알고리즘및실습

[실습 및 과제 4]

KOREA TECH
한국기술교육대학교

과목명/분반	알고리즘및실습/02	담당교수	한연희
학 부 명	컴퓨터공학부	제 출 일	2022/ 06 / 03
학번	2018136121	이름	조원석

INDEX

서론 : Homework의 내용 및 목적

▶ **내용**

: 최소신장트리(minimum_spanning_tree.py)와 최단경로(shortest_path.py)의 구현 및 이해

: 연쇄 행렬 곱의 최적 순서와 비용 계산하기

: 허프만 알고리즘을 통한 최적 이진 트리 구축

▶ **목적**

: prim과 cruscal 알고리즘의 이해

: dijkstra 알고리즘과 Floyd-Warshall 알고리즘의 이해

: 허프만 알고리즘의 이해

본론 : 문제 풀이

[문제1] 최소 비용 신장 트리 알고리즘 구현

코드 설명은 주석으로 대체하겠습니다.

```python
import sys

class Graph:          # 그래프
    def __init__(self, adjacency_list, directed=False):
        self.adjacency_list = adjacency_list    # 인접 리스트
        self.nodes = set()                      # 정점 = 빈 집합
        self.edges = set()                      # 간선 = 빈 집합
        self.num_nodes = 0                      # 정점의 수
        self.num_edges = 0                      # 간선의 수

        if directed:          # 직접 인접 리스트
            for node in adjacency_list:
                for adjacency_node in adjacency_list[node]:
                    weight = adjacency_list[node][adjacency_node]
                    self._add_node_and_edge(node, adjacency_node, weight)
        else:                 # 간접 인접 리스트
            for node in adjacency_list:
                for adjacency_node in adjacency_list[node]:
                    edge_exist_conditions = [
                        (node, adjacency_node, adjacency_list[node][adjacency_node]) in self.edges,
                        (adjacency_node, node, adjacency_list[adjacency_node][node]) in self.edges,
                    ]
                    if any(edge_exist_conditions):
                        assert adjacency_list[node][adjacency_node] == adjacency_list[adjacency_node][node]
                    else:
                        weight = adjacency_list[node][adjacency_node]
                        self._add_node_and_edge(node, adjacency_node, weight)

    def _add_node_and_edge(self, s, d, weight): # 간선과 정점 더하기
        if s not in self.nodes:
            self.nodes.add(s)
            self.num_nodes += 1

        if d not in self.nodes:
            self.nodes.add(d)
            self.num_nodes += 1

        self.edges.add((s, d, weight))
        self.num_edges += 1
```

Union 구현

```python
class DisjointSet:                # 서로소 집합 자료 구조
    def __init__(self, vertices):
        self.vertices = vertices
        self.parent = {}
        self.rank = {}

    def make_set(self):           # 원소 x로만 구성된 집합 만들기
        for v in self.vertices:
            self.parent[v] = v
            self.rank[v] = 0

    def find_set(self, item):     # 원소 x를 가진 집합 알아내기
        if self.parent[item] != item:
            self.parent[item] = self.find_set(self.parent[item])

        return self.parent[item]

    def union(self, x, y):        # 원소 x를 가진 집합과 원소 y를 가진 집합을 하나로 합치기
        xroot = self.find_set(x)          # x' ← Find-Set(x)
        yroot = self.find_set(y)          # y' ← Find-set(y)

        # 이곳에 코딩을 추가하세요. (약 5~7라인)
        if self.rank[xroot] > self.rank[yroot]: # if(rank[x'] > rank[y'])
            self.parent[yroot] = xroot            # then p[y'] ← x'
        else:
            self.parent[xroot] = yroot            # p[x'] ← y'
            if self.rank[xroot] == self.rank[yroot]:    # if(rank[x'] = rank[y'])
                self.rank[yroot] = self.rank[yroot] + 1 # rank[y'] ← rank[y'] + 1
```

최소신장트리 부분

```python
class SpanningTree:                  # 최소 신장 트리
    def __init__(self, graph):
        self.graph = graph
        self.prim_tree = {}
        self.kruskal_tree = []

    def print_solution(self, algorithm=None):
        if algorithm == "prim":
            print("*** Prim Solution ***")
            for v, u in self.prim_tree.items():
                print("{0} - {1}".format(u, v))
        elif algorithm == "kruskal":
            print("*** Kruskal Solution ***")
            for u, v in self.kruskal_tree:
                print("{0} - {1}".format(u, v))
        else:
            raise ValueError()
```

프림알고리즘 구현

```python
def prim(self, start_node='1'): # Prim(G, r)
    S = set() # S <- (공집합) # 트리의 내부 노트 집합
    d = {} # d
    for node in self.graph.nodes: # for each u ∈ V
        d[node] = sys.maxsize #d[u] ← ∞ #
    d[start_node] = 0 #d[r] ← 0
    while len(S) != len(self.graph.nodes): # while ( S ≠ V )
        V_minus_S = self.graph.nodes - S # V - S : 트리의 외부 노드 집합
        # 이곳에 코딩을 추가하세요. (약 8~10라인)
        node = self.extract_min(V_minus_S, d) # u ← extractMin(V-S.d)
        S.add(node)
        for v in graph.adjacency_list[node]:# for each v ∈ L(u) # L(u) 인접 정점 집합
            if v in V_minus_S and graph.adjacency_list[node][v] < d[v]:
                d[v] = graph.adjacency_list[node][v]
                self.prim_tree[v] = node
```

extract_min 구현

```python
def extract_min(self, V_minus_S, d):    # extract_min(Q, d[])
    min = sys.maxsize    # min ← ∞
    selected_node = None # 선택된 노드 ← None
    # 이곳에 코딩을 추가하세요. (약 5~7라인)
    for v in V_minus_S:
        if d[v] < min:
            min = d[v]
            selected_node = v

    return selected_node
```

크루스칼알고리즘 구현

```
def kruskal(self):
    ds = DisjointSet(self.graph.nodes)  # 단 하나의 정점만으로 구성된 n개의 집합을 초기화한다
    ds.make_set()

    i = 0
    e = 0
    # 모든 간선을 가중치의 크기 순으로 정렬하여 배열 A[1...E]에 저장한다
    sorted_edges = sorted(self.graph.edges, key=lambda edge: edge[2])

    while e < self.graph.num_nodes - 1: # while(T의 간선 수 < n - 1)
    # 이곳에 코딩을 추가하세요. (약 9~11라인)
        u_v = sorted_edges.pop(0) # A에서 최소 비용의 간선 (u, v)를 제거한다
        if ds.find_set(u_v[0]) != ds.find_set(u_v[1]):   # if (정점 u와 v가 다른 집합에 속함) then
            # T ← T U {(u,v)}
            # 정점 u와 v가 속한 두 집합을 하나로 합친다.
            ds.union(u_v[0],u_v[1])
            self.kruskal_tree.append((u_v[0],u_v[1]))
            e = e + 1
```

결과 화면

```
[Graph] number of nodes: 7, number of edges: 10
{('2', '5', 14), ('1', '3', 9), ('4', '7', 8), ('3', '6', 12), ('6', '7', 7), ('3', '4', 13), ('3', '5', 5), ('4', '6', 9), ('1', '2', 8), ('1', '4', 11)}
*** Prim Solution ***
1 - 2
1 - 3
1 - 4
3 - 5
7 - 6
4 - 7
*** Kruskal Solution ***
3 - 5
6 - 7
4 - 7
1 - 2
1 - 3
1 - 4

Process finished with exit code 0
```

[문제2] 최단 경로 알고리즘 구현

구현을 실패했습니다.

[문제3] 연쇄 행렬들의 곱 A1×A2×A3×A4×A5를 계산하는 최적 순서와 비용

$$A_1 \times A_2 \times A_3 \times A_4 \times A_5$$

$10\times4 \quad 4\times5 \quad 5\times20 \quad 20\times2 \quad 2\times50$

$p_0=10,\ p_1=4,\ p_2=5,\ p_3=20,\ p_4=2,\ p_5=50$

이라고 할 때,

$A_1 = p_0\times p_1,\ A_2 = p_1\times p_2,\ \cdots\ A_5 = p_4\times p_5$

즉, $A_i = p_{i-1}\times p_i$ (i는 $1\le i\le5$ 인 자연수)

라고 나타낼 수 있다.

따라서 행렬 A_1, A_2, A_3, A_4, A_5 이 구해지는 마지막 행렬 곱셈이 어떻게 이루어지느냐에 따라 다음과 같이 4가지로 나타낼 수 있습니다.

① $A_1(A_2A_3A_4A_5)$

② $(A_1A_2)(A_3A_4A_5)$

③ $(A_1A_2A_3)(A_4A_5)$

④ $(A_1A_2A_3A_4)A_5$

그리고 이 4가지 행렬 곱셈문제는 각각을 나누어 또 다른 행렬 곱셈문제를 가질 수 있습니다.

즉, $(A_1 \cdots A_i)(A_{i+1} \cdots A_5)$ 라고 할 때.

i 개의 행렬 곱셈 문제 $(A_1 \cdots A_i)$와

$5-i$ 개의 행렬 곱셈 문제 $(A_{i+1} \cdots A_5)$를

포함하고 있습니다(i는 $1 \le i \le 4$ 인 자연수).

여기서 행렬 $A_x, \cdots, A_y$ 의 곱 $A_x \cdots A_y$를 계산하는 최소비용을 C_{xy}라고 정의할 때,

$$\boxed{C_{xy} = \begin{cases} 0 & \text{if } x=y \\ \min\limits_{x \le z \le y-1} \{C_{xz} + C_{z+1} + P_{x-1} P_z P_y\} & \text{if } x<y \end{cases}}$$

와 같은 최적 부분구조 관계를 갖습니다.

즉 [부분 1] [부분 2]

가 있다고 할 때 $x<y$의 C_{xy}는

부분 1의 곱 + 부분 2의 곱 + 부분 1과 부분 2의 곱으로

나타낼 수 있습니다.

