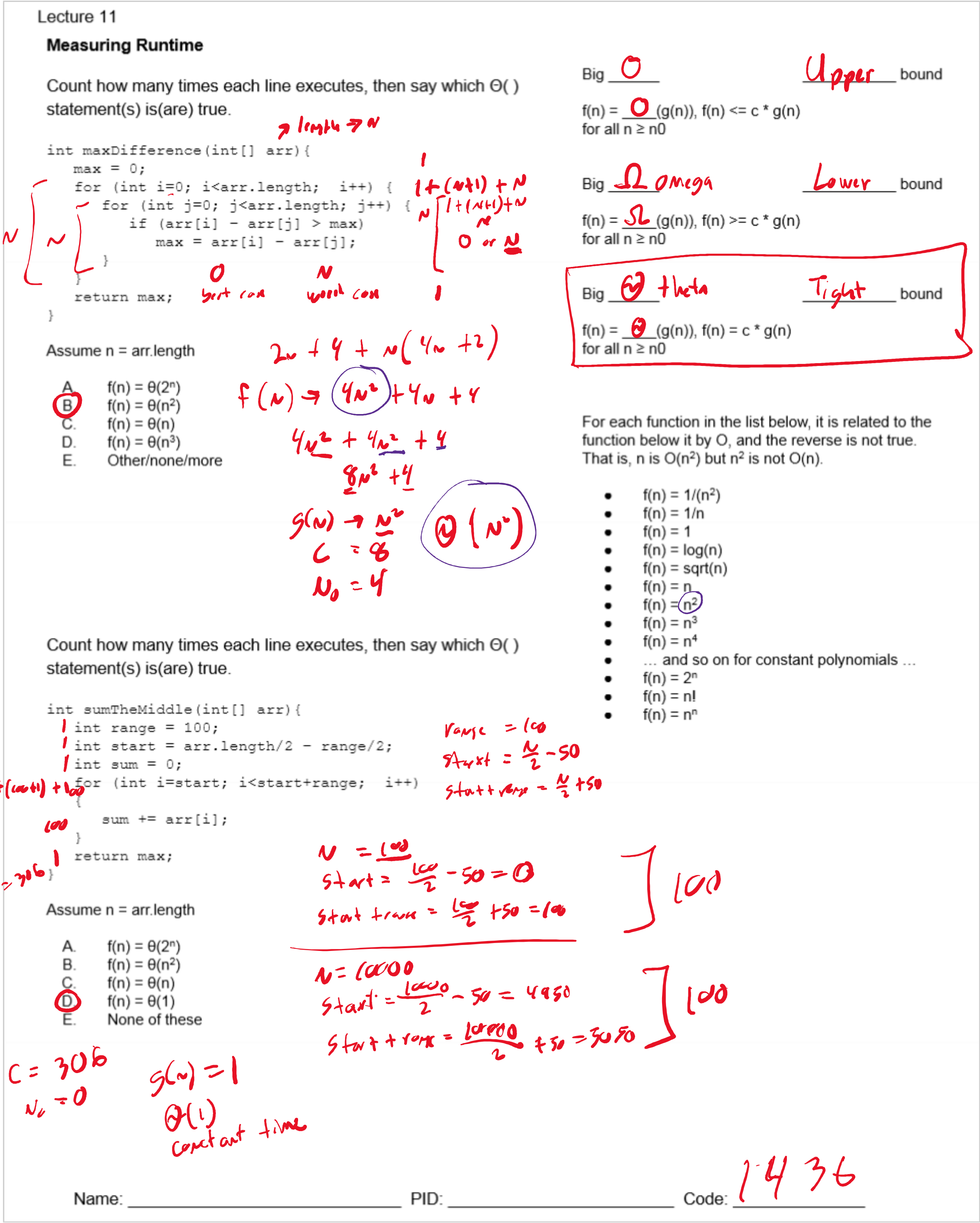
CSE12 - Lecture 11

Friday, April 26, 2024 10:00 AM



Lecture 11

Measuring Runtime

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

→ length → N

```
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

N

1

$1 + (N+1) + N$

N

$1 + (N+1) + N$

N

0 or N

0 best case

N worst case

1

Assume n = arr.length

A. f(n) = θ(2ⁿ)

[✓ B.] f(n) = θ(n²)

C. f(n) = θ(n)

D. f(n) = θ(n³)

E. Other/none/more

$2n + 4 + n(4n + 2)$

$f(n) \to \boxed{4n^2} + 4n + 4$

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

$8n^2 + 4$

$g(n) \to n^2$

$c = 8$

$n_0 = 4$

$\boxed{\Theta(n^2)}$

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(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

Count how many times each line executes, then say which Θ( ) 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 max;
}
```

