[FIELD CAMP COMPETITION]

학령인구 분석을 통한 공공교육시설 위치 최적화

: 2035년 인천광역시 초등 학령인구 예상 데이터를 통한 중구/동구 초등학교 재배치

1조 챔피원

유명준 김유라 심유진 오성빈 유민 이준성 이충현 정세진

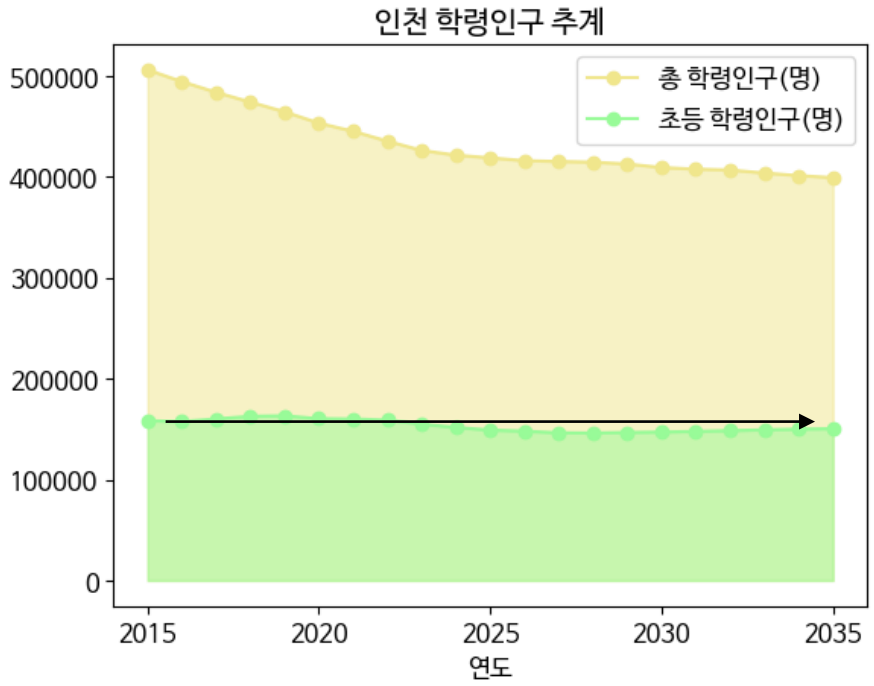
CONTENTS

Introduction

인천 학령인구 데이터 분석

- 인천 전체 학령인구 분석

Q1. "초등 학령인구의 변동"?

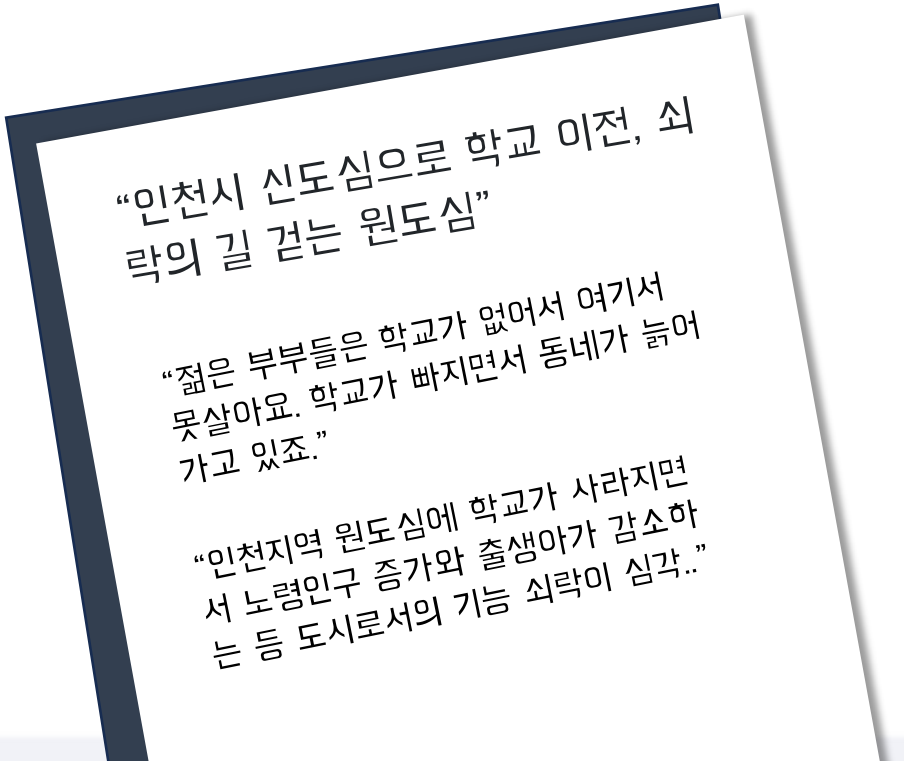


인천 전체의 초등 학령인구는 **일정**한 것을 확인!

신도심의 학령 인구 수는 증가, 원도심의 학령 인구 수는 감소했기 때문

<인천시 신도심으로 학교 이전, 쇠락의 길 걷는 원도심>

Source | 경기일보


"인천시 신도심으로 학교 이전, 쇠락의 길 걷는 원도심"

"젊은 부부들은 학교가 없어서 여기서 못살아요. 학교가 빠지면서 동네가 늙어 가고 있죠."

"인천지역 원도심에 학교가 사라지면서 노령인구 증가와 출생아가 감소하는 등 도시로서의 기능 쇠락이 심각.."

