

**Open Source SW Project**

**Final Report**

**Team5 : B1G4**

**좌석 혼잡도 기반 경로 추천 서비스**

Submitted By

20162406 이설희

20174470 전희수

20172030 김윤희

20163453 현도연

20161344 허정우

June 22 2019

**목차**

1. 프로젝트 소개
   1. 주제
   2. 필요성
2. 관련 기술, 서비스, 오픈소스 소프트웨어 소개
   1. 차별성
   2. 오픈소스 커뮤니티에 미칠 효과
3. 구현물 설명
   1. 아키텍쳐
   2. Workflow / Usage Scenario
   3. 핵심 기술 설명
      1. 버스 데이터 파싱
      2. 혼잡도 데이터 분석
      3. 버스 경로 추천 알고리즘
      4. 안드로이드
      5. 서버
      6. 지하철 경로 추천 알고리즘
   4. Implementation 이후로 개선된 부분
4. 성능 평가, 비교, 최종 구현물 데모
5. Github repository
   1. 최종 commit 수, contributor 수, issue / pull request 수
   2. 다른 학생들이 등록한 issue, pull request에 대한 평가 (부적절한/불필요한 contribution은 아닌지) + 이들을 어떻게 처리했는지
   3. 이외에 얼마나 Github repository를 잘 활용했는지 보여줄 것
6. 기타
   1. 업무분담
7. **프로젝트 소개**
   1. **주제**

이번 프로젝트의 주제는 ‘혼잡도 기반 경로 추천 서비스’ 이다. 기존의 여러 경로 추천 서비스는 보통 최단거리, 혹은 최단시간 소요의 길을 안내한다. 하지만 일부 사람들의 경우, 한 장소에서 다른 장소로 이동할 때에 최단거리 혹은 최단 시간이 우선순위가 아닌 경우도 있다. 몸이 불편하거나 이동이 불편한 교통약자, 혹은 짐이 많거나 피곤에 지친 사람들의 경우이다. 이들은 단순히 빠르게 이동하기보다는 대중교통에서 앉아갈 수 있기를 우선적으로 바라게 된다. 그런데 현재 혼잡도에 기반하여 최대한 앉아서 이동할 수 있도록 길을 추천해주는 서비스는 존재하지 않는다. (‘차별성’ 에서 기존의 서비스와의 비교를 통해 상세하게 다룰 예정이다) 따라서 이 주제로 해당 프로젝트를 진행하게 되었다. 혼잡도와 환승 횟수, 최대 시간 등을 고려하여 사람들이 최대한 편하게 길을 찾을 수 있도록 도와주고, 이 서비스를 최종적으로는 모듈화하여 다른 프로젝트나 서비스에도 쉽게 추가하고 실행할 수 있도록 구현할 예정이다

* 1. **필요성**

먼저, 신체적으로 활동에 제약을 받는 교통 약자는 대중교통을 이용한 이동이 쉽지 않다. 한국교통장애인 협회의 전국 교통약자 현황 통계자료에 따르면, 장애인 250만명, 65세 이상 노인 650만명, 9세 이하 유아 460만명을 포함하여 임산부, 보행 불편자, 신체 약자 등 전체 인구의 약 33%(1,700만명)에 해당하는 비율이 이동제약자인 것으로 확인되었다. 비록 이동에 어려움이 있는 교통약자라도 대중교통 이용은 불가피하다. 실제로 지역 내 외출 시 주로 이용하는 교통수단 실태 조사 결과 임산부(58.9%)는 버스, 고령자는 지하철(46.4%)의 빈도가 가장 높았고 장애인은 버스(24.6%), 지하철(22.4%)을 비슷한 수준으로 이용함을 확인할 수 있었다.

그리하여 교통약자의 이동권 문제를 해결하기 위해 다양한 서비스 및 지원정책들이 꾸준히 제안되고 있으며 각종 시스템이 등장하였다. 대표적인 몇 가지 예로는 교통약자 승차지원시스템과 버스 예약 어플리케이션이 있다. 교통약자 승차지원시스템은 버스 정류소에 설치된 단말기로부터 버스번호, 교통약자 유형 등의 대기 상황을 입력하면 운전기사가 출입문 개방 및 도착 알림 등을 제공하는 서비스로 현재 도입을 앞두고 있다. MyBus는 “한국교통약자버스이용협동조합”에서 제공하는 버스 예약 어플리케이션으로 장애인들을 대상으로 운영되고 있는 시스템이다. 그럼에도 대중교통 내부의 혼잡도와 관련하여, 최대한 앉아 이동할 수 있도록 경로를 제공하는 서비스는 아직 존재하지 않았다.

