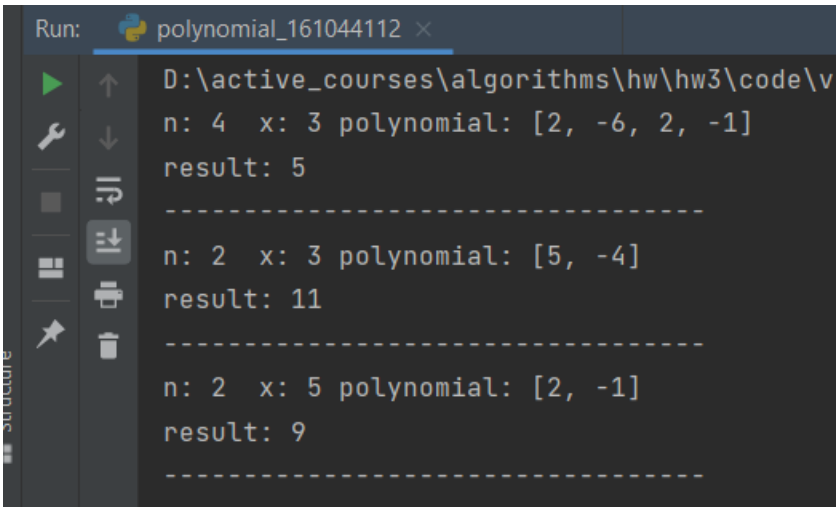
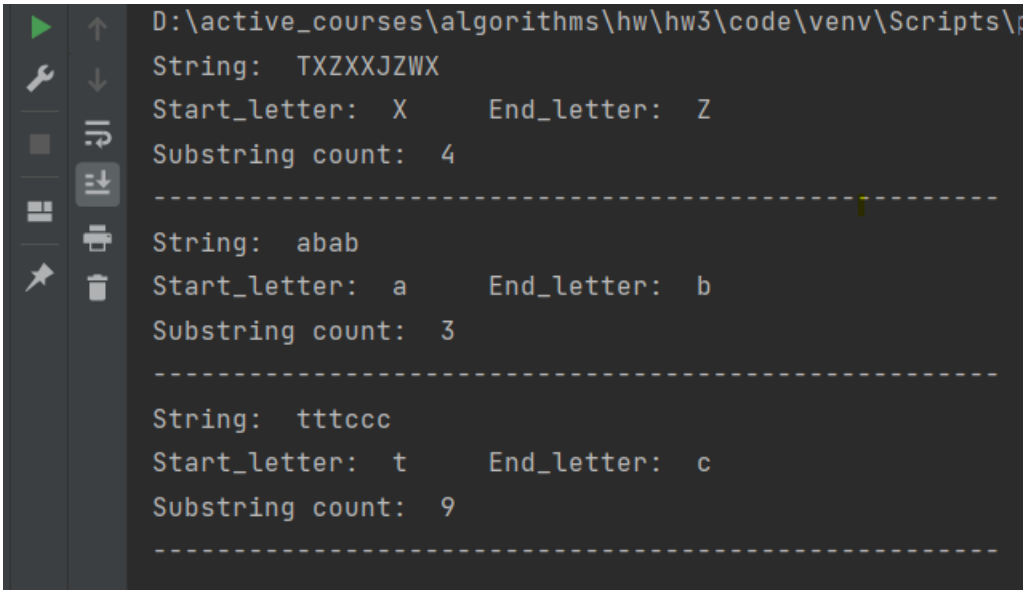
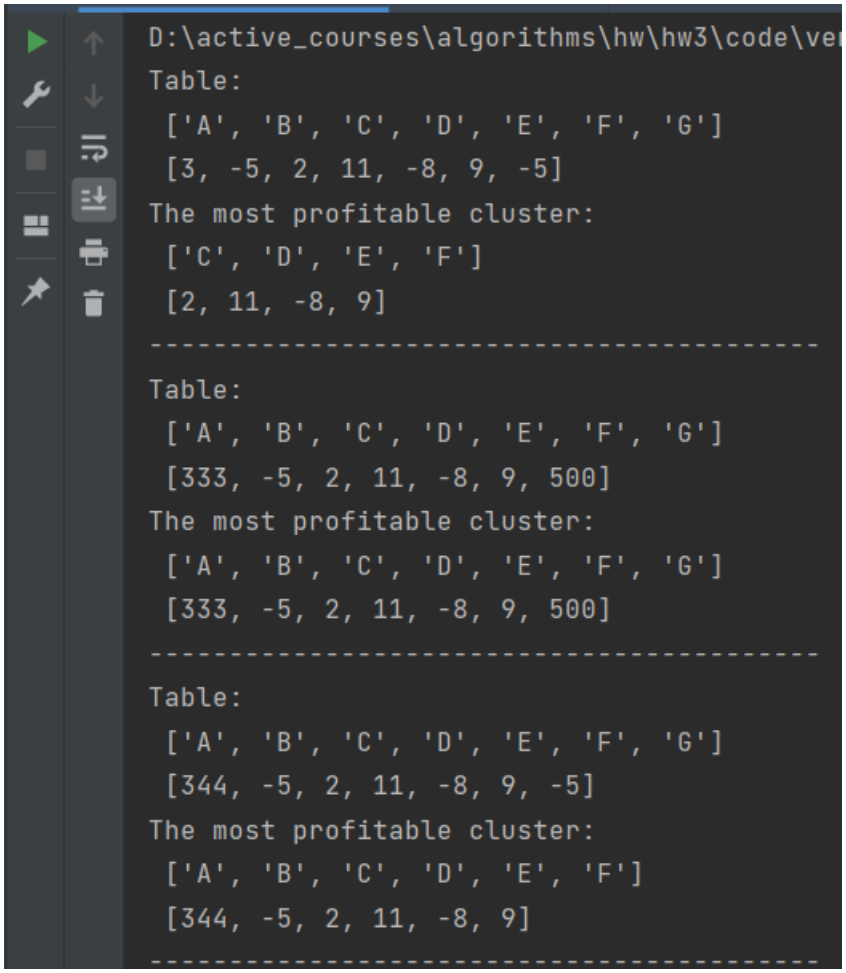
Polynomial(2) test results:



