2018 전기 졸업 과제 중간 보고서

**소스 코드 표절 탐지 및 추적 시스템**

**‘LEECH’**

지도 교수: 조환규 교수님

소속: 전기컴퓨터공학부

팀 이름: 달려라 한이(A)

작성자: 201524551 이하원

201524617 한성혜

**목 차**

**1. 과제 소개** **3**

1.1 과제 선정 배경 3

1.2 시스템 흐름도 3

**2. 수정사항 및 추가사항** **4**

2.1 실시간 유사도 측정 4

2.2 유사도 분석 옵션 4

**3. 설계 상세화** **5**

3.1 함수 실행 순서를 고려한 프로그램 DNA 생성 5

3.2 소스코드 유사도 계산 6

3.3 유사도 그래프 생성 7

3.4 웹 진행 상황 7

**4. 추진 체계** **10**

# 1 과제 소개

* 1. **과제 선정 배경**

컴퓨터 공학 학생에게 코딩 과제는 개발자로서의 역량을 기르는 중요한 역할을 한다. 하지만 과제 검사 시 변수명조차 바꾸지 않은 표절한 코드를 쉽게 발견할 수 있다. 이러한 표절 행위는 개발자로서의 실력을 향상할 수 없게 만들 뿐만 아니라 노력에 맞는 정당한 점수 분배를 어렵게 한다. 본 과제에서는 이러한 상황을 개선하기 위해 함수 호출에 유의하여 토큰을 생성하는 실행 순서 추출 기법과 토큰의 등장 횟수를 고려하여 유사치를 내는 토큰 빈도 가중치를 적용한 소스 코드 표절 탐색 시스템을 제안하며 이름은 ‘LEECH’라 한다.

* 1. **시스템 흐름도**

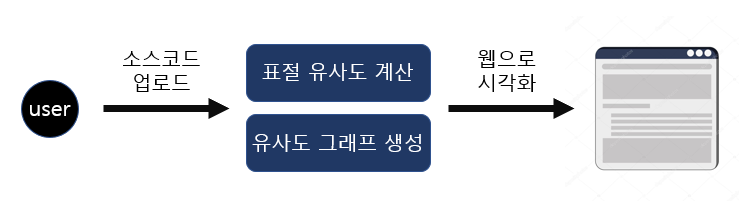


그림 1. 전체 시스템 흐름도

그림?과 같이 user가 웹을 통해 표절을 확인할 다량의 소스 코드 쌍을 업로드 하면 시스템에서 표절 유사도를 계산하고 유사도 그래프를 생성 하여 해당 결과를 웹으로 보여 준다.

1. **수정사항 및 추가사항**

|  |
| --- |
| **2.1  실시간 유사도 측정**  기존의 웹에서는 학생들의 과제를 모두 모아 zip 형식으로 업로드를 하면 과제 유사도 결과를 한번에 보여주는 시스템이었다. 하지만 과제를 따로 모으고 또, LEECH에 그 파일을 다시 업로드하는 과정이 번거로울 것이라 생각해 LEECH에서 학생들에게 직접 과제를 제출할 수 있도록 개선했다. 관리자는 현재 과제를 제출한 학생들의 유사도 결과의 변화를 실시간으로 확인할 수 있다.  또한, 이 과정에서 학생이 과제를 제출한 시간이 기록되면서 표절한 사람과 표절 당한 사람을 비교적 정확하게 분석할 수 있다.  **2.2  유사도 분석 옵션**  관리자는 학생들의 과제 유사도 결과를 확인하는 과정에서 다양한 분석 옵션을 선택할 수 있도록 옵션 사항을 추가했다. 적용할 수 있는 옵션으로는 표절을 체크할 언어 선택(c/c++, jajva, python), 토큰 빈도 가중치 적용/미적용 등이 있다. 옵션을 선택함으로써 더 다양하고 관리자가 원하는 정확한 분석 결과를 볼 수 있다. |

