For the following functions, give the best Big O(   ) descriptions:

```
def foo(n):
    k = 0
    i = n
    while i > 0:
        k += .25**i
        i = i // 2
    return k
```

Answer: O(logn)

```
def bar(n):
    k = 0
    for i in range(n):
        k += 2 + 2*i
    for j in range(n*n):
        k += 10/j
    for m in range(n):
        k += .25*m
    return k
```

Answer: O(n^2)

```
def make_list (n):
    res = []
    for i in range(n):
        val = i**3 - 21
        res.insert(0, val)
    return res
```

Answer: O(n^2)

```
def do_something(n):
    if n <= 0:
        return 42
    return 1.23*do_something(n - 1)
```

Answer: O(n)

```
def do_something2(n, x=0):
    if n <= 0:
        return 0
    for i in range(n):
        x += 2*x + i*1.2
    x += do_something2(n - 1, x//2)
    return x
```

Answer: O(n^2)

```
def do_stuff2(n, x=1.23):
    if n <= 0:
        return 0
    val = 1
    for i in range(n//2):
        for j in range(n//4):
            x += 2*x + j/2 + i*1.2
    while val <= n:
        for i in range(n):
            x += val**2 + i//2
        val *= 2
    x += do_stuff2(n - 1, x/2)
    return x
```

Answer: O(n^3)

```
def do_thing2(n):
    x = 3.25;
    val = 1
    while val <= n:
        for j in range(n):
            x += 2*x + j/2 + val*1.2
        val *= 2
    return x
```

Answer: O(n*logn)

```
def do_thing3(lst):
    res = []
    for i in range(10):
        for j in range(15):
            if i in lst and j in list:
                res.append(i)
                res.append(j)
    return res
```

Answer: O(n)