Homework 1 : Notation

Solution:

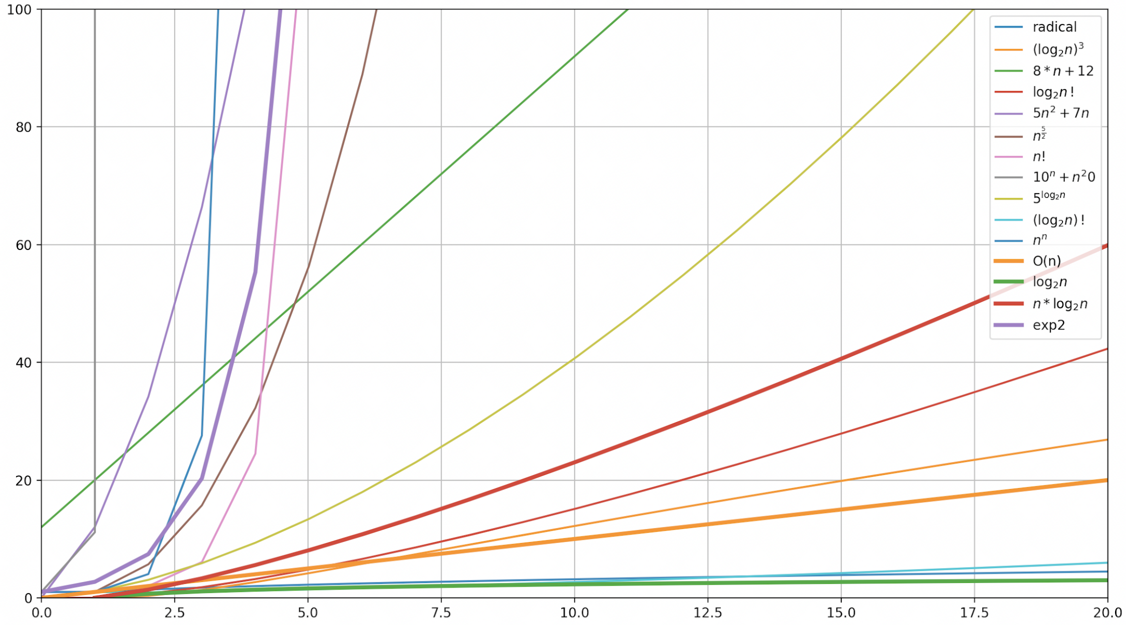
1. quick way to compare growth of two given functions f(x) and g(x) :

$\lim_{x \to \infty} \frac{f(x)}{g(x)} = if\ 0 \leftrightarrow growth\ of\ f(x)\ is\ less\ than\ g(x)\ ,\ \infty \leftrightarrow\ growth\ of\ f(x)\ is\ more\ than\ g($

