**2017년 1학기**

**설계프로젝트C 3차 보고서**

**DBLP 데이터를 이용한 SNS 프로그램 구현**

**및 프로그램 개선**

제출일: 2017년 06월 21일

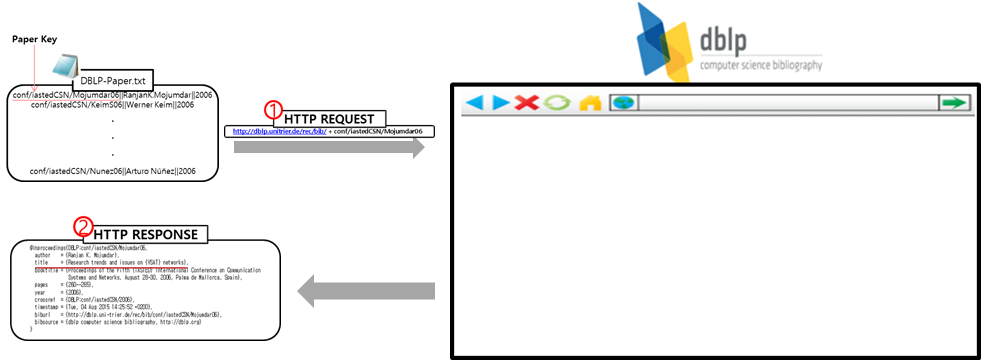
|  |  |
| --- | --- |
| 담당 교수 | 이영구 교수님 |
| 조명 | 한조 |
| 조원 | **임호준 권륜환** |
| 연락처 | EMAIL  권륜환:  임호준: lhj940825@gmail.com  C.P  권륜환:  임호준: 010-6689-1070 |

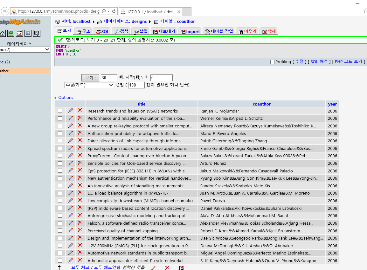
**차례**

**최종 구현된 프로그램 기능**

* DB와 크롤링을 이용한 실시간 파일 스트리밍
* 기본 자료구조
* GUI 개선
* 그래프 시각화 개선 - 가중치에 따른 edge표시
* Top-K Graph 성능 보완
* MinHeap사용 -> PartialSort사용
* Show Chain
* 논문 정보 시각화
* 슬라이더를 이용한 기간별 논문 열람
* 구독 기능
* 속성 기반 논문 검색기능(제목, 년도, 논문 내용)
* 공동 저자 기반 논문검색
* 공동 작업할 저자 추천 기능
* 알 수도 있는 저자 추천
* 파일 스트리밍

구현된 프로그램은 paperkey.txt에서 paper key를 불러온 뒤 이를 이용해서 dblp 사이트에 정보 열람 요청을 한다.





**<Data Base>**

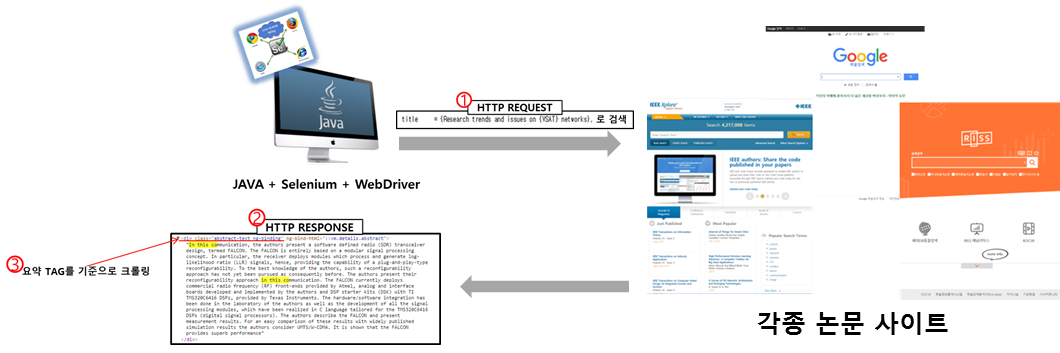
* **특정 정보를 dblp사이트에서 가져와서 mySQL에 저장하는 과정**

특정 키에 대한 데이터가 위의 “HTTP RESPONSE”에서 처럼 출력이 되면, 필요한 정보인 title정보만을 추출한 뒤, 이를 DB에 저장하는 과정을 취하고 있다. paper key를 통해 해당 논문의 세부정보를 불러오게 되어 특정 논문의 제목과 그 논문을 같이 집필한 연구진들까지 DB에 저장하였다.

또한 단순 file stream을 이용했던 1차 프로젝트에서는 실시간 스트리밍 시 특정 시간이 흐른 뒤 기존 데이터를 update할 때, 기존 데이터를 어디까지 읽었는지를 알 수 있는 Iterator를 txt파일에 둘 수가 없었기 때문에, 완전한 의미의 실시간 파일 스트리밍을 구현했다고 보기 어려웠다. 최종구현된 프로그램에서는 DB에 저장되어있는 데이터를 년도를 기준으로 불러오기 때문에 지난 1차때보다 많은 개선을 이루었다.

