1. Prove or disprove the following statements.

(a) n2 − 10n + 2 = O(n2)

Let c=3, N=1. Then for every n ≥ N ,

f(n) = n2 − 10n + 2 <= n2 + 2n2 = 3n2 = cg(n)

(b) (2n)2 = O(22n)

Let c=1, N=1. Then for every n ≥ N ,

f(n) = (2n)2 = 22n = cg(n)

(c) n log2(n) = O(n log10(n))

Let c=log2(10), N=1. Then for every n ≥ N ,

f(n) = n log2(n) = log2(10)nlog10(n) = cg(n)

(d) n log2(n) = O(n)

Let c=1, N=2. Then for every n ≥ N ,

f(n) = n log2(n) <= n = cg(n)

(e) n2n = O(22n)

Let c=1, N=1. Then for every n ≥ N ,

f(n) = n2n <= 2n.2n = 22n = cg(n)

2.

Alg1: The single loop in line 3 <=> 0(n), and the nested loop in loop in line 5, 6

=> O(n+n2), drop O(n) then O(n2).

Alg2: We have a single loop in line 3, but the number of loops remaining is halved after a loop completed because of line “i = i/2” inner the loop.

=> O(log2(n))

3.

majority\_element(A, l, r, k)

{

if l < r | (l =r & A[l] != k)

return 0

if l =r & A[l] = k

return 1

return majority\_element(A, l, (l+r)/2, k) + majority\_element(A, l + (l+r)/2, r, k)

}

num = majority\_element(A, 0, len(A)-1, k)

if num >= len(A)/2

print num

else

print null

Divide&Conquer until only 1 element in array so The runtime of program is O(n).

4.

T (n) = 3T(n/2) + n2 , where T (1) = O(1).

Here a = 3, b = 2 and f(n) = n2

nlogba = nlog23  = O(n1.585) => f(n) = Ω( nlog23+€)

€ = 0.415

af(n/b) = 3f(n/2) = 3n2/4 <= cf(n) = cn2 (with c < 1, set c = 3/4 would cause this condition to be satisfied)

case 3 of the master theorem applies, and the solution is T(n) = Ω(nlog23+€)

5.

T (n) = 2T (n/2) + nlog2(n), where T (1) = O(1).

Suppose n=2m. Then we have

T (2m) = 2T (2m/2) + 2mlog2(2m) = T (2m-1) + m2m, where T (1) = O(1)

Calling T (2m) as f(m)

f(m) = 2f(m-1)+m2m = 2(2f(m-2)+(m-1)2m-1)+m2m

= 4f(m-2)+(m-1)2m+m2m = 4(2f(m-3)+(m-2)2m-2)+(m-1)2m+m2m

= ……

then we have:

f(m) = 2mf(0) + 2m(1+2+3+...+m) = 2mf(0) +m(m+1)2m-1

=> T(n) = nT(1) + n(log2(n)(1+log2(n))/2 = Θ(nlog2n)

6.

Karatsuba algorithm uses 3 recursive call on n/2 size input. T(n) = 3T(n/2)+O(n)

*T(n) = aT(n/4) + n*

Now, when *a* increases, number of subproblems determines the asymptotic running time of the problem and case 1 of master theorem applies. So, in worst case, asymptotic running time of the algorithm will be T(n) = Θ(nlogb(a)) = Θ(nlog4(a)) = Θ(nlog2(a^(1/2)))

nlog2(a^(1/2)) < nlog2(3)

=> a1/2 < 3

=> a < 9

Hence, largest integer value of a is 8