SOLUTIONS for HW #1

1. Order the following functions according to their order of growth (from the lowest to the highest). If any two or more are of same order, indicate which.

$f_1(n) = n^2 + \log n$ $\qquad$ $f_8(n) = n^{12} + n^{10}$

$f_2(n) = \sqrt{n}$ $\qquad$ $f_9(n) = n^{12} \log n$

$f_3(n) = n - 1000$ $\qquad$ $f_{10}(n) = n^{\frac{1}{3}} + \log n$

$f_4(n) = n \log n$ $\qquad$ $f_{11}(n) = (\log n)^2$

$f_5(n) = 2^n + n^{10}$ $\qquad$ $f_{12}(n) = 10^{15}$

$f_6(n) = n^5 + 3^n$ $\qquad$ $f_{13}(n) = \dfrac{n}{\log n}$

$f_7(n) = n^{11}.2^{2\log n}$ $\qquad$ $f_{14}(n) = \log\log n$

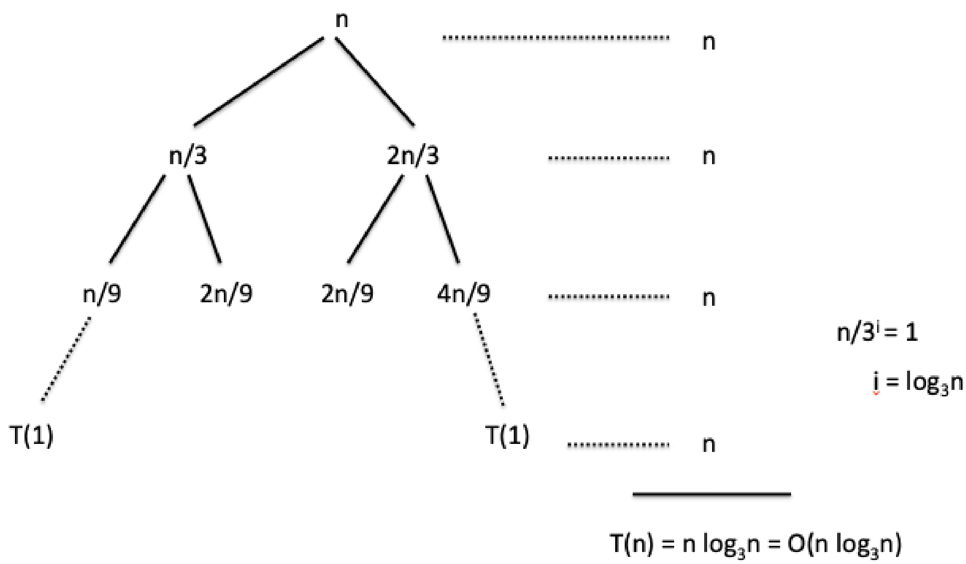
Solution *:*

$f_{12}(n) = 10^{15} = O(1)$, $f_{14}(n) = log\, log\, n = O\,(log\, log\, n)$, $f_{11}(n) = (log\, n)^2 = O(log\,^2 n)$, $f_{10}(n) = n^{\frac{1}{3}} + log\, n = O\left(n^{1/3}\right)$, $f_2(n) = \sqrt{n} = O\left(n^{1/2}\right)$, $f_{13}(n) = \frac{n}{log\, n} = O\left(\frac{n}{log\, n}\right)$, $f_3(n) = n - 1000 = O(n)$, $f_4(n) = n\, log\, n = O(n\, log\, n)$, $f_1(n) = n^2 + log\, n = O(n^2)$, $f_8(n) = n^{12} + n^{10} = O(n^{12})$, $f_9(n) = n^{12}\, log\, n = O(n^{12}\, log\, n)$, $f_7(n) = n^{11}.2^{2\, log\, n} = O(n^{13})$, $f_5(n) = 2^n + n^{10} = O(2^n)$, $f_6(n) = n^5 + 3^n = O(3^n)$

2. Solve the following recurrence relation using recursion tree method.

$$\mathrm{T(n)} = \begin{cases} 1 & , \quad \text{if } \mathrm{n} \leq 2 \\ \mathrm{T}\left(\frac{\mathrm{n}}{3}\right) + \mathrm{T}\left(\frac{2\mathrm{n}}{3}\right) + \mathrm{n}, & \quad \text{if } \mathrm{n} > 2 \end{cases}$$

Solution :



3. What does the following algorithm compute? What is its basic operation? How many times is the basic operation executed? Give the worst-case running time of the algorithm using Big Oh notation.

ALASKA(A $= (\mathbf{a_{ij}})_{\mathbf{nxn}}$**)**
input : an nxn matrix of real numbers
r ← 0
for i = 1 to n-2
    for j = i + 1 to n
        if $a_{ij} \neq a_{ji}$
            return false
return true

Solution : *the algorithm checks whether the given matrix is symmetric or not*

the basic operation $a_{ij} \neq a_{ji}$

how many times the basic operation is executed, $T(n) = \sum_{i=1}^{n-2} \sum_{j=i+1}^{n} 1,$

big-Oh $O(n^2)$

4. Solve the following recurrence relation using Master Theorem.

$$T(n) = \begin{cases} 1 \quad , & \text{if } n \leq 2 \\ 2T\left(\frac{n}{2}\right) + n\log n, & \text{if } n > 2 \end{cases}$$

Solution : $n^{log_a b} = n^{log_2 2} = n, f(n) = n\ logn$ *(second case),* $T(n) = O(nlog^2 n)$

5. What does the following recursive algorithm compute? Set up a recurrence relation for the running time of the algorithm and solve it using backward substitution.

```
SAMSUN(a_i, a_{i+1}, ..., a_j)
input : a sequence of integers
if i = j
   return a_i
else
   mid ← (i + j) / 2
   temp1 ← SAMSUN(a_i, ..., a_mid)
   temp2 ← SAMSUN(a_mid, ..., a_j)
   if temp1 ≤ temp2
       return temp1
   else
       return temp2
```

Solution : *the algorithm finds the minimum of a given sequence*

$$\mathrm{T(n)} = \begin{cases} 1 \quad , & \text{if } \mathrm{n} \leq 1 \\ 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \mathrm{c}, & \text{if } \mathrm{n} > 1 \end{cases}$$

$$T(n) = 2\mathrm{T}\left(\frac{\mathrm{n}}{2}\right) + \mathrm{c} = 2\left(2\mathrm{T}\left(\frac{\mathrm{n}}{4}\right) + \mathrm{c}\right) + c = 2\left(2\left(2\mathrm{T}\left(\frac{\mathrm{n}}{8}\right) + \mathrm{c}\right) + \mathrm{c}\right) + c$$

$$T(n) = 2^i\mathrm{T}\left(\frac{\mathrm{n}}{2^i}\right) + \left(2^i + \cdots + 2^0\right)\mathrm{c}; \textit{for } \frac{\mathrm{n}}{2^i} = 1, i = \log n$$

$$T(n) = 2^{\log n}\mathrm{T}(1) + \left(2^i + \cdots + 2^0\right)\mathrm{c} = n + (2\log n - 1)\, c = O(n)$$