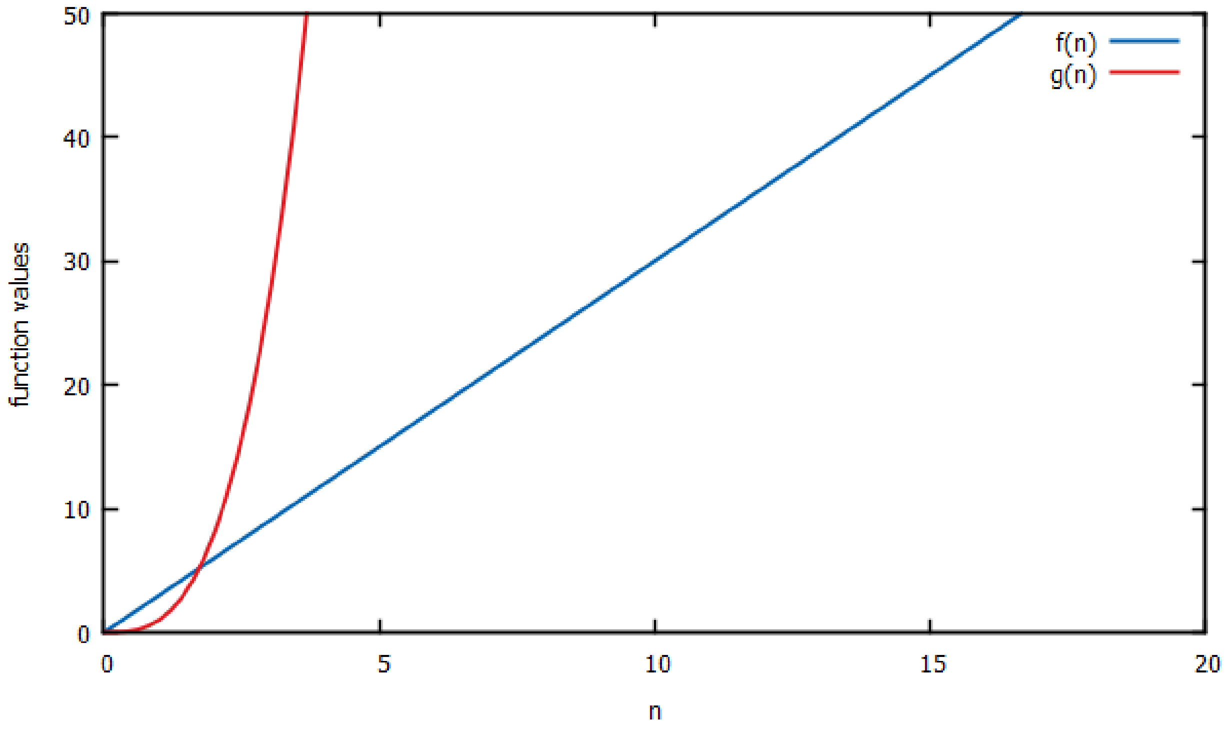
Homework 1

By Farzan Ali Khan

Problem 1.1 Asymptotic Analysis

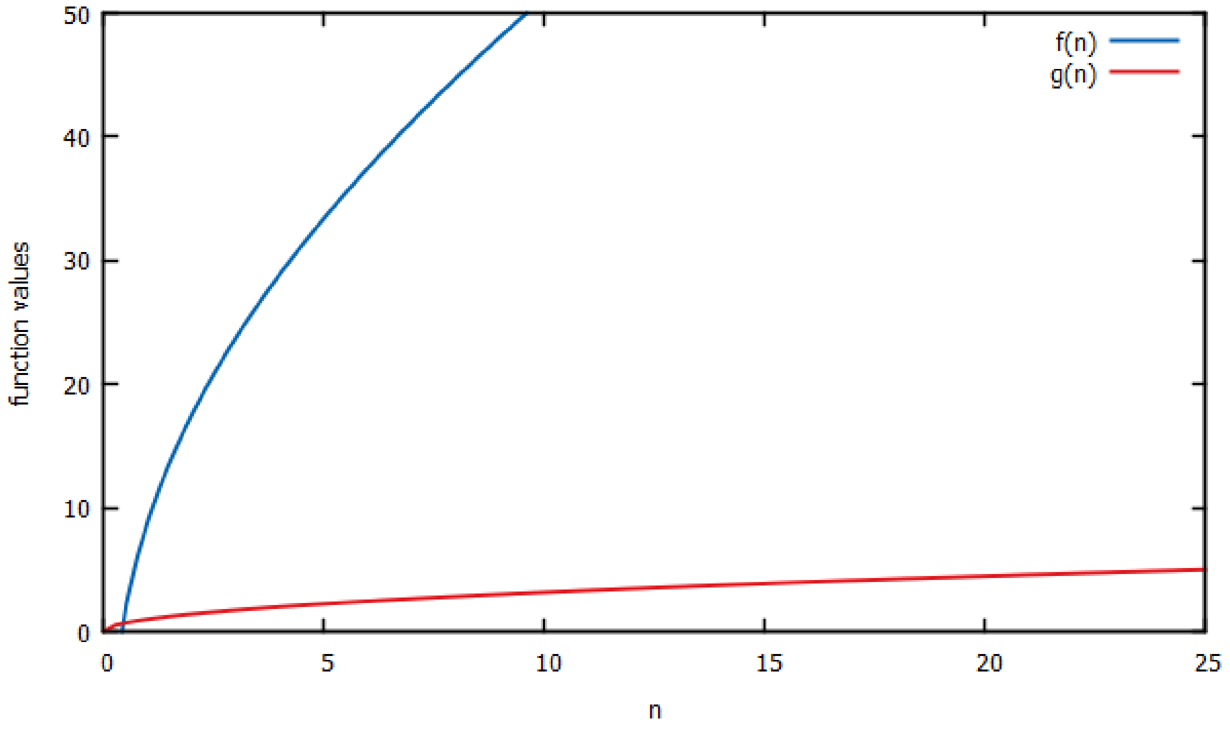
a) Graph:



Where $f(n) = 3n$, $g(n) = n^3$

From the graph we can clearly see that for $n > \sqrt{3}$ g(n) is significantly greater than f(n) hence f(n) ϵ o(g(n)) with $n_0 = \sqrt{3}$

