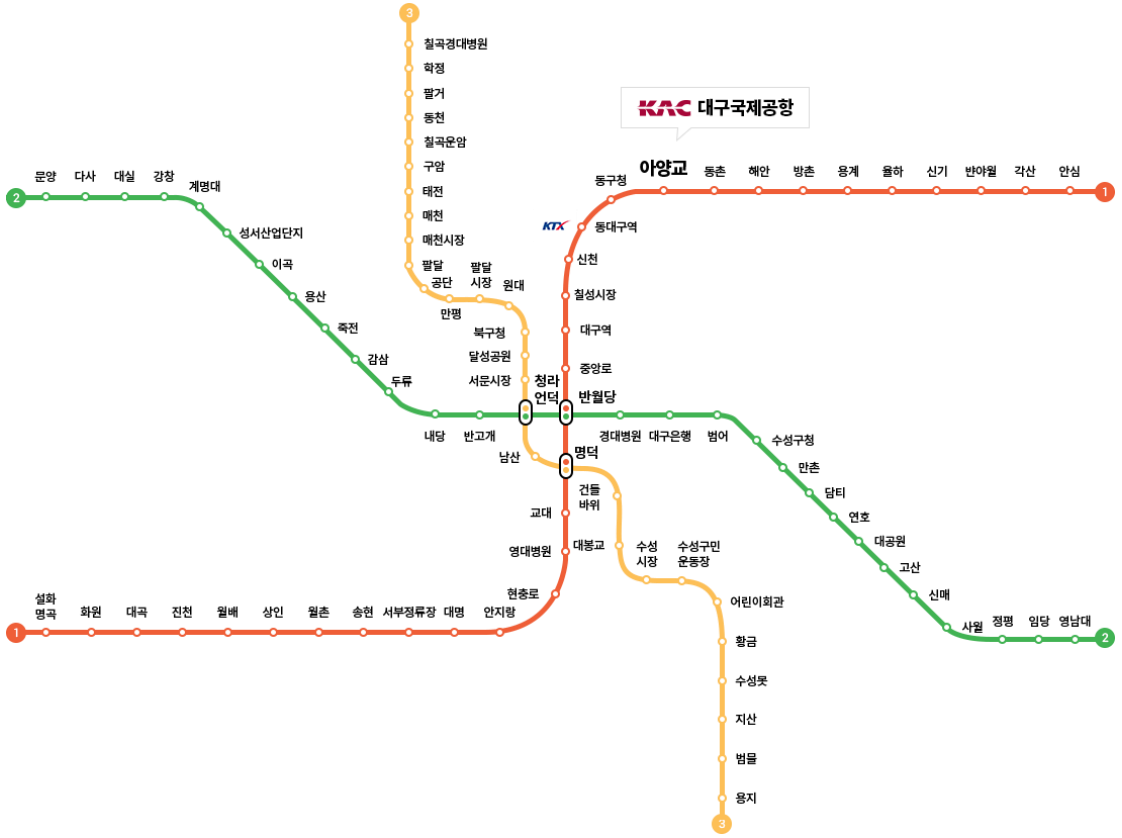
어드벤처디자인 2단계

4번 지하철 길찾기

20200679 안예진

목차

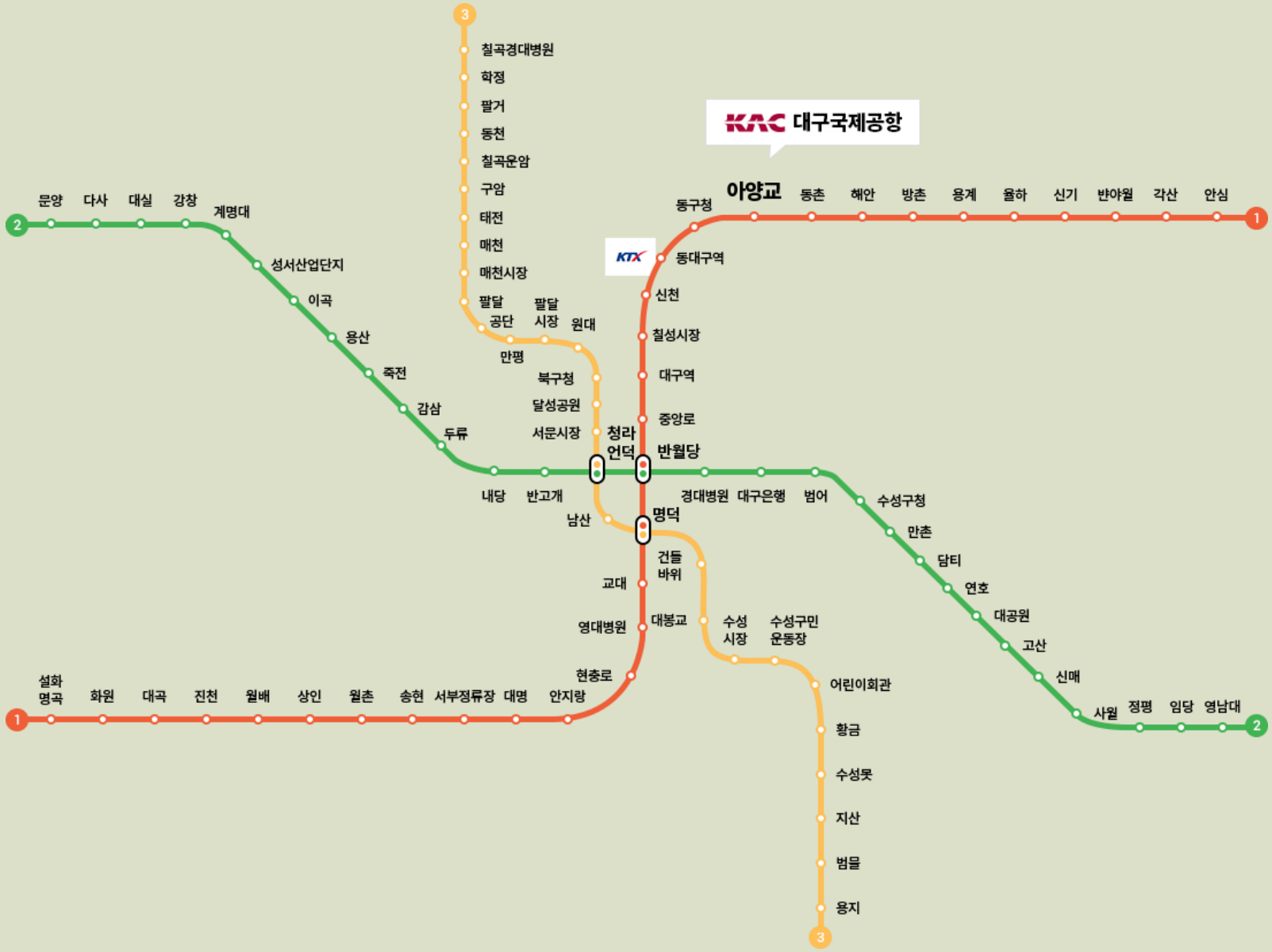
문제 설명



- **대구 지하철의 경로**를 검색하는 프로그램
- 각 역 간 걸리는 시간은 **2분**으로 하되, 환승역은 **환승 시간을 고려**하여 8분 추가
- **출발역**과 **도착역**, **환승역**의 정보를 포함한 경로 정보를 출력

