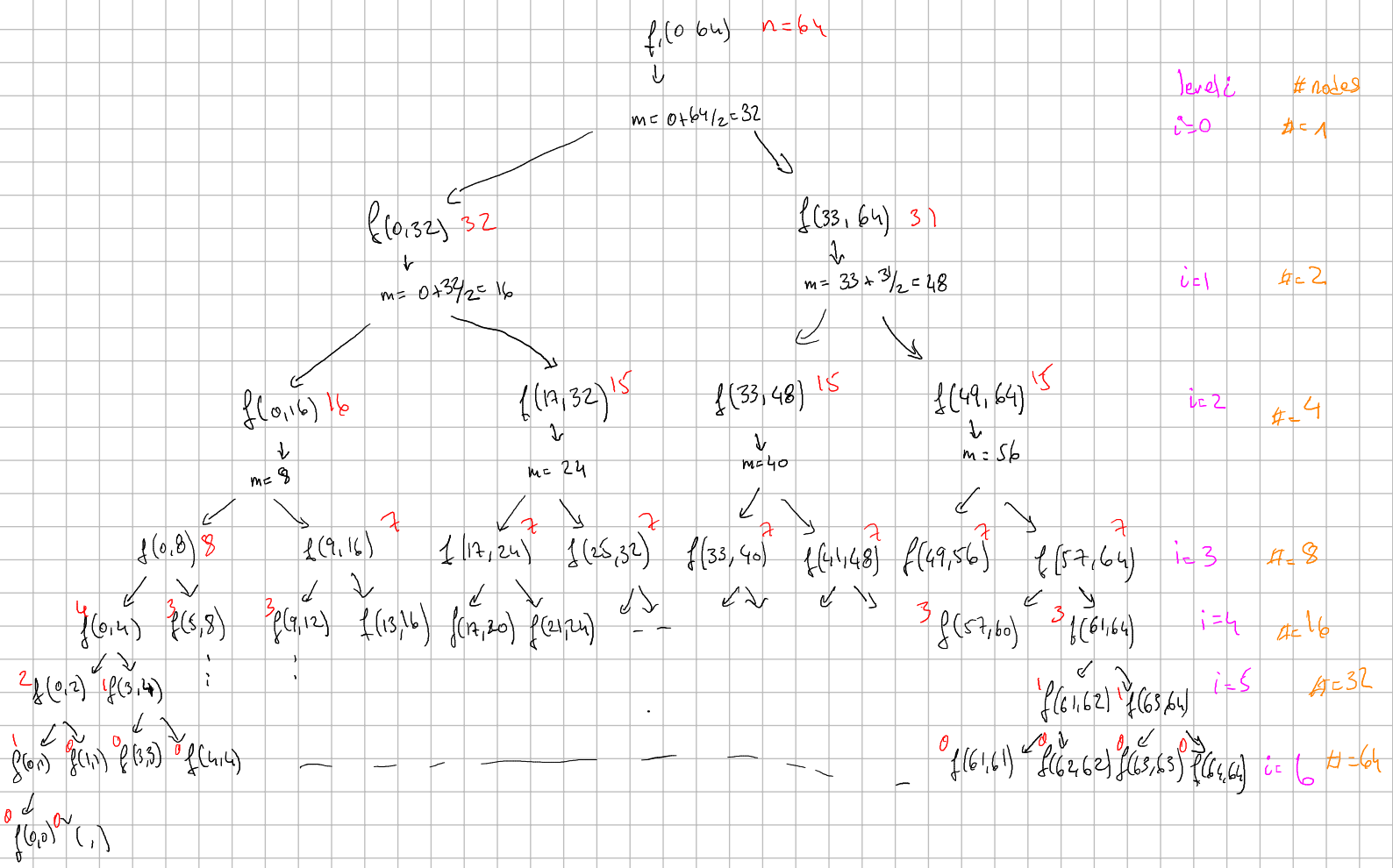
```
// Size of the problem is n = hi - lo.
// You can assume n = 2^p for some integer p.
int f5(int a[], int lo, int hi, int x){
  if(lo >= hi){
    return 0;
  }
  int m = lo + (hi - lo)/2;
  int a = f5(a, lo, m, x);
  int b = f5(a, m+1, hi, x);
  if(a[m] == 0){
    return a + b + 1;
  }else{
    return a + b;
  }
}
```

Just to illustrate, suppose:

$lo = 0$ and $hi = 64$.

we write $f(0,64)$ to mean a call to $f_5(a, 0, 64, x)$ for some a and x.



- There are $(\log_2 n)+2$ levels. The last level (i.e., $i = \log_2 n + 1$) contains 2 nodes. All other levels satisfy # nodes at level i is 2^i (2^0 at first level, 2^1 at second, ...).
- Each node corresponds to an instance of f_5. Each node will have an amount of work between c_1 (just a test and a return) and c_2 (a pair of tests and calls, a return, and a bounded number of arithmetic operations and assignments).
- The amount of work $w(i)$ at level i is therefore bounded by $2^i c_1$ and $2^i c_2$.
- Observe that $\sum_{i=0}^{\log_2 n} 2^i = 2^{(\log_2 n)+1} - 1 = 2n - 1$
- The total amount of work T satisfies: $c_1(2n-1) + k \leq T \leq c_2(2n-1) + k$ where k is the constant amount of work at level $\log_2(n) + 1$ which has 2 nodes.
- The function is therefore in $\Theta(n)$ (both for the best and the worst cases).