뿐만 아니라, 해당 서비스는 교통 약자가 아닌 직장인 또는 대학생의 현대인에게도 꼭 필요하다. 간단히 주변만 둘러보아도 피곤한 몸을 이끌고 혼잡한 대중교통에 몸을 실은 채 언제쯤 자리가 날까 눈치를 보는 수많은 현대인이 흔히 보인다. 뿐만 아니라 짐이 많은 사람, 몸 컨디션이 좋지 않은 사람, 피로에 지친 시민 등 조금이라도 여유 있는 대중교통에서 편안히 이동하고자 하는 모두가 필요로 하는 서비스임을 알 수 있다. 이와 관련한 사용자 니즈를 만족시키고자 2017년 5월부터 서울 시내 버스에서 실시간 버스 차내 혼잡도 안내서비스를 제공하고 있다. 버스 정류소에서, 그리고 어플리케이션에서 실시간 운행되는 버스의 좌석 현황을 확인할 수 있는데 이는 직관적인 혼잡도 이해를 돕지만 실제적으로 혼잡도 고려한 경로추천까지는 이어지지 않는다는 한계가 있었다.

따라서, 해당 프로젝트는 혼잡에 취약한 교통 약자와 지친 현대인에게 여유로운 대중교통에서 되도록 앉아갈 수 있도록 하는 서비스를 제공할 것이다. 이에 추가적으로, 해당 프로젝트가 성공적으로 진행되었을 때 사람들의 이동 경로를 기존의 최단 경로에서 혼잡도 기반 경로로 분산한다면 부분적인 혼잡도 해소까지도 기대할 수 있으므로 더욱 필요성이 부각된다.

1. **관련 기술, 서비스, 오픈소스 소프트웨어 소개**
   1. **차별성**

이미 출시되어 있는 여러 어플리케이션과의 비교를 통해 차별성을 알아보았다

* + 1. 지하철 종결자, 카카오지하철, 또타지하철, 지하철매니저

: 해당 앱들이 제공하는 기능은 대표적으로 실시간 지하철 도착 정보, 하차 알람 기능, 출발시간 기준 / 도착시간 기준 경로 검색, 최소 환승 또는 최소 시간으로 경로를 검색하는 기능이다. 이 경우 대중교통 중 지하철 이용만 고려했다는 점과, 혼잡도를 고려하지 않고 단순히 최소 시간만 생각했다는 점이 특징이다

* + 1. 지하철 혼잡도(바글바글), 지하철혼잡도알리미, R-Subway (<https://mplatform.seoul.go.kr/w/contest/award/2016/wnpz/selectWinner.do>)

: 지하철 혼잡도 통계 값을 제공하는 앱이다. 이 경우 대중교통 중 지하철 이용만 고려했다는 점과, 혼잡도는 제공하지만 구체적인 길 찾기 기능은 제공하지 않는다는 단점이 있다. 또한 앱 자체에 오류가 많다는 후기들이 있었다. R-Subway의 경우 역시 지하철 이용만 고려했지만 부가적으로 통계적인 혼잡도 정보와 역의 사고 여부를 제공한다는 특징이 있다.

* + 1. 전국 스마트 버스, 카카오버스, busconditionServer(<https://github.com/aerain/busconditionServer>)

: 실시간 버스 위치정보와 좌석 혼잡도 현황, 하차 알람 및 일반 경로 찾기를 지원하는 앱이다. 이 경우 대중교통 중 버스 이용만 고려했다는 점과 혼잡도 정보를 제공 만 하고 이를 고려한 경로 추천까지는 해주지 않는다는 한계가 있다.

* + 1. 서울교통포털, 실시간 교통정보, Jamco(<https://github.com/carolrizzi/Jamco>)

