Part 1

$f(n) \in O(g(n)) \rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ ↗ g(n) f(n) den daha hızlı büyür.

$f(n) \in \Omega(g(n)) \rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ ↗ f(n) g(n) den daha hızlı büyür.

$f(n) \in \Theta(g(n)) \rightarrow 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} = C < \infty$

Birbirine yakın

a-) $f(n) = (n^2-3n)^2$ and $g(n) = 5n^3+n$

$$\lim_{n\to\infty} = \frac{f(n)}{g(n)} = \frac{(n^2-3n)^2}{5n^3+n} = \frac{\infty}{\infty} = \text{L'Hospital}$$

$$\lim_{n\to\infty} \frac{2(n^2-3n)(2n-3)}{15n^2+1} = \frac{4n^3-12n^2-12n^2+36n}{15n^2+1}$$

$$\lim_{n\to\infty} = \frac{4n^3}{15n^2} = \frac{4}{15}n$$

$$= \infty$$

Therefore $f(n) \in \Omega(g(n))$.

b-) $f(n) = n^3$ and $g(n) = \log_2 n^4$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n^3}{\log_2 n^4} \Rightarrow \lim_{n\to\infty} \frac{n^3}{4\log_2 n} - \text{L-Hospital}$$

$$\lim_{n\to\infty} \frac{3n^2}{4 \cdot \frac{1}{n\ln 2}} = \lim_{n\to\infty} \frac{3n^3 \ln_2}{4} = \infty$$

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

c-) $f(n) = 5n.\log_2(4n)$ and $g(n) = n\log_2(5^n)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{5\cancel{n}\log_2(4n)}{\cancel{n}.\log_2(5^n)} = 5\lim_{n\to\infty} \frac{\log_2(4n)}{n\log_2 5} = ?$$

$$= 5\lim_{n\to\infty} \frac{\log_5(4n)}{n} \rightarrow (\text{hospital})$$

$$5\lim_{n\to\infty} \frac{\frac{1}{\ln(5)n}}{1} = 5\lim_{n\to\infty} \frac{1}{\ln(5)n}$$

$$= \lim_{n\to\infty} \frac{5}{\infty} \Rightarrow 0$$

Therefore $f(n) \in o(g(n))$

$x^x > x!$ [illegible]

d-) $f(n) = n^n$ and $g(n) = 10^n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^n}{10^n} = \lim_{n\to\infty} \frac{\log n^n}{\log 10^n}$$

$$\lim_{n\to\infty} \frac{\cancel{n} \log n}{\cancel{n} \log 10} = \lim_{n\to\infty} \frac{\log \infty}{\log 10} = \underline{\underline{\infty}}$$

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

e-) $f(n) = 8n \sqrt[5]{2n}$ and $g(n) = n \cdot \sqrt[3]{n}$

$$\lim_{n\to\infty} = \frac{8\cancel{n} \cdot \sqrt[5]{2n}}{\cancel{n} \cdot \sqrt[3]{n}} = \lim_{n\to\infty} \frac{8 \cdot 2^{1/5} \cdot n^{1/5}}{n^{1/3}} = \lim_{n\to\infty} \frac{8 \cdot 2^{1/5}}{n^{2/15}}$$

$$\lim_{n\to\infty} \frac{8 \cdot 2^{1/5}}{\infty} = \underline{\underline{0}}$$

Therefore $f(n) \in O(g(n))$

① static void method.A (String str-array[]) {

```
    for (int i = 0; i < str-array.lenght; i++)
    str-array[i] = " ";
}
```

a-) O(n)
This code loops through each element in an array and assigns an empty array to the element. Each assigment operation takes constent time. As a result, the worst-case time complexity of method A is O(n) because it loops n times, which is the length of the input array str-array

b-) O(n^2)
In the first for loop method A is called n times to measure the size of the str-array. The time complexi of method A is O(n). Since it is called "n" time. in method B, the total time complexity is O(n^2) Elements are written by calling the second for loop n times to the size of the str-array. O(n^2) dominates O(n). So the time complexity of method is O(n^2)

c-) O(n^4)
In method C, the first loop calls the second loop (J. n times. The second loop loops n times for each With this loop, method B is called n^2 times. Th time complexity of method B is O(n^2). Total time complexity is : O(n^2)$\cdot n^2$ = O(n^4). method C's time complexity is O(n^4).

CamScanner ile tarandı

4-) Time complexity is not defined
An infinite loop occurs in method D. The loop never ends due to the i-- statement. I keeps going to 0 or a negative value. In order for this method to work correctly, the i-- statement must be removed from the loop. If the loop worked properly, the time complexity would be O(n). But in this way the time complexity of method D is not defined.

5-) O(n)
In this method, the elements are checked one by one throughout the loop and it checks whether the element is empty or not. When it finds an empty element, it exits the loop. In the worst case, if there are no empty elements in the array, the loop continues until it checks all elements. So it continues for n. Method E's worst case complexity is O(n)

CamScanner ile tarandı

Part B

a-)
```
Algorithm maxDif(arrayA):
    if leght of arrayA < 2:
        return 0
    max_different = arrayA[leght of A-1] - A[0]
    return max_different
```

The running time of the algorithm is fixed, since the string is sequentiel. there is no need to loop. The size of the array does not matter as the first element is always substracted from the last element in a array.

So the big(O) of this algorithm is $O(1)$
constent time complexity

b-)
```
Algorithm maxDif(arrayA):
    if leaght of arrayA < 2:
        return 0
    max-element = arrayA[0]
    min-element = arrayA[0].
    for i from 1 to length of arrayA-1
        if arrayA[i] > max-element
            max_element = arrayA[i]
        if arrayA[i] < min-element
            min-element = arrayA[i]
    max-dif = max-element - min-element
    return maxdif
```
$\hookrightarrow O(n)$