1

1

1

$1 + (100+1) + 100$

100

1

$f(n) = 306$

range = 100

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

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

Assume n = arr.length

A. f(n) = θ(2ⁿ)

B. f(n) = θ(n²)

C. f(n) = θ(n)

[✓ D.] f(n) = θ(1)

E. None of these

$N = 100$

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

$C = 306$

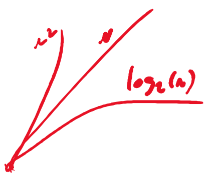
$N_0 = 0$

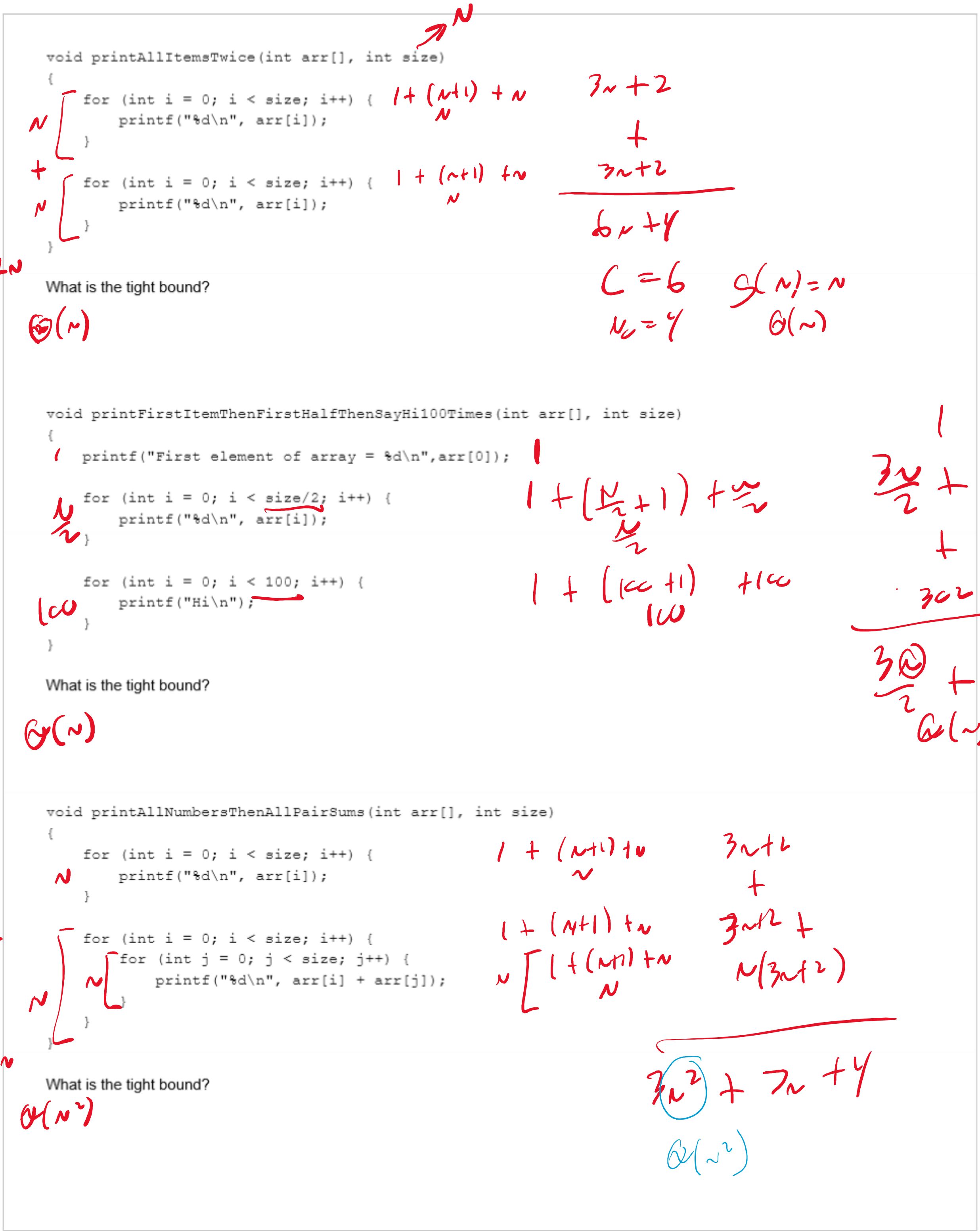
$g(n) = 1$

$\Theta(1)$

constant time

Name: ____________ PID: ____________ Code: 1436





```
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]);
    }
}
```

$\nearrow N$

N + N

$2N$

$1+(N+1)+N$ $\quad N$

$1+(n+1)+n$ $\quad N$

$3n+2$

$+$

$3n+2$

$6n+4$

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

$N_c=4$ $\quad \Theta(n)$

What is the tight bound?

