Middle East Technical University Department of Computer Engineering

CENG 310

Algorithms and Data Structures with Python

Spring 2020-2021

Homework-03 Solution

By Mehmet Taha Şahin

Task-1

R-3.2:

We can find the solution with a simple python code and simplifying the formulas by dividing both with 2*n*. A now is equals to 4log*n* and B is equals to *n*. n needs to be equal 4log*n*. Intersection of to lines is the answer which is approximately 1.5.

```
import matplotlib.pyplot as plt
import math

x = [x/100 for x in range(1, 500)]
y1 = list(map(lambda x: 4*math.log(x),x))
y2 = x
plt.plot(x, y1)
plt.plot(x, y2)
```

R-3.18:

$2^{n+1} = (2)2^n$. Constant 2 does not affect complexity, then 2^{n+1} is $O(2^n)$.

R-3.20

Lets assume, f(n) = n^2 and g(n) = n*logn. Then, f(n) is in Ω(g(n)) if there is c > 0 and n0 > 0 such that: f(n) >= c*g(n) for every n >= n0. That means, “n*n >= c*n*logn”. Than situations when “n >= c*logn” meets the qeuality.

Task-3

R-3.27:

There are 3 nested loop and 2 of them are in range(n). One of them in range(1+j) and j also goes to n. Answer is $O(n^3)$

Task-5

P-3.57:

With 2 diffirent size of n we can compare the ratios of time and complexity.

```
import math
import random
import time

n1 = 1000000
l1 = random.sample(range(1,1000000000),n1)

n2 = 10000000
l2 = random.sample(range(1,1000000000),n2)

start=time.time()
sorted(l1)
end=time.time()
l1time = end-start


start=time.time()
sorted(l2)
end=time.time()
l2time = end-start
print("time to sort l1: " + str(l1time))
print("time to sort l2: " + str(l2time))

bigon1 = n1*math.log(n1)
bigon2 = n2*math.log(n2)

print("sorted ratio: " + str(l2time/l1time))
print("bigoNratio: " + str(n2/n1))
print("bigoNLOGNratio: " + str(bigon2/bigon1))
print("bigoNNratio: " + str((n2/n1)*(n2/n1)))
```

Output: (We can see that sorted ratio is close to nlogn ratio)

```
time to sort l1: 0.43955039978027344

time to sort l2: 5.9454240798950195

sorted ratio: 13.526148725759489

bigoNratio: 10.0

bigoNLOGNratio: 11.666666666666668

bigoNNratio: 100.0
```