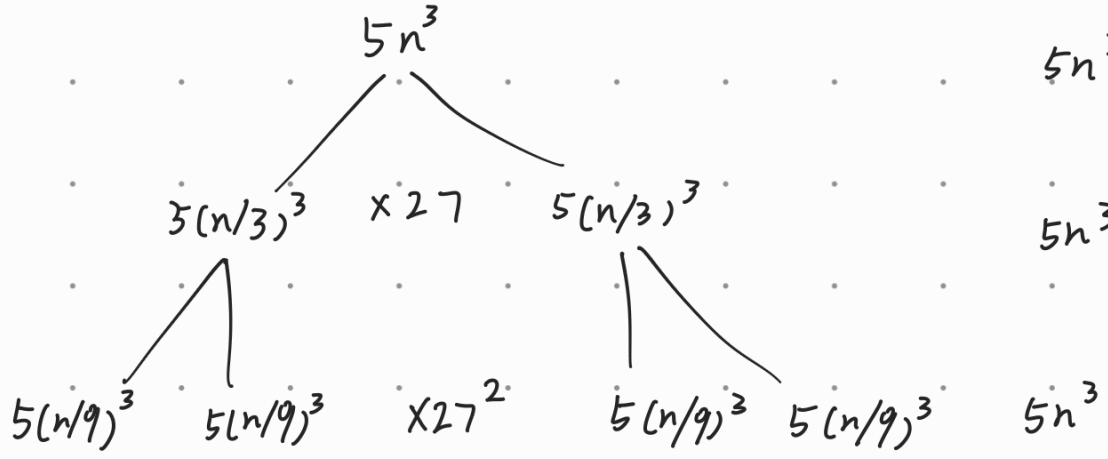
Warm up:

A. $T(n) = 27T(n/3) + 5n^3$



work: $5n^3$, Depth: Largest recursive constant is $\frac{1}{3}$

$\sum_{k=0}^{\log_3(n)} 5n^3 = (1+\log_3(n))(5n^3)$ $k = \log_3(n)$

$= 5n^3\log_3(n) + 5n^3$

$= O(n^3\log(n))$

$\therefore T(n) = O(n^3/\log(n))$

Assuming $T(n) \le cn^3/\log(n)$ for all $n \le k-1$

$\therefore T(\frac{k}{3}) \le c(\frac{k}{3})^3/\log(\frac{k}{3})$

$\therefore T(k) = 27T(\frac{k}{3}) + 5k^3 \le ck^3/\log(\frac{k}{3}) + 5k^3$

$\because ck^3/\log(\frac{k}{3}) + 5k^3 \le ck^3/\log(k)$

$\therefore c\log(\frac{k}{3}) + 5 \le c\log(k)$

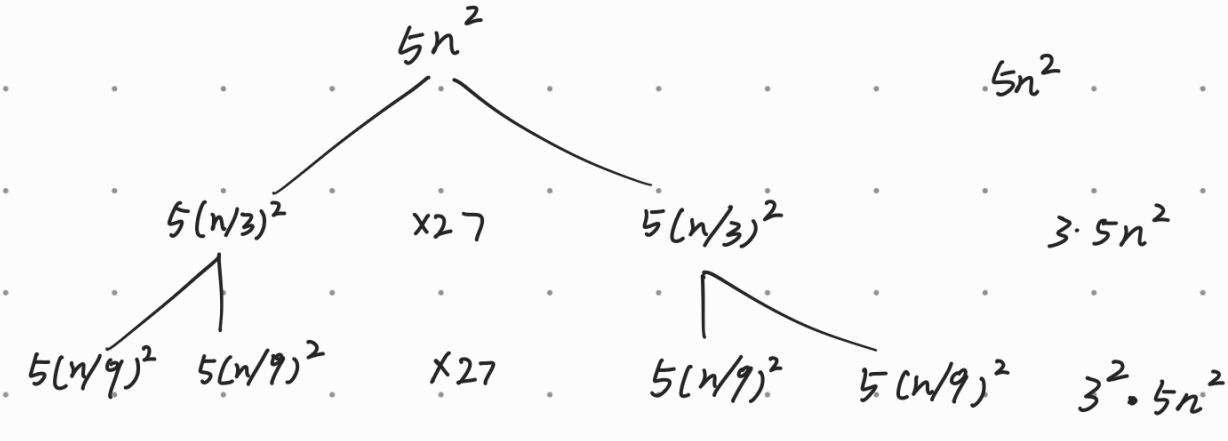
$\therefore c\log(k) - c\log(3) + 5 \le c\log(k)$,