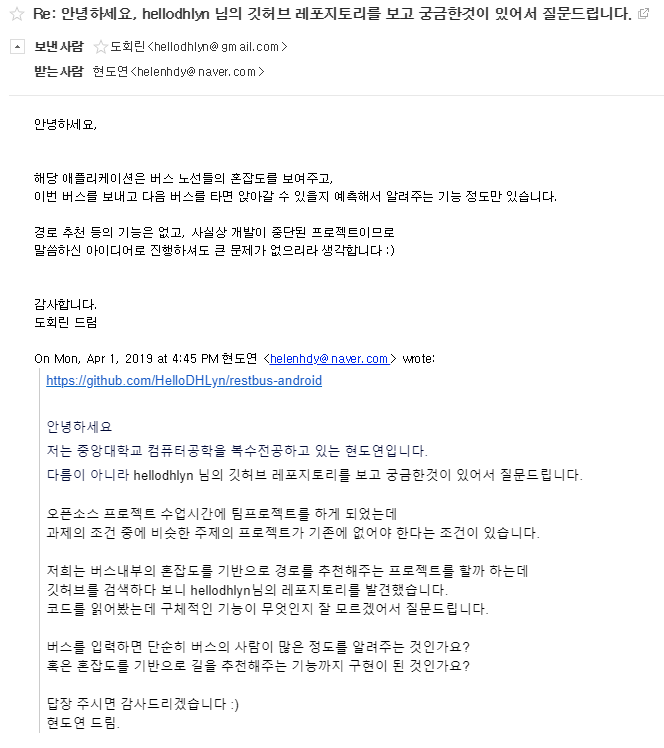
: 주변 지하철 및 정류소를 조회하고, 경로 검색 및 현재 일시 기준 도로통제정보를 확인, 통계 값에 기반한 교통예측정보 확인이 가능한 앱이다. 서울교통포털 앱은 버스와 지하철을 통합하여 길찾기를 최소시간으로 제공하지만 앱 자체가 너무 느리고 오류가 많다는 후기들이 있었다. 또한 실시간 교통정보 앱과 Jamco의 경우 대중교통 사용보다는 자가용 사용에 초점이 맞추어진 도로 별 혼잡도 제공 어플리케이션이었다.

* + 1. 카카오맵, 네이버 지도, 구글 맵 (일반 길찾기 앱)

: 주변 지하철, 정류소 조회 기능과 길찾기를 지하철, 버스, 택시, 도보 등 다양한 경로로 제공하는 앱이다. 버스와 지하철, 도보, 택시 등의 여러 수단을 제공하며 혼잡도 정보는 버스만 제공하고, 경로 찾기는 수단별로 최소 시간 경로들을 제공하였다. 버스 앱과 마찬가지로 혼잡도 정보가 단순히 제공되기만 하고 이를 고려한 경로 탐색은 이루어지지 않았다.

* + 1. Restbus(<https://github.com/HelloDHLyn/restbus-android>)

: github에 있는 해당 프로젝트의 경우, 버스의 혼잡도 정보를 이용한 길찾기 프로젝트였는데 정확히 어떤 기능을 구현하였는지 알 수 없어 해당 레퍼지토리의 소유자에게 이메일을 보내 우리 조의 프로젝트와의 차별성을 확인해 보았다. (자세한 내용은 아래 사진 첨부)



통합해 보았을 때 기존의 여러 앱 중 단순히 최단 시간이 아니라, 최대한 오래 앉아서 편안히 이동할 수 있도록 혼잡도 기반 길 찾기를 제공하는 앱은 없었다. 이 점에 차별성을 두어 본 프로젝트를 진행하고자 한다.

* 1. **오픈소스 커뮤니티에 미칠 효과**

기존의 길찾기 (최단경로 기반)의 경우 다른 여러 프로젝트에 활용할 수 있도록 api가 오픈되어 있다. 마찬가지로 해당 프로젝트 역시 모듈화시켜 이식할 수 있게 제작하고 오픈소스 커뮤니티에 공개한다면, 다른 프로젝트 진행 시에 사람들이 사용할 수 있을 것이다. 단순히 시간을 우선순위로 한 경로들이 아니라, 사용자에 맞게 혼잡도를 우선적인 필터로 고려할 수 있으므로 많은 교통 약자와 피곤에 지친 현대인을 생각해 보았을 때 많은 수요가 예상된다.

1. **구현물 설명**
   1. **아키텍쳐**

ㅇㅇㅇ

* 1. **Workflow / Usage Scenario (사용방법 적어도 좋을 듯)**

ㅇㅇㅇ

* 1. **핵심 기술 설명**
     1. **버스 데이터 파싱**

희수,윤희

* + 1. **혼잡도 데이터 분석**

설희, 도연

* + 1. **버스 경로 추천 알고리즘**

윤희, 도연

* + 1. **안드로이드**

희수, 설희

* + 1. **서버**

현재 구축된 서버는 i5-6600/16GB RAM/100Mbps광랜/Ubuntu 18.04로 구성되어 있습니다. 사양은 굉장히 낮지만 현재 서버로 사용 가능한 컴퓨터가 없어 위 컴퓨터로 진행하였습니다.