C: x \ y	1	2	3	4	5
1	0	200	1200		
2	0	0	400	240	
3	0	0	0	200	700
4	0	0	0	0	2000
5	0	0	0	0	0

C_{12} : $\boxed{\underset{C_{11}}{0} + \underset{C_{22}}{0} + 10\times4\times5 = 200}$

C_{13} : $\underset{C_{11}}{0} + \underset{C_{23}}{4\times5\times20} + \underset{P_0 P_1 P_3}{10\times4\times20} = 0 + 400 + 800 = 1200$

: $\underset{C_{12}}{10\times4\times5} + \underset{C_{33}}{0} + \underset{P_0 P_2 P_3}{10\times5\times20}$

$= 200 + 0 + 1000 = 1200 = 1200$

C_{23} : $\underset{C_{22}}{0} + \underset{C_{33}}{0} + \underset{P_1 P_2 P_3}{4\times5\times20} = 200$

C_{24}:

$$\boxed{\underset{C_{22}}{0} + \underset{C_{34}}{5\times 20\times 2} + \underset{P_1P_2P_4}{4\times 5\times 2} = 0 + 200 + 40 = 240}$$

$$\underset{C_{23}}{200} + \underset{C_{44}}{0} + \underset{P_2P_3P_4}{5\times 20\times 2}$$

$$= 200 + 200 = \underline{400 > 240}$$

C_{34}:

$$\boxed{\underset{C_{33}}{0} + \underset{C_{44}}{0} + \underset{P_2P_3P_4}{5\times 20\times 2} = 0 + 0 + 200 = 200}$$

C_{45}:

$$\boxed{0 + 0 + \underset{P_3P_4P_5}{20\times 2\times 50} = 0 + 0 + 2000 = 2000}$$

C_{35}:

$$\underset{C_{33}}{0} + \underset{C_{45}}{2000} + \underset{P_2P_3P_5}{5\times 20\times 50}$$

$$\boxed{\begin{array}{l} 0 + 2000 + 5000 = 7000 \\ \underset{C_{34}}{200} + \underset{C_{55}}{0} + \underset{P_2P_4P_5}{5\times 2\times 50} \\ = 200 + 0 + 500 = \underline{700 < 2000} \end{array}}$$

C

x \ y	1	2	3	4	5
1	0	200	1200	320	
2	0	0	400	240	640
3	0	0	0	200	700
4	0	0	0	0	2000
5	0	0	0	0	0

$$C_{14} : \boxed{\underset{C_{11}}{0} + \underset{C_{24}}{240} + \underset{P_0 P_1 P_4}{10 \times 4 \times 2} = 0 + 240 + 80 = 320}$$

$$: \underset{C_{12}}{200} + \underset{C_{34}}{200} + \underset{P_0 + P_2 + P_4}{10 \times 5 \times 2}$$

$$= 200 + 200 + 100 = \underline{500 > 320}$$

$$: \underset{C_{13}}{1200} + \underset{C_{44}}{0} + \underset{P_0 P_3 P_4}{10 \times 20 \times 2} = \underline{1600 > 320}$$

$$C_{25} : \underset{C_{22}}{0} + \underset{C_{35}}{700} + \underset{P_1 P_2 P_5}{4 \times 5 \times 50} = 1700$$

$$: \underset{C_{23}}{400} + \underset{C_{45}}{2000} + \underset{P_1 P_3 P_5}{4 \times 20 \times 50} = \underline{6400 < 1700}$$

$$\boxed{: \underset{C_{24}}{240} + \underset{C_{55}}{0} + \underset{P_1 P_4 P_5}{4 \times 2 \times 50} = \underline{640 < 1700}}$$

C

$x \backslash y$	1	2	3	4	5
1	0	200	1200	320	1320
2	0	0	400	240	640
3	0	0	0	200	700
4	0	0	0	0	2000
5	0	0	0	0	0

C_{15} : $C_{11} + C_{25} + P_0 P_1 P_5$

$= 0 + 2000 + 10 \times 4 \times 50 = \underline{4000}$

$C_{12} + C_{35} + P_0 P_2 P_5$

$= 200 + 700 + 10 \times 5 \times 50$

$= 200 + 700 + 2500 = \underline{3400 < 4000}$

$C_{13} + C_{45} + P_0 P_3 P_5$

$= 1200 + 2000 + 10 \times 20 \times 50 = \underline{4200 > 3400}$

$$\boxed{\begin{aligned} & C_{14} + C_{55} + P_0 P_4 P_5 \\ & = 320 + 0 + 10 \times 2 \times 50 = \underline{1320 < 3400} \end{aligned}}$$

C

$x \backslash y$	1	2	3	4	5
1	0	200	1200	320	1320
2	0	0	400	240	640
3	0	0	0	200	700
4	0	0	0	0	2000
5	0	0	0	0	0

A) C_{15}의 최적 순서는

$(A_1A_2A_3A_4)A_5$

B) C_{14}의 최적순서는

$A_1(A_2A_3A_4)$

C) C_{24}의 최적순서는

$A_2(A_3A_4)$

와 같은 식으로 이전 과정에서 도출할 수 있고,

따라서 A), B), C)를 합치면,

$\boxed{((A_1(A_2(A_3A_4)))A_5)}$의 최적순서를 가지고,

최적 비용은 1320이 됩니다.

검증결과

```
#include <cstdio>
#include <limits>
#define min(x,y) ((x)<(y)?(x):(y))

int M[555][2];
int D[555][555];
int main(){
    int N;
    scanf("%d", &N);
    for (int i = 1; i <= N; ++i)
        scanf("%d %d", &M[i][0], &M[i][1]);
    for (int i = N; i > 0; --i)
    {
        for (int j = i + 1; j <= N; ++j)
        {
            D[i][j] = std::numeric_limits<int>::max();
            for (int k = i; k <= j; ++k)
                D[i][j] = min(D[i][j], D[i][k] + D[k+1][j] + (M[i][0] * M[k][1] * M[j][1]) );

        }

    }
    printf("%d", D[1][N]);

}
```

Success #stdin #stdout 0s 5528KB

stdin

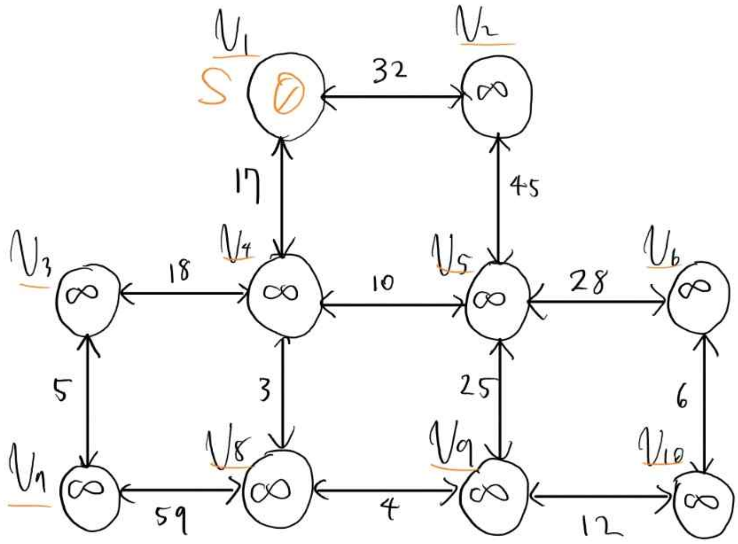
```
5
10 4
4 5
5 20
20 2
2 50
```

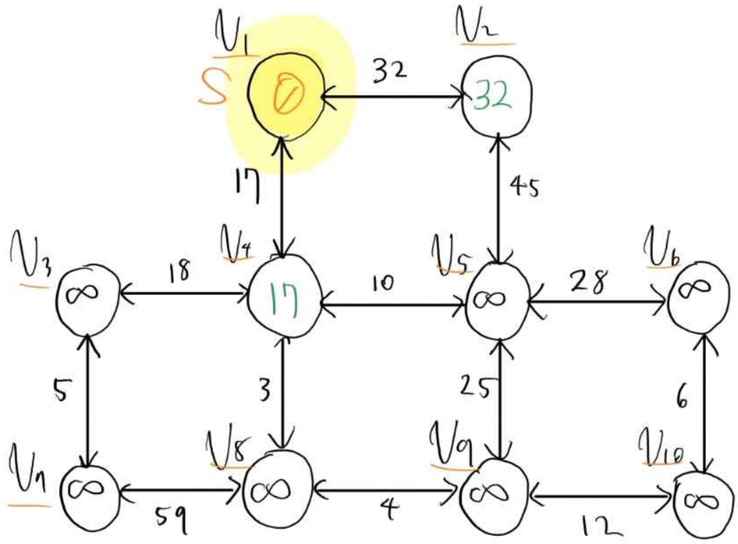
stdout

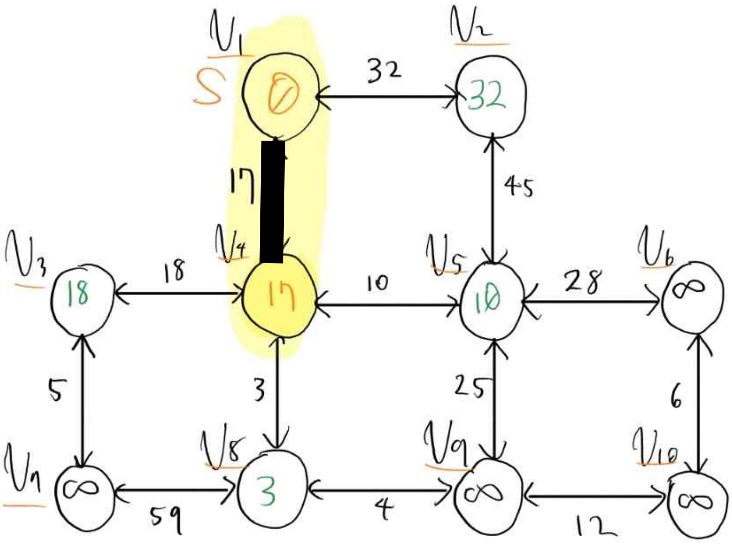
```
1320
```

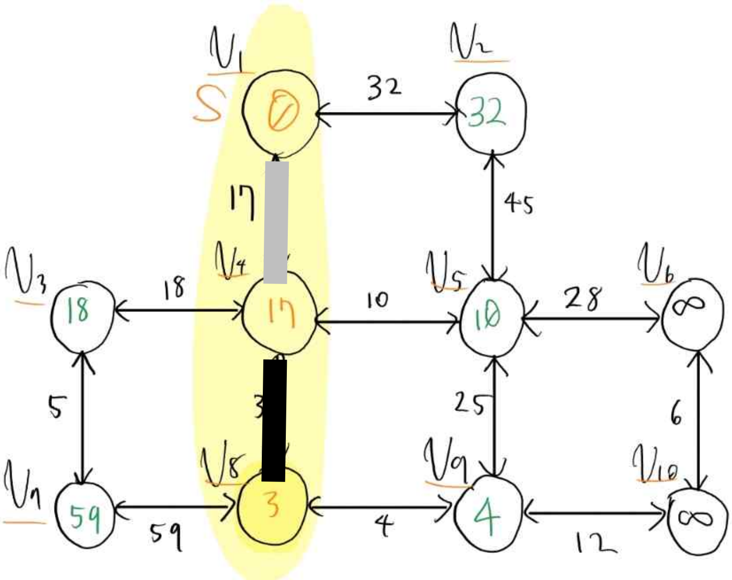
[문제4] 제시문제1

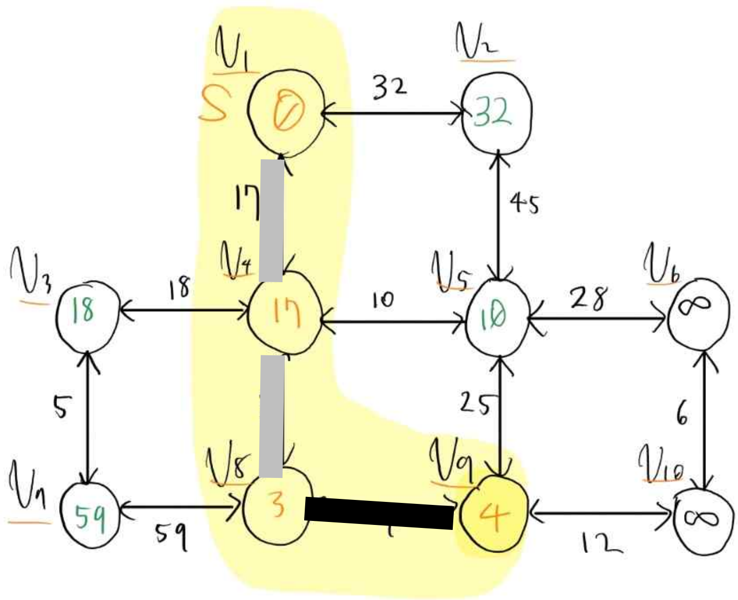
[4-1] Prim 알고리즘을 이용하여 위 그래프의 최소비용 신장 트리를 구하는 과정을 교재 [그림 10-16]과 유사한 형태로 그려 제시(캡쳐 이미지 포함)