$\therefore C \ge 5$

$\therefore$ For $c = 5$ and $n_0 = 1$, $T(n) = O(n^3/\log(n))$ is proved.

$T(n) \le cn^3/\log(n)$

B. $T(n) = 27T(n/3) + 5n^2$



work: $3^i(5n^2)$

$$\sum_{k=0}^{\log_3(n)} 3^k(5n^2) = 5n^2\frac{(3^{\log_3(n)+1}-1)}{3-1}$$

$$= 5n^2\frac{(3n-1)}{2} = \frac{15}{2}n^3 - \frac{5}{2}n^2$$

$$= O(n^3)$$

depth: largest recursive constant is $\frac{1}{3}$. $k = \log_3(n)$

$T(n) = O(n^3)$, Assuming $T(n) \le cn^3$ for all $n \le k-1$

$\therefore T(\frac{k}{3}) \le \frac{ck^3}{27}$

$\therefore T(k) = 27T(\frac{k}{3}) + 5k^2 \le ck^3 + 5k^2$

$\therefore ck^3 + 5k^2 \le ck^3$

$\therefore 5k^2 \le 0$

$T(n) = O(n^3 - n^2)$ Assuming $T(n) \le cn^3 - cn^2$ for all $n \le k-1$

$\therefore T(\frac{k}{3}) \le \frac{ck^3}{27} - \frac{ck^2}{9}$

$\therefore T(k) = 27T(\frac{k}{3}) + 5k^2 \le ck^3 - 3ck^2 + 5k^2$

$ck^3 - 3ck^2 + 5k^2 \le ck^3 - ck^2$

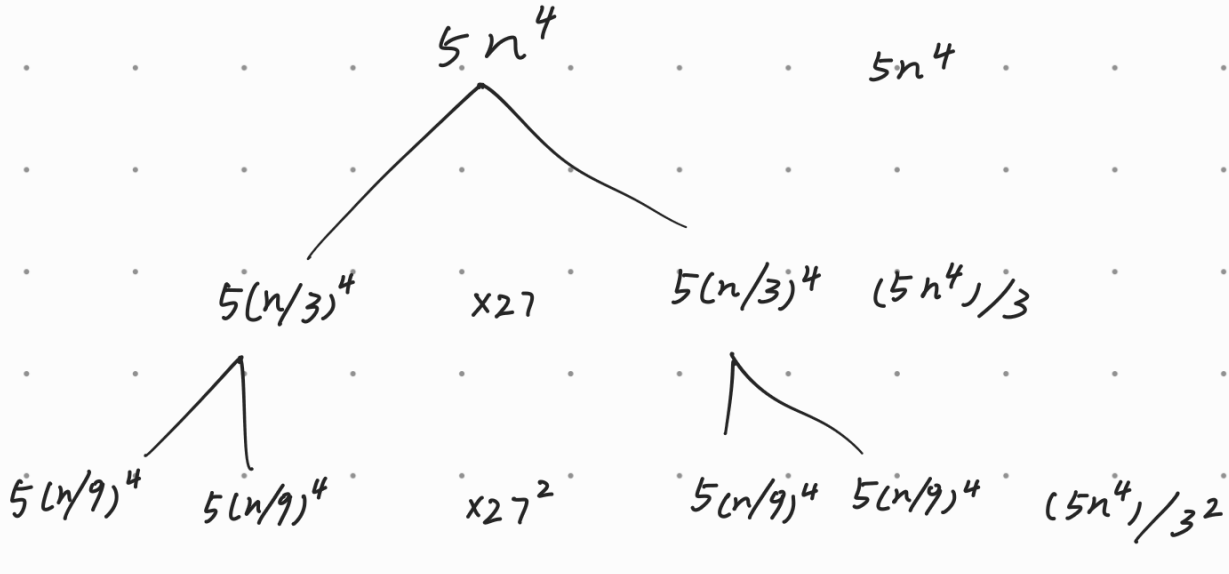
$\therefore 5k^2 \le 2ck^2$

$\therefore c \ge \frac{5}{2}$

$\therefore O(n^3 - n^2) = O(n^3)$

$\therefore$ For $T(n) \le cn^3$ for $c = \frac{5}{2}$ and $n_0 = 1$, $T(n) = O(n^3)$ proved.

C. $T(n) = 27T(n/3) + 5n^4$



work: $\frac{5n^4}{3^i}$

Depth: Largest recursive constant is $\frac{1}{3}$.

$$\sum_{k=0}^{\log_3(n)} \frac{5n^4}{3^k} \leq \sum_{k=0}^{\infty} \frac{5n^4}{3^k} = \frac{5n^4}{1-\frac{1}{3}}$$

$$= \frac{15}{2}n^4 = O(n^4)$$

∴ $T(n) = O(n^4)$ Assuming $T(n) \leq cn^4$ for all $n \leq k-1$

∴ $T(\frac{k}{3}) \leq \frac{ck^4}{81}$

∴ $T(k) = 27T(\frac{k}{3}) + 5k^4 \leq \frac{ck^4}{3} + 5k^4$

∴ $\frac{ck^4}{3} + 5k^4 \leq ck^4$