* + - 1. 데이터 통신

처음으로 서버 구축을 시작했을 때 모든 데이터를 웹으로 주고받으려 하였습니다. 마침 현재 프로그램도 Java를 사용하고 있기 때문에 JSP를 이용해 구축하여 통신을 진행했습니다. 여기서 서버를 담당한 팀원 전체가 JSP를 사용하는데 익숙하지 않고 시간이 부족한 문제가 발생하여 결국 단순 Socket을 이용한 통신을 수행하기로 결정하였습니다.

* + - 1. 코드 수정
      2. 문제점
      3. 현재 서버에서 가장 큰 문제점으로 어느 정도까지 버틸 수 있는지 검증이 되지 않았습니다. 나중에 실제 테스트를 수행하면서 확인 및 수정이 가능하지만 데모에서는 크게 문제가 되지 않기 때문에 일단은 그대로 두었습니다.
      4. 지속적으로 Baidu쪽에서 데이터를 쏘는 작업을 하여 프로그램이 종료되는 경우가 있어 서버에서 프로그램을 24시간 실행시키지 못하는 상황입니다. 시간상 보안 관련 설정은 전부 제외하고 환경설정을 수행한 결과 랜덤하게 뿌리는 패킷에 걸려 예상치 못한 서버 다운을 겪고 있습니다. 이 부분은 보안관련 설정을 하여 보완할 수 있지만 빈도가 높지 않고 데모때만 실행하기 때문에 일단은 놔둔 상태입니다.
    1. **지하철 경로 추천 알고리즘**

버스와 마찬가지로 지하철 경로 또한 시간대별 승/하차 인원을 분석해 나온 혼잡도 수치와 총 소요시간을 계산하여 추천을 하도록 하였다.

* + - 1. **혼잡도**

지하철은 특성상 상/하선 인원 구분이 되지 않습니다. 또, 혼잡한 시간대의 열차라도 어느 차량에 타는지에 따라 혼잡도가 달라집니다. 예를 들어, 7호선 부평구청 방향의 경우 통학하면서 직접 타고 다녔을 때 3,4 번 차량은 사람이 너무 많아서 탈 수가 없었지만 1번차량의 경우 널널해서 탈 수 있었고 심지어 가끔은 앉아서 갈 수 있었습니다. 때문에 지하철은 혼잡도가 버스에 비해 부정확할 수 있습니다.

혼잡도 기준은 현재 차량의 혼잡도를 종합적으로 고려하여 전체 수치가 **33.33 미만일 경우: 여유, 차량에 탑승했을 때 앉을 수 있는 좌석이 존재**

**66.66 미만일 경우: 보통, 차량에 탑승했을 때 손잡이는 잡을 수 있음.**

**66.66 이상일 경우: 혼잡, 손잡이를 잡지 못하고 서있어야 함**.

위 수치들은 실제로 서울시에서 통합 데이터를 제공하면서 혼잡도 관련 사항에서 여유, 보통, 혼잡을 나누는 방법입니다.

지하철을 얘기하고 혼잡도를 얘기했을 때 2호선이 현재 혼잡도 분석 및 안내 서비스를 하고 있다고 하는 피드백을 많이 받았습니다. 하지만 실제로 이용해본 결과 2호선 혼잡도 서비스에 문제가 많음을 알 수 있었습니다.



그림 1 2호선 혼잡도 안내 서비스

그림 1을 보면 모든 차량에 여유라고 기록되어 있습니다. 실제로 해당 차량은 공식적인 기준에 따르면 혼잡으로 나타나야 할 정도로 사람이 많았고 이는 5,6,7,9번 차량도 마찬가지였습니다.

위와 같은 문제를 가진 채 혼잡도를 계산하기 위해 우선 사용할 수 있는 모든 지하철 승/하차 인원 통계자료를 가져와서 계산을 수행하였습니다.

첫번째로 현재 정거장에서 상/하선 별 승/하차 인원의 구분 비율은 현재 정거장에서 다음 정거장으로 넘어갔을 때 어느 방향의 인원이 더 많은지를 판단하였습니다. 부정확한 방법이고 정확도 또한 계산이 불가능하지만, 기본적으로

* + - 1. **기본적으로 많은 사람이 이용하는 노선이 존재**
      2. **1의 노선에서도 특정 시간대별로 많은 사람이 이용하는 방향이 존재**

한다는 내용을 바탕으로 모든 시간대가 아니더라도 대부분의 시간대에서는 어느 정도의 정확도를 얻을 수 있었습니다.