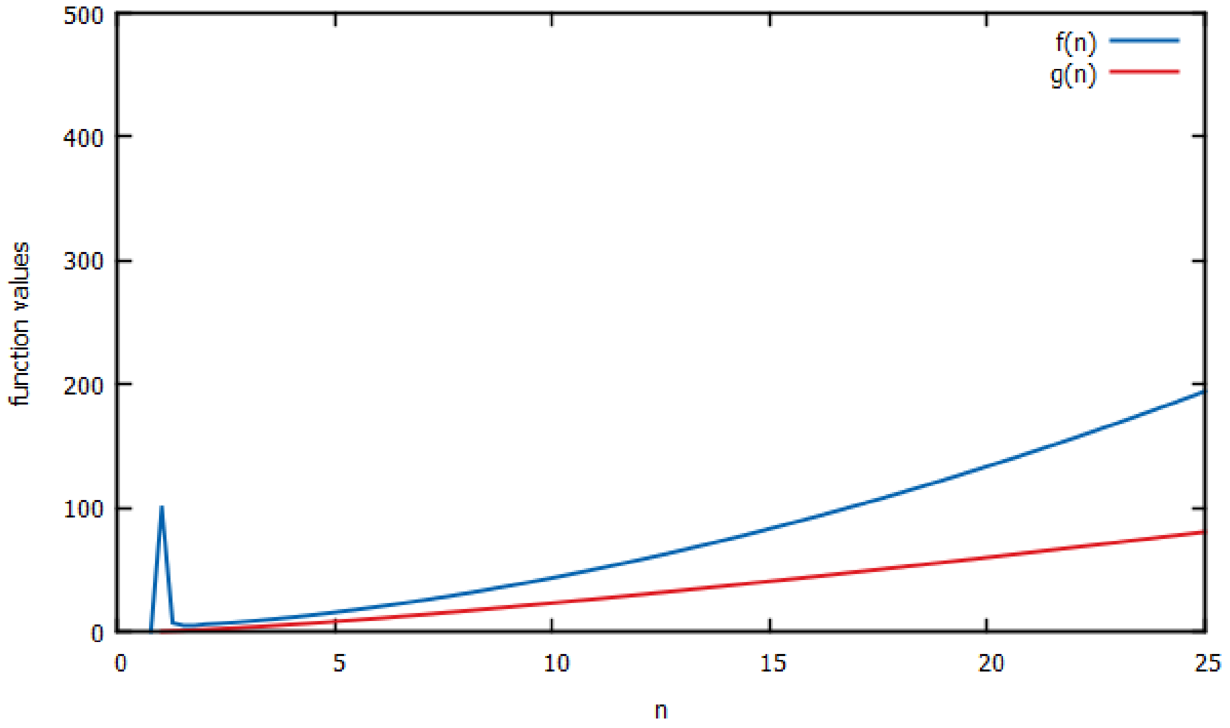
b) Graph



Where $f(n) = 7n^{0.7} + 2n^{0.2} + 13\log n$, $g(n) = \sqrt{n}$

From the graph we can clearly see that for $n > 0.5$ f(n) is significantly greater than g(n) hence (n) ϵ o(f(n)) with $n_0 = 0.5$.

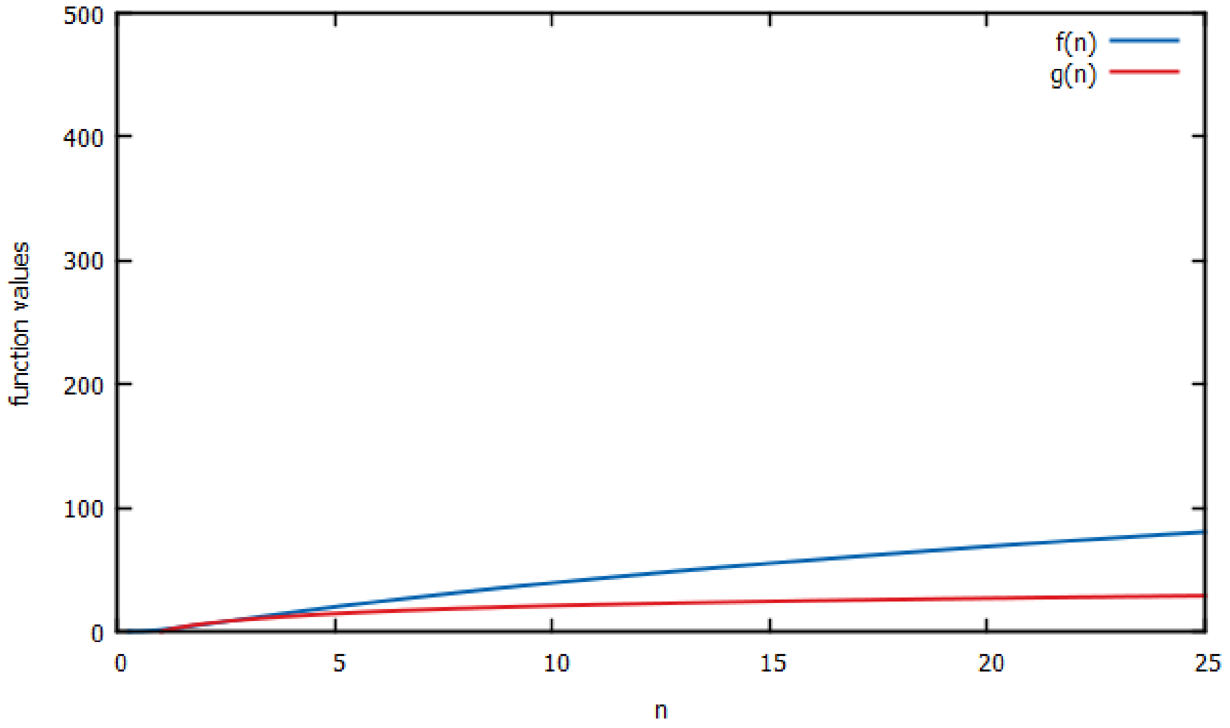
c) Graph



Where $f(n) = n^2/\log n$, $g(n) = \mathrm{n}\log n$

From the graph we can clearly see that for $n > 1.5$ f(n) is a tight upper-bound of g(n) hence f(n) is an asymptotic upper-bound of g(n),g(n) ϵ O(f(n)) with $n_0 = 1.5$.

d) Graph



Where $f(n) = (\log(3n))^3$, $g(n) = 9\log n$

From the graph we can clearly see that for $n > 4$ f(n) is a tight upper-bound of g(n) hence f(n) is an asymptotic upper-bound of g(n),g(n) ϵ O(f(n)) with $n_0 = 4$.