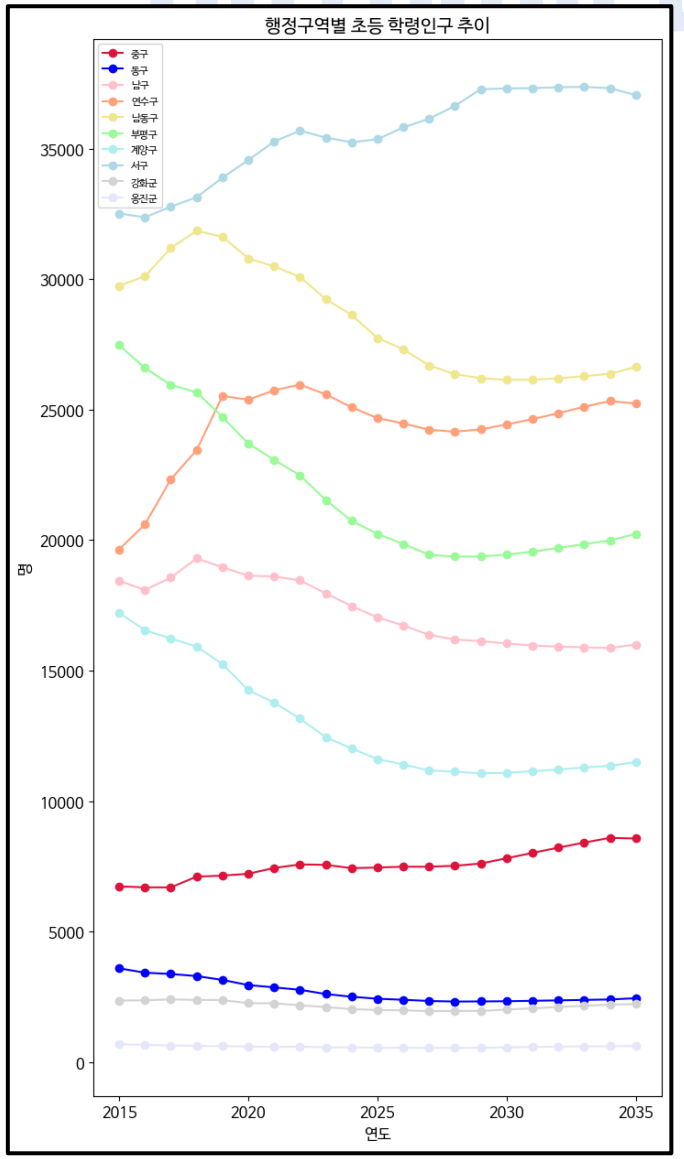
인천 학령인구 데이터 분석

- 행정구역별 학령인구 분석

Q1. “초등 학령인구의 변동”?

전 슬라이드 그래프에서
전체 초등 학령인구는 연도별로 거의 일정하다는 것을 확인 후 각 구별로 추이 시각화 진행

=> 신도심인 ‘서구’와 원도심인 ‘중구‘, ‘동구’, ‘강화군’, ‘옹진군‘ 등과의 초등 학령 인구 차이가 2015년부터 2035년으로 갈수록 증가하고 있음을 알 수 있음.

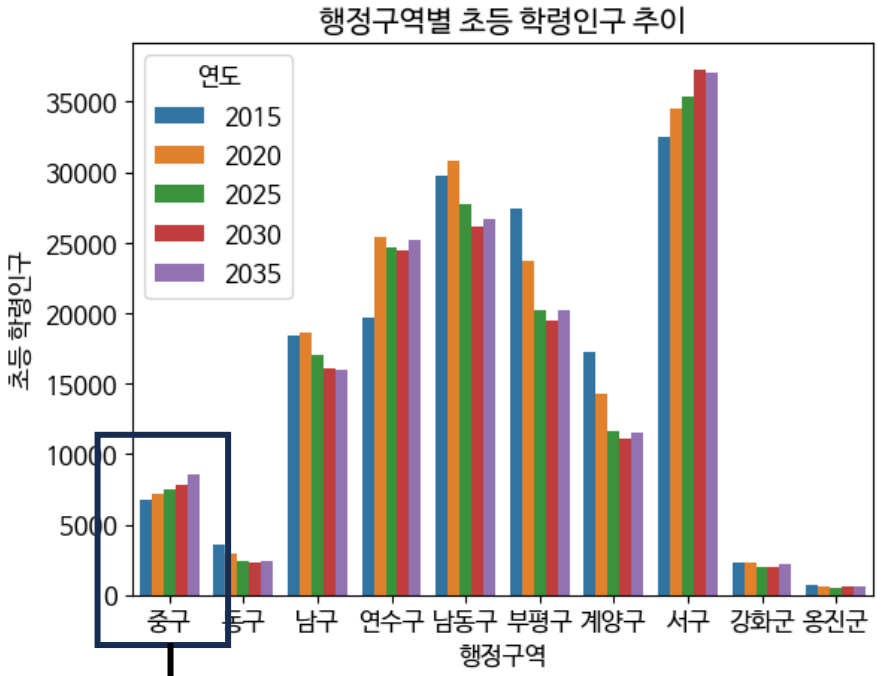


Introduction

인천 학령인구 데이터 분석

- 행정구역별 학령인구 분석

Q2. “인천 학령인구의 양극화”?



중구

중구의 초등학교 12개교 중 **학생수 감소율이 50% 이상인 학교는** 5개교로 **41.7%** 에 이르고 있으며, 동구지역의 경우 **6개교 모두 학생수 감소율이 50% 이상**을 차지하고 있다.

동구

동구 지역의 초등학교 6개교 모두 학생수 **감소율이 50% 이상**을 차지하고 있다.

신도심

동구 지역의 초등학교 6개교 모두 **학생수 감소율이 50% 이상**을 차지하고 있다.

