Algorithms and Lab

한국기술교육대학교
KOREATECH

	Lab 03
ID	2020136149
Name	김태섭

2023. 03. 29

1-1.

```
class sumArray:
    def maxSum1(self, arr):
        mx = sum = 0
        n = len(arr)
        for i in range(n):
            for j in range(i,n):
                sum=0
                for k in range(i, j+1):
                    sum += arr[k]


                mx = max(mx, sum)

        return mx
```

```
def testSum1():
    lab03 = sumArray()
    print(lab03.maxSum1([3,-5,4,3,-6,4,8,-5,3,-7,2,1]))
```

Ans)

```
7
```

$$T(n,m) = \Theta(nm) + \sum_{r1=1}^{n}\sum_{r2=1}^{m}\sum_{c1=1}^{n}\sum_{c2=1}^{m} 1 = \Theta(nm) + n^2m^2 = \Theta(nm) + \Theta(n^2m^2) \in \Theta(n^2m^2)$$

1-2.

```
def maxsum2(self, arr):
    mx = 0
    noo= 0
    n = len(arr)
    for i in range(n):
        sm = 0
        for j in range(i, n):
            sm += arr[j]
        mx = max(mx, sm)
    return mx
```

Ans)

$$\begin{cases} T(1) = 1, & n = 1 \\ T(n) = T\left(\frac{n}{2}\right) + 1, & n > 1 \end{cases}$$

1-3.

```
def maxsum3(self, arr, low, high):
    if (low == high):
        return arr[low]
    mid = (low + high) // 2
    return max(self.maxsum3(arr, low, mid),
               self.maxsum3(arr, mid+1, high),
               self.maxCrossingSum(arr, low, mid, high))
```

Ans)

$$\sum_{i=1}^{k} \sum_{j=1}^{i} 1 = \sum_{i=1}^{k} i = \frac{k(k+1)}{2}, (i \leq k)$$

2-1.

```
def powerSet(self, L):
    if len(L) == 0:
        return [[]]
    else:
        base = self.powerSet(L[:-1])
        print('\n pre-recursive, L= ', L)
        operator = L[-1:]
        print('\n post-recursive, L= ', base + [(b + operator) for b in base])
        return base + [(b + operator) for b in base]
```

```
def testPowerSet():
    lab03 = Lab03()
    S1 = [1,2,3]
    L1 = lab03.powerSet(S1)
    print('Power Set = ', L1)

    S2 = ['A', 'B']
    L2 = lab03.PowerSet(S2)
    print('Power Set = ', L2)
```

Output:

```
pre-recursive, L=  [1]

post-recursive, L=  [[], [1]]

pre-recursive, L=  [1, 2]

post-recursive, L=  [[], [1], [2], [1, 2]]

pre-recursive, L=  [1, 2, 3]
pre-recursive, L=  [1]

post-recursive, L=  [[], [1]]

pre-recursive, L=  [1, 2]

post-recursive, L=  [[], [1], [2], [1, 2]]

pre-recursive, L=  [1, 2, 3]

post-recursive, L=  [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]
Power Set =  [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]
```

complexity Function:

$$T(n) \in \Theta(n2^n)$$

2-2.

```
def intersectionSet(self, A, B):
    C = []
    m = len(A)
    n = len(B)
    i, j = 0, 0
    while i < m and j < n:
        if A[i] < B[j]:
            i += 1
        elif B[j] < A[i]:
            j += 1
        else:
            print(B[j])
            C.append(B[j])
            j += 1
            i += 1

    return C
```

```
def testIntersection():
    lab03 = Lab03()
    A = [1,3,4,5,7,9,11,13,15,17]
    B = [2,4,6,8,10,12,14,16]
    C = lab03.intersectionSet(A,B)
    print('Intersection Set = ', C)
```

Output:

```
4
Intersection Set =  [4]
```

complexity Function:

$$T(n, m) = \Theta(n.log_2(n) + \Theta(m.log_2(m)) + \Theta(n + m)$$
$$T(n, m) \in \Theta(s.log_2(s), s = max(n, m)$$