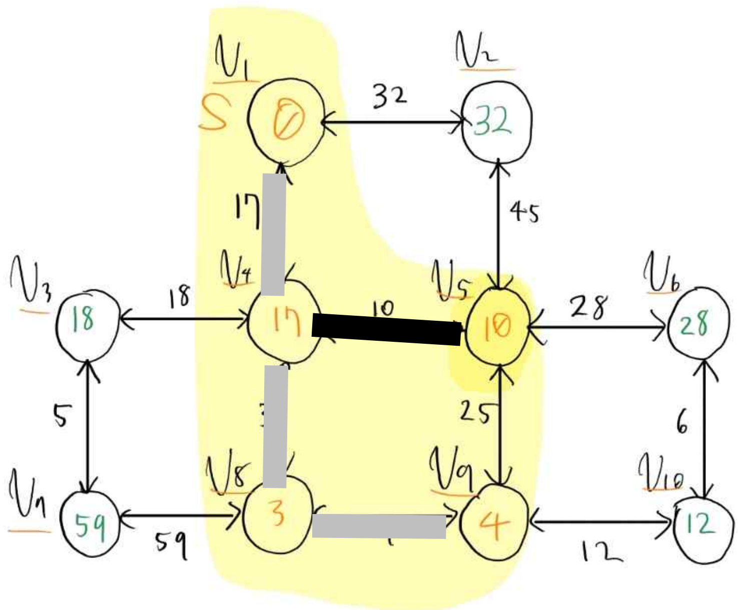


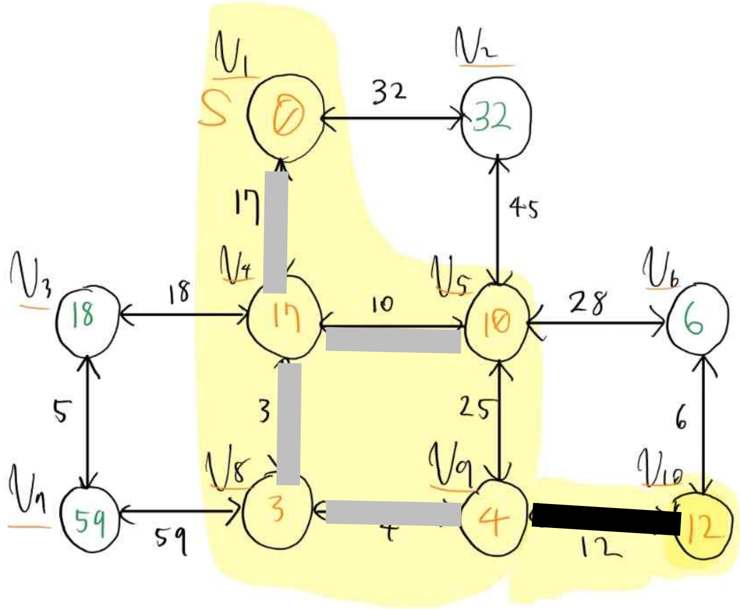


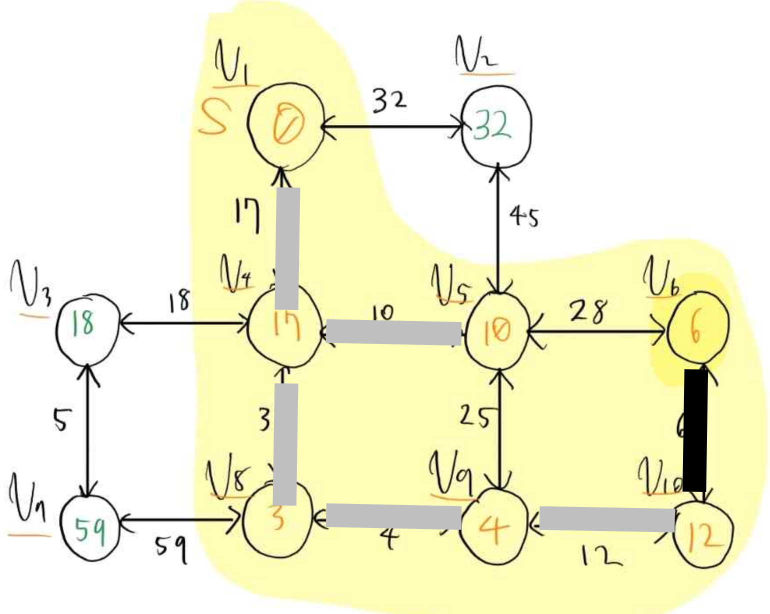

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
∞
5
3
25
6
V7
∞
59
V8
3
4
V9
∞
V10
12
∞

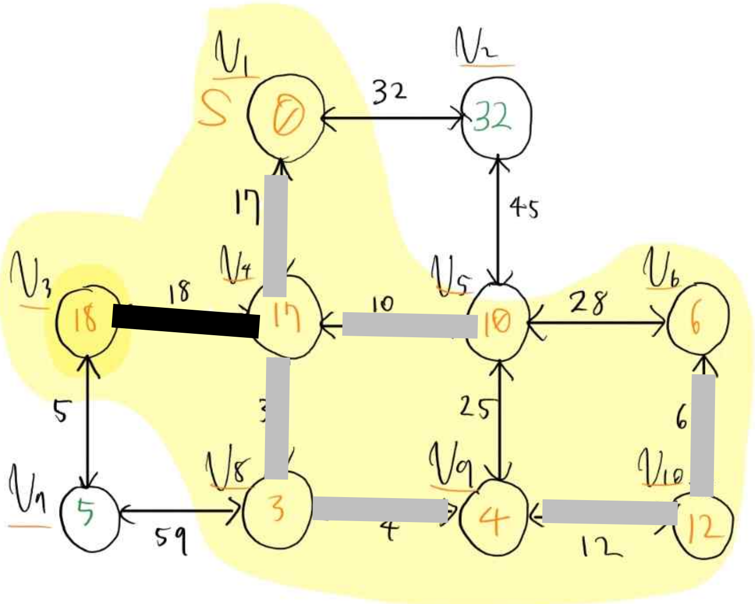

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
∞
5
25
6
V7
59
59
V8
3
4
V9
4
V10
12
∞

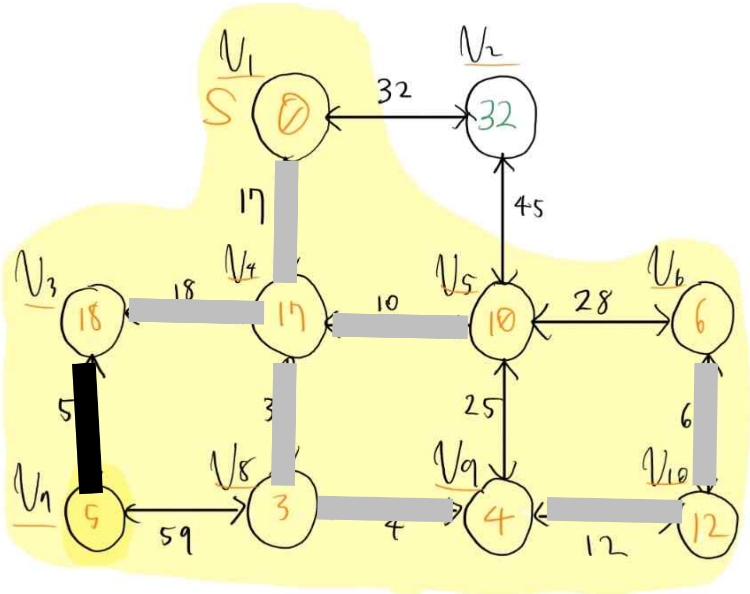

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
∞
5
25
6
V7
59
59
V8
3
V9
4
V10
∞
12

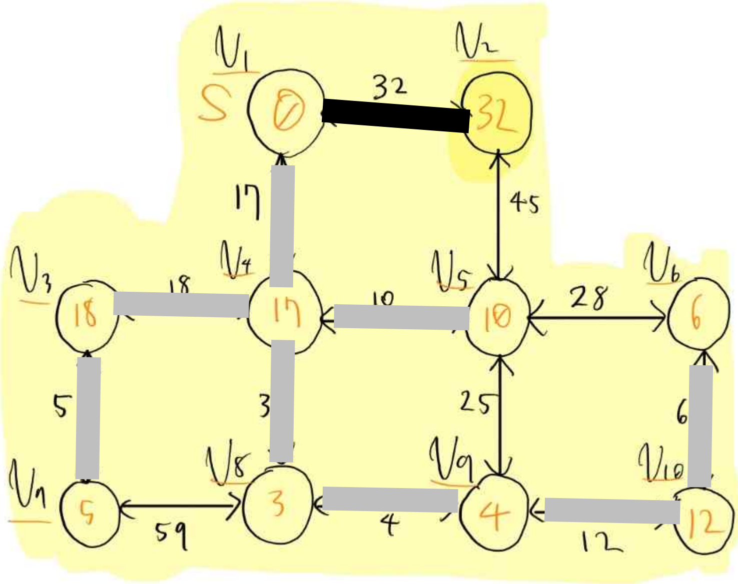

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
28
5
25
6
V7
59
59
V8
3
V9
4
V10
12
12