중구 내 신도시인
‘**영종도**’로 인한 인구 증가

** 출처: 강석진, 강규진, 이경훈.(2013).보행안전과 범죄예방을 고려한 초등학교 주변 위험도 평가연구.대한건축학회 학술발표대회 논문집,33(2),249-250.

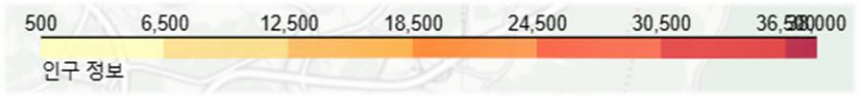
Introduction

인천 학령인구 데이터 분석

- 행정구역별 학령인구 분석

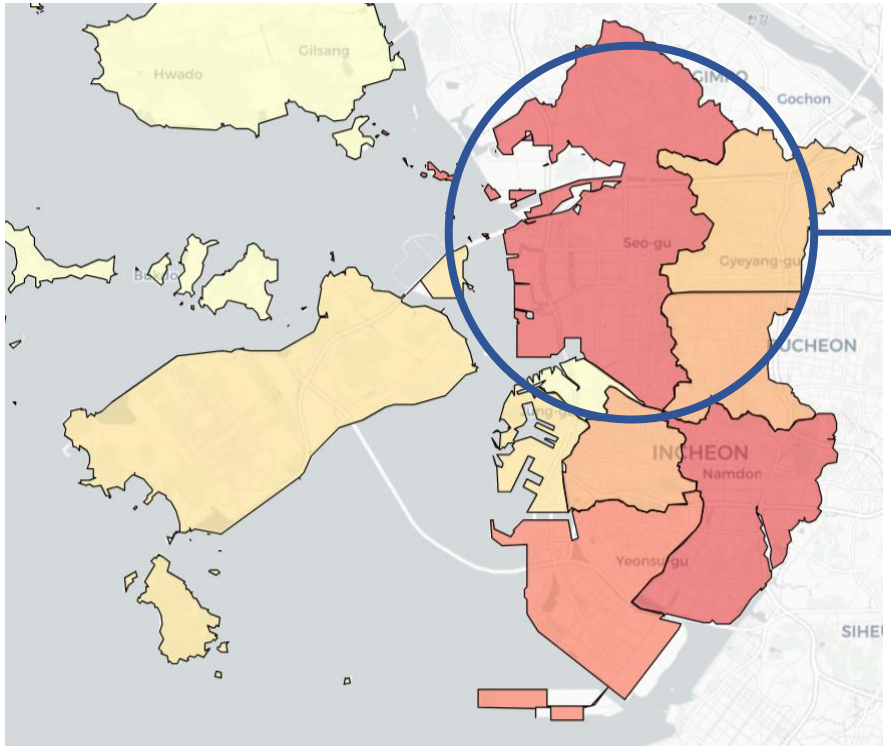
	2020	2035
중구	7214	8564
동구	2950	2446
서구	34557	37052

<scale>

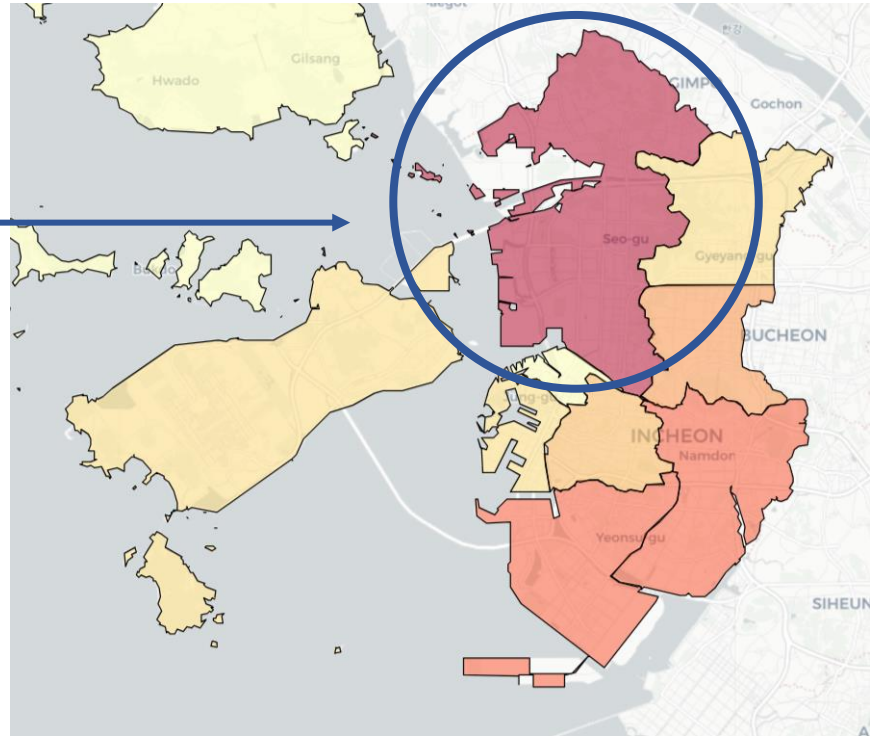


"인천_학령인구_추계.csv" 데이터 활용

Q2. "인천 학령인구의 양극화"?



2020년



2035년

02 후보지 선정 p-median / 결과

후보지 선정 알고리즘

P-median

주어진 도시들 사이의 거리나 비용을 고려하여 특정 위치에 p개의 시설을
배치하여 전체 거리나 비용을 최소화하는 최적화 방법

$$Min \sum_i \sum_j a_i b_{ij} l_{ij} \quad (1)$$

$$\sum_j l_{ij} = 1 \ (\text{for all } i) \quad (2)$$

$$\sum_j k_j = p \quad (3)$$

$$l_{ij} \le k_j \ (\text{for all } i,j) \quad (4)$$

$$l_{ij} \in 0,1 \ (\text{for all } i,j) \quad (5)$$

$$k_j \in 0,1 \ (\text{for all } i,j) \quad (6)$$

a_i : 수요지 i의 수요량

b_{ii} : 수요지 i와 시설물 입지점 j의 거리

p : 시설물 수

k_j : 1 (만약 노드 j에 시설물이 설치된 경우)
0 (그렇지 않으면)

l_{ij} : 1 (만약 노드 j에 시설물이 노드 i의 총 수요를 충속시킨 경우)
0 (그렇지 않으면)

02 후보지 선정 p-median / 결과

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

함수 *haversine_distance*
→ 지구 위 두 점 사이의 실제 거리 계산

① 거리 함수 정의

```python
def haversine_distance(lat1, lon1, lat2, lon2):
    R = 6371  # 지구 반지름 (킬로미터)

    # 위도와 경도를 라디안으로 변환
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])

    # 위도와 경도의 차이 계산
    dlat = lat2 - lat1
    dlon = lon2 - lon1

    # 허버사인 공식
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2)**2
    c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-a))
    distance = R * c

    return distance
```

Python

유클리드 거리 사용 X

02 후보지 선정 p-median / 결과

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

② 모든 수요지와 모든 입지점 사이의 거리를 2차원 리스트인 b에 할당

```
import pulp
from pulp import LpConstraint

# 데이터 초기화 부분
# ---------------------------------------------
# i의 개수, 즉 수요지의 개수
num_i = data.shape[0]

# j의 개수, 즉 시설물 입지점의 개수
num_j = target.shape[0]

# a_i 값의 리스트. 각 수요지 i의 수요량을 나타냅니다.
a = []
for i in data['초등학령인구']:
    a.append(i)

# b_ij 값의 2차원 리스트. 수요지 i와 시설물 입지점 j 사이의 거리를 나타냅니다.
# 예: b = [[b_11, b_12, ...], [b_21, b_22, ...], ...]
b = []

# 각 수요지에서 모든 시설물까지의 거리를 계산
for i, data_row in data.iterrows():
    distances_for_this_demand = []
    for j, target_row in target.iterrows():
        distance = haversine_distance(data_row['Y'], data_row['X'], target_row['Y'], target_row['X'])
        distances_for_this_demand.append(distance)
    b.append(distances_for_this_demand)
```