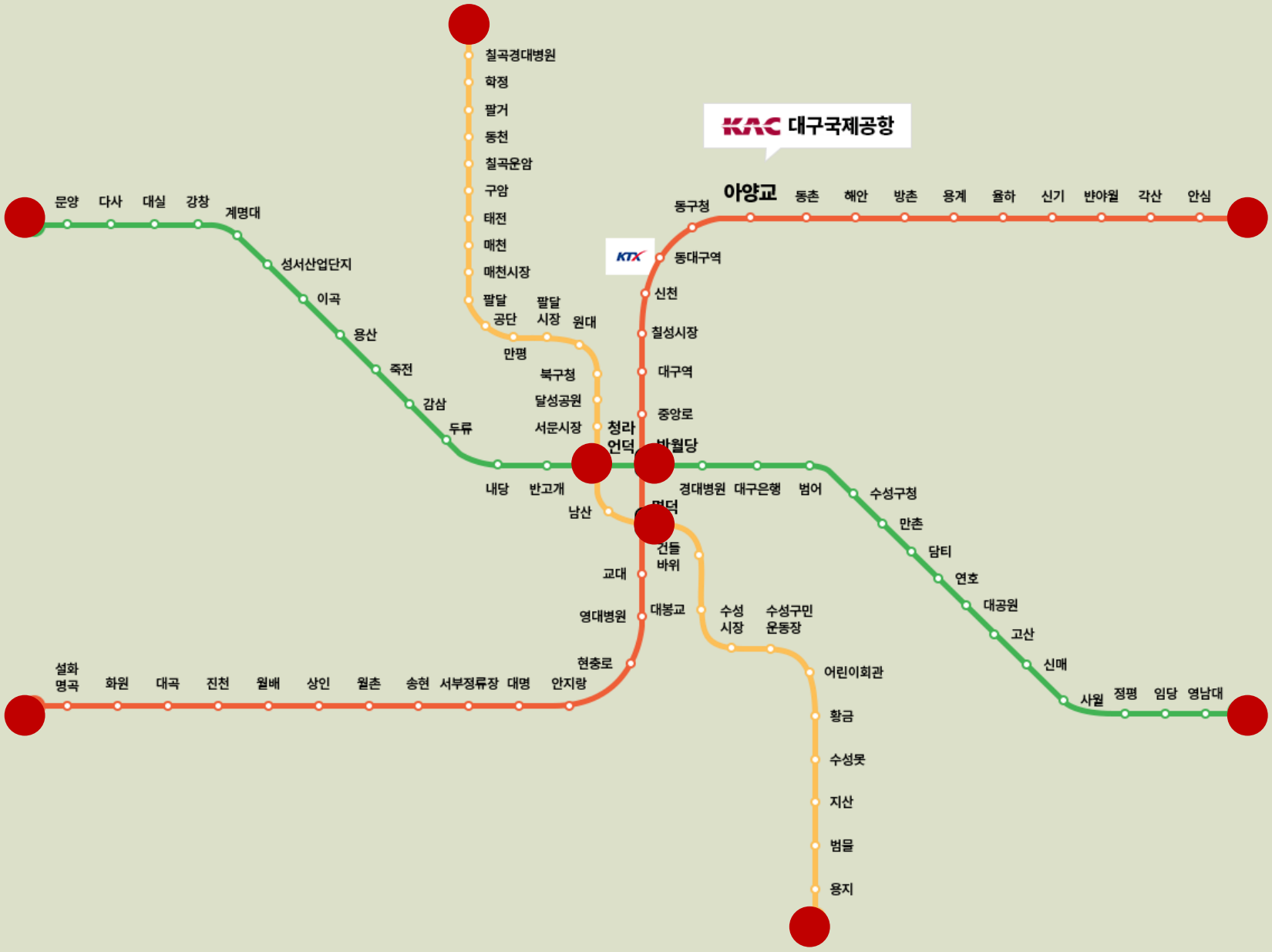
지하철 길찾기

문제 설명


3
칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
KAC 대구국제공항
아양교
동구청
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
1
KTX
동대구역
신천
칠성시장
대구역
중앙로
반월당
2
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
내당
반고개
경대병원
대구은행
범어
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
남산
명덕
건들
바위
교대
영대병원
대봉교
수성
시장
수성구민
운동장
현충로
어린이회관
황금
수성못
지산
범물
용지
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑

지하철 길찾기

문제 설명


칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
KAC 대구국제공항
아양교
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
동구청
KTX
동대구역
신천
칠성시장
대구역
중앙로
바월당
경대병원
대구은행
범어
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
내당
반고개
남산
건들
바위
교대
영대병원
대봉교
수성
시장
수성구민
운동장
현충로
어린이회관
황금
수성못
지산
범물
용지
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑

지하철 길찾기

문제 설명


1
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
내당
반고개
2
칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
남산
KAC 대구국제공항
아양교
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
3
동구청
KTX
동대구역
신천
칠성시장
대구역
중앙로
바월당
경대병원
대구은행
범어
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
4
7
8
9
건들
바위
교대
영대병원
대봉교
현충로
수성
시장
수성구민
운동장
어린이회관
황금
수성못
지산
범물
용지
5
6
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑

지하철 길찾기

문제 설명


1
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
2
칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
7
내당
반고개
남산
8
바월당
경대병원
대구은행
범어
9
교대
영대병원
대봉교
현충로
6
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑
KAC 대구국제공항
아양교
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
3
동구청
KTX
동대구역
신천
칠성시장
대구역
중앙로
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
4
건들
바위
수성
시장
수성구민
운동장
어린이회관
황금
수성못
지산
범물
용지
5

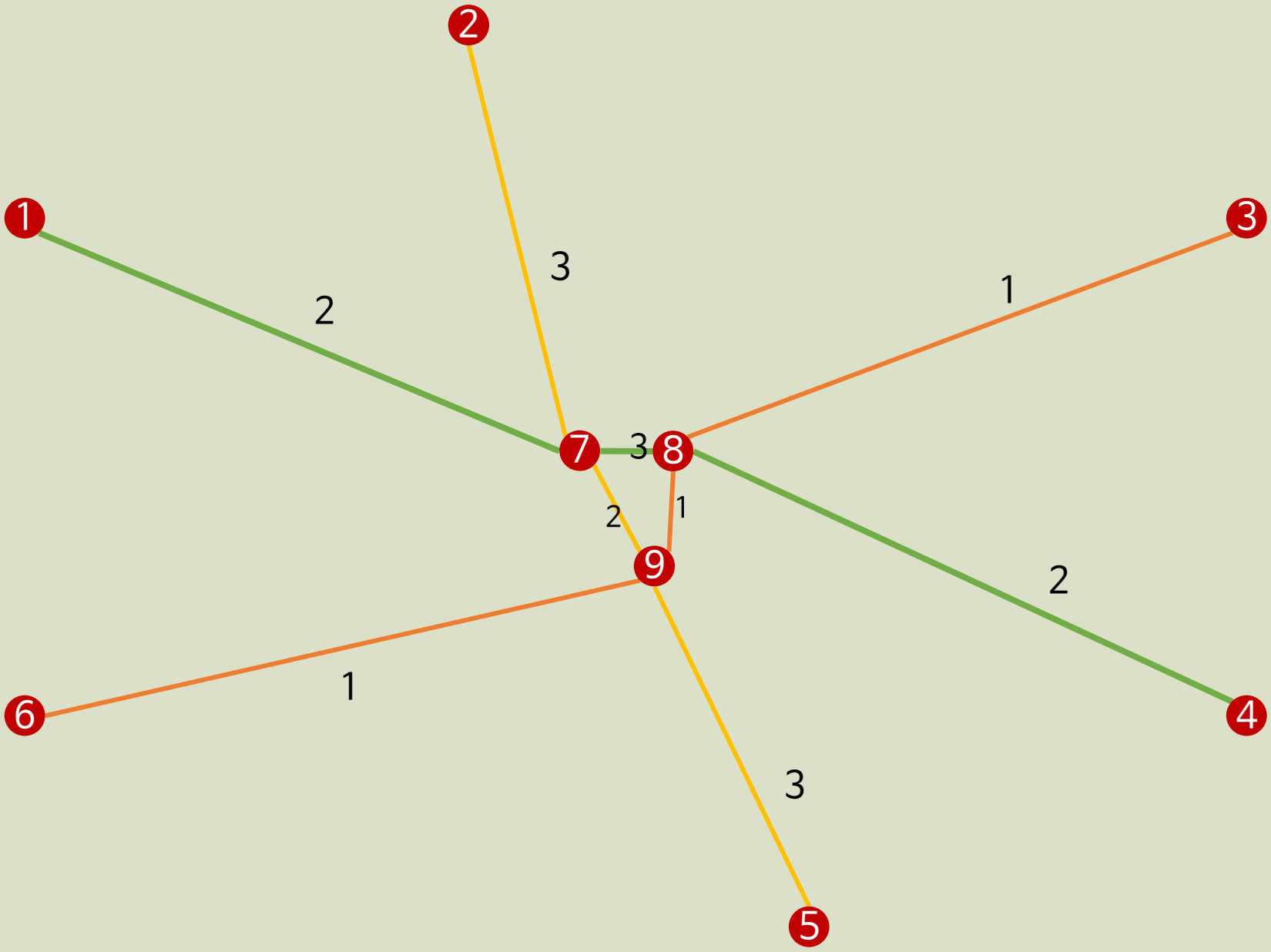
문제 설명

130 안심
131 각산
132 반야월
133 신기
134 율하
135 용계
136 방촌
137 해안
138 동촌
139 아양교
1310 동구청
1311 동대구
1312 신천
1313 칠성시장
1314 대구
1315 중앙로

문제 설명


1
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
2
칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
7
내당
반고개
남산
8
바월당
경대병원
대구은행
범어
9
교대
영대병원
현충로
6
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑
KAC 대구국제공항
아양교
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
3
동구청
KTX
동대구역
신천
칠성시장
대구역
중앙로
건들
바위
대봉교
수성
시장
수성구민
운동장
어린이회관
황금
수성못
지산
범물
용지
5
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
4
호선 노드 리스트 이름
1 3 0 안심

문제 설명



설계 변경

```
class subwayGraph
{
private:
string subway;
stationNode* graph;
int stationCount;
}
```

```
class Station
{
private:
   int subwayNum;
   int nodeNum;
   int nodeList;
   string name;
}

struct Node {
