**알고리즘 보고서**

**도시 설계 프로그램 설계 1차**

**학번 : 12171708**

**이름 : 조승효**

**연락처 : 010-4972-8286**

**이메일 : akak4456@naver.com**

**1. 개요**

- 설계의 목적

국왕은 최소 비용으로 모든 도시들을 자동차로 왕래할 수 있게 연결하

고자 한다. 즉 모든 도시에 도로를 세울 것인데 그것의 비용을 최소화 하고자 한다.

- 요구사항

disjoint set은 연결리스트로 구현하여야 함. Disjoint set의 leader node는 집합의 원소의 수가 더 많은 것 또는 집합의 원소의 수가 같을 시에는 leader id가 작은 것.

최소의 비용으로 mst를 만들어야 함. 이때 비용이 같을 시에는 우선 인구수가 많은 것을 선택해야함. 인구 수도 같을 시에는 id가 작은 것을 선택해야 함.

STL은 vector, list, string, 표준입출력, algorithm 헤더의 sort만 허용한다.

- 개발 환경

windows 10에서 dev-c++을 이용해 compile함

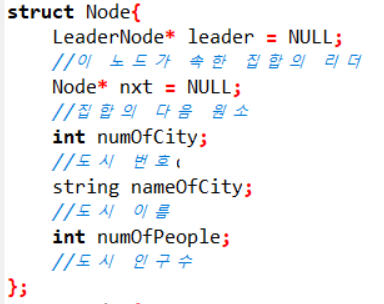
**2. 필요한 자료구조 및 기능**

- 필요한 자료구조, 그 자료구조의 기능

1.연결리스트를 이용한 disjointset:

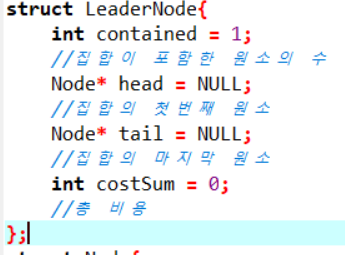
Disjoint set이란 서로 중복되지 않는 부분 집합들로 나눠진 원소들에 대한 정보를 저장하고 조작하는 자료구조.

2.city(vertex)를 저장하는 배열



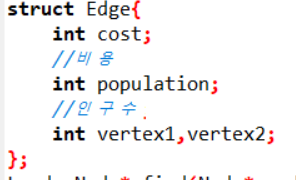


3.city(vertex)가 속한 집합의 leader node를 저장하는 배열





4.비용, 인구수, vertex번호를 포함하는 edge의 vector



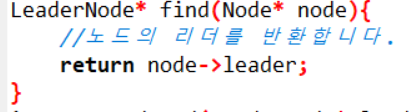


**3. 기능별 알고리즘 명세**

- 각 기능에 대한 알고리즘 설명(시간 및 공간복잡도)

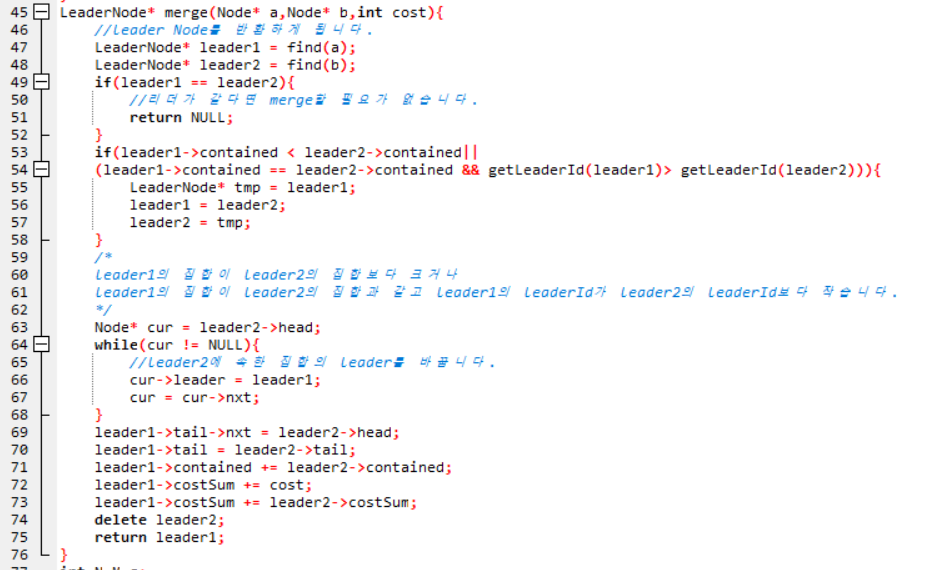
1. 연결리스트를 이용한 disjointset:

Find:집합의 leader를 반환하는 것



한번 실행 시에 시간 복잡도는 O(1), 공간 복잡도는 O(1)

Merge: 서로 다른 집합을 하나의 집합으로 합하는 것



47~48행의 시간 복잡도는 O(1), 공간 복잡도는 O(1)

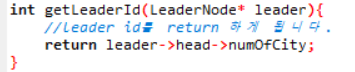
53~58행의 시간 복잡도는 O(1) 공간 복잡도는 O(1)

63~68행의 시간 복잡도는 O(n) 공간 복잡도는 O(1)

69~76행의 시간 복잡도는 O(1) 공간 복잡도는 O(1)

따라서 merge를 한번 실행 시에 시간 복잡도는 O(n), 공간 복잡도는 O(1)

GetLeaderId: leader id를 반환하는 것



한번 실행 시에 시간 복잡도는 O(1) 공간 복잡도는 O(1)

2.city(vertex)를 저장하는 배열

Insert: vertex를 삽입함

각각의 노드에 대하여 시간 복잡도는 O(1), 공간 복잡도는 O(1)

Get: 번호에 해당하는 vertex를 얻어냄

각각의 노드에 대하여 시간 복잡도는 O(1), 공간 복잡도는 O(1)

3.city(vertex)가 속한 집합의 leader node를 저장하는 배열

Insert: leader node를 삽입함

각각의 노드에 대하여 시간 복잡도는 O(1), 공간 복잡도는 O(1)

4.비용, 인구수, vertex번호를 포함하는 edge의 vector

Insert: edge를 삽입함

각각의 edge에 대하여 시간 복잡도는 O(1), 공간 복잡도는 O(1)

GetMin: 우선적으로 처리해야 하는 edge를 뽑음

이 연산을 하기 위해서는 우선 edge를 담는 벡터를 sort해야함. 여기에서 드는 연산의 시간 복잡도는 O(nlgn)

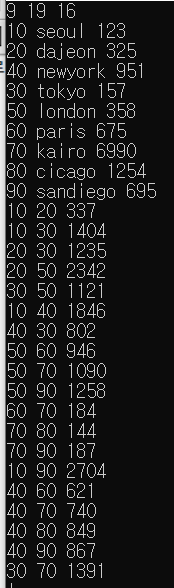
이후에는 정렬할 필요 없이 가장 뒤에 있는 것을 뽑으면 되므로 edge를 뽑는 것에 대해서 시간 복잡도는 O(1), 공간 복잡도는 O(1)