**3 설계 상세화**

**3.1 함수 실행 순서를 고려한 프로그램 DNA 생성**

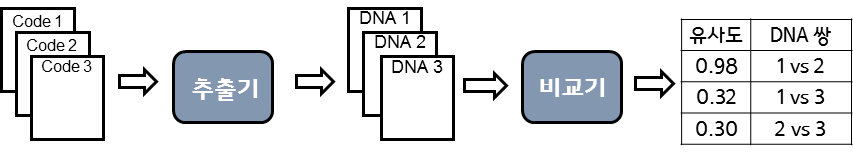


그림 2. ‘표절 유사도 계산’ 구조

그림 2와 같이 사용자에 의해 업로드된 다량의 소스 코드는 추출기와 비교기를 통해 쌍을 이뤄 유사도를 계산해 낸다. 추출기에서는 lex를 통한 소스 코드 토큰을 추출해 내는 작업을 하며 비교기에서는 추출기에서 만든 토큰값을 불러와 유사도 측정 알고리즘을 통해 유사도를 계산한다.

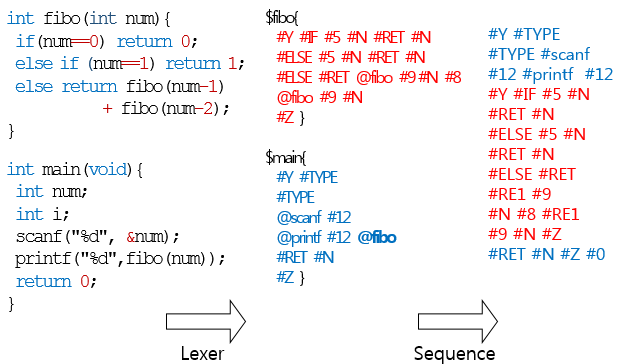


그림 3. Lexer와 Sequence를 이용한 프로그램 DNA 생성

**3.1.1 Lexer**

Lexer(추출기)는 소스코드를 처음부터 끝까지 읽으면서 for, int, 변수 등을 토큰으로 변환해 주는 작업을 한다. 이때 그림?과 같이 함수 생성의 경우 ‘$’, 함수 호출은 ‘@’, 그 외 모든 경우는 ‘#’이 토큰 앞에 위치하여 이들을 구분 할 수 있다. 변수명의 경우 ‘#ID’라는 토큰으로 변환한다.

**3.1.2 Sequence**

Lexer에서 생성된 토큰들을 함수 실행 순서를 고려해 하나의 스트링을 만들며 이를 프로그램DNA라 칭한다. 그림?에서 파란 토큰들 사이에 fibo함수에 해당하는 빨간 토큰들이 삽입된 것을 확인 할 수 있다. 생성된 프로그램DNA 쌍을 얼마나 유사한지 비교하여 유사도를 측정한다.

**3.2 토큰 빈도 가중치를 이용한 소스코드 유사도 계산**

생성된 프로그램 DNA는 비교할 다른 프로그램 DNA와 쌍을 이뤄 알고리즘을 통해 유사도를 낸다. 알고리즘은 최장 공통 문자열 추출에 많이 쓰이는 LCS(Longest Common Subsequence)알고리즘을 사용했다. 단 가산점을 줄 때 동일한 점수를 주는 것이 아니라 소스 코드에서 해당 토큰이 나오는 정도를 고려하여 유사도를 내는 ‘토큰 빈도 가중치’ 기술을 적용했다. 즉 빈번히 등장하는 토큰이 매치될 시 낮은 가중치를 주며, 드물게 나오는 토큰에 대해서는 높은 가중치를 주는 것이다. 사용한 가중치 계산 값은 그림 4과 같다. 표1은 그림3의 fibo.c 코드의 토큰에 따른 가중치 값으로, 숫자에 해당하는 토큰인 ‘N’의 경우 빈번히 나왔기에 낮은 가중치를 보여주며 한번 등장한 ‘printf’, ‘scanf’ 등은 가중치 ‘5’로 높은 값을 보여주는 것을 알 수 있다. ‘토큰 빈도 가중치’를 적용함으로써 보다 유사도 측정의 정확도를 높힐 수 있었다.