Count_str(3) test results:



Cluster(5-a) test results:



1 a

T(n) ← Algorithm alg 1 (L[0 .. n-1])

2 ← [if (n == 1) return L[0]

else

T(n-1) ← [tmp = alg 1 (L[0 .. n-2])

2 ← if (tmp <= L[n-1]) return tmp

2 ← else return L[n-1]

$$T(n) = \begin{cases} 2 & n = 1 \\ T(n-1)+4 & n > 1 \end{cases}$$

Substitute :

$T(n) = T(n-1) + 4$ — (1)

$= [T(n-2)+4] + 4$ — (2)

$= [[T(n-3)+4]+4]+4$ — (3)

$\hookrightarrow T(n-3) + 3 \cdot 4$

⋮ — (n-1)

$= T(n-(n-1)) + (n-1) \cdot 4$

$= T(1) + (n-1) \cdot 4$

$= 4n - 1 + 2 = \Theta(n)$

[redacted]

(1b)

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

$$T(n) = \begin{cases} 1 & l == r \\ 2\left(\frac{T}{2}\right)+3 & l < r \end{cases}$$

$$T(n) = 2\left(\frac{T}{2}\right) + 3$$

According to case III,

$$T(n) = \Theta(n^{\log_b^a})$$

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

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

$a = 2 \geq 1$ ✓

$b = 2 \geq 2$ ✓

$c = 1 > 0$ ✓

$f(n) = 3 = 3n^d$

$\Rightarrow d = 0$

If $f(n) \in \Theta(n^d)$ where $d \geq 0$ then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \quad \text{case I} \\ \Theta(n^d \log n) & \text{if } a = b^d \quad \text{case II} \\ \Theta(n^{\log_b^a}) & \text{if } a > b^d \quad \text{case III} \end{cases}$$

case III $a > b^d$

$2 > 2^0$

$2 > 1$ ✓

* Complexities of the algorithms are the same. Algorithm b allows parallel processing. But Algorithm a have smaller number of function call. Algorithm b may be choosable.