두번째로 현재 정거장에서 환승이 가능할 때 구분입니다. 이 자료가 환승 게이트에서 사람들이 카드를 찍었을 때 해당 내용도 통계가 이루어지는지 알 수 없기 때문에 어려웠습니다. 계산을 위해 현재 정거장에서 한 노선의 누적 승차 인원과 환승 노선의 누적 승차인원을 계산하여 각각 다음 정거장에서 누적 승차인원을 비교해 어느정도 수준의 인원이 보통 환승을 진행하는지 알아보았습니다. 예를 들어, 3호선 고속터미널 역에서 현재 데이터로는 7호선으로만 환승이 가능합니다. 이 때, 3호선의 누적 승차인원이 100이고 7호선이 50이라 했을 때 각각 잠원/반포에서 누적 승차 인원이 110/80 이고 해당 정류장에서 각각 +5/+10 정도 승/하차 인원이 존재했다고 하면 3호선에서 7호선 방향으로 많은 사람들이 환승했음을 의미하고 이를 적절하게 나눠 혼잡도를 계산하였습니다.

여러 기준을 바탕으로 많은 데이터를 이용해 혼잡도를 계산하고 실제로 경로 탐색까지 수행하였지만 여전히 불확실한 값에 의존하는 점이 높아 실제로 적용을 위해서는 실제 카드기에 카드를 찍는 사람을 실시간으로 구하면서 CCTV를 이용해 실제 탑승한 인원의 수를 비교하면서 하는 방법이 아닌 이상 정확한 인원이 구해질 수 없다는 점을 느끼게 되었습니다.

* + - 1. **경로 탐색**

우선 모든 노선과 정류장 정보는 싱글톤 객체인 SubwayInfo라는 Class에 모두 저장되도록 구현되었습니다. 각 노선 정보는 LineInfo라는 Class를 이용해 저장하고, 각 정류장 정보는 StationInfo라는 클래스를 이용하여 저장하였습니다.

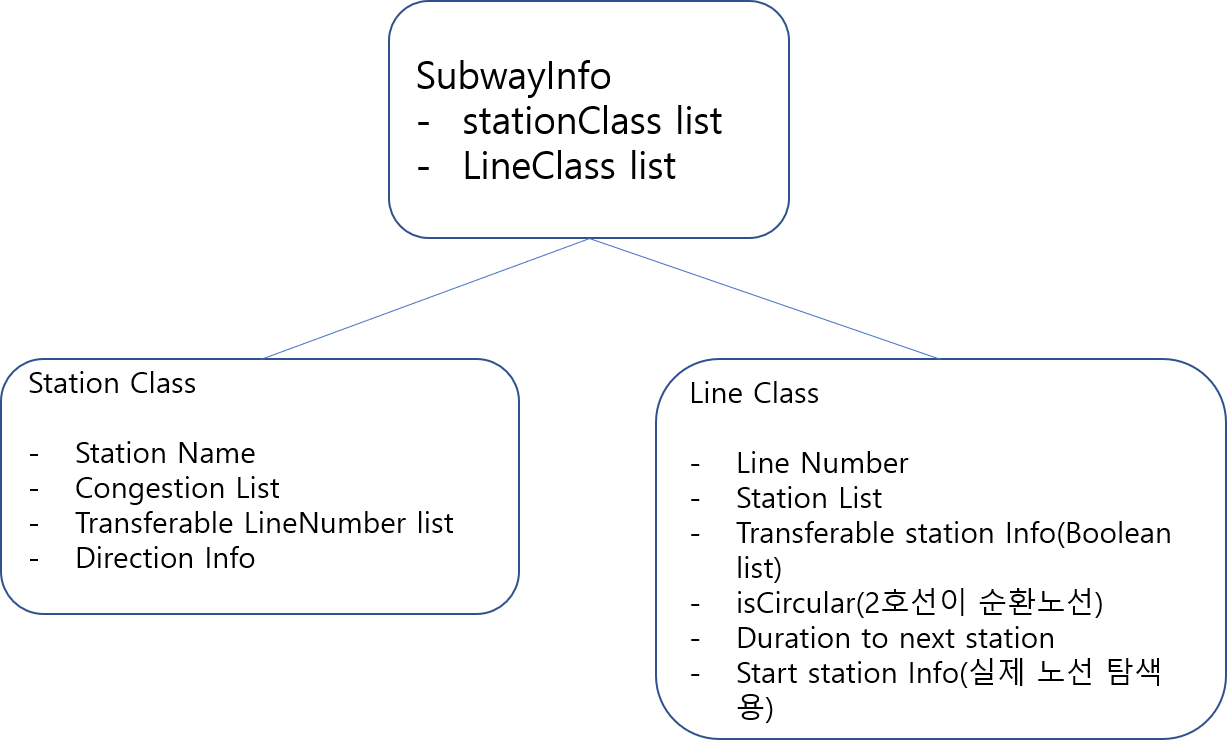


그림 2 지하철 관련 데이터 저장

모든 정거장 연결 관련 정보는 Line Class의 Station List를 순서대로 저장하여 한번에 파악할 수 있도록 짜였습니다.