∴ $\frac{c}{3} + 5 \leq c$

∴ $c \geq \frac{15}{2}$

∴ For $T(n) \leq cn^4$ for $c = \frac{15}{2}$ and $n_0 = 1$

∴ $T(n) = O(n^4)$ proved.

D.

```
Foo arraySum(arr):
  total = 0                          -> O(1)
  for(i=0 to arr.length-1):          } O(n)
    total += arr[i]
  return arrTotal.                   -> O(1)
```

∴ Total runtime: $O(n)$

E.

```
Foo sumUp(arr, start, end):
  if start == end:                   -> O(1)
    return arr[start]
  middle = floor((start+end)/2)      -> O(n/2) -> O(n)
  a = sumUp(arr, start, middle)
  b = sumUp(arr, middle+1, end)
```

∴ $T(n) = 2T(\frac{n}{2}) + O(1)$

return a+b. → O(1) ∴

work: 2^k

Depth: largest recursive constant is $\frac{1}{2}$

$k = \log_2(n)$

$$\therefore \sum_{k=0}^{\log_2(n)} 2^k = \frac{2^{\log_2(n)+1} - 1}{2-1}$$

$$= 2n - 1 = O(n)$$

F.

```
                                    6    20   30
Foo   randomArr(size, min, max):
        array = []
        for(i = 1 to size):
            randomNum = randomInt(min, max)
            randomNum = Append(array)
        return array.

Foo   printArr(arr)
        n = length.arr
        for(i = 0 to n-1):
            for(j = i to n-1):
                print subarrays from index to end.
```

1.

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

Assuming $T(n) \leq cn^2$ for $c > 0$

$\therefore T(n) = T(n-1) + n \leq c(n-1)^2 + n$

$= cn^2 - 2cn + c + n = cn^2 + c(1-2n) + n$

$\therefore \quad cn^2 + c(1-2n) + n \leq 0$

$\therefore \quad c \geq \frac{n}{2n-1}$ for all $n \geq 1$

$\therefore$ For $n_0 = 1$, $c = 1$, proved $T(n) = O(n^2)$

2.

