CSE12 - Lecture 11 - A00

Monday, October 17, 2022 8:00 AM

PA3 due tomorrow

Exam1 → Friday

Measuring Runtime

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

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

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){
   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;
}
```

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

```
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]);
    }
}
```

$1 + (N+1) + N$ → $3N+2$

$1 + (N+1) + N$ → $3N+2$

$6N+4$, $C=6$, $N_0=4$, $g(N) = N$

$\Theta(N)$

What is the tight bound?

```
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");
    }
}
```

$1 + (\frac{N}{2}+1) + \frac{N}{2}$ → $\frac{3N}{2}+2$

$1 + (100+1) + 100$ → 302

$\frac{3N}{2} + 304$

$\Theta(N)$

What is the tight 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]);
        }
    }
}
```

$1 + (N+1) + N$ → $3N+2$

$1 + (N+1) + N + N[1 + (N+1) + N]$ → $2N+2 + N(3N+2)$

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

$3N^2+7N+4$

$\Theta(N^2)$

$N^2 + N$ What is the tight bound?

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

N/2

$N^2/2$

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]
```

5
4
3
2
1
0

N/2

worst case: reverse sorted array
83 53 49 45 15

best case: sorted case
15 45 49 53 83

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

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

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

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
```

15 45 49 53 83

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

Worse case: $O(N^2)$

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