위 정보를 이용하여 경로를 탐색하는 알고리즘은 FindSubwayPath Class의 search\_all이라는 함수를 이용해 Recursive로 작성되었습니다.

기본적으로 현재 프로그램은 가능한 모든 경로를 탐색한 다음 혼잡도를 기반으로 최단경로가 아닌 다른 경로를 추천해 주어야 합니다. 여기서 가장 큰 문제가 생깁니다.

1. 모든 경로의 혼잡도는 종점/기점까지 진행하고 올 경우 낮아진다.
2. 2호선의 경우 혼잡도가 떨어지는 시간까지 지하철에서 내리지 않고 계속 타고 있을 경우 혼잡도가 낮아진다. 즉 순환하는 문제가 발생한다.
3. 환승 계속 진행하면서 혼잡도가 떨어지는 노선을 선택하면 전체 노선의 혼잡도가 떨어진다.
4. 환승을 계속 진행 가능하게 할 경우 프로그램이 끝나지 않는다.

등 위와 같은 여러 문제들이 생기는데 그 중 2,4의 경우 프로그램이 끝나질 않거나 call Stack이 포화상태가 되어 프로그램이 강제로 종료되는 경우가 생겨 매우 치명적이었습니다. 이를 해결하기 위해 몇 가지 기준을 세워 경로 탐색을 수행하였습니다.

1. 어디서 출발하더라도 종점에서 하행, 기점에서 상행을 탐색할 경우탐색을 멈춘다.
2. 순환 노선의 경우 처음으로 탑승한 지점을 저장하고 해당 지점을 다음 정류장으로 탐색하려는 경우 탐색을 멈춘다.
3. 환승은 최대 2번으로 제한한다.

위 기준 중 2는 쉽게 납득할 수 있습니다. 하지만 1,3의 경우 문제가 조금 있는데

1. 1의 경우 탑승 정류장이 종점/기점의 직전 정류장일 경우 종점/기점을 들려서 오는 것이 더 효율적이라고 할 수 있을 수 있습니다. 하지만 이 경우, 합리적인 판단 기준이 서지 않기 때문에 현재 경로 탐색 알고리즘에서는 생각하지 않기로 하였습니다.
2. 3의 경우 환승 제한이 너무 적다고 할 수 있습니다. 하지만 서울 지하철을 전체적으로 분석해보면 3개의 노선을 거치면 거의 모든 지역을 갈 수 있음을 알 수 있습니다. 또, 아무리 혼잡도를 기반으로 추천한다 하여도 30분이면 갈 수 있는 거리를 1시간 30분 이상 가야 하는 거리를 추천하는 것은 굉장히 비효율적이고 합리적이지 않기 때문에 이런 경우를 줄이기 위해 환승 횟수를 최대 2번으로 제한하였습니다.

이러한 기준을 바탕으로 search\_all에 경로 탐색 알고리즘을 작성하였고 실제 수행 시간은 Debug모드/서버 에서 1초 미만으로 확인되었습니다.

* + - 1. **경로 추천**

기본적으로 모든 경로 탐색을 마치면 FindSubwayPath에서는 탐색된 모든 경로의 소요시간, 혼잡도를 받을 수 있습니다. 이를 이용해서 적절한 기준을 세워 경로를 추천해야 합니다. 이 부분은 아직 작성이 되지 않았습니다.

* 1. **Implementation 이후로 개선된 부분**
     1. **서버 구축**
     2. **안드로이드----**
     3. **안드로이드 ----**
     4. **안드로이드----**
     5. **안드로이드----**

1. **성능 평가, 비교, 최종 구현물 데모 (스크린샷같은것도 넣자)**

ㅇㅇㅇㅇ

1. **Github repository**
   1. **최종 commit 수, contributor 수, issue / pull request 수**