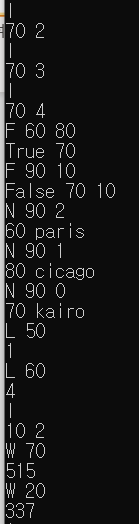
**4. 인터페이스 및 사용법**

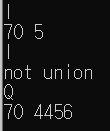
- 간단한 사용법 설명

표준 입출력을 사용함. 그래서 프로그램을 실행한 뒤에 데이터를 입력하면 됨

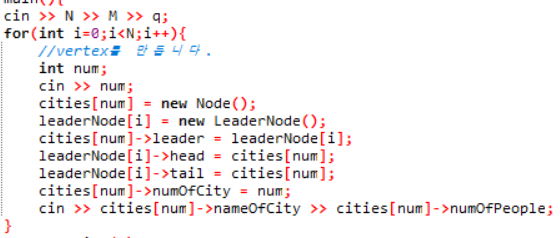
- 실행 화면 캡쳐 포함



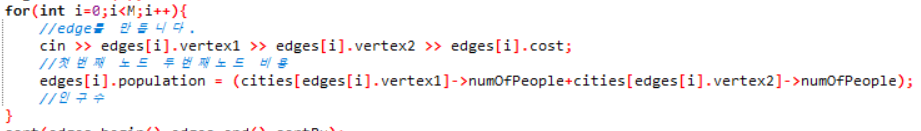




**5. 평가 및 개선 방향**



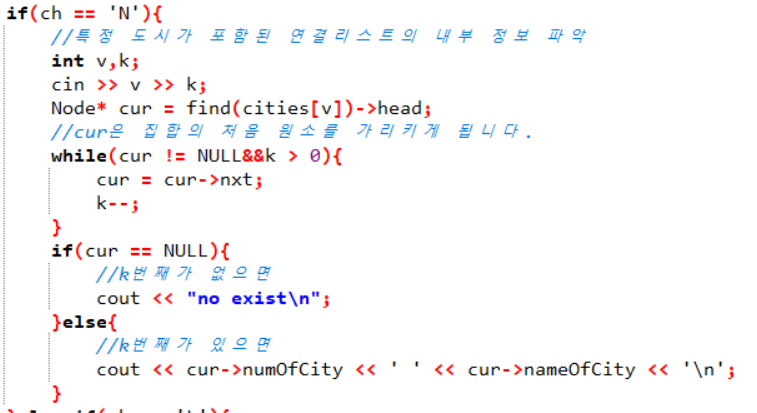
각각의 node(leader node)에 대해서 insert하는데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1) 이다. 따라서 위의 프로세스가 진행될 때 소요되는 전체 시간 복잡도는 O(n), 공간 복잡도는 O(n)이다.



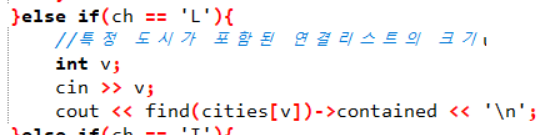
각각의 edge를 insert하는 데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1) 이다. 따라서 위의 프로세스가 진행될 때 소요되는 전체 시간 복잡도는 O(m), 공간 복잡도는 O(m) 이다.



