CS104 HW1

Alexander Deng

September 7, 2022

Problem 3: Runtime Analysis

a) The inner while-loop will terminate when $i \geq n$. With each loop, i is set equal to it's square. This operation and the given operation together take O(2). The loop itself has a runtime of $log(log(n))$. The inner logarithm represents how many powers of 2 it takes to get to n. Since i is being squared instead of simply being raised to an additional power, another logarithm is needed to account for the base (2) being effectively squared as well with each loop. Ignoring constants, the approximate runtime of the function is $log(log(n))$.

b) We first notice that the inner for-loop will run i^3 times. Through plugging in example numbers, we can determine that the modulo if statement will evaluate to true $\sqrt{n}$ times. Lastly, the outer for-loop will run n times. This algorithm can then be expressed in summation form: $\sqrt{n}^3 \sum_{i=0}^{\sqrt{n}} i^3$ which, using the arithmetic series formula, evaluates to $O(n^{7/2})$

c) The inner for-loop will terminate when $m > n$. With each loop, m is doubled. So m = 1, 2, 4, 8, 16 [...] at k iterations we will be increasing by 2^k. Thus, the runtime of the inner loop is log(n) since we are increasing closer to n by a factor of 2. Since this is an analysis of the worst case for runtime, we can assume that the array is sorted from 1 to n, meaning that the if statement and the inner loop will evaluate to true n times. Lastly, the two outer loops each have runtimes of n, which when nested, produce a runtime of n^2. The if statement and subsequent inner for loop evaluate n times which in total produce the runtime of the algorithm: $n^3 log(n)$.

d) We first recognize that if i = size the variable *size* is increased by 3/2 with each passing of the outermost for-loop. However, since the operation inside the inner for-loop is O(1), and *size* is set as 10 and increases by a fixed amount, it's runtime is not determined by the input n, and is thus constant. We can thus conclude that the runtime is only determined by the outermost for-loop. The runtime is $O(n)$.