assignment #2

박진덕

1. (a)

text $\overbrace{0000 \cdots 0}^{10^6 \text{개}}$

pattern $\overbrace{0000}^{w}\,1$ → scan

$\underbrace{0000}_{\text{match}}$ $\underbrace{1}_{\text{unmatch}}$

for brute-force algorithm, we have to compare all the pattern with shifting windo

since till last word '1', it matches to text.

let pattern length w, because it's worst case for brute-force,

Number of comparison : $\underbrace{w}_{\text{comparison per each scan}} \times \underbrace{(10^6 - w + 1)}_{\text{total shifting number}}$

$= 5 \times (10^6 - 4)$

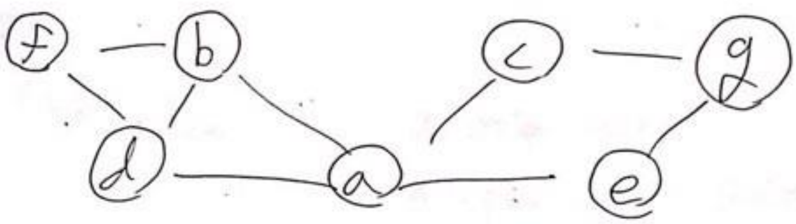
(b) text $\overbrace{0000 \cdots 0}^{10^6 \text{ ea}}$

pattern $0\,1\,0\,1\,0$ → scan

$\underbrace{0}_{\text{match}}$ $\underbrace{1}_{\text{unmatch}}$

In this case, we have to compare till 2nd word for each 5 step, so

Number of comparison : $2 \times (10^6 - 4)$

2. 

(a)

① adjacency matrix

	a	b	c	d	e	f	g
a	0	1	1	1	1	0	0
b	1	0	0	1	0	1	0
c	1	0	0	0	0	0	1
d	1	1	0	0	0	1	0
e	1	0	0	0	0	0	1
f	0	1	0	1	0	0	0
g	0	0	1	0	1	0	0

② adjacency list

a	b c d e
b	d f
c	g
d	f
e	g
f	
g	

$\rightarrow$ [ab, ac, ad, ae
bd, bf
cg
df
eg]

