문제 소개

하천의 모든 곳에 쓰레기가 있지 X 쓰레기가 주로 몰리는 곳이 있음

어떻게 하면 빠른 시간 안에 효율적으로 쓰레기를 많이 모아서 처리할 수 있을까?



사진 출처: 한성대학교 성곽마을 아카이빙

자원회수시설에서 시작하여 하천의 징검다리와 그 주변을 모두 지나면서 쓰레기를 수거할 수 있는 빠른 경로를 찾아보자! => 자원회수시설별 하천 부유 쓰레기 노선 안내 알고리즘

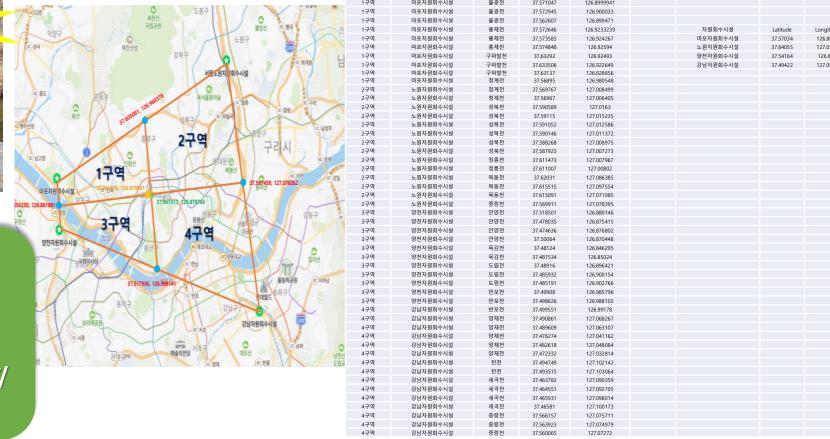
사용 데이터 소개

59명 수강생들의 데이터 중 서울에 위치한 하천 가상은

20211291

	한강의 북쪽에 위치								한강의 남쪽에 위치						
구파발천	불광천	홍제천	성북천	청계천	정릉천	중랑천	묵동천	안양천	목감천	도림천	반포천	양재천	탄천	세곡천	
1	2	2	29	5	2	2	1	6	1	2	1	5	4	1	

수강생의 사진 데이터들을 갖고 징검다리 같은 물과의 접근성이 뛰어난 것만 선별-> 아래와 같이 엑셀로 분류 => **총 20명의 데이터 활용**



해결 방안

- 1. 기존 해결 방안이었던 다익스트라 알고리즘은 모든 지점을 다 지나지 않음
- <반드시 지켜야할 조건>
- 거리를 기준으로 빠른 경로를 탐색할 것
- 모든 지점을 다 한 번씩 거쳐야 함
- 시작 지점은 자원회수시설로 고정
- => 최소 비용 신장 트리 활용!!

크루스칼 알고리즘 vs 프림 알고리즘

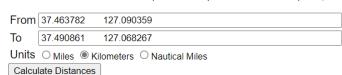
⇒ <u>프림 알고리즘</u>

Why?

- : 그래프 내의 간선이 많을 것으로 예상됨
- 2. 엑셀 파일의 위도, 경도를 통해 가중치 구하기

Coordinate Distance Calculator

Calculate the distance between two points or one point and a number of points, sorted by closest. More Help.,



Distances from N 37 27.827 E 127 5.422:

[Show Map]

Coordinates	<u>Distance</u>		Found By Date
N 37 29.452 E 127 4.096	3.58 kilomete	rs NW (326°)	
Download Waypoints	Map Waypoints		

3. 각 가중치를 지도에 나타내기



현재 노드, 이웃 노드, 가중치(거리)



