Quiz 3 — Data Struct. & More (T. III/21–22)

Name:	Chanat Keratiyutwong
ID:	638981

Directions:

- This quiz is paper-based. Answer all the questions in this booklet.
- 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 60 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)

3

Order the following functions from small to large when n is very large.

~~$n \log n$~~ $O(n \log n)$ ~~2^n~~ $O(b^n)$ ~~$1000n$~~ $O(n)$ ~~n^3~~ $O(n^3)$ ~~$0.25n^2$~~ $O(n^2)$ ~~$9 \log n$~~ $O(\log n)$ ~~$28{,}000{,}000$~~ $O(1)$

Answer in the blanks below.

28,000,000 < $9 \log n$ < $1000n$ < $n \log n$ < $0.25n^2$ < n^3 < 2^n

Problem 2: Basic Facts & Techniques (8 points)

4

(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	$O(n)$ ✓	$O(n^2)$ ✓
Merge Sort	$O(n \log n)$	$O(n^2)$ ✗ $O(n \log n)$
One `isConnected` operation in the disjoint set data structure that uses lazy linking with height control	$O(1)$ ✓	$O(n)$ ✗ n is number of sets. $O(\log n)$

(ii) **(5 points)** Suppose $f(n)$ is $\Theta(n^3)$ and $g(n)$ is $\Theta(n^4)$. Give a mathematical proof using either the limit definition or the for-all-there-exist definition that $h(n) = n^2 \cdot f(n) + n \cdot g(n)$ is $\Theta(n^5)$.

2

$$\lim_{n\to\infty} \frac{h(n)}{n^5} = c \quad (\neq 0)$$

$\lim = c \neq 0$

$$\boxed{\begin{matrix} n^3 + \ldots \in \Theta(n^3) \\ n^4 + \ldots \in \Theta(n^4) \end{matrix}}$$ most dominant terms, (highest powers)

$$= \lim_{n\to\infty} \frac{n^2 \cdot \Theta(n^3) + n \cdot \Theta(n^4)}{n^5}$$

How? How?

$$= \lim_{n\to\infty} \frac{n^2 \cdot n^3 + n \cdot n^4}{n^5} = \lim_{n\to\infty} \frac{2n^5}{n^5} = 2 \neq 0$$

Therefore, $h(n) \in \Theta(n^5)$.

11

Problem 3: Running Time Analysis (12 points)

- **Carefully analyze the following snippets and give the tightest possible big-O for its running time as a function of n.**
- **Optionally, justify your answer very briefly—no more than three short sentences.**
- **Partial credit will be given to correct answers that aren't tight but aren't outrageous.**

(i)

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

3

$O(1) + O(1)$

$O(n^2) \cdot O(n) \cdot O(1)$

$O(1)$

$\boxed{T(n) = O(n^3)}$

(ii)

```
void puzzle1(int[] data) {
    int n = data.length;
    for (int i=0;i<n;i++) {
        int k = n-1;
        while (k > 0) {
            data[k] = data[i];
            k = k / 2;
        }
    }
}
```

3

n times

$O(1)$

$O(n)$

$O(1)$

$O(\log n)$

$O(1)$

$\log_2(n)$

$\boxed{T(n) = O(n \log n)}$

Further Directions: The snippets below are recursive. Write a recurrence. Show how you arrive at the recurrence. You don't need to solve for the final big-O.

(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;
}
```

$\boxed{T(n) = 2T\left(\frac{n}{2}\right) + O(n)}$

3

$O(1)$

$O(1)$

$O(n)$ (allocating array)

$O(n)$

$O(1)$

$O(n)$

$O(1)$

$O(1)$

$2T(n/2)$

call recursive twice, with odd and even arrays being half the size of original input size.

$O(1)$

(iv)