(b) DFS

※ Note) a_{nm}

a : vertex index

n : push order index

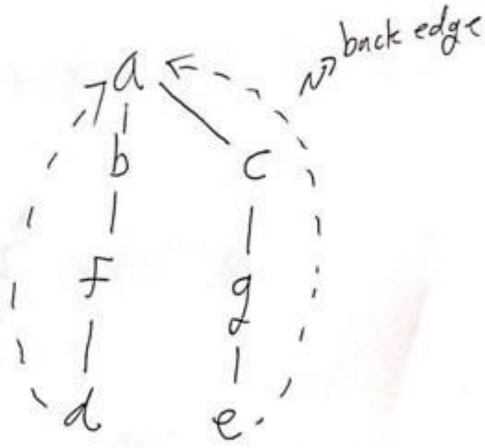
m : pop order index

⟨depth-first search tree⟩

$d_{4}\ 1$
$f_{3}\ 2$
$b_{2}\ 3$
$a_{1}\ 7$

$e_{7}\ 4$
$g_{6}\ 5$
$c_{5}\ 6$

Traversal's stack

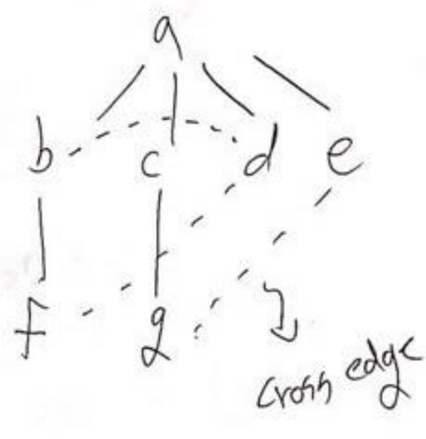


corresponding tree

(c) BFS

⟨breadth-first search tree⟩

$a_1 b_2 c_3 d_4 e_5 f_6 g_7$



3. Strassen formula programming

```python
import numpy as np

# initialize matrix
A = np.array([[1,0,2,1], [4,1,1,0],[0,1,3,0],[2,0,1,1]])
B = np.array([[0,1,0,1], [2,1,1,4],[2,0,1,1],[1,3,5,0]])


#Strassen's algorithm

#divide matrix if len(mat) = 2^n
def StrassenDivide(mat):

    length = len(mat)

    d = length // 2
    dividedmat=[mat[:d, :d], mat[d:, :d], mat[:d, d:], mat[d:, d:]] # Divided matrix

    return dividedmatA


def StrassenProduct(mat1,mat2):

    #Exit recurrence if n == 1
    if (len(mat1)*len(mat2) == 1):

        return (mat1*mat2)

    else:
        #Devide to submatrices
        mat1_set = StrassenDivide(mat1)
        mat2_set = StrassenDivide(mat2)

        #indices for submatrices - A
        a00 = np.array(mat1_set[0])
        a01 = np.array(mat1_set[2])
        a10 = np.array(mat1_set[1])
        a11 = np.array(mat1_set[3])

        #indices for submatrices - B
        b00 = np.array(mat2_set[0])
        b01 = np.array(mat2_set[2])
        b10 = np.array(mat2_set[1])
        b11 = np.array(mat2_set[3])

        #Recursive algorithm for StrassenProduct
        m1 = np.array(StrassenProduct((a00 + a11),(b00 +b11)))
        m2 = np.array(StrassenProduct((a10 + a11), b00))
        m3 = np.array(StrassenProduct(a00,(b01 - b11)))
        m4 = np.array(StrassenProduct(a11, (b10 - b00)))
        m5 = np.array(StrassenProduct((a00 + a01), b11))
        m6 = np.array(StrassenProduct((a10 - a00), (b00 + b01)))
        m7 = np.array(StrassenProduct((a01 - a11), (b10 + b11)))

        #Strassen formula
        mat3_set = [m1 + m4 - m5 + m7, m3 + m5, m2 + m4, m1 + m3 - m2 + m6]

        mat3_set[0] = np.array((m1 + m4 - m5 + m7))
        mat3_set[2] = np.array((m3 + m5))
        mat3_set[1] = np.array((m2 + m4))
        mat3_set[3] = np.array((m1 + m3 - m2 + m6))

        # concatenation just for the matrix form
        concx1 = np.concatenate((mat3_set[0], mat3_set[2]),axis = 1)
        concx2= np.concatenate((mat3_set[1], mat3_set[3]),axis = 1)
        strassenresult= np.concatenate((concx1,concx2),axis = 0)

        return strassenresult
```

Output of Strassen Formula

```
In [3]: A
Out[3]:
array([[1, 0, 2, 1],
       [4, 1, 1, 0],
       [0, 1, 3, 0],
       [2, 0, 1, 1]])

In [4]: B
Out[4]:
array([[0, 1, 0, 1],
       [2, 1, 1, 4],
       [2, 0, 1, 1],
       [1, 3, 5, 0]])

In [5]: StrassenProduct(A,B)
Out[5]:
array([[5, 4, 7, 3],
       [4, 5, 2, 9],
       [8, 1, 4, 7],
       [3, 5, 6, 3]])

In [6]: np.dot(A,B)
Out[6]:
array([[5, 4, 7, 3],
       [4, 5, 2, 9],
       [8, 1, 4, 7],
       [3, 5, 6, 3]])
```