(2) calc_polynomial(poly, x)

```
1 <-- res <- 0
1 <-- n <- len(poly)
n <-- for i <- 0 to n-1 :
        1 <- product = 1
        i <- [ for j <- 0 to i :
                 product = x * product
        3 <- res = res + poly[n-i-1] * product
1 <- return res
```

$$3 + \sum_{i=0}^{n} \sum_{j=0}^{i} 1 \Rightarrow \sum_{i=0}^{n} i \Rightarrow \frac{n \cdot (n+1)}{2} = O(n^2)$$

If we used Horner's method our complexity will be $O(n)$.

Horner's Rule:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$
$$= a_0 + (x(a_1 + x(a_2 + x(a_3 + \cdots x(a_{n-1} + x a_n))))).$$

By using this rule polynomial will be solved with n multiplication and n addition which serve us $O(n)$ complexity.

Note: Phython implementation is in polynomial_16104112.py

(3)

```
count_substrings(string, start_letter, end_letter):
    counter ← 0
    for i ← 0 to len(string)-1 :                          (n)
        if string[i] != start_letter :                    (1)
            continue
        end if
        for J ← (i+1) to len(string):                     (n-i)
            if string[J] == end_letter :                  (1)
                counter ← counter + 1
            end if
        end for
    end for
    return counter
end procedure
```

$$\sum_{i=0}^{n-1}\sum_{J=i+1}^{n} 1 \Rightarrow \sum_{i=0}^{n-1} n-i \Rightarrow n-0+n-1+n-2 \cdots n-(n-1)$$

$$\Rightarrow n+(n-1)+(n-2)+\cdots+1$$

$$= \frac{n\cdot(n+1)}{2} = O(n^2)$$

(4)

Procedure ClosestPair :

min = inf

for i from (1) to (n-1) :

for J from (i+1) to (n) :

{dist = EuclideanDist (L[i], L[J])

if min > dist :

min = dist

end if

end for

end for

end procedure

(n-1) (n-i+1) 1

$$\sum_{i=0}^{n-1} \sum_{J=i+1}^{n} (k+1) \Rightarrow \sum_{i=0}^{n-1} (k+1)(n-i)$$

$$\Rightarrow (k+1)(1) + (k+1)(2) + (k+1)(3) \cdots (k+1)(n)$$

$$\Rightarrow (k+1)(1+2+3 \cdots n) \Rightarrow \frac{n \cdot (n-1)}{2} \cdot (k+1)$$

$$\Rightarrow \Theta(k \cdot n^2)$$

(5) (a)

procedure find_most_prof(reg_prof_list)

```
max_start = (-1)
max_end = (-1)
g_max = -infinity

for i ← 0 to len(reg_prof_list)
    for J ← (i+1) to len(reg_prof_list)
        sum_sublist = sumlist(reg_prof_list[i:J+1])
        if sum_sublist > g_max
            g_max = sum_sublist
            max_start = i
            max_end = J+1
    end for
end for
return reg_prof_list[max_start : max_end]
end procedure
```

(J-i) (n-i) n 1

$$\sum_{i=0}^{n} \sum_{J=i+1}^{n} J-i \Rightarrow \sum_{i=0}^{n} [(i+1)-i] + [(i+2)-i] + \cdots - [n-i]$$

$$\Rightarrow \sum_{i=0}^{n} 1+2+3+\cdots-(n-i)$$

$$\Rightarrow \sum_{i=0}^{n} \frac{(n-i)\cdot(n-i-1)}{2} \approx \Rightarrow \sum_{i=0}^{n} n^2 \Rightarrow n \cdot n^2 \in \underline{\underline{O(n^3)}}$$