4. 프림 알고리즘에 적용하기 -> 가중치 값은 100을 곱하여 정수로 만들었음

```
#include <iostream
                                                                          if (parent[u] != u)
#include <vector>
                                                                                                                                   ifstream fp("FIN_1.txt"); // Input할 파일
#include <fstream:
                                                                                                                                   char line[100];
                                                                             // selected에 저장
#include <string>
                                                                                                                                   int n = 0; // 해당 파일의 라인 수
using namespace std;
                                                                             selected.push_back(make_pair(parent[u], u));
                                                                                                                                   while (!fp.eof()) {
                                                                                                                                      fp >> line:
                                                                          ret += minWeight[u];
                                                                                                                                   fp.close():
                                                                          added[u] = true; // 방문하였음
                                                                                                                                   ifstream fp2("FIN_1.txt"); // 읽을 파일
// vector<pair<이웃 노드, 가증치>> adi[현재 노드]
                                                                                                                                   if (fp2.is_open()) {
vector<pair<int, int> > adj[MAX_V];
                                                                                                                                      // 해당 파일의 라인 수 만큼 반복
                                                                         for (int i = 0; i < adj[u].size(); i++) {
                                                                                                                                      for (int i = \theta; i < n; i + *) {
// selected 변수: 선택한 간선의 정점 정보가 저장됨
                                                                             // v: u와 인접한 정점
                                                                                                                                          // 현재 노드, 이웃 노드, 가중치
int prim(vector<pair<int, int> >6 selected) {
   selected.clear(); // 시작이니까 벡터 비우기
                                                                                                                                          int u, v, w;
                                                                             // weight: 가중치
                                                                                                                                          fp2 >> u >> v >> w;
                                                                             int v = adj[u][i].first, weight = adj[u][i].second;
   // 처음이니까 모두 false로 함
                                                                                                                                          adj[v].push_back(make_pair(v, w));
                                                                             if (!added[v] && minWeight[v] > weight) {
   // 이웃한 정점 중 가장 가까운 정점
                                                                                 parent[v] = u;
                                                                                                                                   else cout << "파일이 존재하지 않습니다." << endl;
   vector<int> minWeight(V, INF), parent(V, -1);
                                                                                 minWeight[v] = weight; // 작으면 수정하기
                                                                                                                                   vector<pair<int, int> > selected;
    //0의 부모는 자기 자신이고 자기 자신과의 거리는 0
   minWeight[0] = parent[0] = 0;
    for (int iter = 0; iter < V; iter++) {
       int u = -1; // u가 첫 시작점일 때
                                                                                                                                   for (int i = 0; i < selected.size(); i++) {
                                                                                                                                      cout << selected[i].first << "-" << selected[i].second << endl
                                                                      return ret: // ret 값 반화
           if (!added[v] && (u == -1 || minWeight[u] > minWeight[v]))
                                                                                                                                   cout << endl;
```

결과



최소 거리 합: 15.86km

<1구역>

<3구역>

mst:2099

4-9 4-3 3-5 5-6 4-10 10-11 6-7 7-8

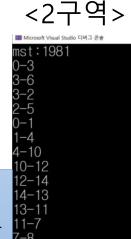
mst:1586

마포자원시설-불광천2-불광천3-불 광천1-홍제천1-홍제천2-홍제천3-청계천1-구파발천3-구파발천2-구 파발천3



최소 거리 합: 20.99km

양천자원시설-안양천1-안양천2-안 양천3-안양천4-도림천1-도림천2-도림천3-목감천2-목감천1-반포천 1-반포천2



<4구역>





최소 거리 합: 19.81km

노원지원시설-묵동천2-묵동천3-묵 동천1-중랑천4-정릉천2-정릉천1-성북천6-성북천5-성북천4-성북천 3-성북천2-성북천1-청계천2-청계 천3



최소 거리 합: 23.98km

강남자원시설-탄천1-탄천2-양재천 5-양재천4-양재천3-양재천2-양재 천1-세곡천4-세곡천3-세곡천2-세 곡천1-반포천3-중랑천1-중랑천2-중랑천3



- 빠른 시간 안에 많은 양의 쓰레기를 수거할 수 있어 효율적
- 시간이 단축되어 인건비가 절약됨
- · 강이나 바다로 흘러가는 쓰레기들을 막아 조금이나마 수질오염에 도움을 줄 수 있음

<깨달은 점>
방법을 찾는 것도 중요하겠지
만 서울의 인구 밀도가 점점
높아지는 만큼 **자원회수시설의 수를 늘릴 필요**가 있음