1) increasing order

$$T_2 - T_1 - T_4 - T_5 - T_3 - T_8 - T_6 - T_7$$

Proof:

$T_2 - T_1$

$$\left.\begin{array}{l} T_2(n) = 4\log(\log n) \\ T_1(n) = 3\log n + 3 \end{array}\right\} \lim_{n\to\infty} \frac{4\log(\log n)}{3\log n + 3}$$

$$= \frac{(4\log(\log n))'}{(3\log n + 3)'} = \frac{\frac{4(\log n)'}{\log n} \cdot \cancel{\log e}}{3 \cdot \frac{1}{n} \cdot \cancel{\log e}}$$

$$= \frac{\cancel{4} \cdot \cancel{\frac{1}{n}} \cdot \log e}{\cancel{3} \cdot \log n \cdot \cancel{\frac{1}{n}}}$$

$$= \lim_{n\to\infty} \frac{\log e}{\log n} = \frac{1}{\infty} = 0$$

so $\boxed{T_2(n) \in O(T_1(n))}$

$$\left.\begin{array}{l} T_1(n) = 3\log n + 3 \\ T_4(n) = 2000n + 1 \end{array}\right\} \lim_{n\to\infty} \frac{3\log n + 3}{2000n + 1}$$

$$= \frac{(3\log n + 3)'}{(2000n + 1)'} = \frac{3 \cdot \frac{1}{n} \cdot \log e}{2000}$$

$$= \lim_{n\to\infty} \frac{3\log e}{2000n} = 0$$

So $T_1(n) \in O(T_4(n))$

$$\left.\begin{array}{l} T_4(n) = 2000n + 1 \\ T_5(n) = \left(\frac{n}{6}\right)^2 \end{array}\right\} \lim_{n\to\infty} \frac{2000n + 1}{n^2 \cdot \frac{1}{36}}$$

$$= \lim_{n\to\infty} \frac{2000}{2 \cdot n \cdot \frac{1}{36}} = 0$$

So $T_4(n) \in O(T_5(n))$

2

$$\left.\begin{array}{l} T_5(n) = \left(\frac{n}{6}\right)^2 \\ T_3(n) = n^5 + 8n^4 \end{array}\right\rangle \lim_{n\to\infty} \frac{\left(\frac{n}{6}\right)^2}{n^5 + 8n^4}$$

$$= \lim_{n\to\infty} \frac{\frac{1}{36}}{n^3 + 8n^2} = 0$$

so $T_5(n) \in O(T_3(n))$ //

$$\left.\begin{array}{l} T_3(n) = n^5 + 8n^4 \\ T_8(n) = 2^n + n^3 \end{array}\right\rangle \lim_{n\to\infty} \frac{n^5 + 8n^4}{2^n + n^3}$$

$$= \lim \frac{\cancel{2^n}\left(\frac{n^5}{2^n} + \frac{8n^4}{2^n}\right)}{\cancel{2^n}\left(1 + \frac{n^3}{2^n}\right)} = \lim_{n\to\infty} \frac{\overset{0}{\cancel{\frac{n^5}{2^n}}} + \overset{0}{\cancel{\frac{8n^4}{2^n}}}}{1 + \underset{0}{\cancel{\frac{n^3}{2^n}}}}$$

$$= \frac{0}{1} = 0$$

so $T_3(n) \in O(T_8(n))$ //

CamScanner ile tarandı

$$\left.\begin{aligned} T_8(n) &= 2^n + n^3 \\ T_6(n) &= 3^n + n^2 \end{aligned}\right\} \quad \lim_{n\to\infty} \frac{2^n + n^3}{3^n + n^2}$$

$$= \frac{\left(\frac{2}{3}\right)^n + \frac{n^3}{3^n}}{1 + \frac{n^2}{3^n}} = \frac{\lim_{n\to\infty} \left(\frac{2}{3}\right)^n + \frac{n^3}{3^n}}{\lim_{n\to\infty} 1 + \frac{n^2}{3^n}} = \frac{0}{1}$$

$$= 0$$

So $T_8(n) \in O(T_6(n))$

$$\left.\begin{aligned} T_6(n) &= 3^n + n^2 \\ T_7(n) &= n^n + 1000n \end{aligned}\right\} \quad \lim_{n\to\infty} \frac{3^n + n^2}{n^n + 1000n}$$

$$= \lim_{n\to\infty} \frac{\frac{3^n}{n^n} + n^{2-n}}{1 + 1000n^{1-n}} = \frac{\lim_{n\to\infty} \frac{3^n}{n^n} + n^{2-n}}{\lim_{n\to\infty} 1 + 1000n^{1-n}}$$

$$= \frac{0}{1} = 0$$

So $T_6(n) \in O(T_7(n))$

4

Q2)

a) $f(n) = 99n$ and $g(n) = n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} \Rightarrow \lim_{n\to\infty} \frac{99n}{n} = 99 > 0$$

so $f(n) \in \Theta(g(n))$ //

---

b) $f(n) = 2n^4 + n^2$, $g(n) = (\log n)^6$

$$\lim_{n\to\infty} \frac{2n^4+n^2}{(\log n)^6} = \lim_{n\to\infty} \frac{\sqrt[6]{2n^4+n^2}}{\sqrt[6]{(\log(n))^{\cancel{6}}}}$$

