SSN College of Engineering
B.E. (Computer Science and Engineering) Semester 4
Unit Test: 1 (23 January 2017)

Time: 8.00–9.30 **CS6402 Design and Analysis of Algorithms** Max marks: 50

Part A $10 \times 2 = 20$

Answer *all* questions.

1. What is an algorithm? (CO1)

   An algorithm is a *composition (sequence) of unambiguous instructions* for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

2. State the characteristic of *basic operations*. Which of the following are *not* basic operations? add, multiply, power, logical or. (CO1)

   A basic operation is executed in *constant* amount of time. It does not depend on the *input size*. `x power n` is not a basic operation. It depends on n.

3. Find the order of growth of the function $10n^2 + 4n + 2$ with suitable values for c and n_0. (CO3)

   Method 1:

$$\begin{aligned} 10n^2 + 4n + 2 &\leq 11n^2 \\ &= 10n^2 + n^2 \\ 4n + 2 &\leq n^2 \\ \frac{4}{n} + \frac{2}{n^2} &\leq 1 \end{aligned}$$

   holds for $n_0 = 5$. Thus, one of the possible values are $c = 11$ and $n_0 = 5$.

$$\begin{aligned} 10n^2 + 4n + 2 &\leq 10n^2 + 4n^2 + 2n^2 \\ &= 16n^2 \end{aligned}$$

   holds for $n_0 = 1$. Thus, one of the possible values are $c = 16$ and $n_0 = 1$.

   Method 2:

$$\begin{aligned} \lim_{n\to\infty} \frac{10n^2 + 4n + 2}{n^2} &= \lim_{n\to\infty} 10 + \frac{4}{n} + \frac{2}{n^2} = 10 \\ 10n^2 + 4n + 2 &= O(n^2) \end{aligned}$$

4. If $f(x) = \frac{x^3}{2}$ and $g(x) = 37x^2 + 120x + 17$, show that $g = O(f)$, but $f \neq O(g)$. (CO3)

$$\begin{aligned} \lim_{n\to\infty} \frac{g(x)}{f(x)} &= \frac{37x^2 + 120x + 17}{x^3} = \frac{37}{x} + \frac{120}{x^2} + \frac{17}{x^3} = 0 \\ g(x) &= O(f(x)) \end{aligned}$$

$$\lim_{n\to\infty} \frac{f(x)}{g(x)} = \frac{x^3}{37x^2+120x+17} = \frac{3x^2}{74x+120} = \frac{6x}{74} = \infty$$
$$f(x) = \Omega(g(x))$$

5. Find the order of growth of the sum $\sum_1^n (i^2+1)^2$ (CO3)

$$\sum_{i=1}^{n} i^4 = \frac{1}{30} n(n+1)(2n+1)(3n^2+3n-1) = O(n^5)$$

6. How many times the body of the inner loop is executed? What is the order of growth of the algorithm? (CO3)

```
for $i \leftarrow 1$ to $m$
    for $j \leftarrow 1$ to $n$
        $c[i,j] \leftarrow a[i,j] + b[i,j]$
    end
end
```

mn times, Time complexity = $O(mn)$

7. Find the time complexity of sum(a) where a is a list. (CO3)

```
Algorithm: sum $a[0:n-1]$
if a = [] then return 0
return a[0] + sum a[1:n-1]
```

$$T(n) = T(n-1) + 1$$
$$T(n) = n$$

8. Solve the recurrence relation: (CO3)

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1)+1 & \text{if } n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= 1 + T(n-1) \\ &= 1 + 1 + T(n-2) = 2 + T(n-2) \\ &= 2 + 1 + T(n-3) = 3 + T(n-3) \\ &\ldots \\ &= n - 1 + T(1) \\ &= n - 1 + 1 \\ T(n) &= n \end{aligned}$$

9. Prove that any comparison sort algorithm requires $\Omega(n \log n)$ comparisons in the worst case. (CO1)

10. For each of the following functions, indicate how much the functions value will change if its argument is increased fourfold. (CO3)

a. $\log_2 n$ b. $\sqrt{n}$ c. n d. n^2 e. n^3 f. 2^n

n	$\log_2 n$	$\sqrt{n}$	n	n^2	n^3	2^n
$4n$	$\frac{\log_2 4n}{\log_2 n}$	$\frac{\sqrt{4n}}{\sqrt{n}}$	$\frac{4n}{n}$	$\frac{(4n)^2}{n^2}$	$\frac{(4n)^3}{n^3}$	$\frac{2^{4n}}{2^n}$
$4n$	$\frac{\log_2 4n}{\log_2 n}$	2	4	16	64	2^{3n}

Part B $6 \times 5 = 30$

Answer any *five* questions.