private:
   int id;
   int subwayNum;
   vector<string> str;
   vector<Station> log;
}
```

설계 변경

```
vector<int> dfs(int x, int goal) {
   visited[x] = true;

   if 출발역 == 도착역{
      int count = 지나온 노드들의 길이

      for 지나온 노드들의 길이 {
         각 노드들을 순서대로 입력(push_back)
      }
   }

   for 각 그래프 노드.size() {
      int y = graph_[x][i].first;
      if (!방문한 노드) {
         dfs(y, goal);
         visited[y] = false;
}
```

구현 내용 : 그래프 순회

```
vector<int> dfs(int x, int goal) {
    visited[x] = true;

    line.push_back(x);

    if (x == goal) {
        int count = line.size();

        for (int i = 0; i < count; i++) {
            route.push_back(line[i]);
        }
    }

    for (int i = 0; i < graph_[x].size(); i++) {
        int y = graph_[x][i].first;
        if (!visited[y]) {
            dfs(y, goal);
            visited[y] = false;
        }
    }
    line.pop_back();
    return route;
}
```

x 방문으로 처리(true)

x를 vector에 새로 삽입

만약 x가 도착역에 해당하는 노드와 동일한 경우,
vector의 크기를 count에 저장
반복문을 활용해 line의 원소를 route에 저장

노드 원소를 하나씩 꺼내서 y에 저장
만약 y가 방문하지 않은 노드라면 재귀함수 호출
dfs에서 빠져나오면 해당 노드는 방문하지 않은 것으로 함 (모든 경로 출력)

구현 내용 : 파일 입력

```
int count = 1;
while (getline(fin, line)) {
    istringstream buffer(line);
    buffer >> subwayNum >> nodeNum >> nodeList >> name;

    if (nodeNum == count) {
        node[count].setLog(subwayNum, nodeNum, nodeList, name);
    }
    if (subwayNum == 0)
        count++;

    if (name == arr.getName()) {
        arr.setSubwayNum(subwayNum);
        arr.setNodeNum(nodeNum);
        arr.setNodeList(nodeList);
    }
    if (name == start.getName()) {
        start.setSubwayNum(subwayNum);
        start.setNodeNum(nodeNum);
        start.setNodeList(nodeList);
    }
}
```