③ 최적화를 위한 목적함수 및 변수 설정

```
# LP 문제 정의. 여기서는 총 거리를 최소화하는 것이 목적입니다.
prob = pulp.LpProblem("Binary_Optimization", pulp.LpMinimize)

# 변수 정의 부분
# ---------------------------------------------
# l_ij는 노드 j의 시설물이 노드 i의 총 수요를 충족하면 1, 아니면 0입니다.
l = pulp.LpVariable.dicts("l", (range(num_i), range(num_j)), 0, 1, pulp.LpBinary)

# k_j는 노드 j에 시설물이 설치되면 1, 아니면 0입니다.
k = pulp.LpVariable.dicts("k", range(num_j), 0, 1, pulp.LpBinary)
# ---------------------------------------------

# 목적함수 정의. 각 수요지와 시설물 입지점 사이의 거리와 해당 수요지의 수요량을 곱한 값을 최소화합니다.
prob += pulp.lpSum([a[i] * b[i][j] * l[i][j] for i in range(num_i) for j in range(num_j)])

# 수동으로 수요지 1을 시설물 입지점 3에 할당합니다.
manually_assigned_facility_for_demand_1 = 2
```

02 후보지 선정 p-median / 결과

수요지1

뒤에 나올 제약조건 2.

Q: 교육재정 지출이 비효율적이다?

A: 같은 액수의 예산을 받더라도 작은학교는 학생 1인당 공교육비가 높기 때문에 비효율적이라는 지적도 있다.

=> 수요지 1의 경우 다른 데이터들에 비해 너무 동떨어져 있어 이상치로 판단, 최적화 과정에서 제외 후 입지 선정이 완료되면 가장 가까운 입지에 할당

02 후보지 선정 p-median / 결과

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

⑤ 기존 수업 영상에서 소개된 기본 제약조건 설정

```
# 수동으로 수요지 1을 시설물 입지점 3에 할당합니다.
manually_assigned_facility_for_demand_1 = 2

# 제약조건 설정 부분
# ----------------------------------------------

# 각 수요지 i는 정확히 하나의 시설물 j에만 할당되어야 합니다.
for i in range(1, num_i): # 1번 수요지는 수동 할당되므로 제외(index=0)
    prob += pulp.lpSum([l[i][j] for j in range(num_j)]) == 1

# 설치된 시설물의 총 개수는 p와 동일해야 합니다.
prob += pulp.lpSum([k[j] for j in range(num_j)]) == p

# l_ij는 k_j에 의존적입니다. 즉, k_j가 1일 때만 l_ij는 1이 될 수 있습니다.
for i in range(num_i):
    for j in range(num_j):
        prob += l[i][j] <= k[j]

# 각 시설물 j에 할당된 총 수요량을 계산하고, 750 이하이록 제약 조건을 설정합니다.(도시 지역 폐교 기준 200명을 넘는 결과 도출)
for j in range(num_j):
    total_demand_for_j = pulp.lpSum([a[i] * l[i][j] for i in range(num_i)])
    prob += total_demand_for_j <= 750 * k[j]  # 시설물이 설치되었을 때만 제약이 활성화되어야 합니다.
```

=> 각 수요지는 하나의 초등학교 후보지에만 할당되어야 한다.

=> 재배치된 초등학교의 총 개수는 p와 동일해야 한다.

=> 초등학교가 재배치되어야만 수요지가 그 위치에 할당될 수 있다.
즉 특정 후보지로의 수요지 할당은 재배치 여부에 의존적

02 후보지 선정 p-median / 결과

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

⑥ 할당될 입지점 개수인 p의 최대/최소 범위 설정

초등학교 **최대 수용 인원** : 750명
도시 지역의 경우 **폐교 기준** : 200명

"폐교 기준은 도시 지역의 경우 학생수 200명, 농촌 지역은 60명이다.

출처 : 경북일보 - 굿데이 굿뉴스(http://www.kyongbuk.co.kr)

```
s=data['초등학령인구'].sum()
s/750
```

```
13.614666666666666
```

```
s/200
```

```
51.055
```

If) 초등학교 수가 13개 이하
-> 모든 학교에 750명씩 할당되어도 모든 학생을 커버하지 못함
If) 초등학교 수가 52개 이상
-> 모든 학교에 200명씩 할당되어도 0명이 할당되는 학교 존재

13 < p < 52

```
# 설치할 시설물의 개수
# 14개 이상 50개 이하
p = 15
```

최적해가 도출되는 최소의 p 찾기

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

새로 추가한 제약조건 설정

제한조건 1. 유흥시설과의 거리는 200m 이상

제89조 4항

위생 교육 보안상 지장을 초래하는 공장, 쓰레기처리장, 유흥업소, 관람장과 소음 진동 등으로 교육활동에 장애가 되는 고속국도 철도 등에 근접한 지역에는 설치하지 아니할 것

교육환경 보호에 관한 법률 제8조(교육환경보호구역의 설정 등)

교육감은 학교 경계 또는 학교 설립 예정지 경계(이하 "학교 경계 등"이라 한다)로부터 200미터의 범위 안의 지역을 다음 각 호의 구분에 따라 교육환경보호구역으로 설정, 고시해야한다.

1. 정대보호구역 : 학교 출입문으로부터 직선거리로 50미터까지인 지역(학교 설립 예정지의 경우 학교 경계로부터 직선거리 50미터까지인 지역)
2. 상대보호구역 : 학교 경계 등으로부터 직선거리로 **200미터**까지인 지역 중 절대보호구역을 제외한 지역

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

새로 추가한 제약조건 설정

제한조건 2. 초등학생의 통학거리는 2400미터 이내

초등학교는 학생들이 안전하고 편리하게 통학할 수 있도록 다른 공공시설의 이용관계를 고려하여야 하며, 통학거리는 **1천5백미터** 이내로 할 것. 다만, 도시 지역 외의 지역에 설치하는 초등학교 중 학생수의 확보가 어려운 경우에는 학생수가 학년당 1개 학급 이상을 유지할 수 있는 범위까지 통학거리를 확대할 수 있으나, 통학을 위한 교통수단의 이용가능성을 고려할 것.