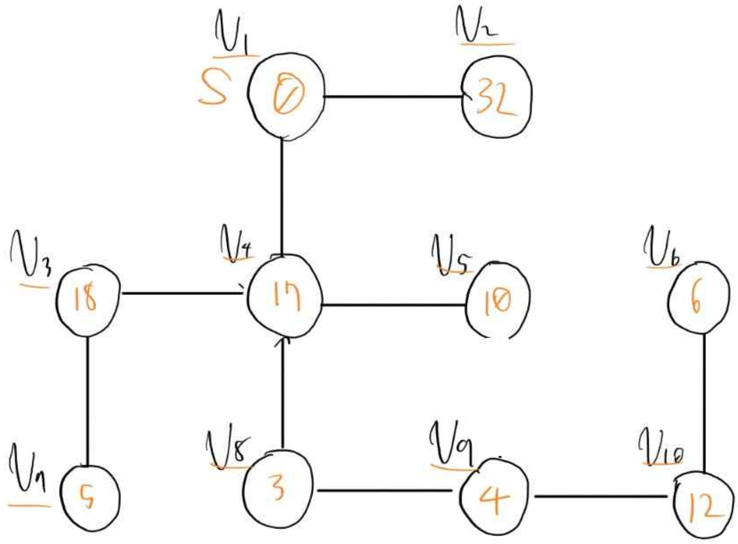

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
6
5
3
25
6
V7
59
59
V8
3
4
V9
4
V10
12
12


V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
6
5
3
25
V7
59
59
V8
3
4
V9
4
V10
12
12

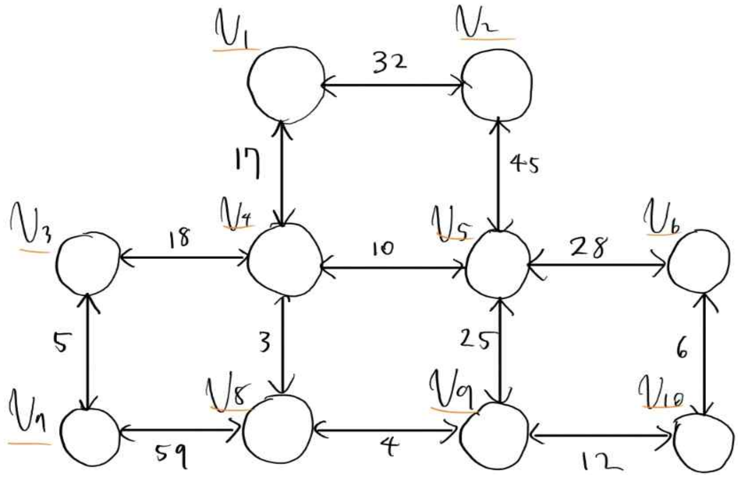

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
6
5
3
25
6
V7
5
59
V8
3
4
V9
4
12
V10
12

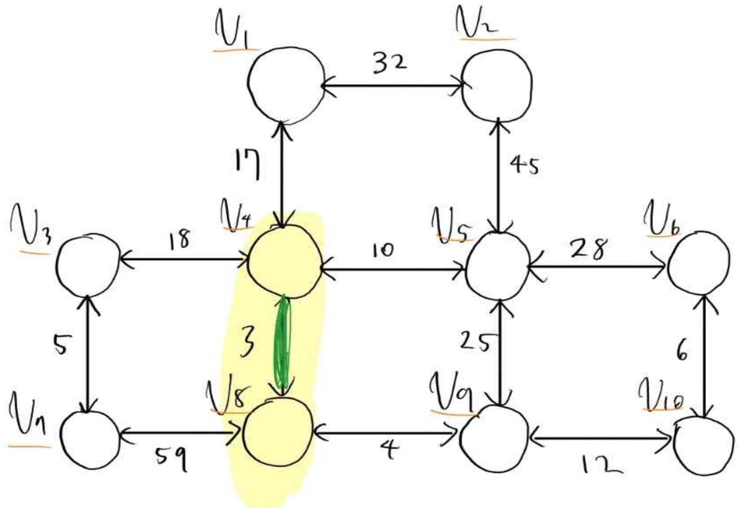

V1
S
0
32
V2
32
17
45
V3
18
18
V4
17
10
V5
10
28
V6
6
5
3
25
6
V7
5
59
V8
3
4
V9
4
12
V10
12

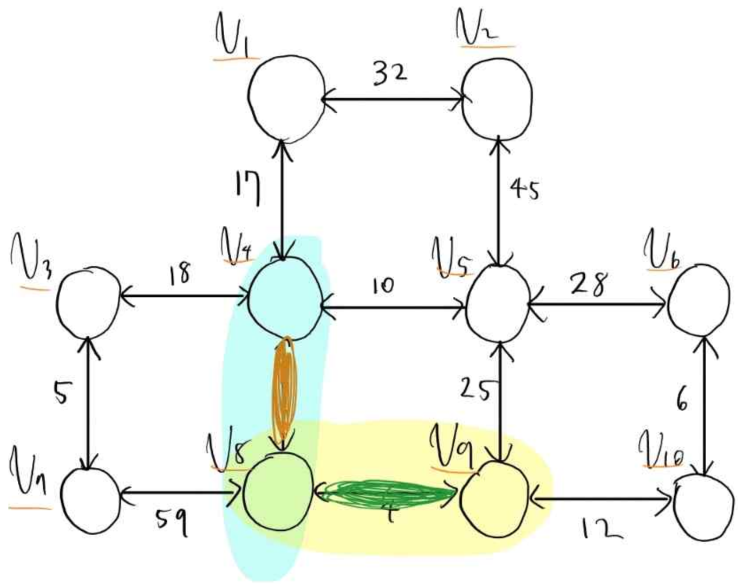

V_1
V_2
S
0
32
32
17
45
V_3
V_4
V_5
V_6
18
18
17
10
10
28
6
5
3
25
6
V_7
V_8
V_9
V_{10}
5
59
3
4
4
12
12

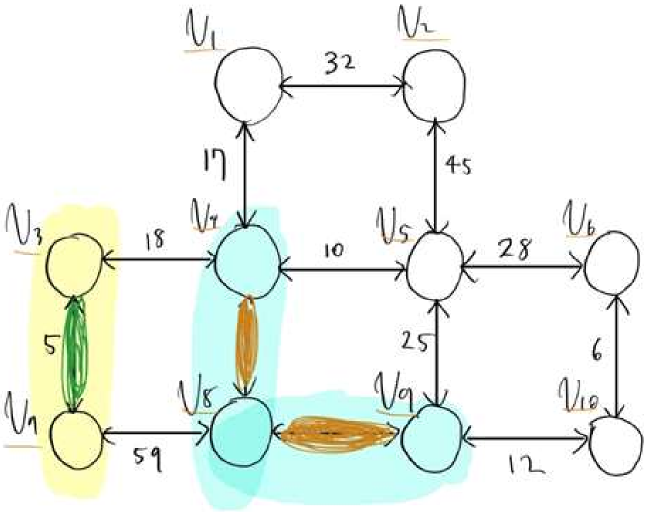

V_1
V_2
S
0
32
V_3
V_4
V_5
V_6
18
17
10
6
V_7
V_8
V_9
V_{10}
5
3
4
12

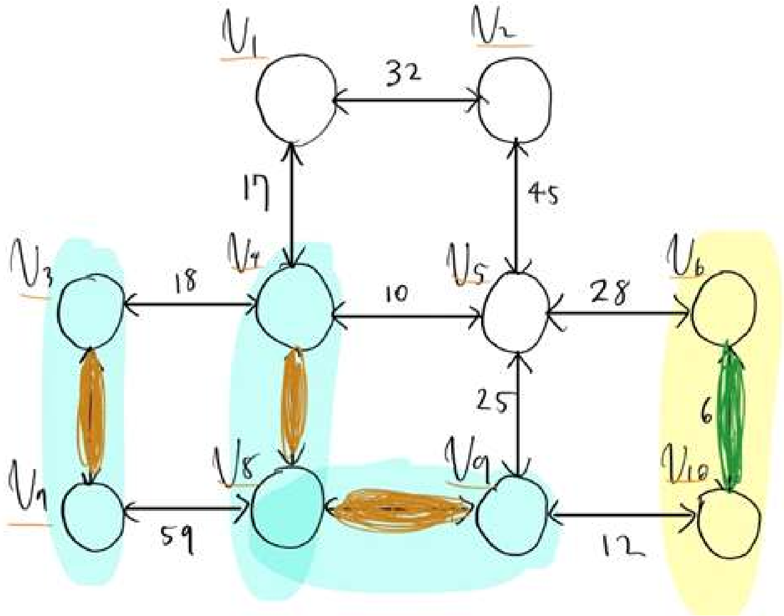
[4-2] 알고리즘을 이용하여 위 그래프의 최소비용 신장 트리를 구하는 과정을 교재 [그림 10-16]과 유사한 형태로 그려 제시(캡쳐 이미지 포함)