$\Theta(n)$

```
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

1

$1+\left(\frac{N}{2}+1\right)+\frac{N}{2}$ $\quad \frac{N}{2}$

$1+(100+1)$ $\quad +100$ $\quad 100$

1 $+$

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

$+$

302

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

$\Theta(n)$

What is the tight bound?

$\Theta(N)$

```
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

$+$

N^2

N $\quad N$

N^2+N

$1+(N+1)+N$ $\quad n$

$3n+2$

$+$

$1+(N+1)+N$

$3n+2$ $+$

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

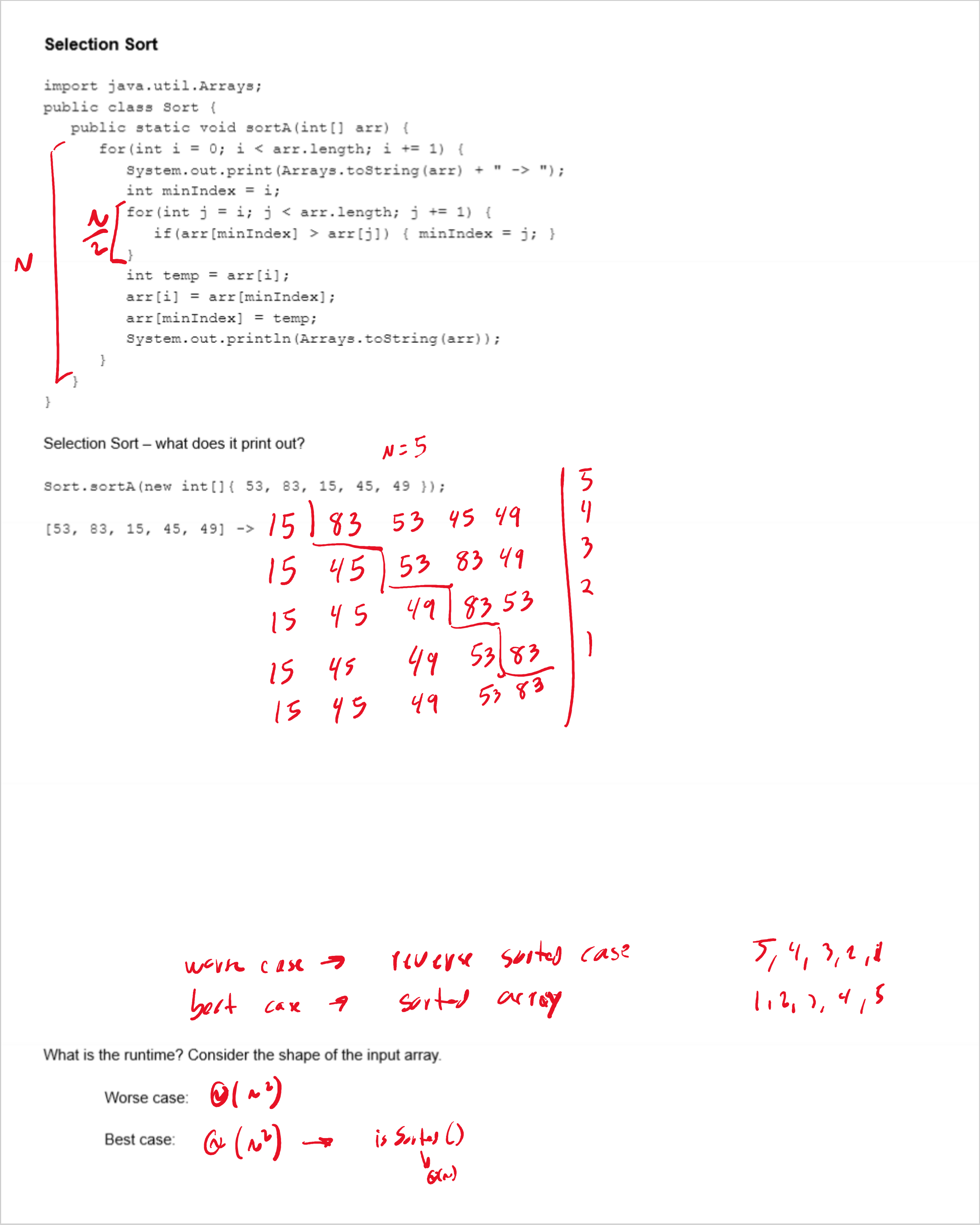
$N(3n+2)$

$3n^2+7n+4$

$\Theta(n^2)$

What is the tight bound?

