Kyle Wolf

Problem Set #4

**1)**

a = 1, b = 2, f(n) = n, h(n) = nlog21

= n

Case 3:

= Ω(n1), 1 > 0

Regularity Condition:

f(n/2) ≤ f(n)

=> T(n) = Ө(n)

**2)**

Master Theorem does not apply, a = 2n, not a constant.

**3)**

a = 5, b = 2, f(n) = n, h(n) = nlog25

Case 1:

= O(n-1), 1 > 0

=> T(n) = Ө(nlog25)

**4)**

a = 1, b = 1, f(n) = c, h(n) = nlog11 = undefined

Mastertheorem does not apply.

**5)**

a = 4, b = 4, f(n) = O(1), h(n) = nlog44 = n

=

Case 1:

= O(n-0.5), 0.5 > 0

=> T(n) = Ө(n)

**6)**

a = 2, b = 2, f(n) = n/logn, h(n) = nlog22 = n

=

Case 1:

= O(n-0.1), 0.1 > 0

=> T(n) = Ө(n)

**7)**

a = 4, b = 2, f(n) = n logn h(n) = nlog24 = n2

= log(n)/n

Case 1:

= O(n-0.5), 0.5 > 0

=> T(n) = Ө(n2)

**8)**

a = 4, b = 2, f(n) = n2, h(n) = nlog24 = n2

= 1

Case 2:

= Ө(log0n), 0 ≥ 0

=> T(n) = Ө(n2logn)

**9)**

Master theorem does not apply, f(n) not bounded above by a polynomial.

**10)**

Root: n

Has two children: n/3, and n/4

Total value of first two children: 7n/12 = 7/12 \* n

At level i of recursion, total value of children is (7/12)in

T(n) = Ө(n)

**11)**

Root: 2n

Has child: 2n/2 = 2(1/2)n

At level i of recursion, total value of children is 2(1/2)^i n

T(n) = Ө(2n)

**12)**

Root: cn

Has four children: c(n/2) each

Total value of first four children: 4c(2n) = 8cn

At level i of recursion, total value of children is (8c)in

T(n) = Ө(n)

**13)**

**a)** T(n) = T(3n/4) + c

**b)**

a = 1, b = 4, f(n) = c, h(n) = nlog41 = n0 = 1

= c

Case 2:

= Ө(log0n), 0 ≥ 0

=> T(n) = Ө(logn)

**14)**

**a)** T(n) = (n/16)T(n/16) + c

**b)** T(n) = Ө(n)

**15)**

**a)** True. Dividing the array continuously into 1/4th and 3/4ths portions leaves the possibility of the target being in the larger of the two portions repeatedly, resulting in more recursive calls than an equal parts division.

**b)** True. Even if in a skewed divide-and-conquer algorithm, like the one described above, you are repeatedly taking the larger of the two portions, you’re still shaving off 1/4th of the array each recursive call. In the worst case, the solution described in problem 14 requires you to search every element n. This will never be the case in skewed divide-and-conquer algorithm.