$$= \lim_{n\to\infty} \frac{\cancel{\frac{1}{6}} \cdot (8n^3+2n)^{-5/6}}{\frac{1}{n} \cdot \cancel{\log e}}$$ → we can remove constants

$$\lim_{n\to\infty} \frac{n}{(8n^3+2n)^{\frac{5}{6}}} \xrightarrow{\text{again}} \left(\frac{n}{(8n^3+2n)^{\frac{5}{6}}}\right)'$$

$$= \frac{1}{\frac{5}{6} \cdot (24n^2+2)^{-1/6}} = \lim_{n\to\infty} (24n^2+2)^{1/6}$$

$$= \infty$$

so $f(n) \in \Omega(g(n))$ //

5

CamScanner ile tarandı

c) $\lim_{n\to\infty} \dfrac{\frac{n(n+1)}{2}}{4n+\log n} \rightarrow \dfrac{\frac{2n+1}{2}}{4n+\frac{\log e}{n}}$

$\rightarrow \dfrac{n.(2n+1)}{2.(4n+\log e)} = \dfrac{2n^2+n}{8n+2\log e} = \dfrac{4n+1}{8}$

$\lim_{n\to\infty} \dfrac{4n+1}{8} = \infty$ So $\underline{f(n) \in \Omega(g(n))}$

---

d) $\lim_{n\to\infty} \dfrac{3^n}{5^{\sqrt{n}}} \Rightarrow \lim \dfrac{(3^{\sqrt{n}})^{\sqrt{n}}}{5^{\sqrt{n}}}$

$= \lim \left(\dfrac{3^{\sqrt{n}}}{5}\right)^{\sqrt{n}} = \left(\lim_{n\to\infty} \dfrac{3^{\sqrt{n}}}{5}\right)^{\sqrt{n}} = \infty$

So $f(n) \in \Omega(g(n))$

Q3)

a) This function ~~takes~~ takes an array and its size as arguments.

This function iterates the array in a nested loop checks if there is a value that is more than half of the array if it founds such a value returns it otherwise return -1

b) worst case: Outer loop will go entire array (count > n/2 will not happen)

so $\Theta(n^2)$

! space complexity is $\Theta(1)$ (no creates allocation)

Best case: $\underbrace{x\ x\ x \ldots x}_{n/2+1}\ \underbrace{a\ b\ c \ldots}_{n/2-1}$

The outer loop will go one iteration but inner loop will go all elements only once

So $\Theta(1) . \Theta(n) = \Theta(n)$

Avarage: $O(n^2)$

A

Q4) a) Takes an array and its size.
first, ~~takes~~ takes the maximum element of the array and creats another array that has the max el size

After that travels the array and whenever see a same value it will increment the same location in map array.
so functions checks if there is an element that is more than half of the array. If its found return that value other-wise return -1,

- - -

Worst case : $\Theta(n)$
Best case : $\Theta(n)$ → In all cases there are at least 2 loops that iterates n times

Avarage : $\Theta(n)$

Space complexity : $\Theta(m)$ → it is not depend on any condition for that reason Θ.
m is the maximum element of the array

8

Q5) The performance of Q4 algorithm is better than Q3 algorithm in terms of <u>time</u>

On the other hand the performance of Q3 algorithm is better than Q4 algorithm in terms of <u>space</u>

The strength of the Q3's algorithm is space. When we want to focus on using less memory we choose it. On the other hand the strength of the Q4's algorithm is time. When we want to focus on time efficiency we choose it.

9

CamScanner ile tarandı

Q6)

a) Initialize max = Integer.MinValue

for i=0 to n-1 do:

for j=0 to m-1 do:

if A[i].B[i] > max then:

max = A[i].B[i]

end if

end for

end for

return max

end

// m is size of B
n is size of A

Time complexity:

Worst: $\theta(n) \cdot \theta(m)$

Best: $\theta(n) \cdot \theta(m)$

Avarage: $\theta(n) \cdot \theta(m)$ so $\theta(n \cdot m)$

10

b) Procedure foo(A: array of items, B: array of items)

```
Declare BigArr[m+n]
for i=0 to n-1 do:
    BigArr[i] = A[i]
endfor

for i=0 to m-1 do:
    BigArr[n+i] = B[i]
endfor

for i=0 to m+n-i-1 do:
    isswapped = false
    for j=0 to m+n-1 do:
        if BigArr[j] < BigArr[j+1] then:
            temp = BigArr[j]                //swap
            BigArr[j] = BigArr[j+1]
            BigArr[j+1] = temp
            isswapped = true
        endif
    endfor

    if not isswapped then
        break
    endif
endfor

end procedure return BigArr
```

Worst case : $O(n^2)$

Best case : $O(n)$

Avarage case : $O(n^2)$

C) function foo (a[1:n], index, element)

    for i = n-1 to index -1 do :
        a[i+1] = a[i]
    end for
    a[index -1] = element
end

Worst case : $\Theta(n)$

Best case : $\Theta(n)$

Avarage : $\Theta(n)$

d) function foo (a [1 : n] , index)

```
function foo (a[1:n], index)
    for i = Index-1 to n-1 do:
        a[i] = a[i+1]
    endfor
end
```

worst case : $\Theta(n)$

Best case : $\Theta(n)$

Average case : $\Theta(n)$

Hacı Hasan Saven

1901042704