ㅇㅇㅇ

* 1. **다른 학생들이 등록한 issue, pull request에 대한 평가 (부적절한/불필요한 contribution은 아닌지) + 이들을 어떻게 처리했는지**

|  |
| --- |
| Issu2 1번과 3번을 등록한 학우가 2번 Pull Request를 해주었다.  <Issue>위치 권한 미설정 시 앱 튕김 현상 #1([**https://github.com/b1g4/Application/issues/1**](https://github.com/b1g4/Application/issues/1) **)**  <Issue> 출발지 미 기입시 튕김 현상 #3 (<https://github.com/b1g4/Application/issues/3> )  <Pull Request> 위치 권한 부여, 출발지 검색 오류 처리 #2 (<https://github.com/b1g4/Application/pull/2> )  위치 권한 설정을 하지 않아 사용자가 위치 권한을 거부하면 앱 튕김 현상이 일어나고 출발지를 기입하지 않고 길 찾기 버튼을 누르면 앱이 종료되는 버그가 있었다. 이를 학우분께서 몇 줄의 코드를 추가해 해결해 주었고 이를 merge하여 Android를 개선했다. |
| <Issue> 최근 검색 목록 기능 추가 #4 (<https://github.com/b1g4/Application/issues/4> ) |
| <Issue> Add the keyword function #7 (<https://github.com/b1g4/Application/issues/7> ) |
| <Issue> 주소 검색 시 지역 범위에 대한 범위 설정 #9 (<https://github.com/b1g4/Application/issues/9> )  ~~기존 Proposal에는 길 찾기 개발 범위를 서울 시내로 한정했다. 하지만 구현하는 과정에서 미처 개발을 하지 못했다. 이는 추후에 수정할 계획이다.~~ |
| <Pull Request> 최근 검색 목록 기능 추가 #5 ( <https://github.com/b1g4/Application/pull/5> |
| <Pull Request> 즐겨찾기 기능 추가 #6 (<https://github.com/b1g4/Application/pull/6> ) |
| <Pull Request> Find the Best path by using keyword. #8 (<https://github.com/b1g4/Application/pull/8> ) |
| <Pull Request> find thd best spot by using keyword #10 (<https://github.com/b1g4/Application/pull/10> ) |
| <Pull Request> Revert "find thd best spot by using keyword" #11 (<https://github.com/b1g4/Application/pull/11> ) |
| 풀리퀘 받은 사람이 알아서 링크 넣고 설명달고 할 것!!!!!!!!!  옮기다가 귀찮아서 관뒀으니까 깃허브 필수로 확인할 것!!!!!!!!!! |

* 1. **이외에 얼마나 Github repository를 잘 활용했는지 보여줄 것**

기본적으로 팀원 전부가 Git, Github에 익숙하지 않아서 실수도 잦고 전체적인 활용도가 숙련자에 비해 떨어질 수 밖에 없습니다.

처음으로 깃허브 사용 수칙을 정했을 때, Branch를 따서 활용하는 것이 익숙하지 않고, merge도중 여러 문제가 발생하여 Github에는 master branch만 활용하고, Local Git에서 Branch를 따서 작업하도록 하였습니다.

