Algorithms and Data Structures I — Assignment 01

Ayse Sude Baki K12211229

1a) „for i in range(n):" the statements that are under it all get executed once for every iteration, so it iterates n times, which means that it has a time complexity of O(n). "for j in range(i, n):" runs n-i times since i is a constant value. Now if we plug in i = 0, the loop will run n times, basically O(n). And since this is a nested loop we have to multiply both loops. O(n) * O(n) = O(n^2)

1b) „i *= 10" from this line we know that the first loop iterates 10 times, which is why we need the log. Using this information we get O(log(n)) for the complexity. The second loop stops once j<n, which gives us a complexity of O(n). The third loop has a complexity of O(1), so we can just ignore it. We again, multiply the loops O(log(n)) * O(n) = O(log(n)*(n))

1c) "if a < b and b < c:" has a complexity of O(a). "if c < a:" would normally iterate c times but in this case it would differentiate the first loop and will therefor iterate b times. With this information we get a time complexity of O(min(a, c)). "elif a > b and b > c:" has a complexity of O(b – c). "else: for i in range(a, a + 5):" is O(1), so we can ignore it. We can see that O(a) is the most dominant complexity which makes it the big O notation of this algorithm.

$S = a(1 - r^n)/(1-r)$
$= 1/99 - 1/(99n^2)$

2a) $T(1) = 1$

$T(n) = 100T(n/10) + n$

→ we substitute the recurrence relation into itself until we reach "Base Case"

$T(n) = 100T(n/10) + n^2$

$= 100(100T(n/100) + n/10) + n^2$

$= 100^2 T(n/100) + n^2(1 + 1/100)$

$= 100^2(100T(n/1000) + (n/100)^2) + n^2(1 + 1/100)$

$= 100^3 T(n/1000) + n^2(1 + 1/100 + 1/10000)$

$= 100^k T(n/10^k) + n^2(1 + 1/100 + 1/10000 + \dots + 1/10^{2k})$

Let $k = \log 10(n)$ be the number of times we can divide n by 10 before we reach the base case

Now, we plug in k so,

$T(n) = 100^{\log 10(n)} T(1) + n^2(1 + 1/100 + 1/10000 + \dots 1/10^{2k})$

$= n^2 + n^2(1/100 + 1/10000 + \dots + 1/n^2)$

$T(n) = n^2 + n^2(1/100 + 1/10000 + \dots + 1n^2)$

$= n^2 + n^2(1/99 - 1(99n^2)$

$= n^2 + n^2/99 - 1/99$

$\leq n^2 + O n^2$

$= O(n^2)$

2b) $\exists c, n_0 : T(n) \leq cn^2 \ \forall n \geq n_0$

from 2a, we have: $T(n) = n^2 + O(n^2)$,

which means $\exists c : T(n) \leq cn^2 \ \forall n \geq 1$

Therefore we can choose $c = c$ & $n_0 = 1$,

So, we have

$T(n) \leq cn^2 \ \forall n \geq 1$

And this proves $\underline{O(n^2)}$

3a) written in the form of the Master Theorem as:

$a = 8$, $b = 2$, $f(n) = n^3$, $d = 3$

$\log\_b(a) = \log\_2(8) = 3$

Since $f(n) = n^3 = \Omega(n^{\log\_b(a)})$, we have Case 1

Thus the result of the recurrence relation is:

$T(n) = \Omega(n^{\log\_b(a)}) = \Omega(n^3)$

3b) written in the form of the Master Theorem as:

$a = 1$, $b = 2$, $f(n) = n \log n$, $d = 1$

$\log\_b(a) = \log\_2(1) = 0$

Since $f(n) = n \log n = \Omega(n^{d \log\_b(a)})$, we have Case 2

Thus, the result of the recurrence relation is:

$T(n) = \Omega(n^{d \log\_b(a)} \log n) = \Omega(n \log n \log n)$ $\Omega(n(\log n)^2)$

3c) written in the form of the Master Theorem

$a = 3$, $b = 3$, $f(n) = \log n$, $d = \log\_3(3) = 1$

$\log\_b(a) = \log\_3(3) = 1$

Since $f(n) = \log n = \Omega(n^d)$, we have Case 2

We also need to verify $af(n/b) \leq cf(n)$ for some constant $c<1$ and sufficiently large n.:

$3(\log(\frac{n}{3})) \leq c(\log n)$

$3\log n - 3\log(3) \leq c\log n$

$3\log n - \log(27) \leq c\log n$

$(3 - \log(27)/\log n)\log n \leq c\log n$

$2.17 \log n \leq c \log n$

Since $2.17 < 1$, the additional condition is satisfied

Thus, the result of the recurrence relation is:

$T(n) = \Omega(n^{d \log\_b(a)}) = \Omega(n^{\log\_3(3)}) = \Omega(n)$