위 1500m를 준수하려 했으나, 너무 근접한 거리라 최적해가 도출 X

→ 설문조사 내용을 반영하여 제약조건을 **2.4km**로 사용

02 후보지 선정 p-median / 결과

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

제약조건 1. 코드

```
# 각 거리 b_ij가 2.4km 이내로만 가능하도록 제약식을 설정합니다.
for i in range(num_i):
    for j in range(num_j):
        prob += l[i][j] <= (2.4 >= b[i][j])
```

제약조건 2. 코드

```
# 각 시설물 입지점에 대하여 유흥시설과의 거리를 체크하고 제약 조건을 추가
for j, target_row in target.iterrows():
    if is_within_200m_of_entertainment(target_row['Y'], target_row['X'], entertainment_df):
        prob += k[j] == 0
```

제약조건 3. 바다 위 입지점 제외 코드

```
# 시설물 입지점 1, 4, 5, 7, 11, 19 가 할당되지 않도록 하는 제약식 추가
excluded_facility_indices = [0, 3, 4, 6, 10, 18]  # 시설물 입지점 인덱스는 0부터 시작

for j in excluded_facility_indices:
    prob += k[j] == 0
# ------------------------------------------------
```

인천 내 초등학교 위치 최적화

- p-median 알고리즘을 활용한 고찰

결과 출력

```
# 최적화 문제 풀기
prob.solve()

# 결과 출력 부분
# ----------------------------------------------
print(f"Status: {pulp.LpStatus[prob.status]}")

# 각 수요지에 할당된 시설물 입지점 출력 (값이 1인 경우만 출력)
for i in range(num_i):
    for j in range(num_j):
        if l[i][j].varValue == 1:
            print(f"수요지 {i+1}는 시설물 입지점 {j+1}에 할당됨")

# 설치된 시설물 입지점만 출력 (값이 1인 경우만 출력)
for j in range(num_j):
    if k[j].varValue == 1:
        print(f"시설물 입지점 {j+1}에 설치됨")

# 할당된 시설물 입지점 및 연결된 수요지 정보를 저장할 리스트
assigned_data = []

# 설치된 시설물 입지점 및 배정된 수요량 정보를 저장할 리스트
facility_demand_data = []
```

```
# 각 수요지에 할당된 시설물 입지점 정보 추출
for j in range(num_j):
    if k[j].varValue == 1:
        connected_demands = [i+1 for i in range(num_i) if l[i][j].varValue == 1]
        assigned_data.append({'시설물_입지점': j+1, '연결된_수요지': connected_demands})

        total_demand_for_j = sum([a[i-1] for i in connected_demands]) # <- 수정된 부분
        facility_demand_data.append({'시설물_입지점': j+1, '배정된_수요량': total_demand_for_j})

# 데이터프레임으로 변환 후 출력
assigned_df = pd.DataFrame(assigned_data)
print(assigned_df)

facility_demand_df = pd.DataFrame(facility_demand_data)
print(facility_demand_df)
```

02 후보지 선정 p-median / 결과

최종 후보지 선정 결과

```
수요지 2는 시설물 입지점 45에 할당됨
수요지 3는 시설물 입지점 45에 할당됨
수요지 4는 시설물 입지점 45에 할당됨
수요지 5는 시설물 입지점 45에 할당됨
수요지 6는 시설물 입지점 45에 할당됨
수요지 7는 시설물 입지점 45에 할당됨
수요지 8는 시설물 입지점 45에 할당됨
수요지 9는 시설물 입지점 45에 할당됨
수요지 10는 시설물 입지점 45에 할당됨
수요지 11는 시설물 입지점 45에 할당됨
수요지 12는 시설물 입지점 45에 할당됨
수요지 13는 시설물 입지점 45에 할당됨
수요지 14는 시설물 입지점 45에 할당됨
수요지 15는 시설물 입지점 45에 할당됨
수요지 16는 시설물 입지점 45에 할당됨
수요지 17는 시설물 입지점 45에 할당됨
수요지 18는 시설물 입지점 45에 할당됨
수요지 19는 시설물 입지점 45에 할당됨
수요지 20는 시설물 입지점 45에 할당됨
수요지 21는 시설물 입지점 45에 할당됨
수요지 22는 시설물 입지점 45에 할당됨
수요지 23는 시설물 입지점 46에 할당됨
수요지 24는 시설물 입지점 45에 할당됨
```

```
수요지 25는 시설물 입지점 46에 할당됨
수요지 26는 시설물 입지점 44에 할당됨
수요지 27는 시설물 입지점 44에 할당됨
수요지 28는 시설물 입지점 44에 할당됨
수요지 29는 시설물 입지점 46에 할당됨
수요지 30는 시설물 입지점 44에 할당됨
수요지 31는 시설물 입지점 46에 할당됨
수요지 32는 시설물 입지점 46에 할당됨
수요지 33는 시설물 입지점 40에 할당됨
수요지 34는 시설물 입지점 46에 할당됨
수요지 35는 시설물 입지점 40에 할당됨
수요지 36는 시설물 입지점 44에 할당됨
수요지 37는 시설물 입지점 46에 할당됨
수요지 38는 시설물 입지점 46에 할당됨
수요지 39는 시설물 입지점 40에 할당됨
수요지 40는 시설물 입지점 46에 할당됨
수요지 41는 시설물 입지점 46에 할당됨
수요지 42는 시설물 입지점 46에 할당됨
수요지 43는 시설물 입지점 46에 할당됨
수요지 44는 시설물 입지점 46에 할당됨
수요지 45는 시설물 입지점 37에 할당됨
수요지 46는 시설물 입지점 37에 할당됨
수요지 47는 시설물 입지점 37에 할당됨
수요지 48는 시설물 입지점 48에 할당됨
```

```
시설물 입지점 2에 설치됨
시설물 입지점 8에 설치됨
시설물 입지점 12에 설치됨
시설물 입지점 21에 설치됨
시설물 입지점 24에 설치됨
시설물 입지점 28에 설치됨
시설물 입지점 31에 설치됨
시설물 입지점 33에 설치됨
시설물 입지점 35에 설치됨
시설물 입지점 37에 설치됨
시설물 입지점 40에 설치됨
시설물 입지점 44에 설치됨
시설물 입지점 45에 설치됨
시설물 입지점 46에 설치됨
시설물 입지점 48에 설치됨
```

