1) a) Algorithm alg1(L[0...n-1])
if (n==1) return L[0] → Θ(1)
else
tmp = alg1(L[0...n-2]) → T(n-1)
if (tmp <= L[n-1]) return tmp → Θ(1)
else return L[n-1] → Θ(1)

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

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

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

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

$T(n-2) = T(n-3) + 1$

$T(n) = T(n-3) + 1 + 1 + 1$

n item

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

Algorithm 1 and Algorithm 2 are the same time complexity $\Theta(n)$ but algorithm 2 has $\Theta(2n-1)$ time complexity so that it is slower so I choose Algorithm 1.

CamScanner ile tarandı

b) Algorithm alg2 (X[l...r])

if (l==r) return X[l] → θ(1)

else

flr = Floor((l+r)/2) → θ(1)

tmp1 = alg2 (X[l...flr]) → T(n/2)

tmp2 = alg2 (X[flr+1...r]) → T(n/2)

if (tmp1 <= tmp2) return tmp1 → θ(1)

else return tmp2 → θ(1)

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

using Master Theorem

$T(n) = aT(\frac{n}{b}) + f(n)$

$T(1) = c$

$T(n) = 2T(n/2) + 1, n^0$

$a = 2 \cdot b = 2 \quad d = 0$

$T(n) = \theta(n^{\log_2 2})$

$= \theta(n)$ //

CamScanner ile tarandı

2) $P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ at a given point x_0.

The algorithm computes the value of polynomial P at a given point x

BruteForceAlgorithm (P[0,n] , x)

a=0 , i=n , j=1 , k=0

for i to 0 → θ(n)

  power = 1 → θ(1)

  for j to i → θ(n)

    power = power * x → θ(1)

  a = a + P[k] * power → θ(1)

  k++

return a → θ(1)

$T(n) = \theta(n^2)$

we can make time complexity better: n log n or n, I think we can traverse the polynomial array lowest to highest term in one for and we can: multiplier the power with given x then we assign to a, a with plus P[i] multiplier with power then we can return a so time complexity is decreasing from θ(n2) to θ(n).

CamScanner ile tarandı

3)

```
countSubString(str, x, y, n)
    totalcount = 0  → Θ(1)
    countx = 0      → Θ(1)
    for i to n      → Θ(n)
        if(str[i] == x)  → Θ(1)
            countx++;    → Θ(1)
        if(str[i] == y)  → Θ(1)
            totalcount += countx → Θ(1)
    return totalcount   → Θ(1)
```

$T(n) = 2 \cdot \Theta(n)$

CamScanner ile tarandı

4) Def Min Closest Point(P, n)

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

```
min_val = float('inf') → Θ(1)
for i in range(n): → Θ(n)
    for j in range(i+1, n): → Θ(n)
        if(dist(P[i], P[j], k) < min_val): → Θ(1)
            min_val = dist(P[i], P[j], k) → Θ(1)
return min_val
```

dist function implementing according to the k dimension
so I send k only because dist function is given.

CamScanner ile tarandı

5) a)

```
if( max_sum >res
Void FindCluster(Arr1[], a1[], a[], Res-lt1[]){
int max_sum = -999; int i=0; int k=i+2; int result=0; int l;
while (k< size) { → θ(n²)
    l=0; → θ(1)
    Arr = List.of([]); → θ(1)
    max_sum += a[i]; → θ(1)
    Arr[l] = a1[i]; l++; → θ(1)
    for(int j=i+1; j<k; j++){ → θ(n)
        max_sum += a[j];
        Arr[l] = a1[j];
        l++;
    }
    if( max_sum > result ){ result = max_sum; System.Arraycopy(Arr,
        0, Res-lt1, Arr.length,
        Arr.length); }
    k++; → θ(1)
    if(k==size){ → θ(1)
        i++;
        k=i+2;
    }
}
print(res-t); → θ(1)
for(i=0; i< Res-lt1.Length; i++){ → θ(n)
    print(Res-lt1[i]);
}
}
```

when k reach the n i increment one by one so i is increasing every θ(n) time later so θ(n). θ(n) increment, size

θ(n²) = while loop time complexity.

$T(n) = \theta(n^2) \cdot \theta(n) + \theta(n) + \theta(1)$

$\rightarrow T(n) = \theta(n^3)$ //

CamScanner ile tarandı