**COMP 2711 Discrete Math Tools for Computer Science**

**2022 Fall Semester - Homework 5**

**Question 1:** Analyze the worst-case time complexity of the following algorithm for finding the first term of a sequence of integers equal to some previous term.

> **procedure** find duplicate($a_1, a_2, \ldots, a_n$ : integers)
> *location* := 0 {no match found yet}
> $i := 2$
> **while** $i \leq n$ and $location = 0$
>     $j := 1$
>     **while** $j < i$ and $location = 0$
>         **if** $a_i = a_j$ **then** $location := i$
>         **else** $j := j + 1$
>     $i := i + 1$
> **return** *location*
> {location is the subscript of the first value that repeats a previous value in the sequence and is 0 if there is no such value}

**Answer:** The worst case is that in which we do not find any term equal to some previous term. In that case, we need to go through all the terms $a_2$ through $a_n$ , and for each of those, we need to go through all the terms occurring prior to that term. Thus the inner loop of our algorithm is executed once for $i = 2$ (namely for $j = 1$), twice for $i = 3$ (namely for $j = 1$ and $j = 2$), three times for $i = 4$, and so on, up to $n - 1$ times for $i = n$. Thus the number of comparisons that need to be made in the inner loop is $1 + 2 + 3 + \cdots + (n - 1)$. This sum is $(n - 1)(n - 1 + 1)/2$, which is clearly $O(n^2)$ but no better. Bookkeeping details do not increase this estimate.

**Question 2:** A computer system considers a string of digits to be a valid code word if and only if it contains an even number of 0's. For instance, 1203045 is valid, whereas 780900 is not. Let $V_n$ be the number of valid code words of length $n$.

It is clear that $V_1 = 9$ because "1", "2", ..., "9" are valid code words of length 1, while "0" is an invalid code word of length 1.

For $n > 1$, find a recurrence for $V_n$ by determining how it is related to $V_{n-1}$. Solve the recurrence to get a closed-form formula for $V_n$.

**Answer:** A valid length $n$ code can be obtained

- From a valid length $n-1$ code by appending "1", "2", ..., or "9"to the end, or
- From an invalid length $n-1$ code by appending "0" to the end.

The total number of length $n-1$ codes is $10^{n-1}$. Hence the number of invalid length $n-1$ codes is $(10^{n-1} - V_{n-1})$. So, we have

$$V_n = 9V_{n-1} + (10^{n-1} - V_{n-1}) = 8V_{n-1} + 10^{n-1}.$$

Iterating the recurrence, we get

$$\begin{aligned}
V_n &= 8V_{n-1} + 10^{n-1} \\
&= 8(8V_{n-2} + 10^{n-2}) + 10^{n-1} \\
&= 8^2 V_{n-2} + 8 \times 10^{n-2} + 10^{n-1} \\
&= \ldots \\
&= 8^{n-1} V_1 + 8^{n-2} \times 10 \ldots + 8 \times 10^{n-2} + 10^{n-1} \\
&= 9 \times 8^{n-1} + 8^{n-2} \times 10 \times (1 + \frac{10}{8} + \ldots + (\frac{10}{8})^{n-2} + (\frac{10}{8})^{n-2}) \\
&= 9 \times 8^{n-1} + 8^{n-2} \times 10 \times \frac{(\frac{10}{8})^{n-1} - 1}{\frac{10}{8} - 1} \\
&= 9 \times 8^{n-1} + \frac{10^n - 10 \times 8^{n-1}}{2} \\
&= \frac{1}{2} 10^n + \frac{1}{2} 8^n.
\end{aligned}$$

**Question 3:** Use induction to prove that, for any integer $n \geq 1$,

$$5^n + 2 \cdot 11^n \text{ is divisible by } 3.$$

**Answer:** **Base case:** When $n = 1$, we have $5^1 + 2 \cdot 11^1 = 27$, which is divisible by 3. So, the statement is true for $n = 1$.

**Induction hypothesis:** Now let $n > 1$. Assume the statement is true for $n - 1$, i.e. $5^{n-1} + 2 \cdot 11^{n-1}$ is divisible by 3.

**Induction step:** Consider the case of $n$:

$$\begin{aligned} 5^n + 2 \cdot 11^n &= 5^{n-1} + 4 \cdot 5^{n-1} + 2 \cdot 11^{n-1} + 20 \cdot 11^{n-1} \\ &= 5^{n-1} + 2 \cdot 11^{n-1} + 4(5^{n-1} + 5 \cdot 11^{n-1}) \\ &= 5^{n-1} + 2 \cdot 11^{n-1} + 4(5^{n-1} + 2 \cdot 11^{n-1} + 3 \cdot 11^{n-1}) \\ &= 3y + 4(3y + 3 \cdot 11^{n-1}) \\ &= 3(5y + 4 \cdot 11^{n-1}) \end{aligned}$$

which is divisible by 3.

By the principle of Mathematical Induction, we conclude that the statement is true for all integer $n \geq 1$.

**Question 4:** You are given a real number $a$ and a positive integer $n$ that is a power of 2, i.e. $n = 2^k$ for some integer $k \geq 0$.

(a) Devise a recursive algorithm to find $a^n$. Your algorithm should use as few multiplications as possible.

(b) Give a recurrence equation of the number of multiplications used in your algorithm in (a).

(c) Solve your recurrence equation in (b).

Note that in this question, we assume

(i) $b^m$ uses $m - 1$ multiplications for any real number $b$ and positive integer $m$.

(ii) $b/m$ is not counted as a multiplication if $m$ is a positive integer.

(iii) In the computation of $(f(n))^m$, $f(n)$ is evaluated only once. But, in the computation of $f(n) \cdot f(n)$, $f(n)$ is evaluated twice.

**Answer:** (a)

**procedure** *twopower*($n$: positive integer, $a$: real number)
**if** $n = 1$ **then return** $a$
**else return** $(twopower(n/2, a))^2$

(b) $T(1) = 0$. $T(2^k) = T(2^{k-1}) + 1$ for $n > 1$.

(c)

$$\begin{aligned} T(2^k) &= T(2^{k-1}) + 1 \\ &= T(2^{k-2}) + 2 \\ &\vdots \\ &= T(2^0) + k \\ &= k \\ &= \log_2 n \end{aligned}$$