텍스트파일을 한 줄씩 읽어들여 공백을 기준으로 각 변수에 저장

각 count에 맞는 노드에 변수들의 집합 객체를 vector로 쌓음

만약 출발역이나 도착역과 동일한 이름의 역을 순회할 경우,
각 객체에 나머지 정보 저장
(노선번호, 노드번호, 노드리스트 정보)

구현 내용 : 역 찾기

```
if (start.getNodeNum() == arr.getNodeNum()) {
    stationCnt = abs(start.getNodeList() - arr.getNodeList());

    cout << start.getName() << "역(" << start.getSubwayNum() << "호선) - "
        << arr.getName() << "역(" << arr.getSubwayNum() << "호선) :" << stationCnt << "역, "
        << stationCnt * 2 << "분 소요" << endl;
}
```

구현 내용 : 역 찾기

```
route_ = dfs(node[start.getNodeNum()].getId(), node[arr.getNodeNum()].getId());

for (int i = 0; i < route_.size(); i++)
{
    if (route_[i] == node[start.getNodeNum()].getId())
    {
        total += abs(start.getNodeList() - node[start.getNodeNum()].getLogSize());
        continue;
    }

    if (route_[i] == node[arr.getNodeNum()].getId())
    {
        total += (node[arr.getNodeNum()].getLogSize() - arr.getNodeList());

        cout << start.getName() << "(" << start.getSubwayNum() << "호선) - ";

        for (int j = 0; j < graph_[route_[i]].size() + 1; j++) {

            if (graph_[route_[i-1]][j].first == route_[i])
                tmp = j;
        }
```