11. Consider two algorithms A and B for solving the same problem running on two machines 1 and 2. Machine 1 executes 10^9 (1 billion) instructions per second, and machine 2 executes 10^7 (10 million) instructions per second. Algorithm A requires $2n^2$ instructions and runs on machine 1; algorithm B requires $50n \log_{10} n$ instructions and runs on machine 2. (CO1)

   (a) Calculate the running time of the two algorithms for inputs of sizes 100, 1000, 10000. (3)

   (b) Which is better — algorithm A on machine 1, or algorithm B on machine 2? Why? (3)

n	Algorithm A (Slow algorithm $2n^2$ on fast machine 10^9 ips)	Algorithm B (Fast algorithm $50n \log n$ on slow machine 10^7 ips)
10^3	$\frac{2 \times (10^3)^2}{10^9} = 2 \times 10^{-3}$	$\frac{50 \times 10^3 \log 10^3}{10^7} = 15 \times 10^{-3}$
10^4	$\frac{2 \times (10^4)^2}{10^9} = 0.2$	$\frac{50 \times 10^4 \log 10^4}{10^7} = 0.2$
10^5	$\frac{2 \times (10^5)^2}{10^9} = 20$	$\frac{50 \times 10^5 \log 10^5}{10^7} = 0.25$

Algorithm B on machine 2 is better than algorithm A on machine 1.

12. (a) Design a brute-force algorithm for finding the two closest points in a set of n points (the closest-pair problem). (CO2, 3)

```
Algorithm: ClosestPair $P$
Input: A list $P$ of $n$ points $P_1(x_1, y_1), \ldots, P_n(x_n, y_n)$
Output: Indices $k1, k2$ of the closest pair of points $\{P_{k1}, P_{k2}\}$
dmin ← $\infty$
for $i \leftarrow 1$ to $n-1$ do                          // $\sum_{i=1}^{n-1}$
    for $j = i+1$ to $n$ do                               // $\sum_{j=i+1}^{n}$
        $d \leftarrow \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$       // 2
        if d < dmin then
            dmin, k1, k2 ← d, i, j
        end
    end
end
return k1, k2
```

(b) Analyze the running time of the algorithm. (CO3, 3)

$$\begin{aligned}
\text{Running time} =& \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2 \\
=& 2\sum_{i=1}^{n-1}(n-(i+1)+1) \\
=& 2\sum_{i=1}^{n-1}(n-i) \\
=& 2\sum_{i=1}^{n-1} n - 2\sum_{i=1}^{n-1} i \\
=& 2n(n-1) - 2(n-1)n/2 \\
=& 2n(n-1) - (n-1)n \\
=& (2n-n)(n-1) \\
=& n(n-1) \\
=& n^2 - n \\
=& O(n^2)
\end{aligned}$$

13. Given two $n \times n$ matrices A and B, write an algorithm for computing their product $C = AB$, and find its time efficiency. (CO2, CO3, 6)

```
Algorithm: MatMult $a, b$
Input: $a$ and $b$ are $n \times n$ matrices
Output: $c = a \times b$
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $n$ do
        $c[i, j] = 0$
        for $k \leftarrow 1$ to $n$ do
            $c[i, j] \leftarrow c[i, j] + a[i, k] * b[k, j]$
        end
    end
end
return c
```

Body of the innermost loop is executed

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1 = n^3$$

times. Time complexity = $O(n^3)$

14. (a) Design an algorithm to merge two sorted lists. Analyze its running time. (CO2, CO3, 3)

```
Algorithm: Merge a, b
Input: a[0:n1-1] and b[0:n2-1] are sorted lists of items.
Output: A sorted list of all items in a and b.
if a = [] then return b
if b = [] then return a
if a[0] < b[0] then
    return a[0]: Merge a[1:n1-1], b
else
    return b[0]: Merge a, b[1:n2-1]
end
```

Passes over all the items of a and b once. Let $n = \text{len}(a) + \text{len}(b)$. Running time = $O(n)$

(b) Design an algorithm to sort a list, dividing it into two almost equal sublists, sorting each sublist recursively, and then merging the two sorted sublists. Analyze the running time of this sort algorithm. (CO2, CO3, 3)

```
Algorithm: MergeSort a
Input: a[0:n-1] is a list of comparable items.
Output: A sorted list of items in $a$.
if #a = 0 or #a = 1 then return a
m ← $\lfloor \#a/2 \rfloor$
return Merge (MergeSort a[0:m]), (MergeSort a[m+1:n-1])
```