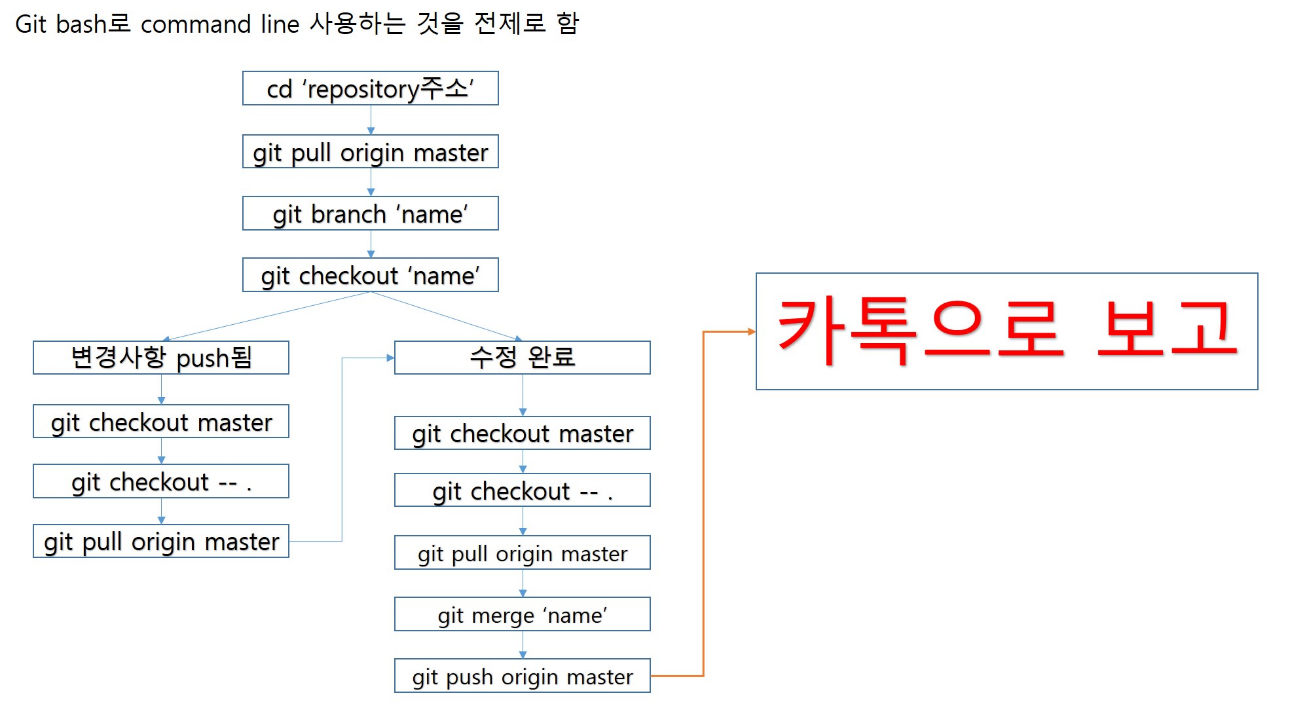


그림 초기 협업 방식

하지만 마스터 브랜치에 코드가 업로드 되면서 되던 코드가 되지 않을 때도 있고, 마스터 브랜치에 릴리즈 불가능한 버전이 자꾸 덮어써지면서 코드가 지저분해지는 문제가 생겼습니다. 때문에 방식을 바꿔 모두 Github 브랜치에서 작업한 뒤 모든 작업 내용이 안정적이고 검증이 가능할 때 마스터 브랜치에 머지를 하도록 진행하였습니다.

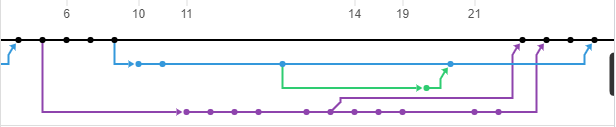


그림 바뀐 작업 내역 2

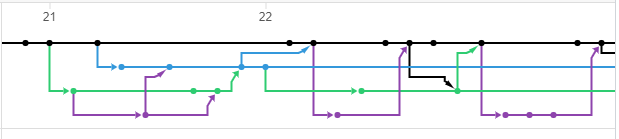


그림 바뀐 작업 내역 1

이렇게 프로젝트를 진행하면서 점점 프로젝트 진행이 안정적으로 바뀌었습니다.

DataParsing Repository를 활용하면서 겪은 시행착오를 바탕으로 Application 개발을 진행하면서는 시작부터 안정적인 개발이 가능했습니다.

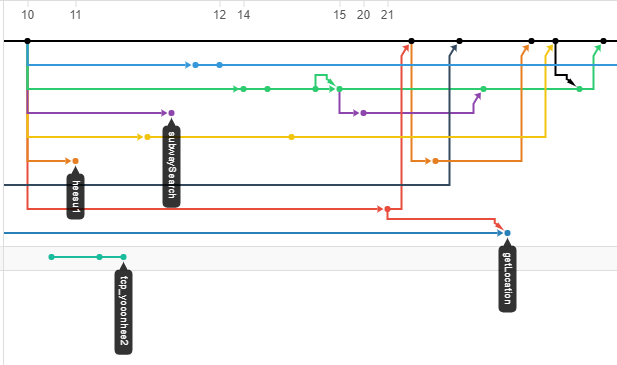


그림 Application Repo 작업 내역

1. **기타**
   1. **업무분담**

기본적으로 모든 업무는 팀원이 함께 진행하며, 진행할 때 메인으로 맡는 사람을 정해 분담하였다.

|  |  |
| --- | --- |
| **역할** | **팀원** |
| 팀장 | 이설희 |
| 개발자 | 데이터 파싱 - 김윤희, 전희수  데이터 분석 - 이설희, 현도연  버스경로 구현- 김윤희, 허정우, 현도연  지하철 구현 - 허정우  안드로이드 구현 - 전희수, 이설희  서버구축 - 김윤희, 현도연, 허정우 |
| 문서작업 | Proposal report- 전희수(메인), 김윤희, 이설희, 허정우, 현도연  Final report - 김윤희, 이설희. 전희수, 허정우, 현도연 |
| 발표자 | Proposal - 전희수  Final - 김윤희 |

3-a 현도연

3-b 이설희

4 허정우

5-a 김윤희

5-c 전희수