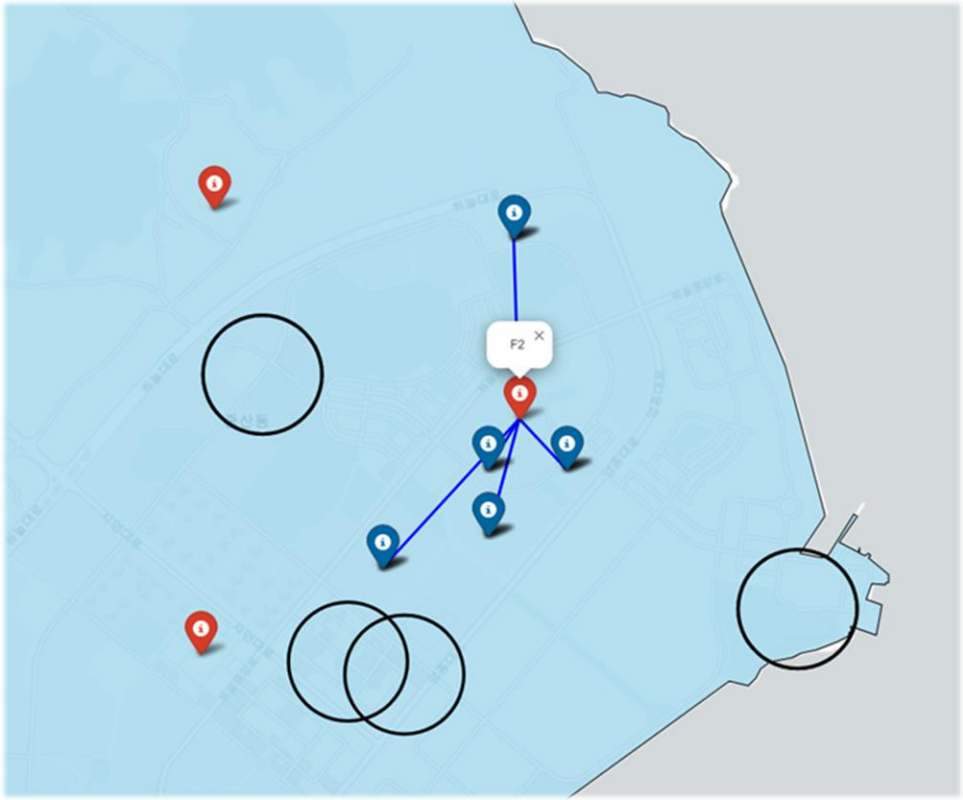
	시설물_입지점	연결된_수요지
0	2	[87, 90, 91, 92, 93]
1	8	[250, 251, 267, 274, 275, 279, 280, 281, 286, ...
2	12	[94, 95, 96, 97, 98, 99, 103, 104, 105, 106, 1...
3	21	[178, 179, 191, 192, 201, 202, 203, 204, 209, ...
4	24	[133, 134, 143, 150, 151, 152, 158, 159, 160, ...
5	28	[100, 101, 102, 116, 138, 219]
6	31	[288, 292, 293, 294, 295, 297, 298, 299, 301, ...
7	33	[74, 80, 81, 83, 86, 88]
8	35	[79, 82, 84, 89]
9	37	[45, 46, 47, 49, 50, 51, 52, 53, 55, 56, 57, 5...
10	40	[33, 35, 39]
11	44	[26, 27, 28, 30, 36]
12	45	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
13	46	[23, 25, 29, 31, 32, 34, 37, 38, 40, 41, 42, 4...
14	48	[48, 54, 68, 75, 77, 78, 85]

	시설물_입지점	배정된_수요량
0	2	721
1	8	750
2	12	710
3	21	750
4	24	728
5	28	580
6	31	750
7	33	520
8	35	750
9	37	664
10	40	737
11	44	736
12	45	741
13	46	436
14	48	637

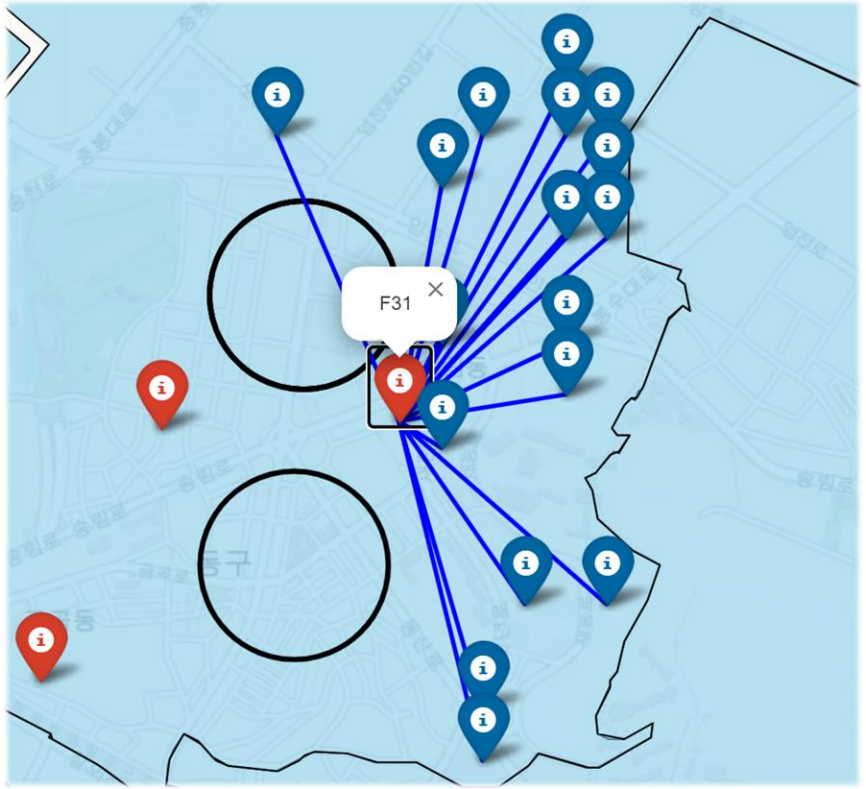
=> 후보지 {2, 8, 12, 21, 24, 28, 31, 33, 35, 37, 40, 44, 45, 46, 48} 에 설치