$T(n) = T(\lfloor n/2 \rfloor) + 1$

Assuming $T(n) \leq c \lg n$ for $c > 0$

$T(n) \leq c \lg (n-2)$

$\therefore$ Let $T(n) = T(\lfloor n/2 \rfloor) + 1 \leq c \lg(\lfloor n/2 \rfloor) + 1$

$\therefore \quad T(n) \leq c \lg(\lfloor n/2 \rfloor - 2) + 1$

$\therefore \quad c \lg(\lfloor n/2 \rfloor + 1 - 2) + 1 \geq c \lg(\lfloor n/2 \rfloor - 2) + 1$

$= c \lg((n-2)/2) + 1$

$= c \lg(n-2) - c \lg 2 + 1 \leq c \lg(n-2)$

$\therefore \quad c \lg n - c + 1 \leq c \lg n$

$\therefore \quad c \geq 1$

$\therefore T(n) \leq c \lg n$

$\therefore T(n) = O(\lg n)$ proved.

3.

Assuming $T(n) \le cn^{\log_3 4}$

$T(n) = 4T(n/3) + n$ is $T(n) = \Theta(n^{\log_3 4})$

$T(n) \le 4(c(n/3)^{\log_3 4}) + n = 4c\left(\frac{n^{\log_3 4}}{4}\right) + n$

$= cn^{\log_3 4} + n$

subtract off a lower-order term

assuming $T(n) \le cn^{\log_3 4} - dn$

$\therefore T(n) \le 4(c(n/3)^{\log_3 4} - dn/3) + n$

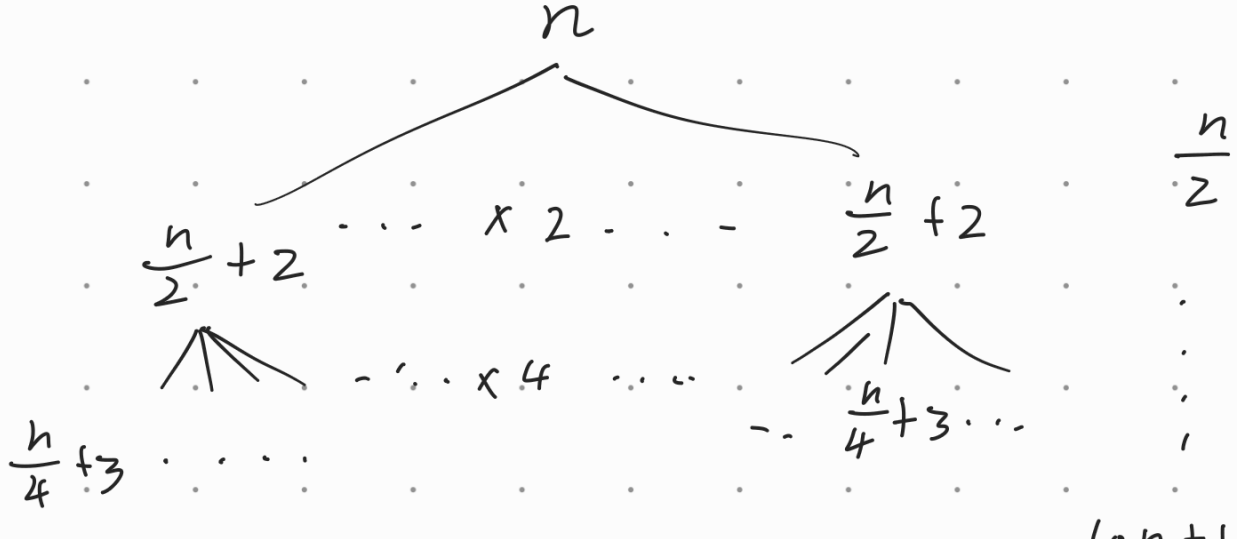
$= 4c\left(\frac{cn^{\log_3 4}}{4} - \frac{dn}{3}\right) + n$

$= cn^{\log_3 4} - \frac{4}{3}dn + n$

$\therefore$ for $d > 3$, $cn^{\log_3 4} - dn \ge cn^{\log_3 4} - \frac{4}{3}dn + n$

4.