```
long puzzle3(long b, long n, long a) {
    if      (n==0) return a;
    else if (n==1) return a*b;
    else {
        long p = n/2;
        long x = puzzle3(b, p, 1);
        return puzzle3(b, n - p, a*x);
    }
}
```

2

$O(1)$

$O(1)$

$O(1)$

$T\left(\frac{n}{2}\right)$

~~$O(1)$~~

$T\left(\frac{n}{2}\right)$

input n is halved.

$p = \frac{n}{2}$

$\boxed{T(n) = 2T\left(\frac{n}{2}\right) + O(1)}$

$n - p = n - \frac{n}{2}$

$= \frac{n}{2}$

$\lfloor n/2 \rfloor = \frac{n-1}{2}$

Problem 4: Correctness (8 points)

8

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/2 is numerically equal to $\lfloor n/2 \rfloor$.)

Predicate: $P(w) \equiv$ puzzle3$(b, w, a) \rightarrow a \cdot b^w$

Base cases: $P(0) \equiv$ puzzle3$(b, 0, a) \rightarrow a \cdot b^0 = a$ } $P(0)$ and $P(1)$ are true.

$P(1) \equiv$ puzzle3$(b, 1, a) \rightarrow a \cdot b^1 = ab$

Inductive Hypothesis: Assume that puzzle3$(b, k, a) \rightarrow a \cdot b^k$ for $k = 0, 1, 2 \ldots w$.

Show that this code works for $k+1$. $(P(k) \rightarrow P(k+1))$.

long puzzle3$(b, k+1, a) \rightarrow a \cdot$ puzzle3$(b, \frac{k+1}{2}, 1)$ ← IH, as $\frac{k+1}{2} < w$.

$\rightarrow a \cdot b^{\lfloor \frac{k+1}{2} \rfloor} \cdot b^{\lfloor \frac{k+1}{2} \rfloor} \cdot 1$ $\lfloor \frac{k+1}{2} \rfloor$ $\lceil \frac{k+1}{2} \rceil$

$\lfloor \frac{n}{2} \rfloor = \frac{n-1}{2}$

$\rightarrow a \cdot b^{(\frac{k+1-1}{2}) + (\frac{k+1-1}{2})}$ not exactly but close enough. :)

$\rightarrow a \cdot b^k$

Therefore, by mathematical induction, puzzle3$(b, w, a) \rightarrow a \cdot b^w$.

Problem 5: Disjoint Sets (4 points)

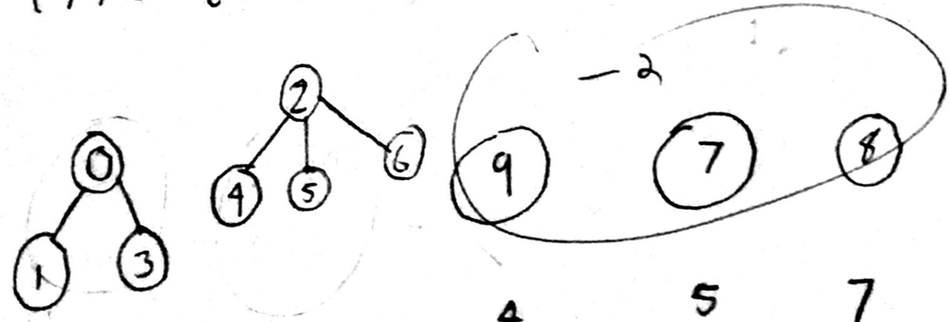
2

Draw a visualization of the disjoint-set structure as we did in class for the following p[] array. The book uses the letter p, but the p array was called backRef in class.

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

1 2 4 5 7

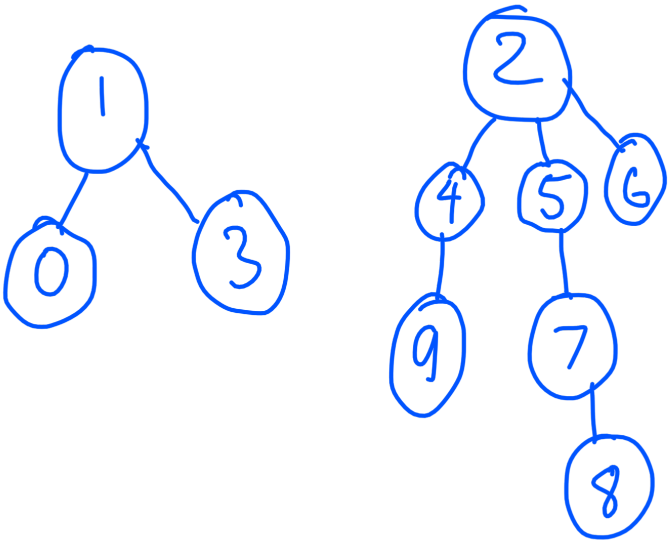
{0, 1, 3} {2, 4, 5, 6} {9} {7} {8}



4 5 7

p[i]: 1 2

5)



(ii) (5 points) Suppose $f(n)$ is $\Theta(n^3)$ and $g(n)$ is $\Theta(n^4)$. Give a mathematical proof using either the limit definition or the for-all-there-exist definition that $h(n) = n^2 \cdot f(n) + n \cdot g(n)$ is $\Theta(n^5)$.

Property

if $f \in \Theta(j(n))$, $g \in \Theta(k(n))$,

then $f \cdot g \in \Theta(j(n) \cdot k(n))$.

2) ii) $f \in \Theta(n^3)$, $g \in \Theta(n^4)$

$\lim_{n\to\infty} \frac{f(n)}{n^3} = c_1$, $\lim_{n\to\infty} \frac{g(n)}{n^4} = c_2$

terms $< n^4$

so $f(n) = c_1 n^3 + \ldots$, $g(n) = c_2 n^4 + \ldots$

terms $< n^3$

$h(n) \in \Theta(n^5)$: $\lim_{n\to\infty} \frac{h(n)}{n^5}$

$= \lim_{n\to\infty} \frac{n^2 \cdot f(n) + n \cdot g(n)}{n^5}$

$= \lim_{n\to\infty} \frac{n^2}{n^5} \cdot f(n) + \lim_{n\to\infty} \frac{n}{n^5} \cdot g(n)$

$= \lim_{n\to\infty} \frac{1}{n^3} \cdot (c_1 n^3 + \ldots) + \lim_{n\to\infty} \frac{1}{n^4} \cdot (c_2 n^4 + \ldots)$

$= \lim_{n\to\infty} c_1 + c_2 + \frac{c_3}{n} + \frac{c_4}{n^2} + \frac{c_5}{n^3}$

$= c_1 + c_2$

So $h(n) \in \Theta(n^5)$.