CS211: Algorithms & Data structures

Dr. Sameer M. Alrehaili

September 28, 2021

srehaili@taibahu.edu.sa
college of computer science and engineering ,yanbu, Taibah University

Assignment 2 Solution

1. Calculate the total number of primitive operations executed for the following algorithm?

Algorithm 1: GCD

```
Input: Two integer numbers a and b
Output: gcd
1: m ← The minimum number of a and b.
2: gcd ← 0
3: i ← 2
4: while (i ≤ m) do
5:    if a mod i = 0 and b mod i = 0 then
6:       gcd ← i
7:    end if
8:    i ← i + 1
9: end while
10: return gcd
```

$T(n) = cn$, where c is some constant and n is the size of the input. At line (1), we will consider that the process of calculating the minimum of a or b may take constant time c and it executed 1 time, so the total of the first line is $1c$. At line (2), we count one unit for initialising *gcd* (1×1). At line (3), we count one unit for initialising i (1×1) . At line (4), we count one unit for each time we go around the while-loop +1 (n) (note that the counter started at 2 and that means n times. At line (5), we count 5 units for each time we go around the loop $(5(n-1) = 5n-5)$. At line (6), we count $n-1$. At line(8), we count $2(n-1) = 2n-2$. At line (10), we count 1.

T(n) $= c + 1 + 1 + n + 5n - 5 + n - 1 + 2n - 2 + 1 =$ **9n-5+c**

Since $9n$ is the fastest growing term in the function we can say $T(n)$ grows at the order of n and we write: $\mathbf{T(n)} = \mathcal{O}(n)$.

To estimate the process of selecting the minimum of two given numbers, here we include the process of selecting minimum inside the algorithm

Algorithm 2: GCD

Input: Two integer numbers a and b
Output: gcd
if $a < b$ **then**
    m $\leftarrow a$
else
    m $\leftarrow b$
end if
gcd $\leftarrow 0$
i $\leftarrow 2$
while $(i \leq m)$ **do**
    if $a \; mod \; i = 0$ and $b \; mod \; i = 0$ **then**
        $gcd \leftarrow i$
    end if
    i $\leftarrow i + 1$
end while
return gcd

The process of calculating the minimum of a or b may here is 3.
T(n) $= 1 + 1 + 1 + 1 + 1 + (n) + 5 * (n-2) + (n-2) + 2 * (n-1) =$ **9n-2**

Since $9n$ is the highest term in the function we can say $T(n)$ grows at the order of n and we write: **T(n)** $= \mathcal{O}(n)$.