$T(n) = 4T(n/2 + 2) + n$



$\therefore i = \log n$, $\log n = \log^{2i}$

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

$T(n) = cn(2) + cn(8) + \cdots + O(n^2)$

$= n\left(\sum_{i=0}^{i=\log n} cn(2^i)\right) + O(n^2)$

∴ depth: $3^i(n/2^i) = (3/2)^i n$

∴ $T(n) \le cn^2 + 2n$

∴ $T(n) \le 4c(n/2)^2 + 2n/2 + n$

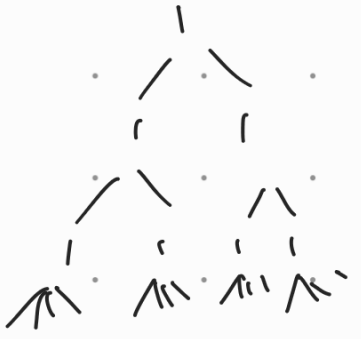
∴ $T(n) \le cn^2 + 2n = O(n^2)$

∴ $T(n) = O(n^2)$

∴ $O(n^2)$.

6.

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



1

1

2

4

⋮

2^{n-1}

∴ $T(n) = \frac{2^n - 1}{2-1} = 2^n - 1 = O(2^n)$

∴ $T(n) = O(2^n)$

when $c > 0$.

∴ $T(n) \le 2(c2^{n-1}) + 1$

$= \frac{2c2^n}{2} + 1 = c2^n + 1$

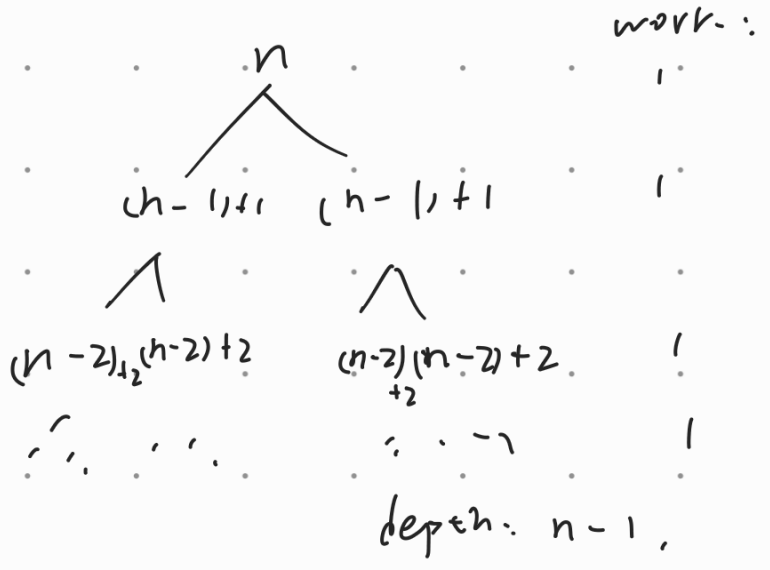
∴ $T(n) \le 2c^n - 1$ for $c > 0$,

∴ $T(n) \le 2(c2^{n-1} - 1) + 1 \le c2^n - 2 + 1 \le c2^n$

∴ $O(2^n)$.

6.