4. recursive relation of strassen's addition

$$T(n) = 7T(n/2) + 18(n/2)^2 , \quad T(1) = 1$$

let $n = 2^k$

$$T(2^k) = 7\,T(2^{k-1}) + 18\,(2^{k-1})^2$$

$$= 7 \cdot (7 \times T(2^{(k-2)}) + 18 \cdot (2^{k-2})^2)) + 18(2^{k-1})^2$$

$$= \ldots$$

$$= 7^k [T(1) + \sum_{j=1}^{k} 18 \times (2^j)/7^j]$$

for $7^k = n^{\log_2 2} = 2^{\log_2 7}$,

$T(n) =$

$$n^{\log_2 7} [T(1) + \sum_{j=1}^{\log_2 n} (2^2/7)^j]$$

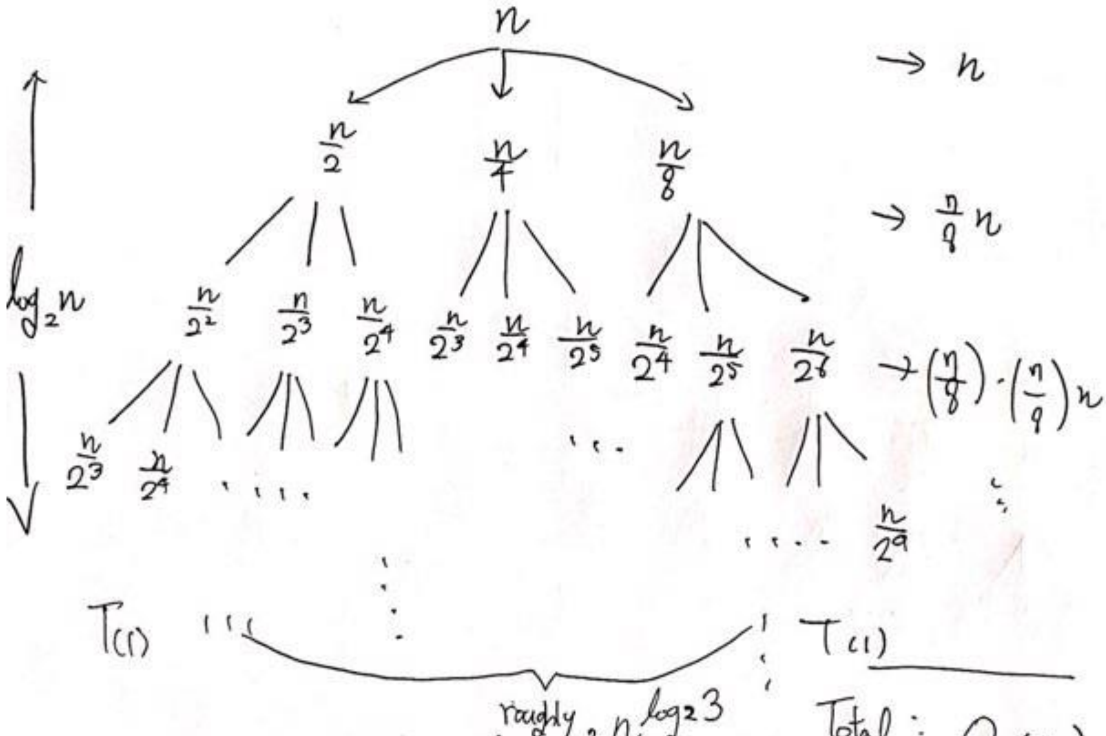
$$= n^{\log_2 7} \left(1 + \frac{+}{\frac{4}{7}-1} (n^{2-\log_2 7} - 1) \right)$$

$$= \frac{10}{3} n^{\log_2 7} - \frac{7}{3} n^2$$

since $\log_2 7 > 2$,

$$T(n) \in \Theta(n^{\log_2 7})$$

5. $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + T(\frac{n}{8}) + n$



Total: $O(n)$

$$T(n) = n \times \left(\frac{1 - \left(\frac{7}{8}\right)^{\log_2 n}}{1 - \frac{7}{8}} \right) + \Theta(n^{\log_2 3}) < \sum_{i=0}^{\infty} \left(\frac{7}{8}\right)^i \cdot n + \Theta = O(n)$$

$$T(n) = \sum_{i=1}^{\log_2 n} n \left(\frac{7}{8}\right)^i + \Theta(n^{\log_2 3}) < \sum_{i=1}^{\infty} n \left(\frac{7}{8}\right)^i + \Theta(n^{\log_2 3}) = \Theta(n^{\log_2 3})$$