**Masters Theorem**

Let T(n) be a monotonically increasing function that satisfies:

| T(1) = c  T(n) = ,  where,  a>=1, b>=1, c>0  n = Size of the problem  a = Number of subproblems in the recursion and a>=1  = Size of each subproblem  f(n) = Cost of work done outside the recursive calls  There are 3 cases  If f(n) **∈** Θ () where d>=0, then   1. T(n) = Θ() if a < () 2. T(n) = Θ(log n) if a = () 3. T(n) = Θ() if a > () |
| --- |

Time complexities for

1. T (n) = 3T (n/2) + n

**Solution:**

**T(n) = Θ() = Θ()**

a = 3, b = 2, d= 1

1. T (n) = 64T (n/8) − n^2(log n)

**Solution:**

Time complexity **can not be calculated**

1. T (n) = 2nT (n/2) + n^n

**Solution:**

Time complexity **can not be calculated**

1. T (n) = 3T (n/3) + n/2

**Solution:**

**T(n) = Θ()**

a = 3, b = 3, d = 1

1. T (n) = 7T (n/3) + n^2

**Solution:**

**T(n) = Θ(**

a = 7, b = 3, d =2