02 후보지 선정 p-median / 결과

최종 후보지 선정 결과



F2 - 선정 후보지 결과



F31 - 선정 후보지 결과

03 클러스터링 변수 설정 / 클러스터링 기법 / 결과 비교 / 타겟 클러스터 선정

변수 설정

법령에 근거

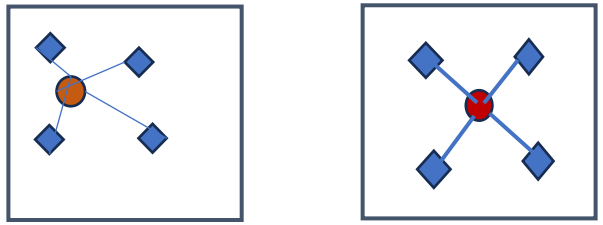
제 89조 14항

"초등학교는 보행자전용도로, 자전거전용도로, 공원 및 녹지축과 연계하여 설치"

출처: 도시 군계획시설의결정 구조및설치기준에관한규칙 (law.go.kr)

클러스터링 방법 소개

K-means



데이터를 비슷한 특성을 가진 K개의 그룹으로 나누는 알고리즘으로, 각 데이터는 가장 가까운 클러스터 중심에 할당

Hierarchical

순차적, 계층적으로 유사한 그룹을 통합하여 k개의 클러스터에 할당

분석 과정

거리 측정에 사용하는 함수

In [95]:
```python
# 공원과 학교간의 거리를 측정하기 위한 함수 생성
from math import radians, sin, cos, sqrt, atan2

def haversine_distance(lat1, lon1, lat2, lon2):
    # 지구 반지름을 이용한 거리 계산
    # 경위도 라디안으로 변경
    lat1, lon1, lat2, lon2 = map(radians, [lat1, lon1, lat2, lon2])

    # Haversine formula
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * atan2(sqrt(a), sqrt(1-a))
    distance = 6371 * c # Radius of earth in kilometers
    return distance

def nearest_park_distance(target_lat, target_lon, park_df):
    distances = park_df.apply(lambda row: haversine_distance(target_lat, target_lon, row['위도'], row['경도']), axis=1)
    return distances.min()
```

분석 과정

```
n_df = pd.read_csv('시설 정보 현황.csv',encoding='cp949')
```

```
# 법령에 근거해서 도시숲, 수변공원, 체육시설의 위치와 target간의 거리 확인
f_df = n_df[n_df['유형'] == '도시숲']
g_df = n_df[n_df['유형'] == '체육시설']
```

```
f_d_list =[]
for idx, row in target_df.iterrows():
    target_lon = row['X']
    target_lat = row['Y']
    d =nearest_park_distance(target_lat, target_lon, f_df)
    f_d_list.append(d)
```

```
g_d_list =[]
for idx, row in target_df.iterrows():
    target_lon = row['X']
    target_lat = row['Y']
    d =nearest_park_distance(target_lat, target_lon, g_df)
    g_d_list.append(d)
```

```
df = pd.DataFrame({'숲' : f_d_list,'체육시설': g_d_list})
```

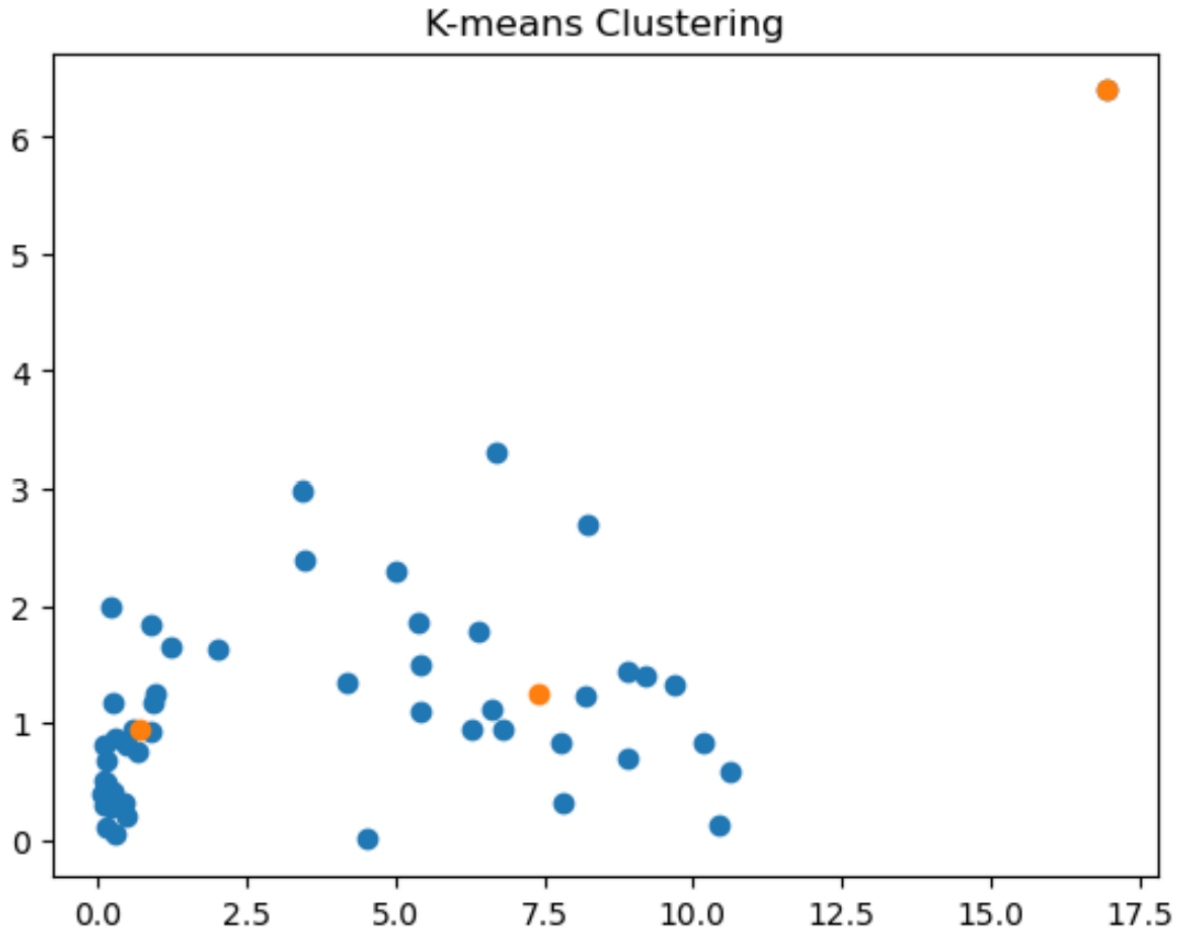
K-means

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(df)
```

```
KMeans(n_clusters=3)
```

```
plt.scatter(df.values[:, 0], df.values[:, 1])

centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1])
plt.title("K-means Clustering")
plt.show()
```

클러스터링 결과 : K-means

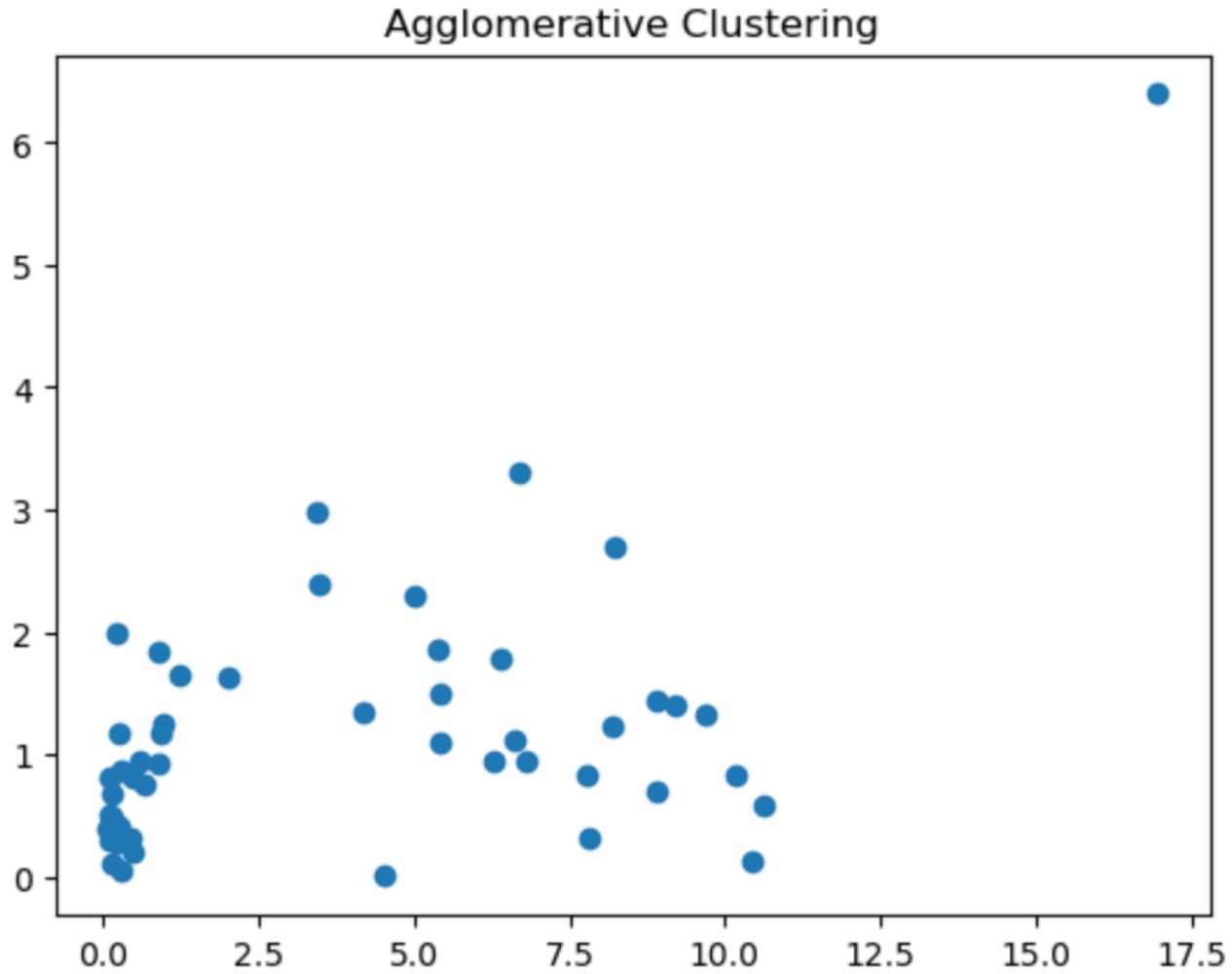


분석 과정

Hierarchical

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from sklearn.cluster import AgglomerativeClustering

# AgglomerativeClustering 객체 생성
agg_cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='average')

# 군집화 수행
clusters = agg_cluster.fit_predict(df)

# 결과 시각화
plt.scatter(df.iloc[:, 0], df.iloc[:, 1], cmap='viridis')
plt.title("Agglomerative Clustering")
plt.show()

# 덴드로그램으로 군집 구조 시각화
def plot_dendrogram(model, **kwargs):
    children = model.children_
    distance = np.arange(children.shape[0])
    position = np.arange(children.shape[0])
    linkage_matrix = np.column_stack([children, distance, position]).astype(float)
    dendrogram(linkage_matrix, **kwargs)

plt.figure(figsize=(10, 5))
plot_dendrogram(agg_cluster, truncate_mode='level', p=3)
plt.title("Hierarchical Clustering Dendrogram")
plt.show()
```

클러스터링 결과 : Hierarchical



최종 결론

In [108]:
```
labels = agg_cluster.labels_
print(labels)
# k-means의 라벨과 같다는 걸 알 수 있다.
```

```
[0 0 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 0 2 2 2 0 0 0 0 0
 0 2 0 0 0 0 0 0 0 0 0 0 0]
```

In [109]:
```
idx2 = []
for i in range(len(labels)):
    if labels[i] == 2 :
        idx2.append(i)
print(idx2)
```

```
[3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 29, 30, 31, 38]
```

In [110]:
```
# 최적화에서 최종 선정 된 시설지 후보 중에서 가중치를 받는 입지들
for i in [1 ,7 ,11 ,20 ,23 ,27, 30, 32, 34, 36, 39, 43, 44, 45, 47]:
    if i in idx2:
        print(i)
```

```
7
11
20
23
27
30
```

7
11
20
23
28
30

"기대 효과 및 활용 방안"

[FIELD CAMP COMPETITION]

감사합니다.

1조 챔피원