Jake Goldstein Written HW 01

Written Part:

1. Write each of the following functions in Big-Oh notation:
   1. O(n)
   2. O(n2)
   3. O(n3)
   4. O(n2)
   5. O(n3)
   6. O(n)
   7. O(n3log(n))
   8. O(n2)
   9. O(n3)
2. For each of the following code fragments, determine the worst case running time using BigOh notation as a function of n.
   1. O(1)
   2. O(n)
   3. O(n2)
   4. O(n2)
   5. O(n3)
   6. O(n)
   7. O(n2)
   8. O(n3)
   9. O(log(n))
   10. O(nlog(n))
   11. O(log(n))
3. Ordered from fastest to slowest below (n, √n, n1.5, n2, nlog(n), nlog2(n), n/2, n3):

√n, n/2, n, nlog(n), nlog2(n), n1.5, n2, n3

where n and n/2 have the same big-oh, but they do not grow at the same rate.

1. Suppose a program takes 0.05 seconds to run on input size of 2048. Estimate how long it would run for an input size of 213 if:
   1. 211 yields 0.05 running O(n). We are trying to find 213, so we can say (213/211)n\*t will equal the run time. Since the run time is O(n), n=1, and the original run time was 0.05, t = 0.05. Then: (213/211)1\*.05 = 0.2
   2. 211 yields 0.05 running O(n2). We are trying to find 213, so we can say (213/211)n\*t will equal the run time. Since the run time is O(n2), n=2, and the original run time was 0.05, t = 0.05. Then: (213/211)2\*.05 = 0.8
   3. 211 yields 0.05 running O(n4). We are trying to find 213, so we can say (213/211)n\*t will equal the run time. Since the run time is O(n4), n=4, and the original run time was 0.05, t = 0.05. Then: (213/211)4\*.05 = 12.8
2. Suppose you had a very complicated code that was difficult to analyze. To get a quick idea of your algorithm’s running time you ran your program on different sized inputs. Suppose the following are the timing results for your algorithm. Using the timing results below, indicate the most likely running time in big-Oh notation; choose one from the following list. O(1), O(n), O(n2), O(n3), O(n4).

Since the time is increasing by about a factor of 4 from each n to the next, the most likely running time is O(n2).

1. Using the definition of Big-Oh, show that 3n2 + 2n log(n) + 6n + 19 = O(n2).

3n2 + 2n2 + 6n2 + 19n2 = 30n2 so….

3n2 + 2n log(n) + 6n + 19 ≤ 30n2 when n ≥ 1

3 + 0 + 6 + 19 ≤ 30

28 ≤ 30

1. Look at the output from the programming part 1 of this assignment. Create a chart of your answers including times for all three algorithms and for all the specified input sizes

Actual times:

|  |  |  |  |
| --- | --- | --- | --- |
| n | maxSubsequenceSum1 O(n^3) | maxSubsequenceSum2 O(n^2) | maxSubsequenceSum4 O(n) |
| 128 | 0.001137 | 0.000039 | 0.000002 |
| 256 | 0.008603 | 0.000129 | 0.000003 |
| 512 | 0.061773 | 0.000506 | 0.000004 |
| 1024 | 0.474599 | 0.002004 | 0.000005 |
| 2048 | 3.772521 | 0.007974 | 0.000010 |
| 4096 | 30.054885 | 0.031828 | 0.000018 |

1. Create another chart that estimates the running time for each of the algorithms using the method presented in class, and using the time your computer took for the algorithms when n = 27 . (If you did not receive valid answers, use the run times from question 7.) Display your answer in a chart

Perdicted times:

|  |  |  |  |
| --- | --- | --- | --- |
| n | maxSubsequenceSum1 O(n^3) | maxSubsequenceSum2 O(n^2) | maxSubsequenceSum4 O(n) |
| 256 | 0.009096 | 0.000156 | 0.000004 |
| 512 | 0.072768 | 0.000624 | 0.000008 |
| 1024 | 0.582144 | 0.002496 | 0.000016 |
| 2048 | 4.657152 | 0.009984 | 0.000032 |
| 4096 | 37.257216 | 0.039936 | 0.000064 |

1. Using the method given in class, predict how long each algorithm would take if n = 218 . Show how you made your prediction by providing the formula (presented in class), and then evaluate your formula.
   1. So for 212 (which is 4096), the algorithms took 30.054885 for O(n3). We are trying to find the run time for 218, so we can say (218/212)n\*t will equal the run time. Since the run time was O(n3), n=3, and the original time was 30.054885, so t = 30.05488. Then: (218/212)3\*30.05488 = 7878707.77344
   2. 212 took 0.031828 for O(n2). We are trying to find the run time for 218, so we can say (218/212)n\*t will equal the run time. Since the run time was O(n2), n=2, and the original time was 30.054885, so t = 0.031828. Then: (218/212)2\*0.031828 = 130.367488
   3. 212 took 0.000018 for O(n). We are trying to find the run time for 218, so we can say (218/212)n\*t will equal the run time. Since the run time was O(n), n=1, and the original time was 0.000018, so t = 0.000018. Then: (218/212)1\*0.000018 = 0.001152
2. Using your answer from the previous question, rewrite your answer cumulatively in seconds, minutes, hours, days, and weeks. (To clarify: you are not going to re-express that time in terms of weeks and then again in terms of days and then again in terms of hours. You should only express it once, using weeks, days, hours, minutes, and seconds.)
   1. 7878707.77344 seconds is the same as 13 weeks 0 days 4 hours 31 minutes and 47.77344 seconds.
   2. 130.367488 seconds is the same as 2 minutes and 10. 367488 seconds.
   3. 0.001152 seconds is 0.001152 seconds.
3. Look at the output from the programming part 2 of this assignment. Create a chart of your answers including times for all four algorithms and for all the specified input sizes, e.g.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | b | c | d | e |
| 256 | 0.000002 | 0.000152 | 0.000075 | 0.034566 |
| 512 | 0.000001 | 0.000522 | 0.000259 | 0.266353 |
| 1024 | 0.000002 | 0.002047 | 0.001024 | 2.132196 |
| 2048 | 0.000005 | 0.008165 | 0.004108 | 17.190329 |
| 4096 | 0.000010 | 0.032732 | 0.017065 | 135.652686 |

1. Circle ***True*** or False. I understand that if I don’t submit a .pdf file for the written portion of this assignment, I will receive no credit for the written part of this assignment.
2. Circle ***True*** or False. I understand that if I don’t submit code that compiles for the programming portion of this assignment, I will receive no credit for the programming part of this assignment.