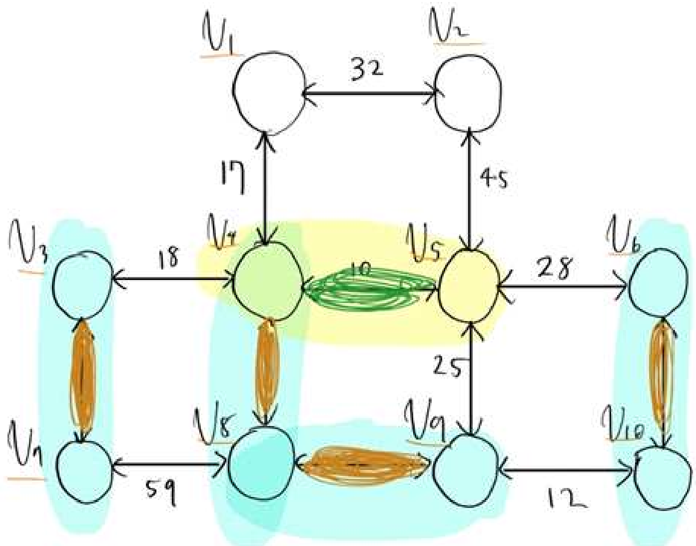


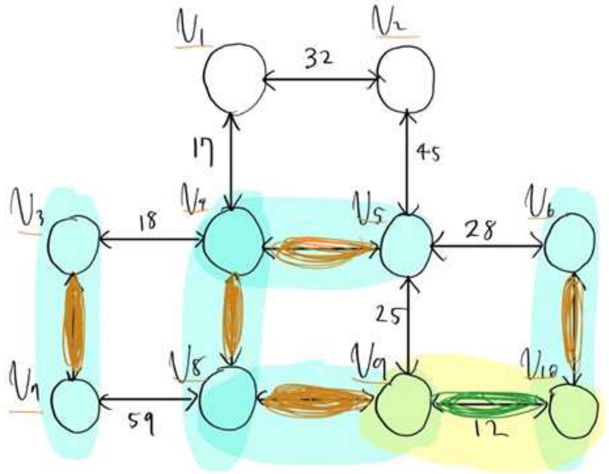


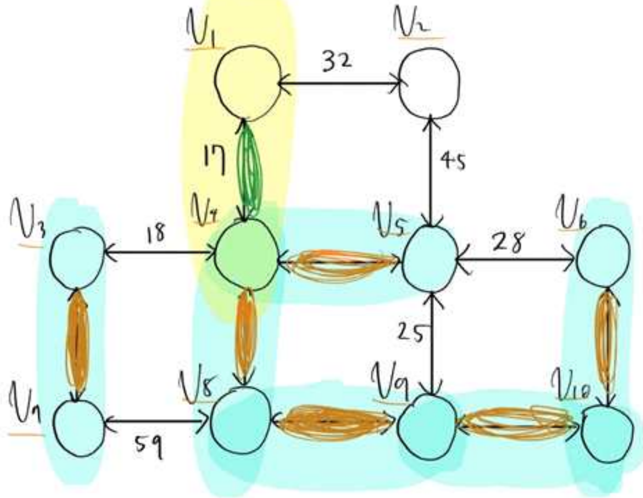


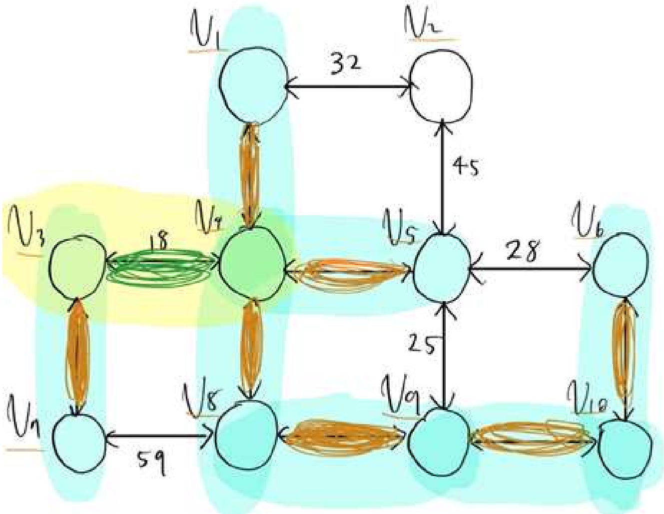

V1
V2
32
17
45
V3
18
V4
10
V5
28
V6
5
25
6
V7
59
V8
V9
V10
12

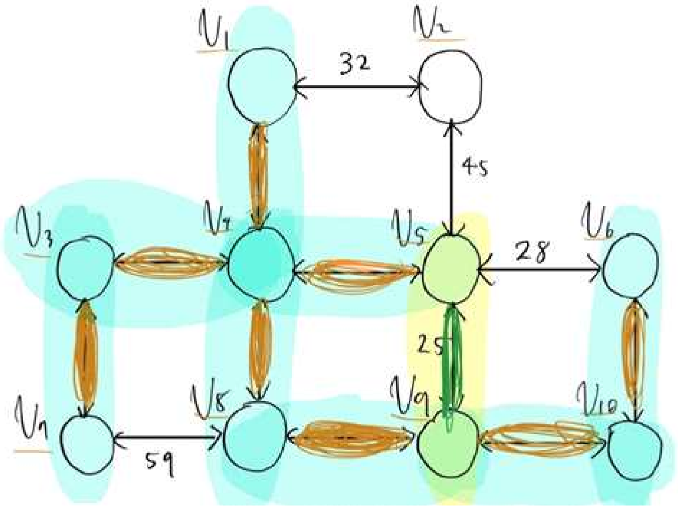

V1
V2
32
17
45
V3
18
V4
10
V5
28
V6
25
6
V7
59
V8
V9
V10
12

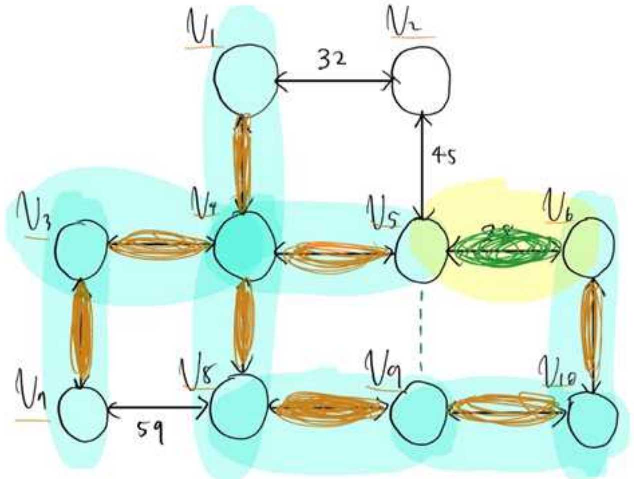

V1
V2
32
17
45
V3
18
V4
10
V5
28
V6
25
V7
59
V8
V9
V10
12

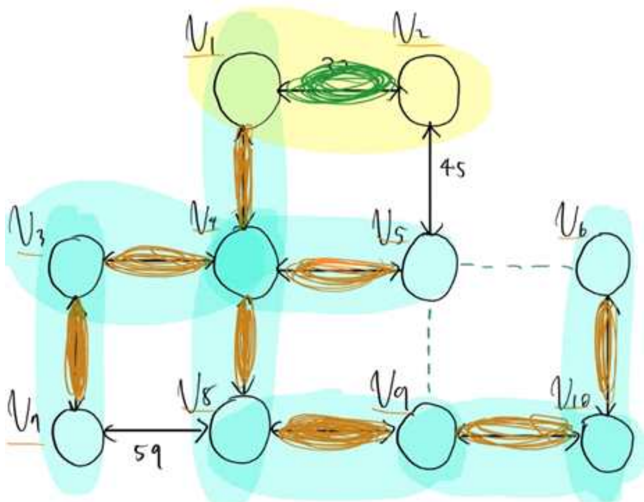

V1
V2
32
17
45
V3
V4
V5
V6
18
28
25
V7
V8
V9
V10
59
12

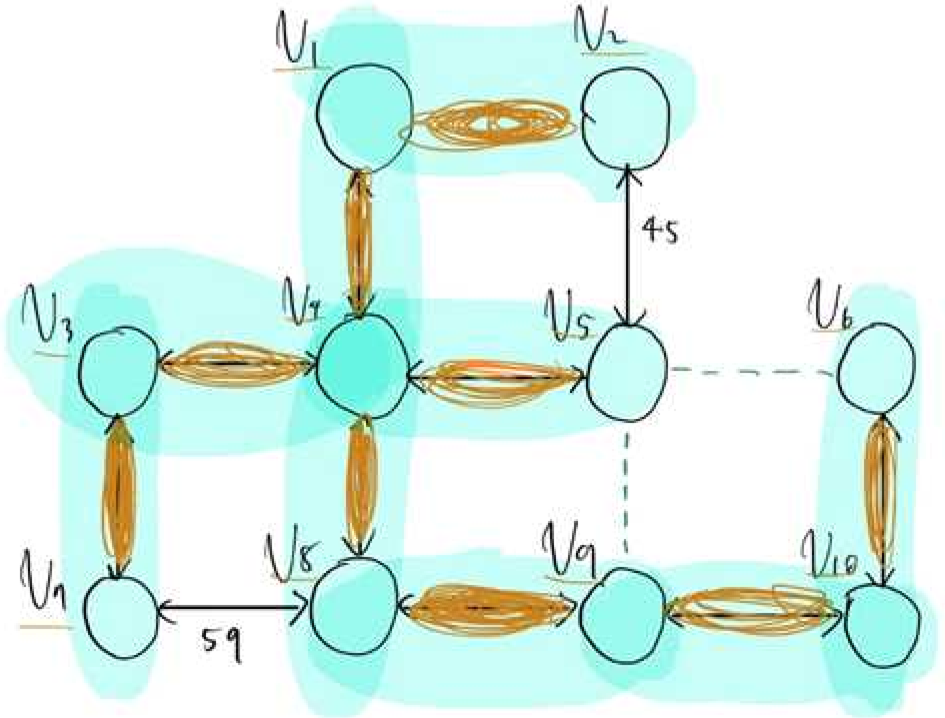

V1
V2
32
17
45
V3
V4
V5
V6
18
28
25
V7
V8
V9
V10
59

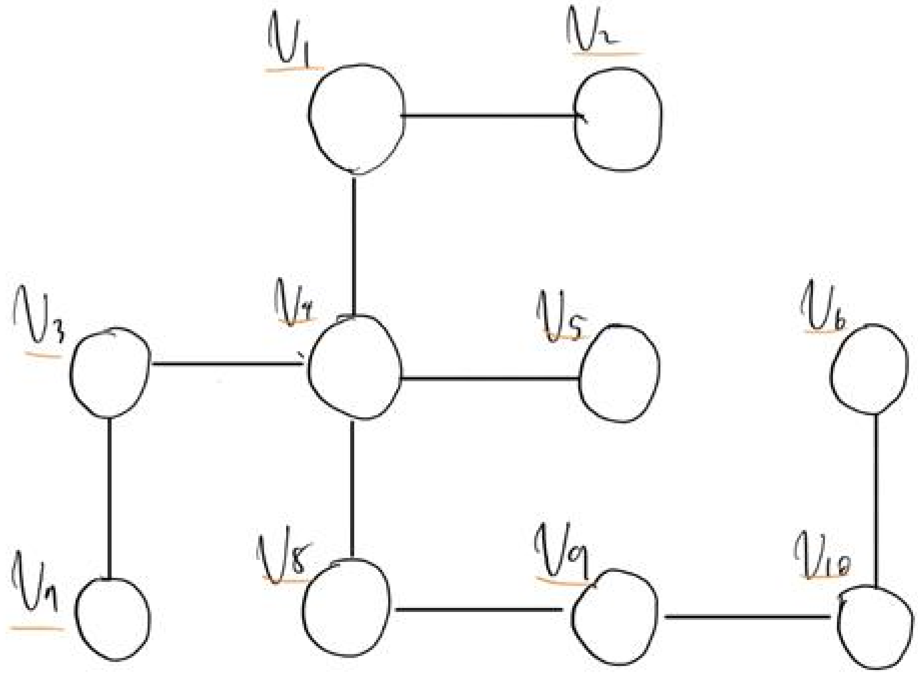

V1
V2
32
45
V3
V4
V5
V6
18
28
25
V7
V8
V9
V10
59


V1
V2
32
45
V3
V4
V5
28
V6
25
V7
59
V8
V9
V10


V1
V2
32
45
V3
V4
V5
V6
V7
59
V8
V9
V10


V1
V2
45
V3
V4
V5
V6
V7
59
V8
V9
V10


V1
V2
45
V3
V4
V5
V6
V7
59
V8
V9
V10

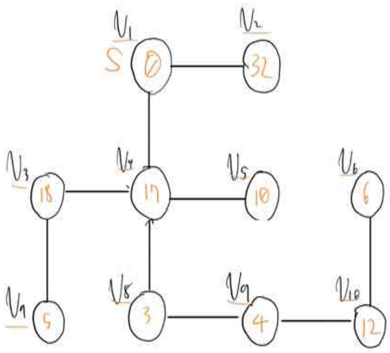

V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

[4-3] 앞선 [문제 1]에서 작성한 최소 비용 신장 트리 알고리즘 파이썬 코드를 이용하여 위 [4-1] 및 [4-2]에 제시한 해답을 검증(캡쳐 이미지 포함)

