9

Name: Hanna Hahn
ID: 6481334

Quiz 3 — Data Struct. & More (T. III/22–23)

Directions:

- This exam is paper-based. Answer all the questions in the space provided.
- No consultation with other people, notes, books, nor the Internet is permitted. Do **not** use an IDE or run Java code.
- This quiz is worth a total of 35 points, but we'll grade out of 30. Anything above 30 is extra credit. You have 65 minutes. Good luck!

Summation Formulas:

- $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$
- $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$
- $1 + 2 + 2^2 + 2^3 + \cdots + 2^n = 2^{n+1} - 1$

Big-O: $f(n)$ is $O(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ for some constant $c \geqslant 0$. Equivalently, $f(n)$ is $O(g(n))$ if and only if there's a real constant $c > 0$ and an integer constant $n_0 \geqslant 1$ such that $f(n) \leqslant c \cdot g(n)$ for all $n \geqslant n_0$.

Theta Θ: $f(n)$ is $\Theta(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ for some constant $c > 0$. Equivalently, $f(n)$ is $\Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.

Problem 1: Growth Rate (3 points)

Order the following functions from small to large in terms of their growth rate.

∅

2^n $\quad n \log n \quad 0.001 \cdot n^3 \quad 28{,}000{,}000{,}000 \quad n^{49} \quad 999n \quad 9 \log n$

Answer in the blanks below.

28,000,000,000 < 0.001×n³ < 999n < n^{49} < 9logn < nlogn < 2^n

Problem 2: Basic Facts & Techniques (8 points)

2

(i) **(3 points)** For each of the following algorithms from lecture, indicate its best-case running time and worst-case running time for input of size n in terms of the *tightest* big-O.

2

	Best Case	Worst Case
Insertion Sort	O(n) ✓	O(n) ✗
Quicksort	O(nlogn) ✗	O(n²) ✓
One `link` operation in the disjoint set data structure that uses lazy linking (point one root to another root) with height control (small into large)	O(n) ✗	O(logn) ✓

(ii) **(5 points)** Suppose $f(n)$ is $\Theta(n \log n)$ and $g(n)$ is $\Theta(n^3)$. Give a mathematical proof using either the limit definition or the for-all-there-exist definition that $h(n) = n^3 \cdot f(n) + 9n^2 \cdot g(n)$ is $\Theta(n^5)$.

$h(n) = n^3 \cdot (n\log n) + 9n^2 \cdot n^3$

∅

$= n^3 (n \log n) + 9n^5$

$= n^5 \log n + 9n^5$

5

Problem 3: Running Time Analysis (12 points)

- Carefully analyze each of the following snippets and give the tightest possible big-O for its running time as a function of n.
- Also, briefly justify your answer.
- Partial credit will be given to correct answers that aren't tight but aren't outraged.

(i)

```
int puzzle0(int[] data) {
    int n = data.length, answer = 0;
    for (int i=0;i<4*n;i++) {
        for (int j=0;j<5;j++) {
            answer += data[(i/4 + j)%n];
        }
    }
    return answer;
}
```

+3 —?

i=0 to i < 4*n with i increasing with 1 it will run until i is less than 4*n (n = data.length), it is constant therefore, O(n)

(ii)

```
void puzzle1(int[] data) {
    int n = data.length;
    for (int i=0;i<n;i++) {
        for (int j=i;j>=0;j=j/2) {
            int k = j;
            while (k > 0) {
                data[k] = data[i];
                k--;
            }
        }
    }
}
```

∅

The outer loop and the inner loop will run iteration with a constant amount. However, the middle loop j is divided by 2 each time. Therefore, it is O(n) (only because it is divided by i)

Further Directions: The snippets below are recursive. Write a recurrence and explain your recurrence briefly.

(iii)

```
int puzzle2(int[] data) {
    int n = data.length;
    if (n == 1)  return data[0];
    else if (n > 1) {
        int[] odd = new int[n/2];
        int[] even = new int[n - n/2];
        int u = 0, v = 0;
        for (int i=0;i<n;i++) {
            if (i%2==0) even[u++] = data[i];
            else odd[v++] = data[i];
        }
        return puzzle2(odd) + puzzle2(even);
    }
    return 0;
}
```

2

$2T(\frac{n}{2}) + O(n)$

since odd and even are created as a new int[]. However it takes n and divides it, also it runs trough a for-loop.

explain?

(iv)

```
int puzzle3(int b, int w, int a) {
    if (w==0) return a;
    if (w==1) return a*b;
    if (w==2) return a*b*b;
    int p = w/3;
    int x = puzzle3(b, p, 2);
    int y = puzzle3(x, 2, a) 4
    return puzzle3(b, w - 2*p, y);
}
```

∅

$2T(\frac{n}{2}) + O(\log n)$

There are 3 new int created, which runs trough the puzzle3 therefore the 2T. Since it involves the division p = w/3 it is O(logn)

1

Problem 4: Correctness (6 points)

The function puzzle3 above does compute something interesting. Prove using induction that for $b, a, w \in \mathbb{Z}$ with $w \geqslant 0$, puzzle3(b, w, a) returns $a \cdot b^w$. (*Hint:* strong induction. also, remember that in Java, the expression n/3 is numerically equal to $\lfloor n/3 \rfloor$.)

P(n), For all b, w, a with $w \geq 0$ puzzle(b, w, a) return $a \cdot b^w$

base case P(0): w=0, it will return a since if(w==0) return a.

inductive step: Assume P(k) is true, we will prove P(w+1)

To find puzzle3(b, w+1, a) There are 2 cases to consider

case 1: w+1 is even can be written as 2m where m is an integer

puzzle3(b, w+1, a) = puzzle3(b, 2m, 2) which should return (

case 2: w+1 is odd can be written as 2m+1 where m is an integer

puzzle3(b, w+1, a) = 2m+1 + puzzle3(b, [w/3], 2) - 1

it should return

j-j

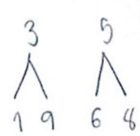
1

Problem 5: Disjoint Sets (6 points)

(i) **(3 points)** Draw a visualization of the disjoint-set structure as we did in class for the following p[] array.

i	0	1	2	3	4	5	6	7	8	9
p[i]	1	3	1	3	6	5	5	2	5	3

+1



(ii) **(3 points)** Suppose `link(i, j)` is the method as discussed in class that implements lazy linking with height (depth) control (i.e., point small into large). It does *not* use path compression. Draw a visualization after `link(0,4)` is called on the disjoint-sets data structure with the `p[]` array above.

∅

(0, 4)

0	1	2	3	4
1	3	1	3	6

0	1	2	3	4
1	3	1	3	2

4

0	2	3	4
1	1	1 3	4 2