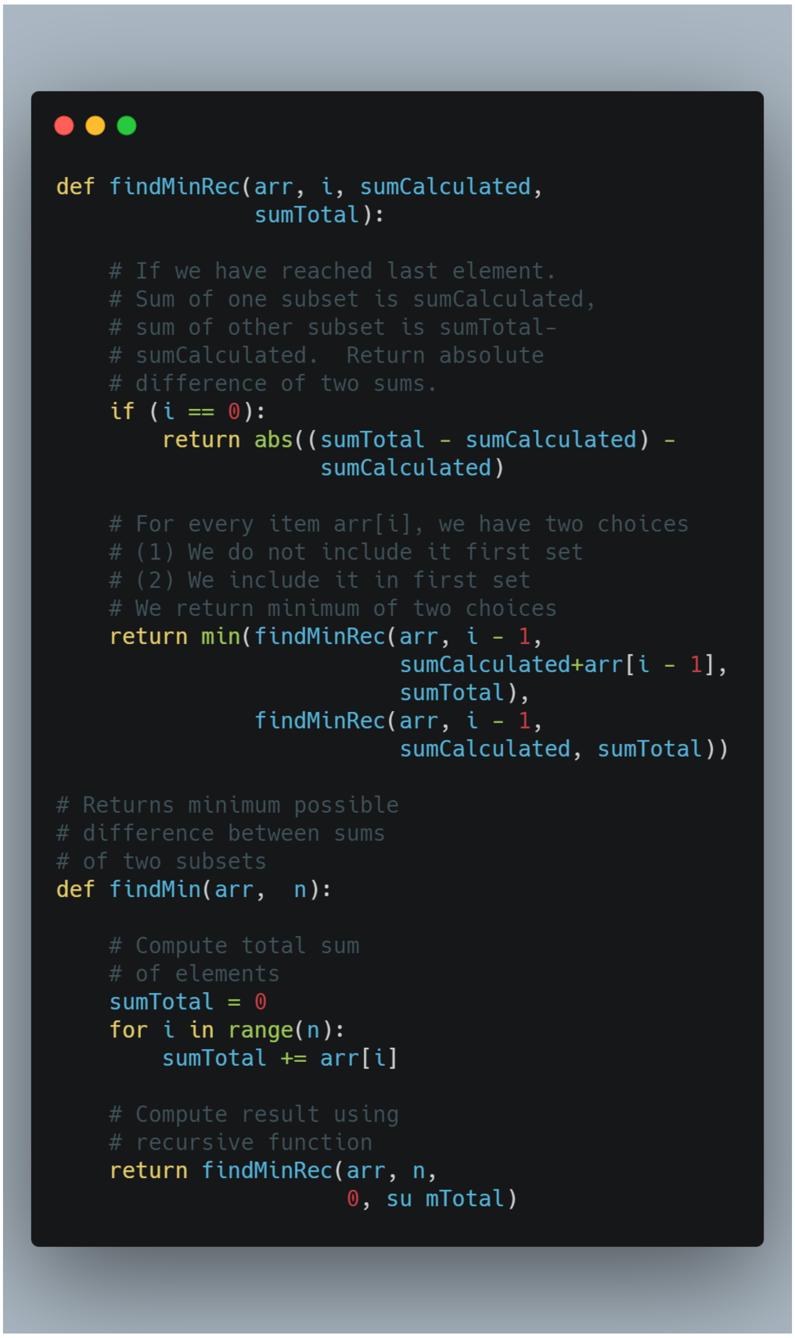
$,\ k\ \neq 0 \leftrightarrow they\ both\ grow\ equally$

- $\lim_{n \to \infty} \frac{log(n)}{n} = 0 \rightarrow log(n) \in O(n)$
- $\lim_{n \to \infty} \frac{n}{nlog(n)} = 0 \rightarrow n \in O(nlog(n))$
- $nlog(n) \leq n^2\ N = 0\ ,\ c = 1,\ n \geq 0$
- $2^n \geq 5^{ln(n)}\ \ N = 0\ ,\ c = 1\ ,\ n \geq 0$

2. $logn < \sqrt{n} < (logn)^3 < 8n + 12 < (logn)! < log(n!) < n.logn < 5n^2 + 7n$
$< n^{\frac{5}{2}} < n^3 < 5^{logn} < e^n 10n + n^{20} < n! < n^n < n^n + ln(n)$



3. The first loop is going from 0 to n , so repetition of “i<n” will be n times . second loop is going to execute for $2^k = n$, so the “j>1” will be executed $logn$ times. In the end the hole code is going to run in $O(nlog(n))$order .
4. Answers:
   a. 25+32 = 57
   b. Each loop is going to execute n times and they’re not nested so the function runs in O(n) order .
   c. We can do the k’s multiplication in the first loop .
5.



```
def findMinRec(arr, i, sumCalculated,
               sumTotal):

    # If we have reached last element.
    # Sum of one subset is sumCalculated,
    # sum of other subset is sumTotal-
    # sumCalculated.  Return absolute
    # difference of two sums.
    if (i == 0):
        return abs((sumTotal - sumCalculated) -
                   sumCalculated)

    # For every item arr[i], we have two choices
    # (1) We do not include it first set
    # (2) We include it in first set
    # We return minimum of two choices
    return min(findMinRec(arr, i - 1,
                          sumCalculated+arr[i - 1],
                          sumTotal),
               findMinRec(arr, i - 1,
                          sumCalculated, sumTotal))

# Returns minimum possible
# difference between sums
# of two subsets
def findMin(arr,  n):

    # Compute total sum
    # of elements
    sumTotal = 0
    for i in range(n):
        sumTotal += arr[i]

    # Compute result using
    # recursive function
    return findMinRec(arr, n,
                      0, su mTotal)
```

6. The first computer will solve the problem of n=1000 size in one minute . the new computer will solve n=1000 size problem in $\frac{1}{1000}$minute , so it can solve a problem of size n = 10^6 in one minute .

    a) T(n)∈$\theta(n)$ n=1000 n = 1000*1000 = 1000000 n* = 10^6
    b) T(n)∈$\theta(n^3)$ n=100 n = 100*100*100 = 1000000 n* = 10^6
    c) T(n)∈$\theta(10^n)$ n=6 n=10^6 10^n = 10^6