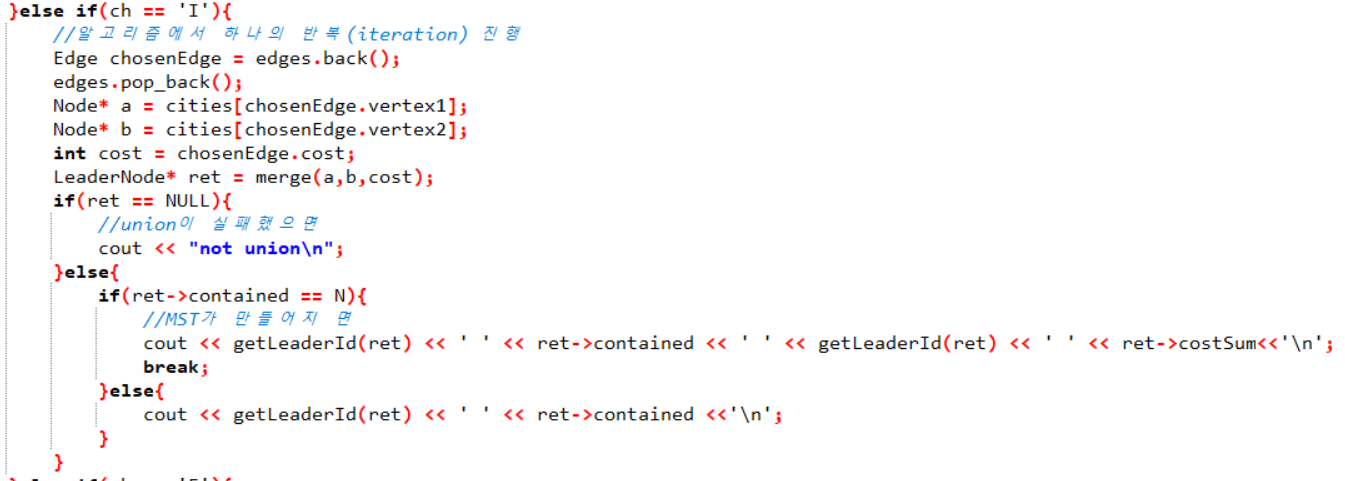
Edge를 정렬하는데 드는 시간 복잡도는 O(mlgm)이다



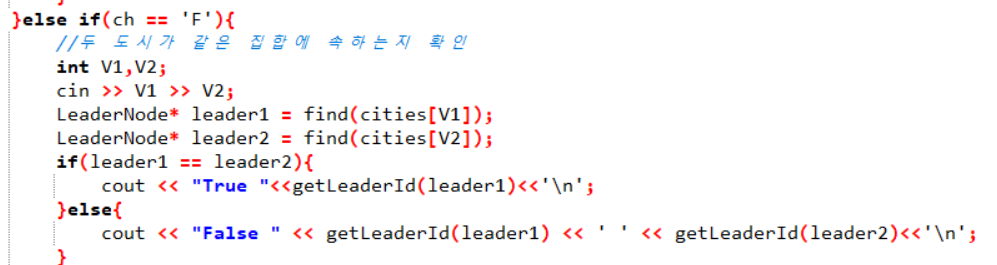
Disjoint set에서 find하는데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다 따라서 위의 쿼리가 진행될 때 걸리는 전체 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다.



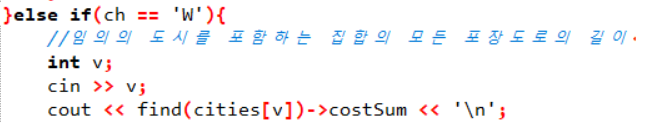
Disjoint set에서 find하는데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다. 따라서 위의 쿼리가 진행될 때 걸리는 전체 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다.



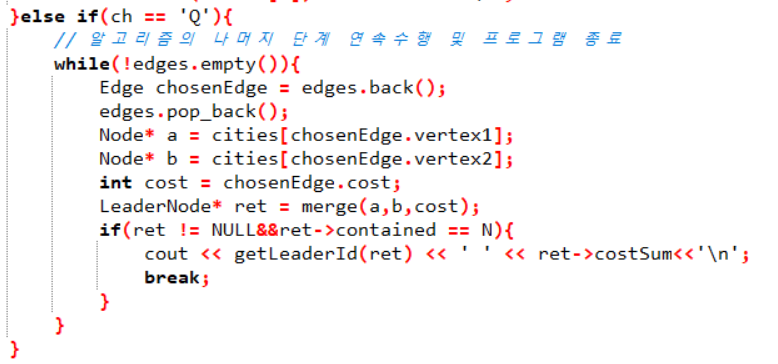
Disjoint set을 merge하는데 걸리는 시간 복잡도는 O(n), 공간 복잡도는 O(1)이다. getLeaderId의 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다. 따라서 위의 쿼리가 진행될 때 걸리는 전체 시간 복잡도는 O(n), 공간 복잡도는 O(1)이다.



Disjoint set에서 find하는데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다. getLeaderId의 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다. 따라서 위의 쿼리가 진행될 때 걸리는 전체 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다.

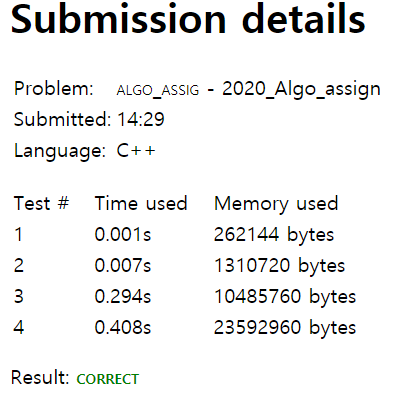


Disjoint set에서 find하는데 걸리는 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다. 따라서 위의 쿼리가 진행될 때 걸리는 전체 시간 복잡도는 O(1), 공간 복잡도는 O(1)이다.



While 문 내에 실행하는데 걸리는 시간 복잡도는 O(n), 공간 복잡도는 O(1)이다. Edge를 비울 때까지 실행하는 것이 최악의 경우이므로 위 쿼리를 실행하는 데 걸리는 최악의 시간 복잡도는 O(n+m), 공간 복잡도는 O(1)이다.

위와 같은 코드를 실행 했을 때의 결과는 아래와 같다.



- 본 결과의 장점

비교적 빠른 시간에 풀 수 있다.

- 본 결과의 단점

불필요한 공간이 든다. 예를 들어 city를 저장하는 부분에서 실제로 필요한 것은 50,000개 인데, cities 배열의 크기는 약 1,000,000개 이다. 여기에서 공간이 낭비되는 부분을 확인 할 수 있다.

- 향후 개선 방향

공간을 절약하는 방향으로 가야할 꺼 같다.