CS 2420
Fall semester section 01
Ricardo Gonzalez Mendez

Chapter 5 Written

Written:

5.1: a. i=5, k=5, j=6

b. i=5, k=6, j=6

c. i=3, k=3, j=3

d. i=5, k=5, j=6

5.4: x^2, x, x^2 + x, x^2 – x, and (x^3 / (x – 1)) group with big oh

Group 1 "x": x

Group 2 "x^2": x^2, x^2 + x, x^2 – x, (x^3 / (x – 1))

Last term bottom big oh is x and top is x^3 so it cancels out one of the x being x^2 also.

5.6: the total cost of solving the problem is in the first algorithm=O(N) second algorithm=O(N) also so O(N)+O(N) = O(N+N) or just O(N) which is the total cost.

5.7: the total cost is first algorithm =O(N^2) and second algorithm =O(N) so O(N^2) +O(N) is just O(N^2 + N) and with big oh that is just O(N^2) for total cost.

5.8: Doing an O(N) on a running the n binary searches on N element is N*O(logN) so it turns into O(NlogN) then another algorithm running is O(N) so equation is O(N) + O(NlogN) + O(N) or just O(N + NlogN + N) which with big oh it is just O(NlogN) being the total cost.

Theory:

5.14 skip B: equation 500/100*.5 a. would take 2.5 ms 4 times more. c. would take 12.5 ms raise by 2 slower. d. would take 62.5 ms raise by 3 and would be slower.

5.15 skip B: 1 minute is 60000 ms a. 60000/.5*100 is 12000000 larger. c. n^2=60000/.5*(100)^2 is 1200000000 but square root It because n^2 so answer is 34641.06 larger d. n^3 = 60000/.5*(100)^3 is 120000000000 cube root it to become 4932.42 larger.

5.16 skip D: equation would be 2000/1000*5 a. for machine b linear would take 10 seconds twice as many items twice the speed. b. (2000)^2/(1000)^2*5 would take 20 seconds. c. (2000)^3/(1000)^3*5 would be 40 seconds.

5.19 skip N log 2 N: from fastest to slowest would be 37 constant, N and 2/n which would be the linear. Then N log N, N log log N are the same after is N log(N^2), then N^1.5 and N^2), N^2 log N all the same quadratic, cubic after which is N^3, and last is exponential 2^N, 2^N/2.

5.20 no b or c: a. fragment 1 is linear O(N). Fragment 2 is linear also. Fragment 3 is quadratic O(N^2). Fragment 4 is linear has 2 separate for loops O(N). Fragment 5 is cubic looks at it 3 times O(N^3). Fragment 6 is linear. Fragment 7 is O(N^5). fragment 8 is log(N)

5.23: so unlucky joe starts off in his own company by himself he to increase in work size he has to be larger then employs N by 1 so with that said the max companies' unlucky joe could've worked for is log2(N+1)

5.29: version 1 is the fastest because big oh is O(N) while the other 2 are O(N^2)

5.36: so, we know that if its being double it most likely is a log so the max number of times it can be increased is log2(N)

5.39: so, in the code I believe it is O(N) it runs through the loop with an enhanced for loop to add the string into another array so it just reads through each element once then moves on.