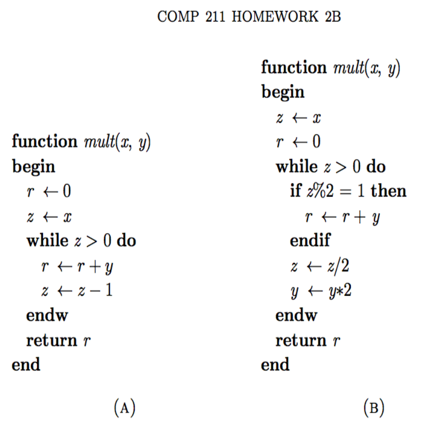
Fabien Bessez

COMP 211: Computer Science I

Homework 2B: Searching and Sorting



COMP 211 HOMEWORK 2B

```
function mult(x, y)
begin
  r ← 0
  z ← x
  while z > 0 do
    r ← r + y
    z ← z − 1
  endw
  return r
end
```

(A)

```
function mult(x, y)
begin
  z ← x
  r ← 0
  while z > 0 do
    if z%2 = 1 then
      r ← r + y
    endif
    z ← z/2
    y ← y*2
  endw
  return r
end
```

(B)

FIGURE 1. Two algorithms for multiplication.

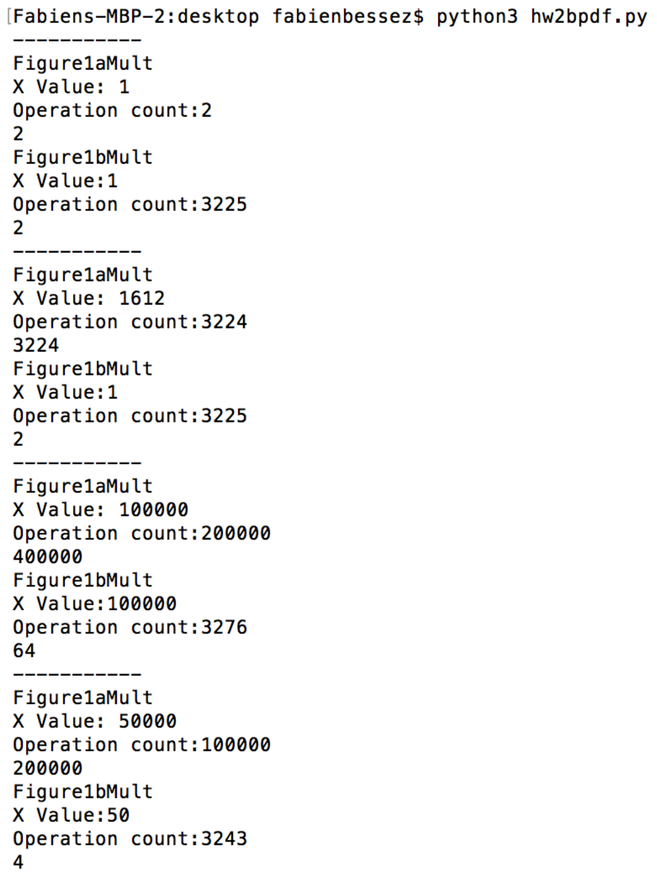
Problem 1. Compute the cost of the mult function in Figure 1(a). Count the number of arithmetic operations and describe the cost in terms of the value of x. Explain your reasoning.

The cost of the mult function in Figure 1(a) should be linear. In other words, it is O(x). In the function, there are two operations: the addition in " r ← r + y " and the subtraction in " z ← z – 1 ". Thus, every time the loop runs through, it will perform two operations. Then, we know that the loop will run z times because whatever value z is to begin with, it is subtracted by 1 each loop through the list until it reaches 0. So, that means that it is two operations every z times. However, since at the beginning of the function, we assign the value of x onto z, we know that the cost is O(x).

Problem 2. Compute the cost of the mult function in Figure 1(b). Count the number of arithmetic operations and describe the cost in terms of the value of x. Explain your reasoning.

The cost of the mult function in Figure 1(b) is logarithmic. In other words, it is O(logX). In the function there are 3 operations: The addition in "r ← r + y ", the division in "z ← z / 2 ", and the multiplication in " y ← y * 2 ". Thus, every time the loop runs through it will perform three operations. However, I discovered that the function, even with an input for x of 1, loops through 1075 times (3225 operations). This loop count value does not change much even with an input for x of 100,000. In that scenario, the mult function in Figure 1(b) loops through 1092 times (3276 operations). Therefore, it seems as if there is a pretty hefty initial cost for low values of x but as x increases, the cost only increases negligibly. I ran a few tests and found that the mult function in Figure 1(a) is better cost-wise for all x values less than or equal to 1612. For any x value greater than 1612, however, it would be beneficial to use the function shown in figure 1(b). The results of which I speak can be seen below:

Fabien Bessez



```
[Fabiens-MBP-2:desktop fabienbessez$ python3 hw2bpdf.py
-----------
Figure1aMult
X Value: 1
Operation count:2
2
Figure1bMult
X Value:1
Operation count:3225
2
-----------
Figure1aMult
X Value: 1612
Operation count:3224
3224
Figure1bMult
X Value:1
Operation count:3225
2
-----------
Figure1aMult
X Value: 100000
Operation count:200000
400000
Figure1bMult
X Value:100000
Operation count:3276
64
-----------
Figure1aMult
X Value: 50000
Operation count:100000
200000
Figure1bMult
X Value:50
Operation count:3243
4
```