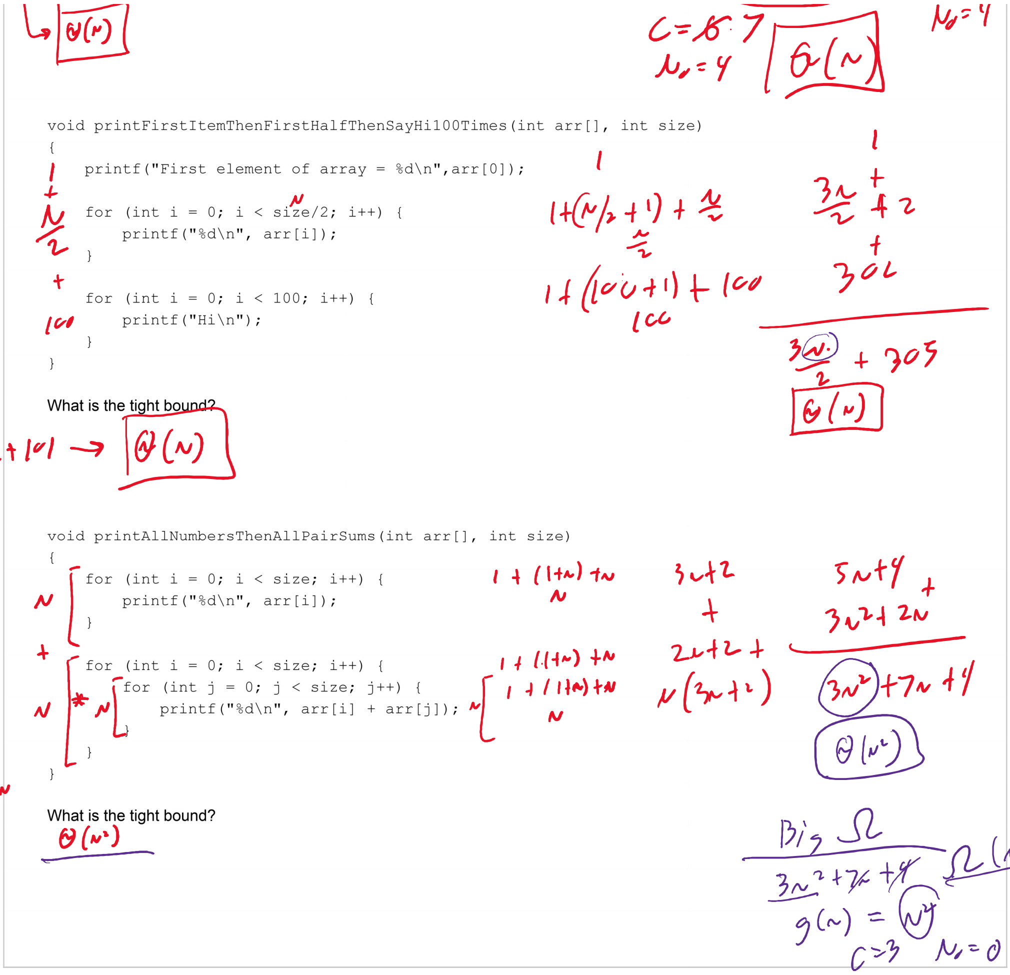
$\boxed{\Theta(N)}$

$C = \cancel{6}\ 7$

$N_0 = 4$

$\boxed{\Theta(N)}$

$N_0 = 4$



```
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");
    }
}
```

$1 + \frac{N}{2} + 100$

N

1

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

$\frac{N}{2}$

$1 + (100 + 1) + 100$

100

$1 + \frac{3N}{2} + 2 + 302$

$\frac{3N}{2} + 305$

$\boxed{\Theta(N)}$

What is the tight bound?

$\frac{N}{2} + 101 \rightarrow \boxed{\Theta(N)}$

```
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]);
        }
    }
}
```

$N + N^2$

$N + N * N$

$N^2 + N$

$1 + (1+N) + N$

N

$1 + (1+N) + N$

$N \left[1 + (1+N) + N \right.$

N

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

$5N+4 + 3N^2 + 2N$

$3N^2 + 7N + 4$

$\Theta(N^2)$

What is the tight bound?

$\Theta(N^2)$

Big O

$3N^2 + 7N^2 + 4N^2$

$14N^2$

$C = 14$

$N_0 = 0$

$g(N) = N^2$

$O(N^2)$

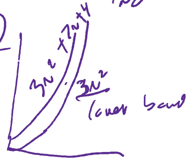
Big Ω

$3N^2 + 7N + 4$

$\Omega(N^2)$

$g(N) = N^2$

$C = 3 \quad N_0 = 0$



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

$\frac{N}{2}$

Selection Sort – what does it print out? $N=5$

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

[53, 83, 15, 45, 49] -> 15 | 83 53 45 49     5
                        15 45 | 53 83 49     4
                        15 45 49 | 83 53     3
                                             2
                        15 45 49 53 | 83     1
                        15 45 49 53 83
```

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: $\Theta(N^2)$

I → 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) {        3
            if(arr[j] < arr[j-1]) {           7
               int temp = arr[j-1];
               arr[j-1] = arr[j];              8
               arr[j] = temp;
            }
         }
         System.out.println(Arrays.toString(arr));
      }
   }
}
```

N $\frac{N}{2}$

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
```

15 45 49 53 83

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

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

Best case: $\Theta(N^2$