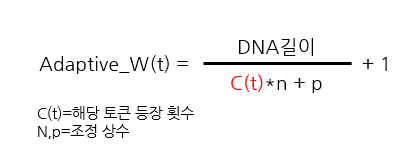


그림 4. 토큰에 따른 가중치 계산 함수

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 토큰 명 | 가중치 | 토큰 명 | 가중치 | 토큰 명 | 가중치 |
| IF | 5 | Z | 3 | TYPE | 3 |
| RET | 2 | ELSE | 3 | printf | 5 |
| scanf | 5 | Y | 3 | N | 1 |
| RE1 | 3 | 5 | 3 | 8 | 5 |
| 9 | 3 | 12 | 3 |  |  |

표 1. Fibo.c 코드의 토큰에 따른 가중치 값

**3.3 유사도 그래프 생성**

위의 3.1과 3.2의 과정으로 도출된 학생들간의 유사도 결과를 분석하기 쉽게 시각화한다. 소스 코드 유사도 결과는 json 파일로 전체 학생들의 이름과 각각의 유사도 관계를 볼 수 있다. 이 파일을 먼저 제출한 학생을 기준으로 유사도가 비슷한 학생을 모아서 phylogenetic tree 형태로 출력한다. Pylogenetic tree는 d3 라이브러리를 사용했다. 현재는 1개의 계층까지만 유사도 관계를 보여주지만 이후에는 더 심화시켜서 여러 개의 계층으로 표현할 것이다.