15. Derive a recurrence relation for Fibonacci series algorithm; also, carry out the time complexity. (CO3,6)

```
Algorithm: Fib n
Input: $n$ is a non-negative integer.
Output: $F(n)$
if n = 0 then return 0
if n = 1 then return 1
return Fib(n-1) + Fib(n-2)
```

$$
\begin{aligned}
&T(n) = T(n-1) + T(n-2) + 1 \\
&T(n) - T(n-1) - T(n-2) - 1 = 0 \\
&(T(n)+1) - (T(n-1)+1) - (T(n-2)+1) = 0 \\
&F(n) - F(n-1 - F(n-2) = 0 \text{ where } F(n) = T(n) + 1
\end{aligned}
$$

This is a second-order homogeneous linear recurrence with constant coefficients. Its characteristic equation is

$$r^2 - r - 1 = 0$$

with roots

$$r_{1,2} = \frac{1 \pm \sqrt{5}}{2}$$

The solution is

$$F(n) = \alpha \left(\frac{1+\sqrt{5}}{2}\right)^n + \beta \left(\frac{1-\sqrt{5}}{2}\right)^n$$

Solve for α, β, with $F(0) = 0$ and $F(1) = 1$. We get

$$\alpha = \frac{1}{\sqrt{5}}, \quad \beta = -\frac{1}{\sqrt{5}}$$

Thus

$$F(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

and

$$T(n) = F(n) - 1 = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n) - 1 = O(\phi^n)$$

16. Analyze the best-case, the worst-case, and the average-case running times of the linear search algorithm for an array a of size n? (CO3, 2+2+2)

    Algorithm: LinearSearch $a[0 : n-1], x$

    Input: Array $a[0 : n-1]$ of n numbers, and a target x to search for.
    Output: i such that $a[i] = x$ if x is in the array; $i = n$, otherwise.

```
$i \leftarrow 0$ until  $i = n$ or $a[i] = x$ do          // 1 to $n+1$ times
    $i \leftarrow i + 1$
end
return i
```

Worst case occurs when the target is not found.

$$T(n) = n + 1 = O(n)$$

Best case occurs when the target is the 0th item.

$$T(n) = 1 = O(1)$$

Average case: The target is equally likely to be in any of the n positions: 0, 1, . . ., $n-1$.

$$P(0) = P(1) = \ldots = P(n-1) = \frac{1}{n}$$

On an average, the number of iterations

$$\begin{aligned}
&= P(0) \times 0 + P(1) \times 1 + P(2) \times 2 + \ldots + P(n-1) \times (n-1) \\
&= \frac{1}{n} \times (1 + 2 + \ldots + (n-1)) \\
&= \frac{1}{n} \times \frac{(n-1)n}{2} \\
&= \frac{n-1}{2} \\
&= O(n)
\end{aligned}$$

17. (a) Solve the recurrence relation (CO3, 3)

$$T(n) = \begin{cases} 2T\left(\dfrac{n}{2}\right) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

(b) What is the order of growth of $T(n)$? (CO3, 3)

$$T(n) = \begin{cases} 2T\left(\dfrac{n}{2}\right) + n & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

Assume that n is a power of 2, say, $n = 2^h$, and hence, $h = \log_2 n$.

$$\begin{aligned}
T(n) &= n + 2T\left(\frac{n}{2}\right) \\
&= n + 2\left[\frac{n}{2} + 2T\left(\frac{n}{2^2}\right)\right] = n + n + 2^2T\left(\frac{n}{2^2}\right) = 2n + 2^2T\left(\frac{n}{2^2}\right) \\
&= 2n + 2^2\left[\frac{n}{2^2} + 2T\left(\frac{n}{2^3}\right)\right] = 2n + n + 2^3T\left(\frac{n}{2^3}\right) = 3n + 2^3T\left(\frac{n}{2^3}\right) \\
&\dots \\
&= nh + 2^hT\left(\frac{n}{2^h}\right) \\
&= nh + 2^hT\left(1\right) \\
&= n\log_2 n + n \\
&= O(n\log n)
\end{aligned}$$

18. Design an algorithm to find all the common elements in two sorted lists of numbers. For example, for the lists 2, 5, 5, 5 and 2, 2, 3, 5, 5, 7, the output should be 2, 5, 5. What is the maximum number of comparisons your algorithm makes if the lengths of the two given lists are m and n, respectively? (CO2, CO3, 3+3)

Algorithm: Common a, b

Input: a[0:n1-1] and b[0:n2-1] are sorted lists of items.
Output: c is a list of common items in a and b.

```
if a = [] or b = [] then  return []
if a[0] = b[0] then
    return a[0] + Common a[1:], b[1:]
else if a[0] < b[0] then
    return Common a[1:], b
else
    return Common a, b[1:]
end
```

Passes over all the items of a and b once. Let $n = \text{len}(a) + \text{len}(b)$. Running time = (n)
