Module 2: Critical Thinking

Algorithm Analysis

Nolan Byrnes

CSC506 – Design and Analysis of Algorithms

Colorado State University – Global Campus

Professor Isaac Gang

September 24th, 2022

Module 2: Critical Thinking

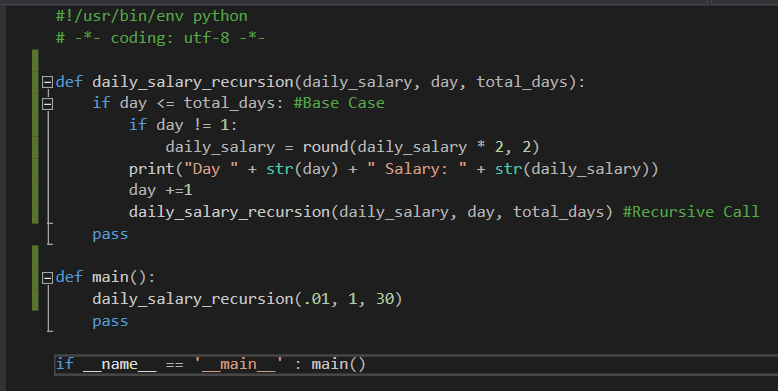
When designing algorithms, it is important to consider the potential errors that may occur at runtime when a computer is executing the developed algorithm. I developed two algorithms that prints the daily salary of a worker who is paid double of what they made the previous day, where one is based on a loop structure, and the other using recursion. Although both algorithms solve the same problem, we need to consider which algorithm that would be the most effective and the problems we may run into when implementing the solution on an actual machine. This paper discusses the two algorithms that I developed, and the potential errors that can occur when running an algorithm that is based on a loop and in a recursive structure.

Recursion Algorithm

A recursive algorithm is defined as “an algorithm that breaks the problem into smaller subproblems and applies the algorithm itself to solve the smaller subproblems” (Lysecky, 2019). In a recursive function, it requires a base case, and the recursive call. In the implementation of the recursive function to print the daily salary of a worker who is paid double each day, I had the function recursively call itself until it met the base case, which is the condition of exceeding the total days of the employee working. The resulting recursive function is shown in figure 1 below.

Figure 1.

Python implementation of Recursive Algorithm



Note. This algorithm prints the daily salary of a worker who is paid twice the amount of the previous day’s salary by using recursion.

When the program is running a recursive function, it is calling itself many times, introducing each function call into the call stack. When the program is run, the daily\_salary\_recursive() function is placed on the stack and will not be removed from the stack until all the actions within the function are completed. With the function recursively stacking the function over and over until the recursive function does not recursively call itself. Only then will the function calls begin to be removed from the stack. Sheldon explains that we receive a stack overflow “when a computer program tries to use more memory space in the call stack than has been allocated to that stack” (Sheldon, 2022, para. 1).

Another common issue that is run into when calling recursive functions is an Out of Memory Error. Every time that the recursive function is called, the computer will allocate memory for all the parameter values.

If the parameters of the recursive function are resource intensive, or there are too many parameters involved, the storage of all the variables could take up all the memory that the computer has, causing an out of memory error.

Loop-based Algorithm

In the algorithm based on a loop structure, the function takes in the first day’s salary, and the total amount of days being worked, and iterates through a loop for the amount of total days being worked. At each iteration of the loop, it reassigns the daily salary to be double of the previous number. I have the value of daily\_salary rounded to two decimal places to account for the fact that the American dollar only goes down to two decimal digits. This algorithm is displayed in figure 2 below.

Figure 2.

Python implementation of Loop-based Algorithm

A screenshot of a computer

Description automatically generated with medium confidence

Note. This algorithm prints the daily salary of a worker who is paid twice the amount of the previous day’s salary by using a loop-based structure.

When running the daily\_salary\_loop() function, I noticed that the maximum day that the function was able to process is 1031, which came up with the value of 1.1505236063118822e+308. When the function tried to calculate day 1032, it printed “inf”. Simplilearn mentions that “Python float values are represented as 64-bit double-precision values” and if a float “exceeds the max value, Python returns an error with string inf (infinity)” (Simplilearn, 2021, para. 3).

Microsoft states that the maximum value of the float data type is 3.402823466 E + 38 (Microsoft, 2021). When the function tried to calculate the daily salary for day 1032, it exceeded the maximum value of the float, resulting us the value of “inf”.

Conclusion

Comparing the two functions that I created to calculate the daily salary of a worker that gets paid double each day, I noticed that the daily\_salary\_loop() function would be the better solution to use if you were to implement it on a machine. It was able to calculate up to 1031 days of salary, whereas the daily\_salary\_recursion() was only able to calculate up to 995 days. The recursive function takes up more of the computers resources as it must include each function call into the stack until it reaches the base case. This increases the risk of running into a stack overflow or an out of memory error. The function based on a loop does not run into a stack overflow error but could run into the issue where the daily salary is higher than what a Float datatype can represent in 64-bits. Although both algorithms solve the same problem, when it comes to implementing them on a machine, the potential problems that the algorithms can produce must be considered to ensure that the machine that is planned to run the algorithms can do so effectively without error.

**REFERENCES**

Lysecky, R., & Vahid, F. (2019). 2.7 Recursive definitions. In Design and Analysis of Algorithms. zyBook.

Microsoft. (2021, August 3). Type float. Microsoft Learn: Build skills that open doors in your career. <https://learn.microsoft.com/en-us/cpp/c-language/type-float?view=msvc-170>

Sheldon, R. (2022, July 27). What is a stack overflow error? WhatIs.com. <https://www.techtarget.com/whatis/definition/stack-overflow>

Simplilearn. (2021, April 27). Understanding float in Python [with examples]. Simplilearn.com. <https://www.simplilearn.com/tutorials/python-tutorial/float-in-python>