Algorithm ***mergeSort***(***S, C***)

Input sequence ***S*** with ***n*** elements, comparator ***C***

Output sequence ***S*** sorted according to ***C***

```
if S.size() > 1 then                 // O(1)
    (S1, S2) ← partition(S, n/2)     // O(n)
    mergeSort(S1, C)                 // T(n/2)
    mergeSort(S2, C)                 // T(n/2)
    merge(S1, S2, C, S)              // O(n)
```

T(n) = 2 T(n/2) + c n

Recurrence equation looks like T(n)= a T(n/b) + f(n) so Merge-sort: T(n) = 2 T(n/2) + cn

(where a=#recursive calls, 1/b is fraction of n in recursive calls, f(n) is time to do all but recursive calls)

Guess and Test:

T(n) <= c n log n

Proof:
T(n) = 2 T(n/2) + k n
T(n) <= 2 (c n/2 log (n/2)) + k n
T(n) <= (c n (log n – log 2)) + k n
T(n) <= c n log n – c n + k n
T(n) <= c n log n , for all c >= k

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 1. T(n) = 4 T(n/2) + n

$\log_2 4 = 2$

Case 1: Is f(n)= n O(n^{2-e})? yes for 0<e<=1, pick e=1

Conclusion: Therefore, T(n) is $\Theta(n^2)$ (To receive full credit, you must give conclusion!)

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 2. T(n) = 2T(n/2) + bn log n

$\log_2 2 = 1$ => O(n)

Case 1: Is f(n)=bn log n   O(n $^{1\text{-}e}$)? (no)

Case 2: Is f(n)=bn log n $\Theta(n^1 \log^k n)$ for some k $\geq$ 0? Yes for k=1

Conclusion: Therefore, T(n) is $\Theta(n^1 \log^{k+1} n) = \Theta(n \log^2 n)$

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 3. T(n) = T(n/3) + n log n

$\log_3 1 = 0$

Case 1: Is f(n)=n log n   O(n $^{0\text{-}e}$)? (no)

Case 2: Is f(n)=n log n $\Theta(n^0 \log^k n)$ for some k $\geq$ 0? No

Case 3: Is f(n)=n log n $\Omega(n^{0+e})$ for some e > 0? Yes for e=1

Show that: a f(n/b) $\leq$ δ f(n) for some δ < 1

1 f(n/3)= (n/3) log (n/3) $\leq$ δ n log n

(n/3) log (n/3) = (n/3) (log n – log 3) < n/3 log n

< (n/3) log n $\leq$ δ n log n for δ = 1/3 < 1

Conclusion: Therefore, T(n) is Θ(n log n)

(Must give conclusion and solve for δ to receive full credit for Case 3!)

Fastest way to decide which case: (fewer mistakes if we try case 2 first, then compare f(n) vs. $n^{\log_b a}$ as follows)

Case 2 applies if there exists a *k* such that f(n) is $\Theta(n^{\log_b a} \log^k n)$;
Case 1 applies if f(n) grows slower than $n^{\log_b a}$ (cannot be the same growth rate which is the reason e must be greater than 0);
Case 3 applies if f(n) grows faster than $n^{\log_b a}$ (cannot be the same rate); then solve for δ.

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 4:

$T(n) = 8\, T(n/2) + n^2$

$\log_2 8 = 3$

Case 2 does not apply since $f(n)= n^2$ is not $\Theta(n^3 \log^k n)$ for any k, so compare growth of $f(n)=n^2$ vs. n^3.

Answer f(n) grows slower, so case 1 applies.

Case 1: Is n^2 $O(n^{3-e})$? yes, for e=1

Therefore, T(n) is $\Theta(n^3)$

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 5:

$T(n) = 9\, T(n/3) + n^3$

$\log_3 9 = 2$

Case 2 does not apply (why?), so compare growth of $f(n)=n^3$ vs. n^2. Answer f(n) grows faster, so case 3 applies.

Case 3: Is n^3 $\Omega(n^{2+e})$? yes, for e=1

Must solve for δ:

$9\,(n/3)^3 \le \delta\, n^3$

$1/3\,(n)^3 \le \delta\, n^3$ which is true for $\delta = 1/3 < 1$

Therefore, T(n) is $\Theta(n^3)$

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 6: Recursive Binary Search and DownHeap

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

$\log_2 1 = 0$

$n^0 = 1$

Case 2: compare f(n)=1 to $\Theta(n^0 \log^k n)$? true for k=0

Therefore, T(n) is $\Theta(\log n)$

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b) \le \delta f(n)$ for some $\delta < 1$.

Example 7: Quick Select

T(n) = T(3n/4) + 2bn

$\log_{4/3} 1 = 0$

Case 2 does not apply, so compare f(n)=2bn to n^0; f(n) grows faster so case 3 applies, so solve for δ.

$2b(3n/4) \le \delta 2bn$ which is true for $\delta = 3/4 < 1$

Therefore, T(n) is $\Theta(n)$.

Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b)\le \delta f(n)$ for some $\delta<1$.

Example 8:
T(n) = T(n/2) + log n
$\log_2 1 = 0$
$n^0 = 1$
Case 2: compare f(n)=log n to $\Theta(n^0 \log^k n)$? true for k=1
Therefore, T(n) is $\Theta(\log^2 n)$

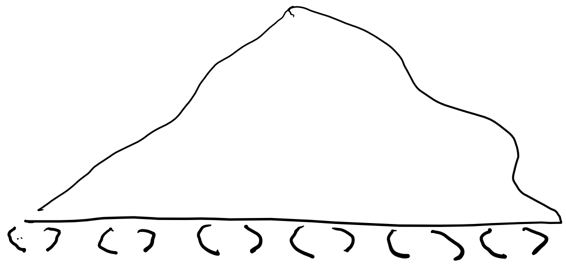
Master Theorem

T(n) = a T(n/b) + f(n)

1. if $f(n)$ is $O(n^{\log_b a-\varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$
2. if $f(n)$ is $\Theta(n^{\log_b a}\log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a}\log^{k+1} n)$
3. if $f(n)$ is $\Omega(n^{\log_b a+\varepsilon})$, then $T(n)$ is $\Theta(f(n))$,
   provided $af(n/b)\le \delta f(n)$ for some $\delta<1$.

Example 9: Heap
T(n) = 2T(n/2) + log n
$\log_b a = 1$
f(n) = log n
$n^{1-.25}$, i.e., e=1/4 > 0
Case 2 does not apply since f(n)=log n is not $\Theta(n^1 \log^k n)$ for any k.
Case 1: n^1 grows faster than log n, i.e. log n is $O(n^{1-.25})$
Therefore, T(n) is $\Theta(n)$ since $n^{\log_b a} = n^1$

(1, 2, 3)
(2, 1, 3)
(3, 1, 2)
(1, 3, 2)
(2, 3, 1)
(3, 2, 1)



```
                    S[0] ≤ S[1]
              /                      \
       S[1] ≤ S[2]                S[1] ≤ S[2]
        /        \                 /        \
 (1, 2, 3)    S[0] ≤ S[1]    (2, 1, 3)    S[0] ≤ S[1]
               /       \                   /       \
        (1, 3, 2)   (2, 3, 1)       (3, 1, 2)   (3, 2, 1)
```