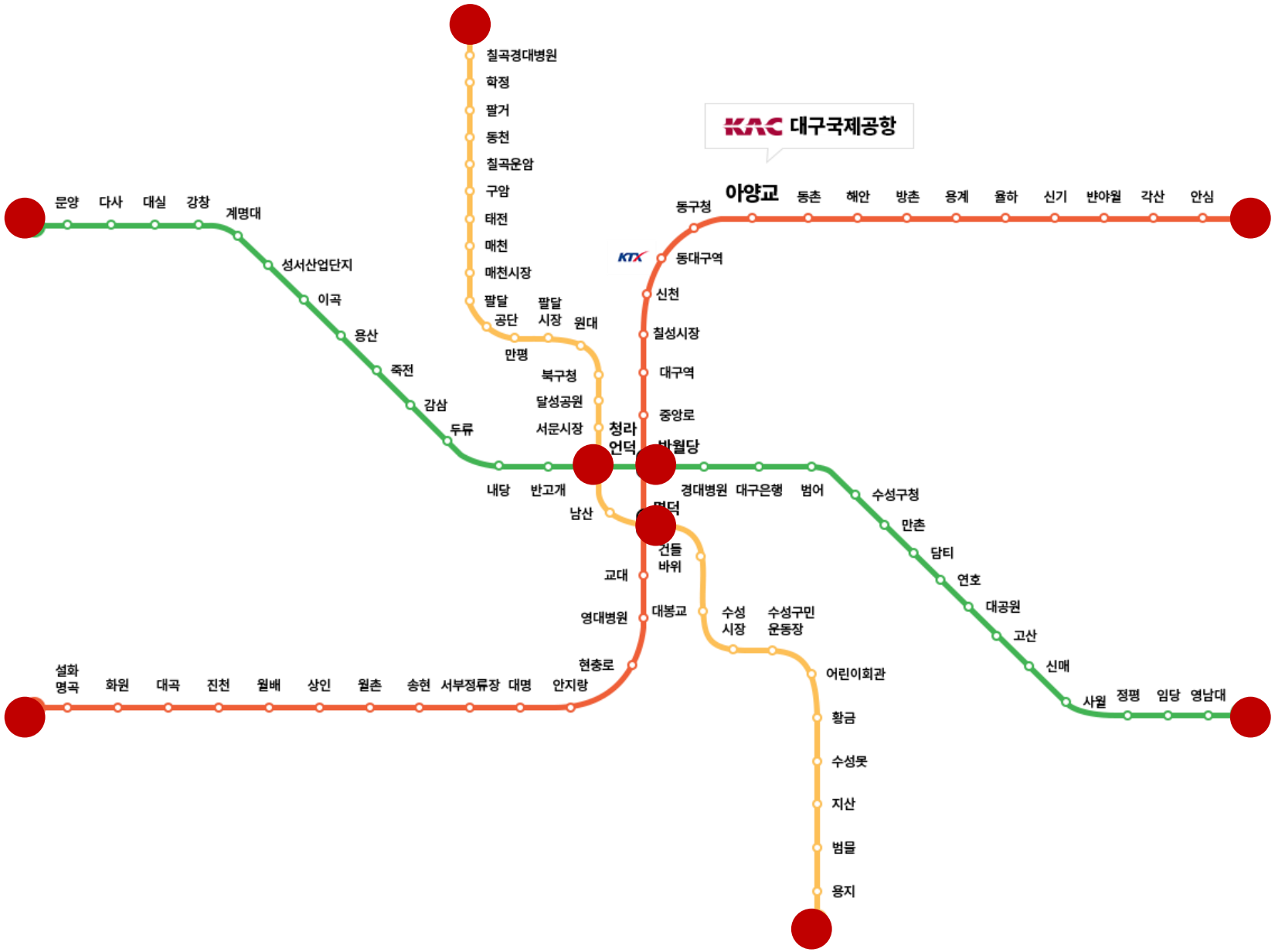
dfs 결과를 route_로 받음

route의 노드가 시작 node정보와 일치할 경우
시작 노드에 저장된 배열에서 환승역까지의 거리를 구함

환승역이 포함될 경우, graph의 직전 노드와 다음 노드를 통해
그래프의 간선 정보를 구함(index)

지하철 길찾기

문제 설명


칠곡경대병원
학정
팔거
동천
칠곡운암
구암
태전
매천
매천시장
팔달
공단
팔달
시장
원대
만평
북구청
달성공원
서문시장
청라
언덕
내당
반고개
남산
KAC 대구국제공항
아양교
동구청
동촌
해안
방촌
용계
율하
신기
반야월
각산
안심
KTX
동대구역
신천
칠성시장
대구역
중앙로
반월당
경대병원
대구은행
범어
수성구청
만촌
담티
연호
대공원
고산
신매
사월
정평
임당
영남대
문양
다사
대실
강창
계명대
성서산업단지
이곡
용산
죽전
감삼
두류
교대
건들
바위
영대병원
대봉교
수성
시장
수성구민
운동장
현충로
어린이회관
황금
수성못
지산
범물
용지
설화
명곡
화원
대곡
진천
월배
상인
월촌
송현
서부정류장
대명
안지랑

구현 내용 : 역 찾기

```
    for (int j = 0; j < route_name.size(); j++) {
        if (graph_[route_[j]][tmp].second ≠ graph_[route_[j + 1]][tmp].second) {
            cout << route_name[j] << "(환승 " << graph_[route_[j]][tmp].second << "호선) - ";
            trans++;
        }
    }

    cout << arr.getName() << "(" << arr.getSubwayNum() << "호선) :" << total << "역, "
        << trans<< " 환승, " << total * 2 + ((trans+1)*8) << "분 소요" << endl;

    total = 0;

    vector<string>().swap(route_name);
    vector<int>().swap(route_num);
    trans = 0;

    continue;
}
```

dfs가 모든 경로를 같은 벡터에 저장하므로
한 경로를 출력하고 나면 이전에 기록된 이름들을 모두 삭제

```
total += node[route_[i]].getLogSize();
if (route[i - 1] == 7 && route[i] == 8)
    total--;
route_name.push_back(node[route_[i]].getNodeName());
route_num.push_back(route_[i]);
```

모든 이동이 배열에 접근
node[7]에서 node[8]으로 이동할 경우 1 감산

감사합니다