Problem 1.2

a) See code file

b) The loop invariant of the selection sort states that the right side of the array at any time is always sorted. This remains true until the code end. At the start the right side is empty i.e { }{a_0, a_1. . . . a_n}.

The loop compares the right most variable in the left side of the array (a_0) with all the other members of the array, finds the smallest member of the array and swaps it with the right most variable. The new right most variable of the left array is now the first and only member of the right array hence the right array remains sorted. { a_{min} }{ a_1. . a_0 . . a_n}.

The process repeats with a_1 as the right most variable of the left array and the new minimum added on the left side of the right array { a_{min}, a_{min2} }{ a_2. .

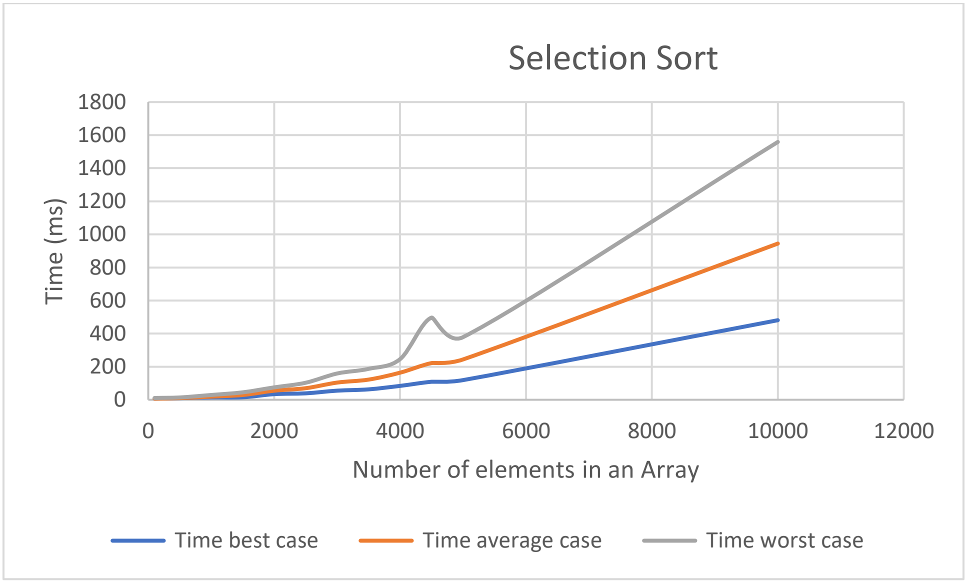
a_0 .a_1. a_n}. Since the new minimum of the left array is larger than the previously found minimum, the right array remains sorted.
The above process goes on until there are no more elements in the left array and we get a completely sorted right array.

c) I used the srand and rand function in C++ to get random numbers up to 1000 for my array of size 100. This represented the average case. The selection sort algorithm was then run on the array to produce a sorted array. This sorted array was regarded as the best case. The sorted array was then reversed which represented the worst case.

d) The table for the data obtained is as follows:

Number of elements	Time best case	Time average case	Time worst case
100	6.993	6.993	10.899
500	9.99	9.99	13.986
1000	13.991	18.986	28.981
1500	14.995	28.98	44.971
2000	33.991	54.966	74.945
2500	38.99	69.961	102.928
3000	54.991	103.945	158.891
3500	62.995	121.933	186.878
4000	83.991	163.91	245.836
4500	108.99	221.891	496.773
5000	118.95	243.876	378.751
10000	480.99	944.541	1558.86

The Graph for the data is as follows:



e) According to the graph ploted, the selection sort is bound asymptotically by n^2 i.e f(n) = Θ(n^2) with $n_0 = 2000, c_1 = 0.4875$ and $c_2 = 3.178$