# CS 5633: Analysis of Algorithms

# Solution of Homework-1

**1. a)** The loop invariant for the while loop is: at the beginning of the c-th iteration, k = 2c

**b)** The invariant holds before the first iteration

Before the first iteration of the loop, c = 0 and k = 1. Hence, the output is 20 = 1 which is exactly the same as k.

If the invariant holds before an iteration, then it also holds before the next one

Let’s say, the invariant holds at the beginning of the i-th iteration. So, k = 2i

During the iteration, we multiply k with 2 again. So, we get k = 2i \* 2 = 2i+1

Which is the same as the beginning of the i+1-th iteration.

The invariant holds at the end of the iteration

At the end of the iteration, c = n, and k = 2n

The loop invariant holds before the first iteration, if the invariant holds before an iteration then it also holds before the next one and the invariant holds at the end of the iteration. So, the algorithm is correct.

**c)**  function POW(n)

k = 1 c1 1

c = 0 c2 1

while c<n do c3 n+1

k = k.2 c4 n

c = c+1 c5 n

end while

return k c6 1

end function

Assume line i takes ci time, where ci is a constant and independent of the input.

Tn = c1+c2+(n+1)c3+n(c4+c5)+c6

= c1+c2+c3+c6+n(c3+c4+c5)

= a + bn where a,b are constants

So, the runtime is Ө(n)

2. **a)** Time complexity of the program is 3n/4 . log3(20n)

As the change in the base of the log only affects the result by a constant factor,

log3(20n) = c.log(n) where c is a constant

So, 3n/4 . log3(20n) <= n . log3(20n) = c.nlog(n)

So, 3n/4 . log3(20n) ɛ О(nlog(n))

Again, 3n/4 . log3(20n) >= ¾ . nlog3(n) = c. nlog(n) [As the change in the base of the log only affects the result by a constant factor]

So, 3n/4 . log3(20n) ɛ Ω(nlog(n))

As 3n/4 . log3(20n) ɛ О(nlog(n)) and 3n/4 . log3(20n) ɛ Ω(nlog(n)), we can conclude that,

3n/4 . log3(20n) = Ө(nlog(n))

**b)** Time complexity of the program is 3n2 . log2(3n2)

So, 3n2 . log2(3n2) <= 10 . 3 . n2 . log(3n2) = 10.3.n2 c.log(n) = k. n2log(n)

So, 3n2 . log2(3n2) ɛ О(n2log(n)) where k is a constant

Again, 3n2 . log2(3n2) >= n2 . log2(3n2) >= n2 . c.log(n) = c.n2log(n)

So, 3n2 . log2(3n2) ɛ Ω(n2log(n))

As 3n2 . log2(3n2) ɛ О(n2log(n)) and 3n2 . log2(3n2) ɛ Ω(n2log(n)), we can conclude that,

3n2 . log2(3n2) ɛ Ө(n2log(n))

**3. a)** 4n5 - 50n2 + 10n ɛ Ө(n5)

First, I will prove that, 4n5 - 50n2 + 10n ɛ О(n5)

4n5 - 50n2 + 10n <= 4n5 + 10n <= 4n5 + 10n5 = 14n5 for all n >= 1

So, 4n5 - 50n2 + 10n <= 14n5

So, 4n5 - 50n2 + 10n ɛ О(n5) where c = 14 and g(n) = n5 for all n >= 1

Now, I will prove that, 4n5 - 50n2 + 10n ɛ Ω(n5)

4n5 - 50n2 + 10n >= 4n5 - 50n2 = n5 + 3n5 - 50n2 >= n5 for all n >= 3

So, 4n5 - 50n2 + 10n ɛ Ω(n5) where c = 1 and g(n) = n5 for all n >= 3

As, 4n5 - 50n2 + 10n ɛ О(n5) and 4n5 - 50n2 + 10n ɛ Ω(n5) we can conclude that,

4n5 - 50n2 + 10n ɛ Ө(n5)

**b)** 5n2/3 +8logn ɛ o(n)

lim (5n2/3 +8logn) / n

n⇾∞

= lim 5n-1/3 + lim (8logn)’ / (n)’

n⇾∞ n⇾∞

= lim 5n-1/3 + lim (8. 1/n) / 1

n⇾∞ n⇾∞

= 5 . lim 1 / n1/3 + 8. lim 1/n

n⇾∞ n⇾∞

= 0 + 0

= 0

So, 5n2/3 +8logn ɛ o(n)

**c)** n5 + 4n2 + 15 ɛ Ω(n3)

n5 + 4n2 + 15 >= n5 + 4n2 = n3 + n5 - n3 + 4n2  >= n3 when n >= 1

So, n5 + 4n2 + 15 ɛ Ω(n3)

**4.** log2n <= 2√log2n <= n1/3 <= n5 <= 10n <= nn