Numerical Linear Algebra & Parallel Computing

Mohammed VI Polytechnic University

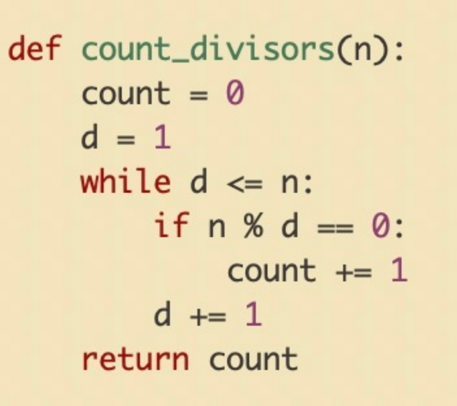
Nouredine Ouhaddou ■ Mohamed Jalal Maaouni ■ Ibrahim El Mountasser

Complexity Analysis

- ***Problem:***

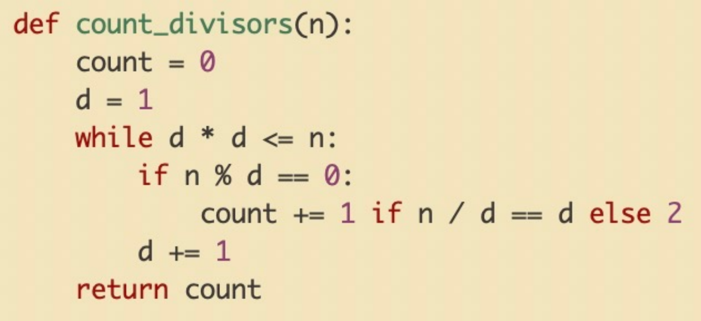
Given an integer n, count the number of its divisors.

Solution 1:



```
def count_divisors(n):
    count = 0
    d = 1
    while d <= n:
        if n % d == 0:
            count += 1
        d += 1
    return count
```

Solution 2:



```
def count_divisors(n):
    count = 0
    d = 1
    while d * d <= n:
        if n % d == 0:
            count += 1 if n / d == d else 2
        d += 1
    return count
```

Introduction

1) Describe solution 1
2) Describe solution 2
3) Run the two programs for different values of n and measure which algorithm is faster.
4) Calculate the number of operations executed by each of the programs for different values of n and generalize for any n.

Big-O notation

1) $T(n) = \ 3n^3 + 2n^2 + \frac{1}{2}n + 7$ prove that $\mathrm{T(n)} = \mathrm{O(n^3)}$
2) Prove: $\forall\, \mathrm{k} \geq 1,\ \mathrm{n^k}$ is not $\mathrm{O}\left(\mathrm{n^{k-1}}\right)$

Merge sort

1) Given two sorted arrays, write a function (with a language of your choice) that merge the two arrays into a single sorted array.

   Ex: def merge(A,B):

   …

   …

   return C

2) Analyse the complexity of your function using Big-O notation.

The master method

1) Using the master method analyse the complexity of merge sort.
2) Using the master method analyse the complexity of binary search

Bonus

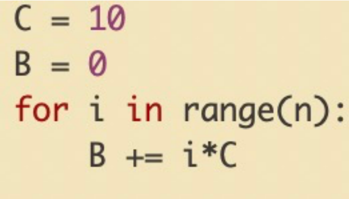
1) Write a function called merge sort (using a language of your choice) that takes two arrays as parameters and sort those two arrays using the merge sort algorithm.
2) Analyse the complexity of your algorithm without using the master theorem.
3) Prove the 3 cases of the master theorem.
4) Choose an algorithm of your choice and analyse it’s complexity using the Big-O notation.

Matrix multiplication

1) Write a function using python3 that multiply two matrices A,B (without the use of numpy or any external library).
2) What’s the complexity of your algorithm (using big-O notation)?
3) Write the same function in C. (bonus)
4) Optimize this multiplication and describe each step of your optimisation.

Quiz

1) What will be the time complexity for the following fragment of code?



```
C = 10
B = 0
for i in range(n):
    B += i*C
```

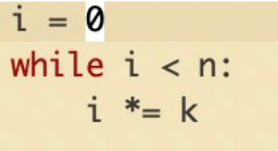
A) $O(n)$

B) $O(B)$

C) $O(log_n B)$

D) $O(log_C n)$

2) What will be the time complexity for the following fragment of code?



```
i = 0
while i < n:
    i *= k
```

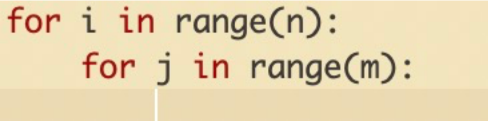
A) $O(n)$

B) $O(k)$

C) $O(log_n K)$

D) $O(log_k n)$

3) What will be the time complexity for the following fragment of code?



```
for i in range(n):
    for j in range(m):
```

A) $O(n)$

B) $O(n^2)$

C) $O(n * m)$

D) $O(n * \log (n))$