



**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Solving Recurrences

The Master Theorem gives a general solution to recurrence relations that are of the form   
*T*(*n*) = *aT*(*n/b*) + *f* (*n*)where *f*(*n*)∈Θ(*nd*), *d ≥* 0- this is called the general divide and conquer recurrence

Master Theorem: If *a < bd* or logb (a) < d, *T*(*n*) ∈ Θ(*nd*)

If *a = bd* or logb (a) = d, *T*(*n*) ∈ Θ(*nd* log *n*)

If *a > bd* or logb (a) > d, *T*(*n*) ∈ Θ(*n*log *b a* )

Note: The same results hold with O instead of Θ

The key to understanding what is going on is the relationship between logb (a) and d.

Solve the following recurrences **using the Master Theorem** giving the Θ() complexity class for each.   
**Show the values for a, b, and d for each**

1. T(n) = 2T(n/2) + c a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
2. T(n) = T(n/2) + c a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
3. T(n) = 4T(n/2) + c a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
4. T(n) = 2T(n/2) + n a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
5. T(n) = T(n/2) + n a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
6. T(n) = 4T(n/2) + n a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
7. T(n) = 2T(n/2) + n2 a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
8. T(n) = T(n/2) + n2 a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )
9. T(n) = 4T(n/2) + n2 a = \_\_\_\_\_\_ b = \_\_\_\_\_\_ d= \_\_\_\_\_\_ Θ( )

Solve the following recurrences **using back substitution or the recursion tree method** giving the Θ() complexity class for each. You may need to consult the text or the Internet for necessary summation formulas. Show your work!

1. T(n) = T(n-1) + 2
2. T(n) = T(n-1) + n
3. T(n) = T(n-1) + 2n
4. T(n) = 2T(n-1) + 1