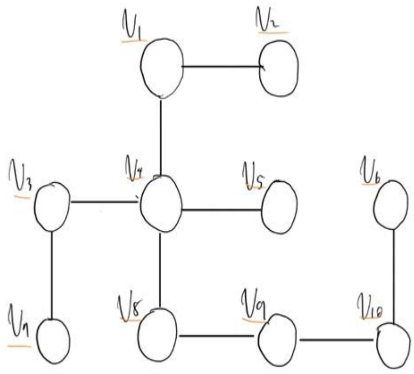
```
*** Prim Solution ***
1 - 2
1 - 4
4 - 3
4 - 5
4 - 8
3 - 7
8 - 9
9 - 10
10 - 6
```

```
*** Kruskal Solution ***
4 - 8
8 - 9
3 - 7
6 - 10
4 - 5
9 - 10
1 - 4
3 - 4
1 - 2
```

[4-1] 결과



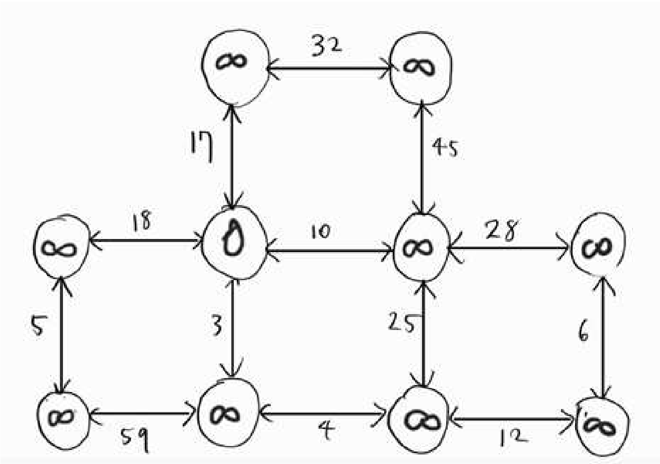
[4-2] 결과

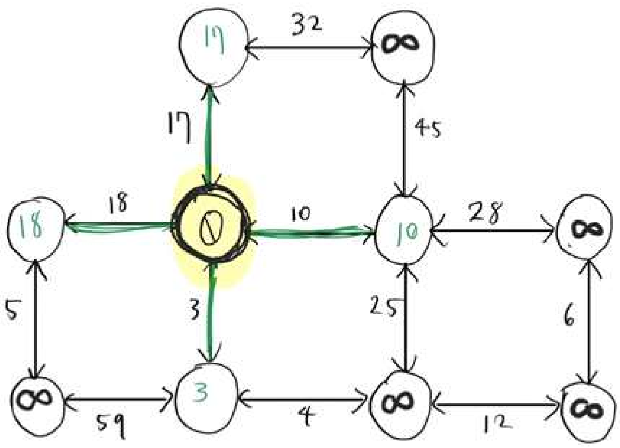


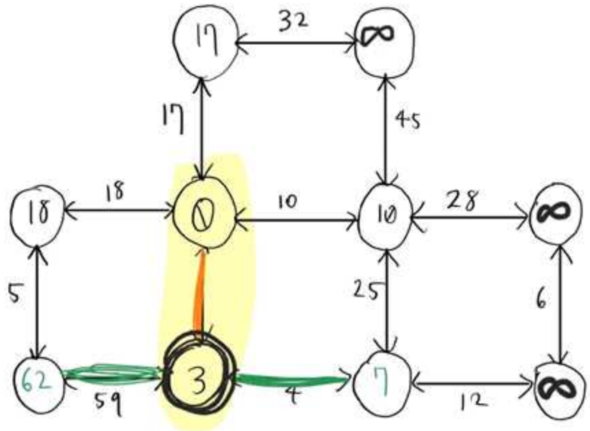
수행결과 [4-1]과 [4-2]와 동일하게 설정되었음을 알 수 있습니다.

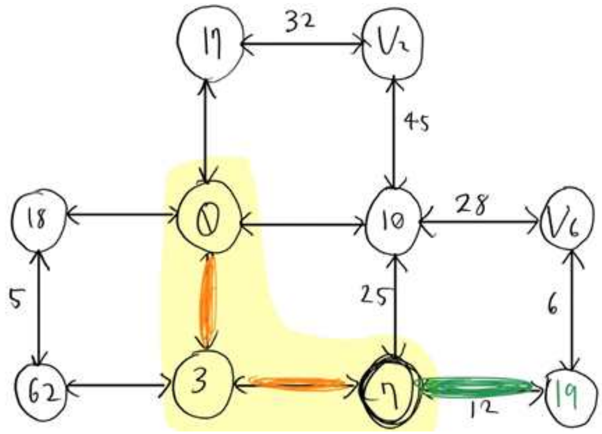
[문제5] 제시문제2

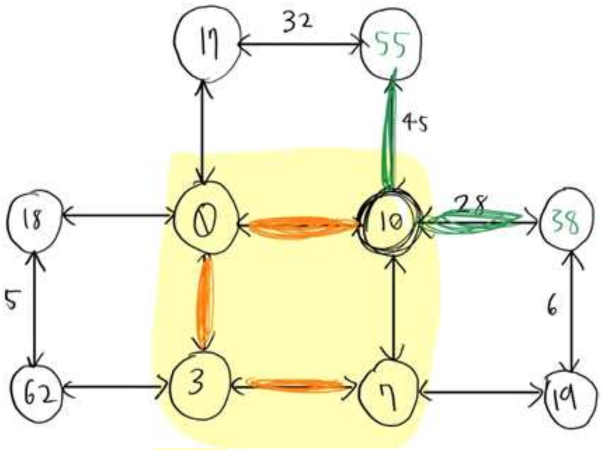
[5-1] Dijkstra 알고리즘을 이용하여 위 그래프(앞선 [문제 4와 동일한 그래프)에서 정점 $v4$에서 다른 모든 정점으로 가는 최단경로를 구하는 과정을 교재 [그림 10-23] 과 유사한 형태로 그려 제시하기. 여기서 각 무향 간선은 같은 가중치를 가진 2개의 쌍방향 간선을 나타낸다고 가정하기.(캡쳐 이미지 포함)