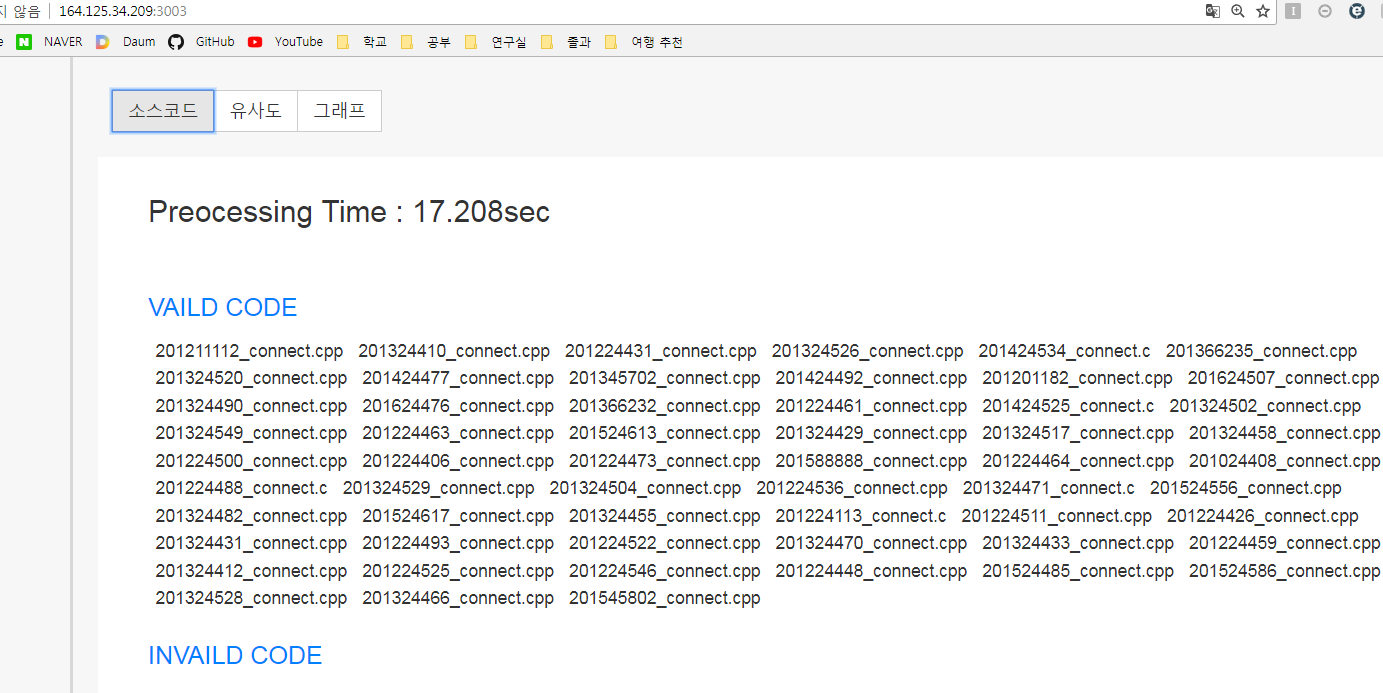
**3.4 웹 진행 상황**

현재 웹은 다량의 소스코드가 담긴 zip파일을 업로드 할 시 ‘소스 코드’, ‘유사도’, ‘그래프’ 버튼을 통해 각각의 결과를 확인 할 수있다. 구현은 python의 flask와 js, ajax를 사용했다.



< 웹 메인 페이지 >

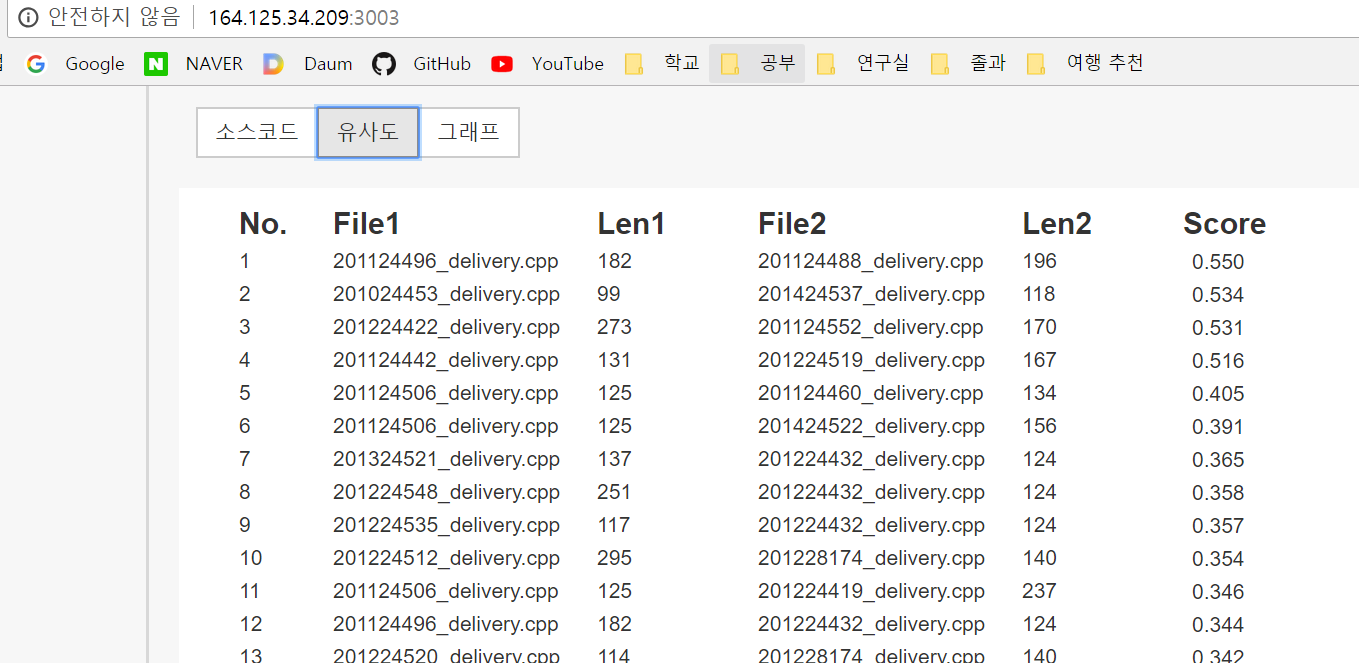
파일 선택 버튼을 통해 파일을 업로드 한다.



