(1)

```
void f1(int n){
  int i=2
  while(i<n){
    (O(1) time)
    i=i*i
  }
}
```

after iter 1: $i = 2^2 = 4$

2 $4^2 = 16$

3 $16^2 = 256$

k $2^{2^{k-1}}$

$\log(i) = 2^{k-1}$

$\log(\log(i)) = k-1$

if $i \approx n$ when loop ends

$\log(\log(n)) = k-1$

$k = \log(\log(n)) + 1$

f1: $\boxed{\Theta(\log(\log(n)))}$

(2)

```
void f2(n){
  for(i=1; i<=n; i++)
    if(i%sqrt(n)==0){
      for(int k=0; k<pow(i,3); k++){
        (O(1) time)
      }
    }
}
```

times inner loop runs × max work per iteration

upper bound $\leq \sqrt{n} \times n^3 = n^{3.5}$

$\boxed{\Theta(n^{3.5})}$

outer loop runs $\sqrt{n}$ times

condition met every $\sqrt{n}$

(3) for int i=1, i<=n, i++

for int k=1, k<=n, k++

if A[k]==1

for m=1, m<2=n, m=m+m

O(1) time

- outermost: i=1→n, n iters

second: k=1→n, n iters

if condition: O(1)

innermost loop: increments m=m+m, $\log_2(n)$ iters

m=1 1 iter

2 2

4 3

$O(n \times n \times \log_2(n)^2) = \Theta(n^2 \times \log_2(n)^2)$

(4) - init array O(1)

loop n times

inside if i=size, multiple times depending on n

size=10

i=10, size=15

15 22

22 33

33 49

$size \times (3/2)^k > = n$

$k = \Theta(\log(n))$

size proportionally ↑ to n

$= O(n + n\log(n) \times n)$

$= \Theta(n \log n)$

Part 2

// rec: input: in1 & in2
- linked list pointers

returns: pointer to merged list

if in1 == nullptr return in2
in2 == nullptr return in1

- 1st recursive: in1 = 1, in2 = 5→6
in1→next
= 2→3→4

(a.) = 1→5→6→2→3→4

second recursive: in1 = 2, in2 = 5→6→2→3→4 in1→next
= 2→5→6→2→3→4

final: 1→5→6→2→3→4

b: if in1 = nullptr & in2 =
↳ returns in2
if in2 = 2
returns 2