CSC 212: Data Structures and Abstractions

Spring 2018

University of Rhode Island

Weekly Problem Set #5

Due Thursday 3/1 before class. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. At the top of each sheet you hand in, please write your name, and ID. The only library you're allowed to use in your answers is `iostream`.

1. Mark each of the following as true or false.

Code	Big O	T/F	Big Omega	T/F	Big Theta	T/F
$3n^2 + 10n \log n$	$O(n \log n)$	F	$\Omega(n \log n)$	T	$\Theta(n \log n)$	F
$3n^2 + 10n \log n$	$O(n^2)$	T	$\Omega(n)$	T	$\Theta(\log n)$	F
$n \log n + n/2$	$O(2^n)$	T	$\Omega(n \log n)$	T	$\Theta(n \log n)$	T
$10\sqrt{n} + \log n$	$O(\log n)$	F	$\Omega(n)$	F	$\Theta(\log n)$	F
$\sqrt{n} + 10 \log n$	$O(\sqrt{n})$	T	$\Omega(1)$	T	$\Theta(\sqrt{n})$	T

2. Complete the following table.

Code	Big Theta
$\log n + 200n \log n$	$\Theta(n \log n)$
$2^n + n^2$	$\Theta(2^n)$
$\sqrt{n} + \log n$	$\Theta(\sqrt{n})$
$2n + 3n + 4n + 5n + 6n$	$\Theta(n)$
$\sqrt{n} + 10 \log n$	$\Theta(\sqrt{n})$
$200n * 10n + \log n$	$\Theta(n)$
Selection Sort	$\Theta(n^2)$
Insertion Sort	$\Theta(n^2)$
Bubble Sort	$\Theta(n^2)$

3. Given the array `A` with elements $[22, 15, 36, 44, 10, 3, 9, 13, 29, 25]$, illustrate the performance of the selection-sort algorithm from the lecture slides on `A`. To illustrate the performance, depict the status of the array after line 15 at every iteration.

   (i:0) $[3, 15, 36, 44, 10, 22, 9, 13, 29, 25]$

   (i:1) $[3, 9, 36, 44, 10, 22, 15, 13, 29, 25]$

   (i:2) $[3, 9, 10, 44, 36, 22, 15, 13, 29, 25]$

(i:3) $[3, 9, 10, 13, 36, 22, 15, 44, 29, 25]$

(i:4) $[3, 9, 10, 13, 15, 22, 36, 44, 29, 25]$

(i:5) $[3, 9, 10, 13, 15, 22, 36, 44, 29, 25]$

(i:6) $[3, 9, 10, 13, 15, 22, 25, 44, 29, 36]$

(i:7) $[3, 9, 10, 13, 15, 22, 25, 29, 44, 36]$

(i:8) $[3, 9, 10, 13, 15, 22, 25, 29, 36, 44]$

4. Given the array `A` with elements $[22, 15, 36, 44, 10, 3, 9, 13, 29, 25]$, illustrate the performance of the insertion-sort algorithm on `A`. Again, use the function provided in the lecture notes, and depict the status of the array after line 14 at every iteration. (Line 14 signifies the moment after the if statement terminates)

(i:0) $[22, 15, 36, 44, 10, 3, 9, 13, 29, 25]$

(i:1) $[15, 22, 36, 44, 10, 3, 9, 13, 29, 25]$

(i:2) $[15, 22, 36, 44, 10, 3, 9, 13, 29, 25]$

(i:3) $[15, 22, 36, 44, 10, 3, 9, 13, 29, 25]$

(i:4) $[10, 15, 22, 36, 44, 3, 9, 13, 29, 25]$

(i:5) $[3, 10, 15, 22, 36, 44, 9, 13, 29, 25]$

(i:6) $[3, 9, 10, 15, 22, 36, 44, 13, 29, 25]$

(i:7) $[3, 9, 10, 13, 15, 22, 36, 44, 29, 25]$

(i:8) $[3, 9, 10, 13, 15, 22, 29, 36, 44, 25]$

(i:9) $[3, 9, 10, 13, 15, 22, 25, 29, 36, 44]$

5. How many inversions are present in each of the following arrays?

A: [1, 5, 4, 3, 3, 7] -¿ 5

B: [5, 4, 3, 2, 1] -¿ 10

C: [1, 2, 4, 3, 5] -¿ 0

D: [5, 1, 3, 2, 4] -¿ 5

E: [6, 9, 1, 4, 10] -¿ 4

6. Write a recursive function that sums all of the elements of a given array, matching this signature:t int sum(int* arr, int n);

```
int sum(int* arr, int n) {
    if (n == 0) return 0;
    else {
        return arr[n-1] + (sum(arr, n-1));
    }
}
```

The following items are considered optional.

1. Rewrite recursive sum function to only sum odd numbers within the array.

```
int sum(int* arr, int n) {
    if (n == 0) return 0;
    else if (n % 2) {
        return arr[n-1] + (sum(arr, n-1));
    } else {
        return sum(arr, n-1);
    }
}
```

2. Write a recursive function that can find the minimum of a given array.

```
int arr_min(int* arr, int n) {
    if (n == 1) return arr[n-1];
    else return min(arr[n-1], arr_min(arr, n-1));
}
```

3. Briefly describe the principles behind Bin Sort. (Refer to OpenDSA)

   (a) Bin sort focuses on breaking up the problem by initially sorting objects into distinct "bins". Imagine working at a library, it might make sense to sort books by genre, then sort each genre by authors last name, this would be an example of a bin sort.

4. What is the Big O, Big Omega, and Big Theta of Radix Sort? (Refer to OpenDSA)

   (a) $O(n \log n)$, $\Omega(n)$, $\Theta(n \log n)$. As explained on openDSA, this algorithm is $n \log n$, however it typically runs in near linear time, especially when larger bases are used. The exact formula is $O(wn)$ where w is the base.