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



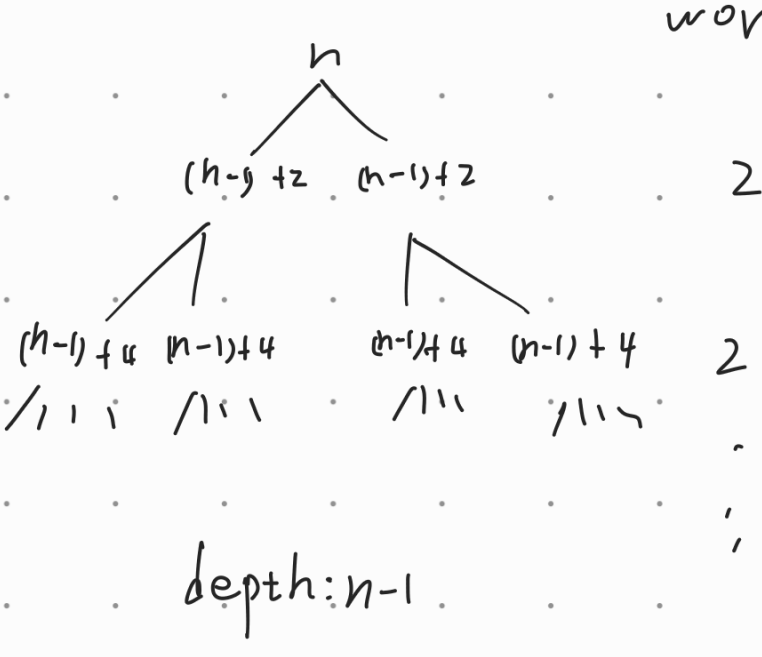
$\therefore c > 0$

$\therefore T(n) = n + c$

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

7.

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



for $c > 0$

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

$= 2n - 2 + c$

$= O(n)$

8.

a. $T(n) = 2T(n/4) + 1$ ↙ $f(n)$

$\therefore \log_4 2 = \frac{1}{\log_2 4} = \frac{1}{2\log_2 2}$

$= \frac{1}{2}$

$\because n^{\log_b a} = n^{\log_a b} = n^{\frac{1}{2}}$

$\because T(n) = 2T(n/4) + 1$

$\therefore T(n) = \Theta(\sqrt{n})$

b.

$T(n) = 2T(n/4) + \sqrt{n}$ ↙ $f(n)$

$\log_4 2 = \frac{1}{\log_2 4} = \frac{1}{2\log_2 2} = \frac{1}{2}$

$\therefore n^{\log_b a} = \sqrt{n}$

$\therefore T(n) = \Theta(\sqrt{n} \lg n)$

c. $T(n) = 2T(n/4) + n$ ↙ $f(n)$

$\therefore \log_b a = \log_4 2 = \frac{1}{2}$, $\varepsilon = \frac{1}{2} > 0$

$n^{\log_b a} < n$

$\therefore T(n) = \Theta(n)$

d. $T(n) = 2T(n/4) + n^2$ ↙ $f(n)$

$\log_a b = \log_4 2 = \frac{1}{\log_2 4} = \frac{1}{2}$

$\because n^{\log_b a} = \frac{1}{2}$, $\varepsilon = \frac{3}{2} > 0$

$\frac{1}{2} < n^2$

$\therefore T(n) = \Theta(n^2)$

9.

$T(n)=4T(n/2)+n^2\lg n$

$a=4,\ b=2,\ f(n)=n^2\lg n \neq O(n^{2-\varepsilon}) \neq \Omega(n^{2-\varepsilon})$

$\therefore T(n)\le cn^2\lg^2 n$

$\therefore\ T(n)\le 4T(n/2)+n^2\lg n$

$=4c(n/2)^2\lg^2(n/2)+n^2\lg n$

$\therefore\ cn^2\lg^2 n+(1-c)n^2\lg n-cn^2\lg(\frac{n}{2})\le cn^2\lg^2 n-cn^2\lg(\frac{n}{2})$