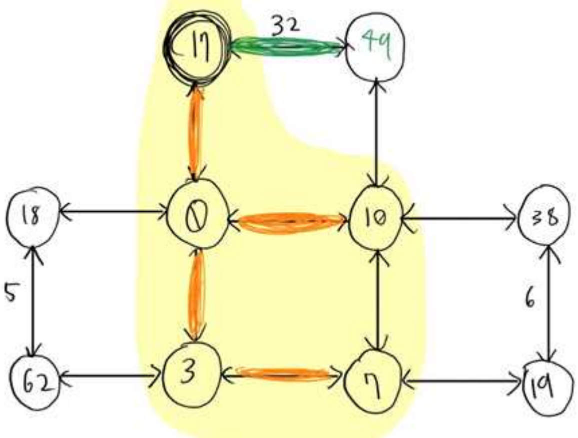


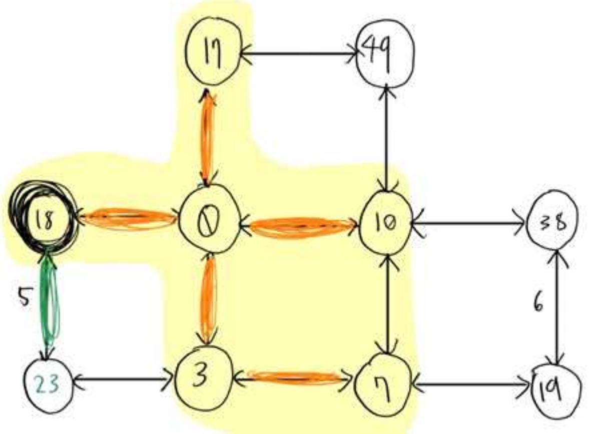


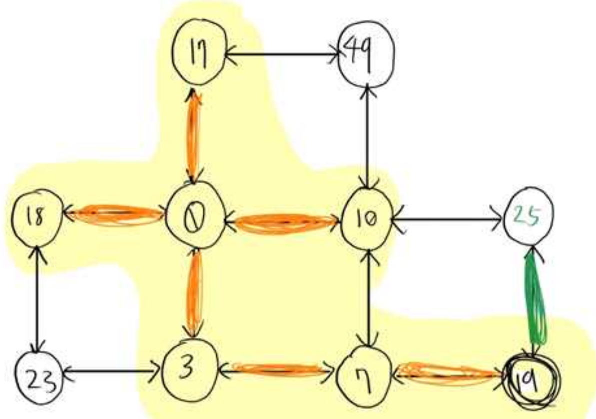


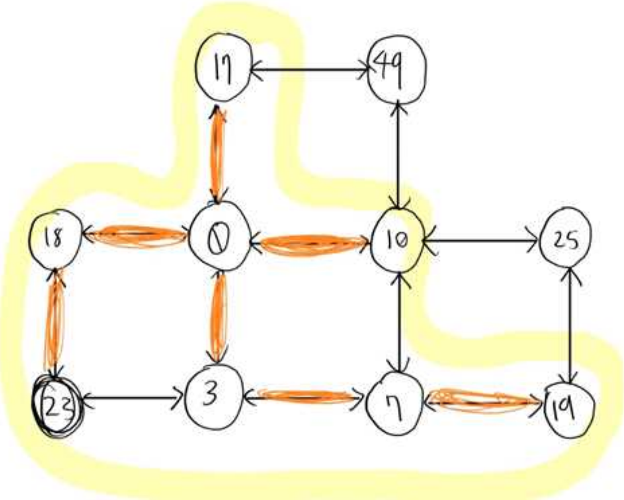

17
32
V2
45
18
0
10
28
V6
5
25
6
62
3
7
12
19

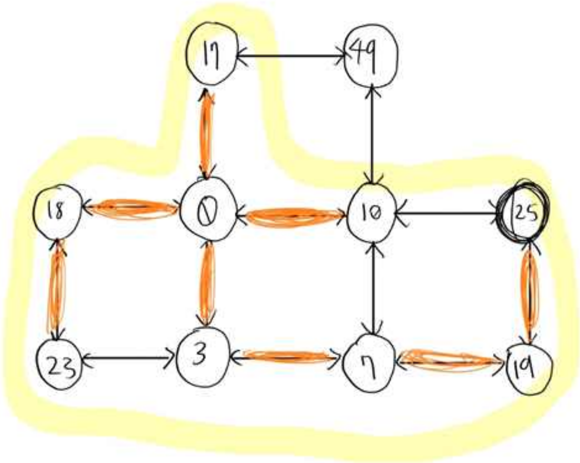

17
32
55
45
18
0
10
28
38
5
6
62
3
7
19

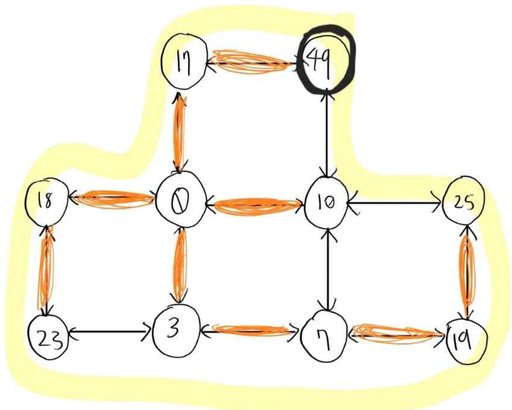

17
32
49
18
0
10
38
5
6
62
3
7
19

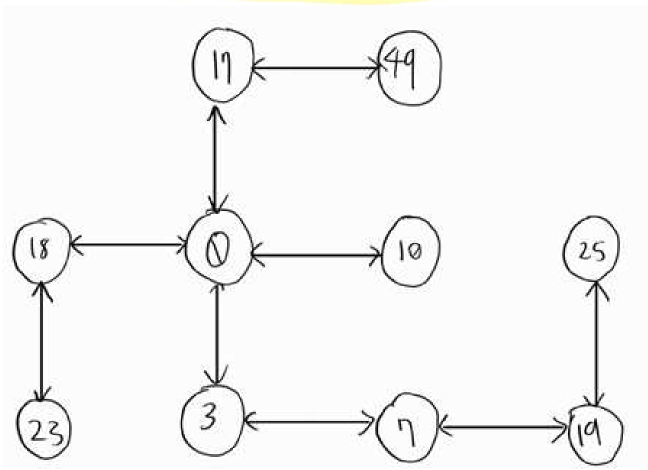

17
49
18
0
10
38
5
6
23
3
7
19

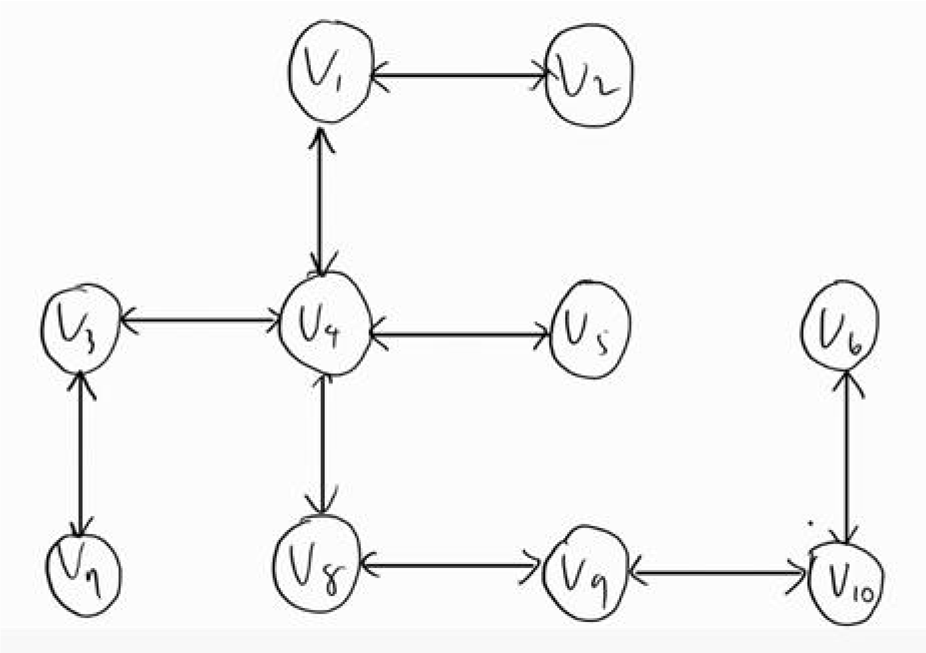

17
49
18
0
10
25
23
3
7
19


17
49
18
0
10
25
23
3
7
19


17
49
18
0
10
25
23
3
7
19


17
49
18
0
10
25
23
3
7
19


17
49
18
0
10
25
23
3
7
19


V1
V2
V3
V4
V5
V6
V7
V8
V9
V10

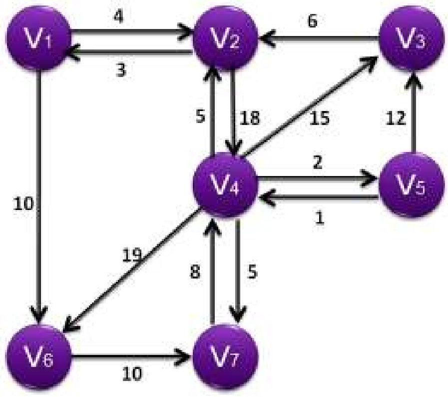
[5-2] 앞선 [문제 2]에서 작성한 Dijkstra 알고리즘 파이썬 코드를 이용하여 위 [5-1] 에 제시한 해답을 검증하기(캡쳐 이미지 포함)

문제2를 구현하지 못했습니다.

[문제6] 제시문제3

[6-1] 오른쪽 주어진 그래프에서 Floyd-Warshall 알고리즘에서 d_{ij}^k와 p_{ij}^k가 만들어지는 과정을 그림으로 제시하시오.

- **d_{ij}^k와 p_{ij}^k 각각 2차원 행렬로 간주하고 총 7 단계별로 2차원 행렬 각각 1개씩 차례로 제시하시오.**



d_{ij}^k = 노드집합 p_{ij}^k = 최단경로

d_{ij}^k을 $D(k)$ $p_{ij}^k = P(k)$라고 할 때

D(0)								P(0)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	∞	∞	∞	10	∞	1	0	1	0	0	0	1	0
2	3	0	∞	18	∞	∞	∞	2	2	0	0	2	0	0	0
3	∞	6	0	∞	∞	∞	∞	3	0	3	0	0	0	0	0
4	∞	5	15	0	2	19	5	4	0	4	4	0	4	4	4
5	∞	∞	12	1	0	∞	∞	5	0	0	5	5	0	0	0
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	∞	∞	∞	8	∞	∞	0	7	0	0	0	7	0	0	0

D(1)								P(1)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	∞	∞	∞	10	∞	1	0	1	0	0	0	1	0
2	3	0	∞	18	∞	**13**	∞	2	2	0	0	2	0	**1**	0
3	∞	6	0	∞	∞	∞	∞	3	0	3	0	0	0	0	0
4	∞	5	15	0	2	19	5	4	0	4	4	0	4	4	4
5	∞	∞	12	1	0	∞	∞	5	0	0	5	5	0	0	0
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	∞	∞	∞	8	∞	∞	0	7	0	0	0	7	0	0	0

// 2 -> **1** -> 6 : 3 + 10 = 13 // **1**

D(2)								P(2)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	∞	**22**	∞	10	∞	1	0	1	0	**2**	0	1	0
2	3	0	∞	18	∞	13	∞	2	2	0	0	2	0	1	0
3	**9**	6	0	**24**	∞	**19**	∞	3	**2**	3	0	**2**	0	**1**	0
4	**8**	5	15	0	2	**18**	5	4	**2**	4	4	0	4	**1**	4
5	∞	∞	12	1	0	∞	∞	5	0	0	5	5	0	0	0
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	∞	∞	∞	8	∞	∞	0	7	0	0	0	7	0	0	0

//1 -> **2** -> 4 : 4 + 18 = 22 // **2**

//3 -> **2** -> 1 : 6 + 3 = 9 // **2**

//3 -> **2** -> 4 : 6 + 18 = 24 // **2**

//3 -> 2 -> **1** -> 6 : 6 + 3 + 10 = 19 // **1**

//4 -> **2** -> 1 : 5 + 3 = 8 // **2**

//4 -> 2 -> **1** ->6 : 5 + 3 +10 = 18 // **1**

D(3)								P(3)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	∞	22	∞	10	∞	1	0	1	0	2	0	1	0
2	3	0	∞	18	∞	13	∞	2	2	0	0	2	0	1	0
3	9	6	0	24	∞	19	∞	3	2	3	0	2	0	1	0
4	8	5	15	0	2	18	5	4	2	4	4	0	4	1	4
5	**21**	**18**	12	1	0	**31**	∞	5	**2**	**3**	5	5	0	**1**	0
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	∞	∞	∞	8	∞	∞	0	7	0	0	0	7	0	0	0

//5 -> 3 -> **2** -> 1 : 12 + 6 + 3 = 21 // **2**

//5 -> **3** -> 2 : 12 + 6 = 18 // **3**

//5->3->2->**1**->6 : 12 + 6 + 3 + 10 = 31 // **1**

D(4)								P(4)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	**37**	22	**24**	10	**27**	1	0	1	**4**	2	**4**	1	**4**
2	3	0	**33**	18	**20**	13	**23**	2	2	0	**4**	2	**4**	1	**4**
3	9	6	0	24	**26**	19	**29**	3	2	3	0	2	**4**	1	**4**
4	8	5	15	0	2	18	5	4	2	4	4	0	4	1	4
5	**9**	**6**	12	1	0	**19**	**6**	5	**2**	**4**	5	5	0	1	**4**
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	**16**	**13**	**23**	8	**10**	**26**	0	7	**2**	**4**	**4**	7	**4**	**1**	0

//1 -> 2 -> **4** -> 3 : 4 + 18 + 15 = 37 // **4**

//1 -> 2 -> **4** -> 5 : 4 + 18 + 2 = 24 // **4**

//1 -> 2 -> **4** -> 7 : 4 + 18 + 5 = 27 // **4**

//2 -> **4** -> 3 : 18 + 15 = 33 // **4**

//2 -> **4** -> 5 : 18 + 2 = 20 // **4**

//2 -> **4** -> 7 : 18 + 5 = 23 // **4**

//3 -> 2 -> **4** -> 5 : 6 + 18 + 2 = 26 // **4**

//3 -> 2 -> **4** -> 7 : 29 // **4**

//5 -> 4 -> **2** -> 1 :9 // **2**

//5 -> **4** -> 2 : 6 // **4**

. . .