$\Theta(N^2)$



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

$\frac{N}{2}$

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

15 45 53 83 49

15 45 49 83 53

15 45 49 53 83

15 45 49 53 83

5
4
3
2
1

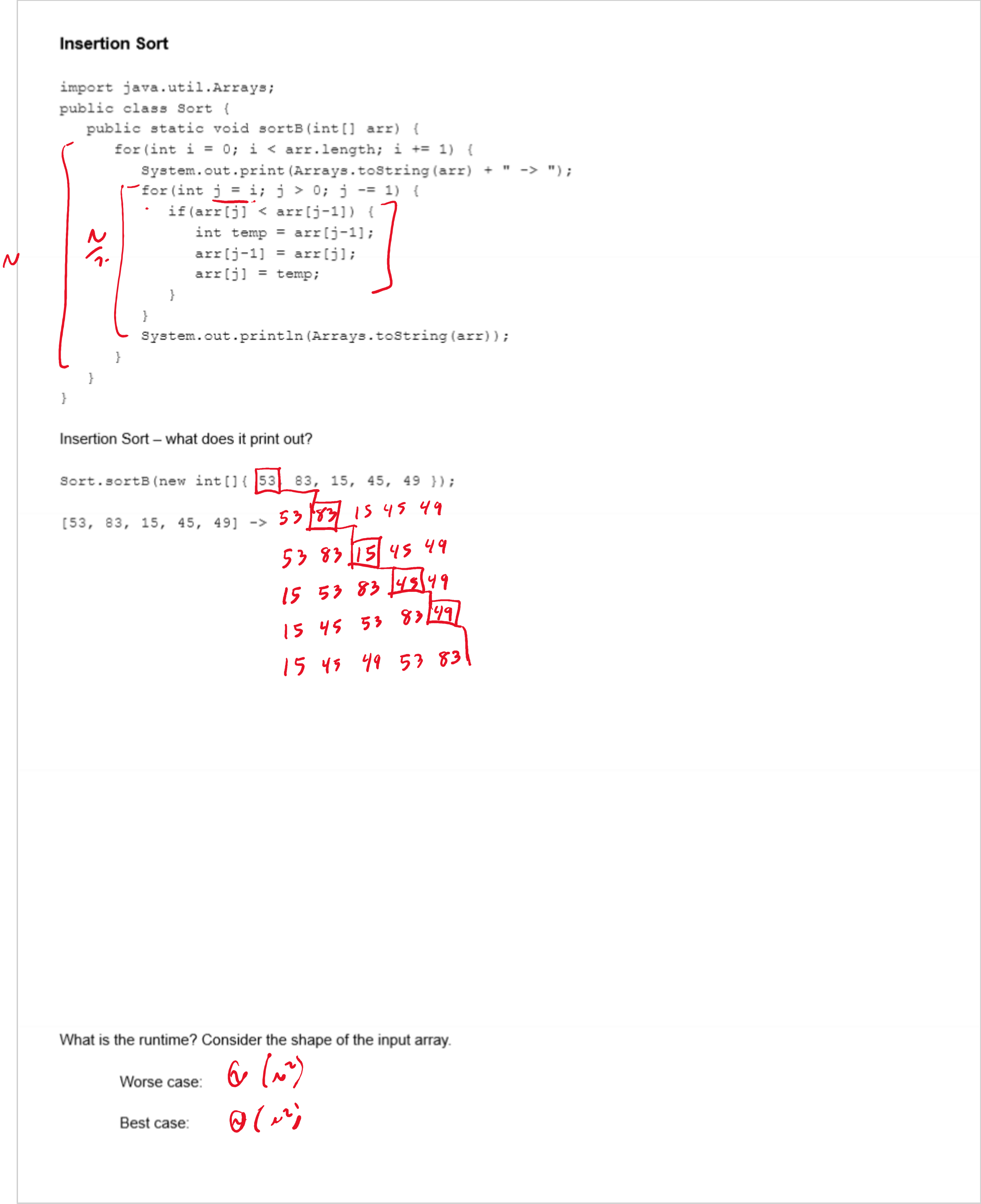
worse case → reverse sorted case   5, 4, 3, 2, 1

best case → sorted array   1, 2, 3, 4, 5

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

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

Best case: $\Theta(n^2)$ → isSorted()
↓
$\Theta(n)$



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));
      }
   }
}
```

-N

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

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

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

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