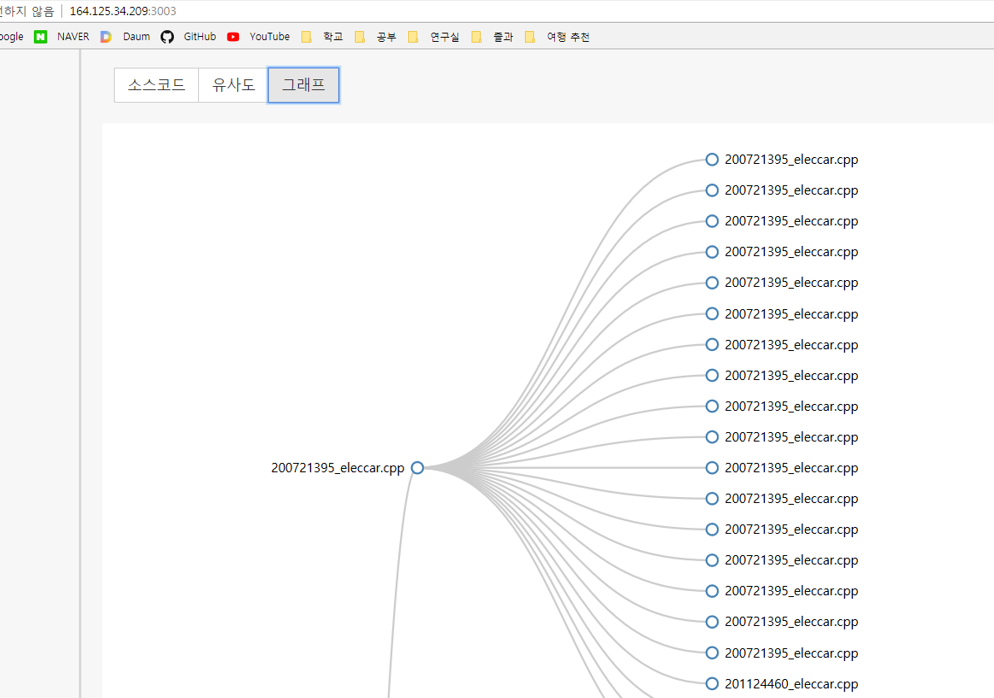
< 파일 업로드 결과 페이지 >

시스템 상의 문제로 인해 ‘LEECH’가 유사도를 계산 하지 못하는 코드의 경우

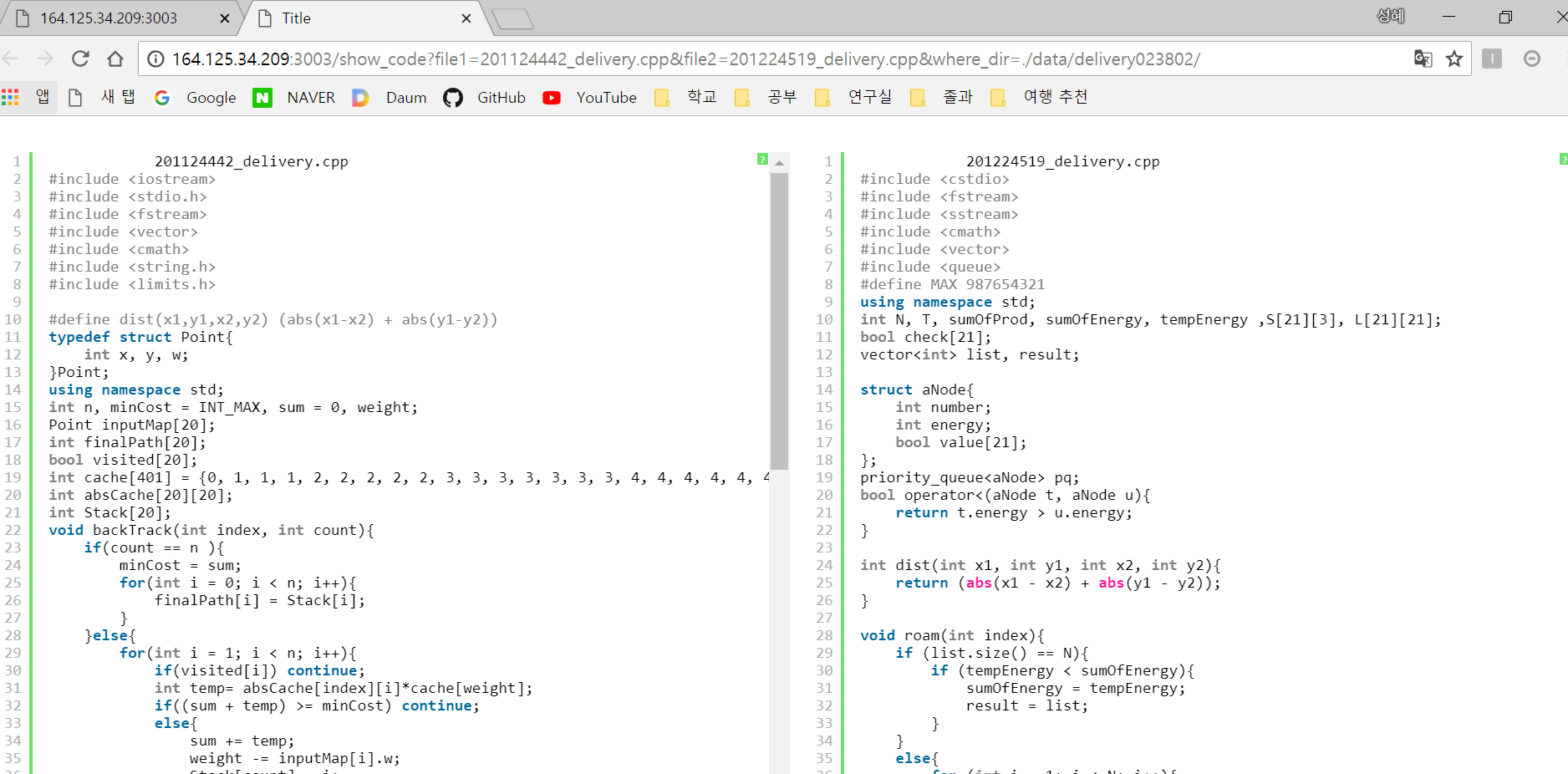
‘invalid code’에 들어간다.



< 유사도 확인 페이지 >



< 유사도 그래프 확인 페이지 >



< 유사도 확인 페이지에서 각 열의 score를 클릭하면 해당 소스 코드 쌍을 확인할 수 있다.

**4 추진 체계**

**4.1 구성원별 진척도**

|  |  |
| --- | --- |
| **구성원** | **역할** |
| **한성혜** | **- C/C++, JAVA 토큰 분리**, **python 토큰 분리**  - **유사도 측정 알고리즘 및 토큰 빈도 가중치 구현**  - **웹 학생 모드 구현** |
| **이하원** | - **유사도 그래프 생성**  - **웹 관리자 모드 구현**  - **전체 WEB UI** |
| **공통** | - 보고서 및 PPT 작성  - 표절 데이터 제작  - 테스트 |

**- 한 것** **- 해야할것**

표 2. 구성원별 역할 분담

**4.2 추진 일정**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **월**  **진행 항목** | **~5** | | | | **6** | | | | **7** | | | | **8** | | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| C/C++ 토큰 분리 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 유사도 측정 알고리즘 개발 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| JAVA 토큰 분리 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 유사도 그래프 생성 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 웹 듀얼 모드 제작 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 유사도 측정 옵션 적용 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 웹 디자인 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 데이터 베이스 제작 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 테스트 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

표 3. 추진 일정