$\therefore,\ cn^2\lg^2 n-cn^2\lg(\frac{n}{2})\le cn^2\lg^2 n.$

$\therefore\ T(n)=O(n^2\lg^2 n)$

10.

a. $T(n)=T(\frac{n}{2})+\theta(1)$

$a=1, b=2, f(n)=1=\theta(n^{\log_2 1})=\theta(0)$

$T(n)=\theta(\lg n)$

$\therefore\ T(n)=T(\frac{n}{2})+\theta(N)=T(\frac{n}{8})+3\theta(N)$

$\sum_{i=0}^{\lg n-1}(i+1)\frac{\theta(n)}{2^i}=\theta(N)\lg n=\theta(N\lg N)$

$\therefore\ T(n)=2T(\frac{n}{2})+\theta(n),\ a=1,\ b=2,\ f(n)=n$

$f(n)=\Omega(n^{\log_2 1+\varepsilon}),\ \varepsilon=1>0$

$\therefore\ af(n/b)\le cf(n)$

$\therefore\ \frac{n}{2}\le cn$

$\therefore\ T(N)=\theta(N),\ \text{for } \frac{1}{2}\le c<1$

b. $T(n) = 2T(\frac{n}{2}) + \Theta(1)$

$a=2, b=2, f(n)=1$

$\therefore f(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_2 2}) = \Theta(1)$

$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(n \lg n)$

$T(n) = 2T(\frac{n}{2}) + cn + 2\Theta(N)$

$= 8T(\frac{n}{8}) + 3cn + 8\Theta(N)$

$$\sum_{i=0}^{\lg n - 1} (i+1)cn + n\sum_{i=0}^{\lg n - 1} 2^i = cn\lg n + nN - N$$

$$= \Theta(N^2)$$

$\therefore T(N) = 2T(\frac{n}{2}) + cn + n = 2T(\frac{n}{2}) + (c+1)n$

$= \Theta(n \lg n)$

11:

a. three ordered list of size $\frac{n}{3}$

```
Foo mergeLists(list1, list2, list3):
    result = []
    i = 0
    j = 0
    k = 0
    n = len(list1) + len(list2) + len(list3)
while i < len(list1) and j < len(list2) and k < len(list3)
    if list1[i] <= list2[j]:
        if list1[i] <= list3[k]
            result.append(list1[i])
            i += 1
```

```
            else:
                result.append(list3[k])
                k += 1
        else if list2[j] <= list3[k]:
            result.append(list2[j])
            j+=1
        else:
            result.append(list3[k])
            k += 1
    result.extend(list1[j:])
    :
    result.extend(list3[k:])
return result.
```

b.

$n=1 \rightarrow O(1)$, $T(1)=0$

$\therefore$ B := mergesort(B, $\frac{n}{3}$)
C := mergesort(C, $\frac{n}{3}$)
D := mergesort(D, $\frac{n}{3}$)
} $T(n)=3T(\frac{n}{3})+f(n)$

$\therefore f(n) \leq 2n$

$\therefore T(n)=3T(\frac{n}{3})+2n$

$\therefore a=3, b=3, f(n)=2n$

$\therefore n^{\log_3 3}=n$

$\because f(n)=2n=\Theta(n)$

$\therefore f(n)=\Theta(n^{\log_b a}\cdot\log n)=\Theta(n\log n)$

$\therefore T(n)=\Theta(n\log n)$.

12.

```
def smallest(arr):
    recurrsive(first, last):
        if last - first == 1:                    O(1)
            return arr[first]
        if last-first == 2:                      O(1)
            return min(arr[first], arr[last+1]),
            max(arr[first], arr[first+1])

    mid = (first + last) / 2                     O(1)
    left_min, left_second = recurrsive(first, last)
    right_min . . . . . .

    compare[0] += 2
        if left_min < right_min:
            return left_min, min(left_second,
                                 left_min)
    result = recurrsive(0, len(arr))
    return result, compare[0]
```

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