1. y = 11

Asymptotic runtime: $Boo(n) = O(n^{11})$

```
1. void Foo(int A[]) {
2.     let n = A.size(); ->O(1)
3.     for (i = 1 to n) {  ->O(n)
4b.         if(i==7){
5b.             Boo(i); ->O(n^11)
            }//endif
6.     } //end for
7.     j = 1; ->O(1)
8.     while (j < n) {
9.         k=1; ->O(1)
           while( k<7) ->O(1)
           { print "hello"; k++;}
10.        j=j*y;  j=j*11
11.    } //end while
12. }
```

O(n)

when i == 7
one time operation

+

$O(\log_{11} n)$

hello one time.
stop.

11 times

∴ asymptotic runtime Foo: $O(\log_{11} n) + O(n) + O(1) \ldots$

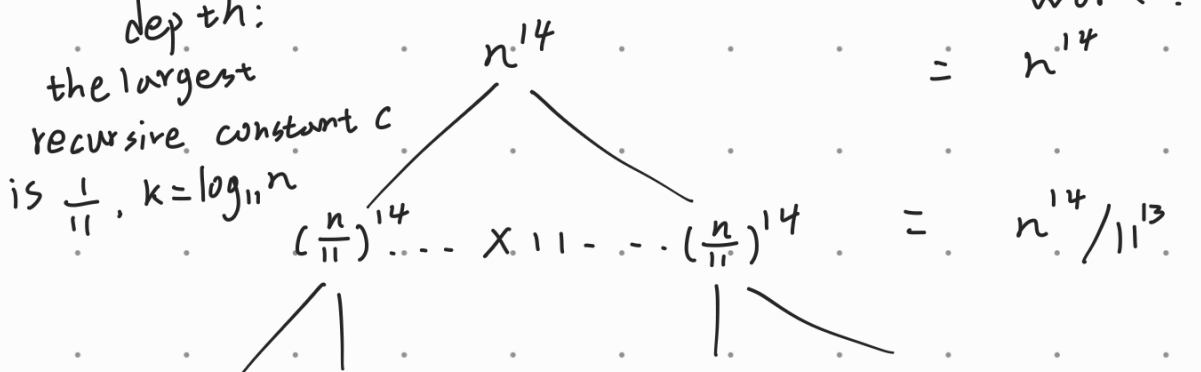
∴ The runtime of Foo is $O(n)$

2. $y = 11$

$$T(n) = 11\,T\left(\frac{n}{11}\right) + n^{14}$$

depth: the largest recursive constant c is $\frac{1}{11}$, $k = \log_{11} n$

work:




$$\sum_{k=0}^{\log_{11} n} \frac{n^{14}}{11^{13k}} = T(n) = n^{14} + \frac{n^{14}}{11^{13}} + \frac{n^{14}}{11^{26}} + \cdots + \frac{n^{14}}{11^{13i}} + \frac{n^{14}}{11^{13\log_{11} n}} + cn$$

$$= \Theta(n^{14})$$

when $n = 1$

$T(n) = n^{14} = 1$ ∴ The close form:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ \Theta(n^{14}) & \text{otherwise.} \end{cases}$$

3.

$y = 11$

```
1. int Foo(int A[]) {
2.    let n = A.size(); if(n==1) return A[1];
3.    for (i = 1 to n) {
44.        print hi
6.    } //end for ....
7.    int x=foo(A[1..n/y]);..
8.    int z=foo(A[1..2n/y]);..
9.    return x+z;
12. }
```

O(1) → O(1)

O(n) → O(n) + T(n) = T(n/11) + T(2n/11) + O(n)

y = 11

Assuming $T(n) \leq cn$

$\therefore T(n) \leq c(\frac{n}{11}) + c(\frac{2n}{11}) + ck$

$= \frac{3}{11}cn + ck$

$\therefore \frac{3}{11}cn + ck \leq cn \quad , c \geq 8$

$\therefore$ proved $T(n) \leq cn$

$\therefore$ The code is $O(n)$

4.

```
Int foo(int arr, first, last):
    if (first == last):
        return arr[first]
    if (last - first == 1):
        return max(arr[first, arr[last])
```

} O(1)

```
Int n = (last - first + 1) / 4
    mid_1 = first + n
    mid_2 = first + 2*n
    mid_3 = first + 3*n
```

} O(1)

```
    a = max(arr, first, mid_1 - 1)
    b = max(arr, mid-1, mid-2-1)
    c = max(arr, mid2, mid3-1)
    d = max(arr, mid-3, last)
return max(a, b, c, d) .→ O(1)
```

} O(nlogn)

b.

```
Int foo(int arr):
    n = length(arr)
    for i in range(n):
        for last in range(i to n):
            print(arr[first])
            :
            arr[last+1]
```