(a)

```
Algorithm alg1(L[0..n-1])
    if (n==1) return L[0]
    else
        tmp = alg1(L[0..n-2])
        if (tmp<=L[n-1]) return tmp
        else return L[n-1]
```

$O(1)$

$T(n-1)$

$O(1)$

$O(1)$

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

(b)

```
Algorithm alg2(X[l..r])
    if (l==r) return X[l]
    else
        flr = floor((l+r)/2)
        tmp1 = alg2(X[l..flr])
        tmp2 = alg2(X[flr+1..r])
        if (tmp1<=tmp2) return tmp1
        else return tmp2
```

$O(1)$

$O(1)$

$T(n/2)$

$T(n/2)$

$O(1)$

$O(1)$

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

1) As it can be seen in the calculations, the first algorithm is faster than. The second algorithm has more recursion relationships.

---

2) Polynomial(n, x)

```
Polynomial(n, x)
    temp = 1
    result = 0

    for i = 1 to (n+1)              ) Θ(n)
        temp = temp * x             ) Θ(1)
        result = result + i * temp  ) Θ(1)
    return result;                  ) Θ(1)
```

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

The operations that can be done are minimized, it is not possible to find the result without going through all the coefficients. So this is the best time complexity.

---

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

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

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

$T(n-3) = T(n-4) + O(1)$

$\vdots$

$T(1) = T(0) + O(1)$

$T(n) = T(0) + (n-1) \cdot O(1)$

$T(n) = O(n)$

Okan Torun
1801042662

Okan Torun
1801042662

3) bruteForce(letters, start, end)

```
bruteForce(letters, start, end)
    count = 0
    for i = 0 to len(letters)):                  ) Θ(n)
        for j = i+1 to len(letters)              ) O(m)
            if(letters[i] != start)              ) Θ(1)
                break
            else
                if(letters[j] == end)            ) Θ(1)
                    count = count + 1
    return count
```

$T(n) = O(n \cdot m)$

4) bruteForce(arr)

```
min = square(arr[0][0], arr[0][1], arr[1][0], arr[1][1])
for i=0 to len(arr)  )Θ(n)
    for j=i+1 to len(arr)  )n
        result = square(arr[i][0], arr[i][1], arr[j][0], arr[j][1]
        if result<min
            min=result  )Θ(1)
return min
```

$$T(n) = \sum_{i=0}^{n} \left(\sum_{j=i+1}^{n} \Theta(1) \right) \rightarrow \Theta(n)$$

$$\sum_{i=0}^{n} \Theta(n) = \Theta(n^2)$$

Okun Torun
180104,2662

5)

```
mostProfitableCluster(branchName, branchNo)
max1 = 0
max2 = " "

for i = 0 to len(branchName)          ) O(n)
    for j = i to len(branchName)      ) n
        temp_profit = 0               )
        temp_cluster = " "            ) O(1)
        for k = i to j+1
            temp_profit = temp_profit + branchNo[k]        \
            temp_cluster = temp_cluster + branchName[k]     |
            if temp_profit > max1                           |  O(1)
                max1 = temp_profit                          |
                max2 = temp_cluster                        /

Print(max2)
```

$$\sum_{0}^{i} \sum_{i}^{len} \sum_{i}^{j+1} 1 = \Theta(n^3) \Rightarrow T(n) = \Theta(n^3)$$

Okan Toran
8010242662