2-3.

```
def binomialCoeffDP(self, n ,k):
    B = [[0 for x in range(k+1)] for x in range(n+1)]
    for i in range(n+1):
        for j in range(min(i, k) + 1):
            if j == 0 or j == i:
                B[i][j] = 1
            else:
                B[i][j] = B[i-1][j-1] + B[i-1][j]

    return B[n][k]
```

```
def testBC():
    lab03 = Lab03()
    print(lab03.binomialCoeffRec(15,5))
    print(lab03.binomialCoeffDP(15,5))
```

Output:

```
3003
3003
```

complexity Function:

$$T(n,k) = T_1(n,k) + T_2(n,k) \in \Theta(nk)$$

2-4.

```
def maxZeroSumSubmatrix(self,matrix):
    (M, N) = (len(matrix), len(matrix[0]))
    S=self.integralMatrix(matrix)
    maxMS=rowStart = rowEnd = colStart = colEnd = 0

    for i in range(M):
        for j in range(i, M):
            for m in range(N):
                for n in range(m, N):
                    ssum = S[j + 1][n + 1] - S[j + 1][m] \
                                - S[i][n + 1] + S[i][m]
                    ms=((j-i)+1)*((n-m)+1)
                    if ssum == 0 and maxMS < ms:
                        maxMS=ms
                        rowStart = i
                        rowEnd = j
                        colStart = m
                        colEnd = n

    A = [ [ matrix[ i ][ j ] for j in range( colStart, colEnd+1 ) ]
                                for i in range(rowStart ,rowEnd+1 ) ]

    print("Submatrix is formed by rows", rowStart, "to", rowEnd,
        "and columns from", colStart, "to", colEnd)
    return A
```

```
def testLSZM():
    lab03 = Lab03()
    matrix = [
        [9,7,16,5],
        [1,-6,-7,-3],
        [1,8,7,-9],
        [7,-2,0,12]
    ]
    for r in matrix:
        print(r)
    Z = lab03.maxzeroSumSubmatrix(matrix)
    for r in Z:
        print(r)
```

Output:

```
[9, 7, 16, 5]
[1, -6, -7, -3]
[1, 8, 7, -9]
[7, -2, 0, 12]
```

complexity Function:

$$T(n,m) = \Theta(nm) + \sum_{r1=1}^{n}\sum_{r2=1}^{m}\sum_{c1=1}^{n}\sum_{c2=1}^{m} 1 = \Theta(nm) + n^2m^2 = \Theta(nm) + \Theta(n^2m^2) \in \Theta(n^2m^2)$$

2-5.

```
def Sm(self, txt, pat):
    m = len(pat)
    n = len(txt)
    for i in range(n-m+1):
        j = 0
        while(j < m):
            if (txt[i + j] != pat[j]):
                break
            j += 1

        if (j == m):
            print('Pattern {} found at index {} ', format (txt[i:i+m], i))
```

```
def testSM():
    lab03 = Lab03()
    txt = 'AABAAABABCCACABAAAAAAAAABBBAAABAA'
    pat = 'AABA'
    lab03.SM(txt, pat)
```

Output:

```
Pattern AABA found at index 0
Pattern AABA found at index 4
Pattern AABA found at index 28
```

complexity Function:

$$T(n,m) = \sum_{i=0}^{n-m} \sum_{j=0}^{m-1} 1 = (n-m+1)m = nm - m^2 + 1 \in \Theta(nm)$$

2-6.

```python
def closest_pair(self,pList):
    cp=[]
    n = len(pList)
    mindist = float("inf")
    for i in range(n-1):
        for j in range(i+1, n):
            dist = self.distance(pList[i], pList[j])
            if dist < mindist:
                mindist = dist
                cp=(pList[i],pList[j])
    return cp
```

```python
def testClosestPair():
    lab03 = Lab03()
    pList = [(2,3), (12,30), (40, 50), (5,1), (12, 10), (4,4), (3,3)]
    cp = lab03.closest_pair(pList)
    print('closest pair [{}, {}] '.format(cp[0], cp[1]))
    print('Distance between closest pair =  {:.2f}   '.format(lab03.distance(cp[0], cp[1])))
```

Output:

```
closest pair [(2, 3), (3, 3)]
Distance between closest pair =  1.00
PS C:\Users\kyung\OneDrive\Desktop>
```

complexity Function:

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1 \\
&= \sum_{i=1}^{n-1} (n-i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i \\
&= n(n-1) - \frac{n(n-1)}{2} \\
&= \frac{n(n-1)}{2} \in \Theta(n^2)
\end{aligned}
$$