$T(n) = O(n \log_2 n)$

```
int foo(1)
  int i, j, sum = 0.
  if (n = = 1)
    return 1;
  else {
    for(i = 1; i <= n; i++) {
      for (j = 1; j <= n; j++)
        sum += j; }
    return sum + foo(n-1); }}
```

$O(1)$

$O(n^2)$

MASTER THEOREM $T(1) = O(1)$

$T(n) = O(n^2) + 1T(n-1)$

$= n^2 + T(n-1)$

$n^2 + n-1 + T(n-2)$

$= n^4 + n-1 + T(n-2)$

$n^2 + n-2 + T(n-3)$

$= n^4 + n-2 + T(n-3)$

$= n^2 + 2 + T(n-3)$

$= n^2 + n-1 + n-2 \dots + T(n)$

$= \frac{n^3}{3} + 1$

$T(n) = O(n^3)$

```
int fun2(int n)
    int i, sum=0;
    if (n==1)          → T(1) = O(1)
        return 1;
    else
        return n + fun2(n/2) + fun2(n/2); }
```

$\boxed{\text{MASTER THEOREM}}$

$T(n) = O(n^1) + 2T(n/2)$; $a=2, b=2, k=1$

$b^k = 2^1$

Thus $O(T(n)) = O(n^k \log_b n)$ $\therefore a = b^k$

$= O(n^1 \log_2 n)$

$= O(n \log_2 n)$

$T(1) = O(1)$

$T(n) = O(n) + 2 \cdot T(n/2)$

$= n + 2\left(n/2 + 2 \cdot T(n/2^2)\right)$

$= 2n + 2^2 \cdot T(n/2^2)$

$= 2n + 2^2\left(n/2^2 + 2 \cdot T(n/2^3)\right)$

$= 3n + 2^3 \cdot T(n/2^3)$

$= in + 2^i \cdot T(n/2^i)$

where $n = 2^k$, $k = \log_2 n$

as i approaches k

$T(n) = k \cdot n + 2^k \cdot T\left(\cancel{\frac{2^k}{2^k}}\right) 1$

$= O(n \log_2 n)$

```
int fun3(int n)
    int i, j, sum=0;
    if (n==1)  → O(1)
        return 1;
    else {
        for (i=1; i<=n; i++) → O(n^2)
            for (j=1; j<=n; j++)
                sum += 1; } }
        return sum + fun3(n/2); } }
```

$\boxed{\text{MASTER THEOREM}}$

$T(n) = O(n^2) + T(n/2)$; $a=1, b=2, k=2$

$b^k = 4$

Thus $O(n^k) = O(n^k)$ $\therefore a < b^k$

$T(n) = O(n^2)$

$T(1) = O(1)$

$T(n) = O(n^2) + T(n/2)$

$= n^2 + n^2/2 + T(n/2^2)$

$= \frac{3n^2}{2} + T\left(\frac{n}{2^2}\right)$ $\quad 2^2 = 4$

$2^2 - 1 = 3$

$= \frac{(2^i - 1)n^2}{i} + T\left(\frac{n}{2^i}\right)$ $\quad 2^i - 1 = 3$

$n = 2^k$ as i approaches k

$= \frac{(2^k - 1)n^2}{k} + T\left(\frac{n}{2^k}\right) 1$

$\frac{(n - 1)n^2}{n}$

$T(n) = O(n^2)$