//7 -> 4 -> 2 -> **1** -> 6 : 26 // **1**

D(5)								P(5)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	**36**	22	24	10	27	1	0	1	**5**	2	4	1	4
2	3	0	**32**	18	20	13	23	2	2	0	**5**	2	4	1	4
3	9	6	0	24	26	19	29	3	2	3	0	2	4	1	4
4	8	5	**14**	0	2	18	5	4	2	4	**5**	0	4	1	4
5	9	6	12	1	0	19	6	5	2	4	5	5	0	1	4
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	16	13	**22**	8	10	26	0	7	2	4	**5**	7	4	1	0

// 1 -> 2 -> 4 -> 5 -> 3 : 36 // 5
// 2 -> 4 -> 5 -> 3 : 32 // 5
// 4 -> 5 -> 3 : 14 // 5
// 7 -> 4 -> 5 -> 3 : 22 // 5

D(6)								P(6)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	36	22	24	10	**20**	1	0	1	5	2	4	1	**6**
2	3	0	32	18	20	13	23	2	2	0	5	2	4	1	4
3	9	6	0	24	26	19	29	3	2	3	0	2	4	1	4
4	8	5	14	0	2	18	5	4	2	4	5	0	4	1	4
5	9	6	12	1	0	19	6	5	2	4	5	5	0	1	4
6	∞	∞	∞	∞	∞	0	10	6	0	0	0	0	0	0	6
7	16	13	22	8	10	26	0	7	2	4	5	7	4	1	0

// 1 -> **6** -> 7 : 20 // **6**

D(7)								P(7)							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	36	22	24	10	20	1	0	1	5	2	4	1	6
2	3	0	32	18	20	13	23	2	2	0	5	2	4	1	4
3	9	6	0	24	26	19	29	3	2	3	0	2	4	1	4
4	8	5	14	0	2	18	5	4	2	4	5	0	4	1	4
5	9	6	12	1	0	19	6	5	2	4	5	5	0	1	4
6	**26**	**23**	**32**	**18**	**20**	0	10	6	**2**	**4**	**5**	**7**	**4**	0	6
7	16	13	22	8	10	26	0	7	2	4	5	7	4	1	0

// 6 -> 7 -> 4 -> 2 -> 1 : 26 // 2
// 6 -> 7 -> 4 -> 2 : 23 // 4
// 6 -> 7 -> 4 -> 3 : 32 // 4
// 6 -> 7 -> 4 : 18 // 7
// 6 -> 7 -> 4 -> 5 : 20 // 4

결과

Dijk								Pijk							
i \ j	1	2	3	4	5	6	7	i \ j	1	2	3	4	5	6	7
1	0	4	36	22	24	10	20	1	0	1	5	2	4	1	6
2	3	0	32	18	20	13	23	2	2	0	5	2	4	1	4
3	9	6	0	24	26	19	29	3	2	3	0	2	4	1	4
4	8	5	14	0	2	18	5	4	2	4	5	0	4	1	4
5	9	6	12	1	0	19	6	5	2	4	5	5	0	1	4
6	26	23	32	18	20	0	10	6	2	4	5	7	4	0	6
7	16	13	22	8	10	26	0	7	2	4	5	7	4	1	0

결과 검증

Success #stdin #stdout 0.01s 5312KB

stdin

```
Standard input is empty
```

stdout

```
 0   4  36  22  24  10  20
 3   0  32  18  20  13  23
 9   6   0  24  26  19  29
 8   5  14   0   2  18   5
 9   6  12   1   0  19   6
26  23  32  18  20   0  10
16  13  22   8  10  26   0
```

i \ j	1	2	3	4	5	6	7
1	0	4	36	22	24	10	20
2	3	0	32	18	20	13	23
3	9	6	0	24	26	19	29
4	8	5	14	0	2	18	5
5	9	6	12	1	0	19	6
6	26	23	32	18	20	0	10
7	16	13	22	8	10	26	0

[6-2] [3-1]에서 제시한 p_{ij}^{k}의 마지막 행렬을 이용하여 다음 2가지 경우에 대한 경로를 풀이과정과 함께 제시하시오

- Case1] **출발**: v_1, **도착**: v_5
- Case2] **출발**: v_3, **도착**: v_6

Pijk							
i \ j	1	2	3	4	5	6	7
1	0	1	5	2	4	1	6
2	2	0	5	2	4	1	4
3	2	3	0	2	4	1	4
4	2	4	5	0	4	1	4
5	2	4	5	5	0	1	4
6	2	4	5	7	4	0	6
7	2	4	5	7	4	1	0

Case1)
출발 : v1, 도착 : v5
위 표에서 i는 출발점 j는 도착점을 의미합니다.
$P^{k}ij$를 P[i][j]라고 나타냈을 때 i=1, j = 5이므로
P[1][5]은 4이고, 이는 v1에서 v5를 가기 위해 4를 반드시 지나친다는 의미이고
즉 중간 경로에서 P[1][4]이 포함된다고 할 수 있습니다.
같은 과정으로 P[1][4]은 2, P[1][2]은 1이므로

v1에서 v5까지 가는 최단 경로는
v1 -> v2 -> v4 -> v5
와 같습니다.

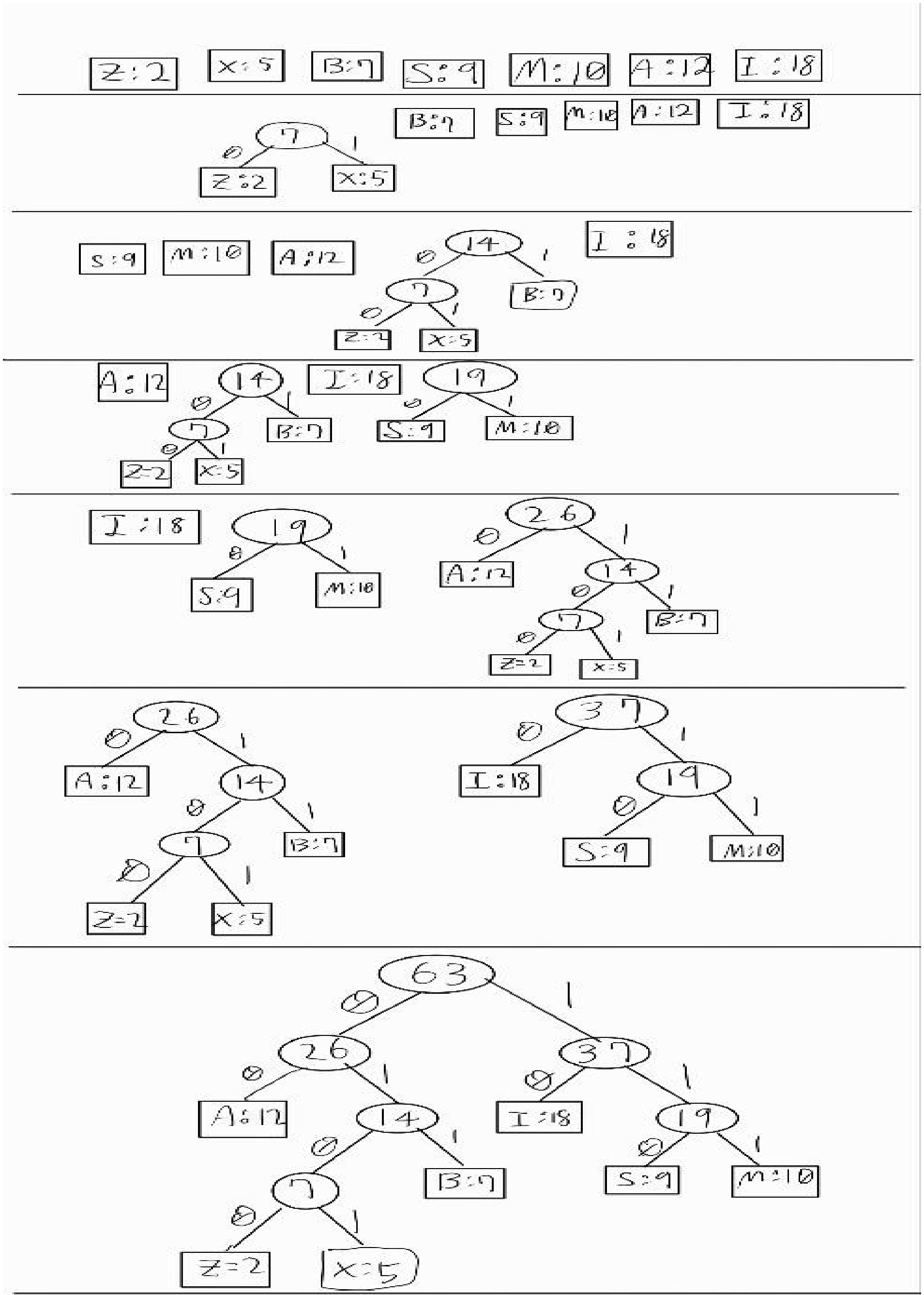
Case2)
출발 v3, 도착 v6
위 표에서 i는 출발점 j는 도착점을 의미합니다.
$P^{k}ij$를 P[i][j]라고 나타냈을 때 i=3, j = 6이므로
P[3][6]은 1이고, 이는 v3에서 v6를 가기 위해 1을 반드시 지나친다는 의미이고
즉 중간 경로에서 P[3][1]이 포함된다고 할 수 있습니다.
같은 과정으로 P[3][1]은 2, P[3][2]는 3이므로

v3에서 v6까지 가는 최단 경로는
v3 -> v2 -> v1 -> v6
와 같습니다.

[문제7] 제시문제4

[문제7-1] 허프만의 알고리즘을 사용하여 다음 표에 있는 글자들에 대한 최적 이진 트리를 구축하는 과정을 제시하기.(캡쳐 이미지 포함)

글자	A	B	I	M	S	X	Z
빈도수	12	7	18	10	9	5	2



[7-2] 위 [7-1]에서 제시한 최적 이진 트리를 기반으로 각 주어진 문자에 대한 최적 전치 코드(허프만 코드)를 제시하기.

[7-1]의 결과를 정리하면 아래 표와 같은 결과가 나옵니다.

문자	허프만 코드
A	00
B	011
I	10
M	111
S	110
X	0101
Z	0100

결론

고찰

아직 이해가 부족하다는 것을 느꼈습니다. 3번과 7번를 통해 행렬곱셈순서문제와 허프만 코드를 공부할 수 있었고 나머지 문제로 최소 신장 트리 등의 개념을 익힐 수 있었지만 아직은 완전한 이해는 부족했던 것 같습니다. 그래서 시험공부를 진행하며 부족한 점을 채워야 한다는 것을 알았습니다.

이번 과제가 많이 어려웠던 것 같습니다.