Master's theorem

$T(n) = aT(n/b)+ø(n^k \log^p n)$

Where, a = number of sub problems
n = size of problem
b = size of sub problem
ø = work done outside

$a≥1$, $b≥1$, $k≥0$, p is real number

case1: if $a>b^k$ then $T(n)=ø(b^{\log_b a})$

case2: if $a=b^k$

1. if $p>-1$, then $T(n)=ø(n^{\log_b a} \log^{p+1} n)$
2. if $p=-1$, then $T(n)=ø(n^{\log_b a} \log \log n)$
3. if $p<-1$, then $T(n)=ø(n^{\log_b a})$

case3: $a<b^k$

1. if $p≥0$, then $T(n)=ø(n^k \log^p n)$
2. if $p<0$, then $T(n)=O(n^k)$

Space complexity

Input is not calculated in space.
For iterative: see the amount of extra variables required.
For recursive: see function calls using tree or stack method.

***Tree Method:** Making tree of calls; only to be used with small code
***Stack method**: Making stack and push pop methods; to be used with small code