* 크롤링

대부분의 논문 사이트 들이 브라우저에 HTMP이 로딩된 이후 내부 자바스크립트로 다시 서버에 논문 정보를 요청해 받아오는 방식을 취하였기 때문에 이러한 정보를 단순한 HTTP Connection으로는 받아올 수 없었다. 따라서 Selenium(셀레늄) Library와 browser driver를 이용해 동적으로 받아와 처리하였다.

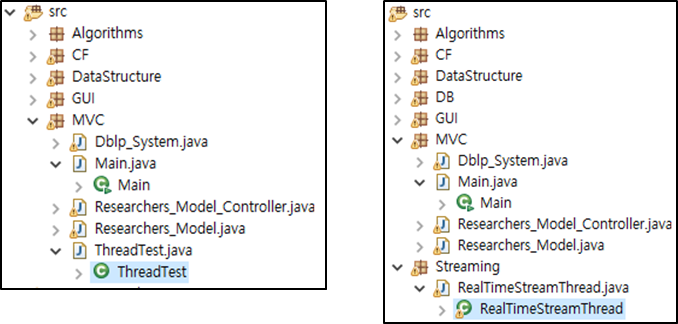


* **크롤링 과정**

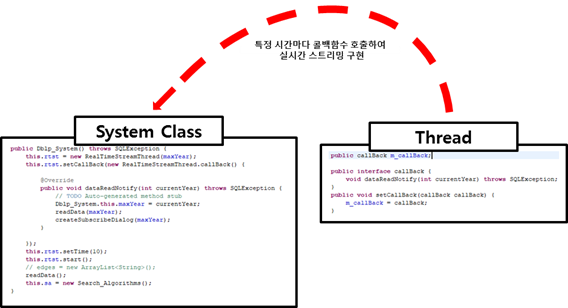
위 과정은 앞에서 paper key를 통해 가져온 title 정보를 통해서 논문 사이트 들에서 검색을 수행하고 그 결과물인 html을 특정 tag기준으로 크롤링하는 과정이다. 특정 논문사이트만 정해서 크롤링을 하려고 했지만, 그 사이트에는 해당 논문이 없고 다른 사이트에만 있는 경우도 있었다. 따라서 세 가지 사이트를 정해서 해당 tag가 존재하는 사이트에서 크롤링을 하고, 만약 해당 tag가 두 사이트 모두에도 없는 경우에는 구글 학술검색을 통해서 요약 내용을 가져올 수 있도록 구현하여 논문 Abstract의 정확도를 높였다.

이렇게 뽑아온 정보는 해당 title을 가지는 논문의 세부내용으로 저장되고, 이번에 구현한 논문 요약내용 출력기능에서 사용되었고, 또한 구현할 TF-IDF 알고리즘에도 사용하였다.

* 실시간 스트리밍

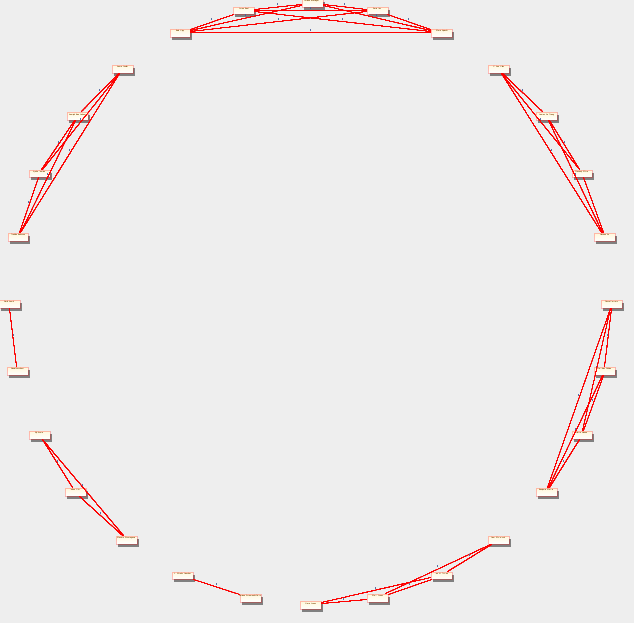
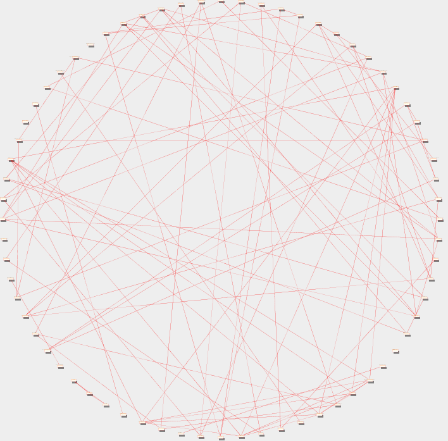


좌측 사진은 1차 프로젝트 시 클래스 구조이고, 우측 사진은 2차 프로젝트 작업을 했을 때 수정한 클래스 구조이다. 1차 프로젝트 시에는 Thread를 관리하는 class가 MVC model class에 종속되어 있었고, 이를 호출하는 방식도 Thread가 System class의 객체를 가지고 있으며, 이를 특정 시간마다 호출하며 데이터를 읽는 방식이었다. 이는 실시간 스트리밍을 수행하기는 하지만 Thread class의 단독적인 사용이나 재사용성을 기대하기는 어려운 단점이 있었다.



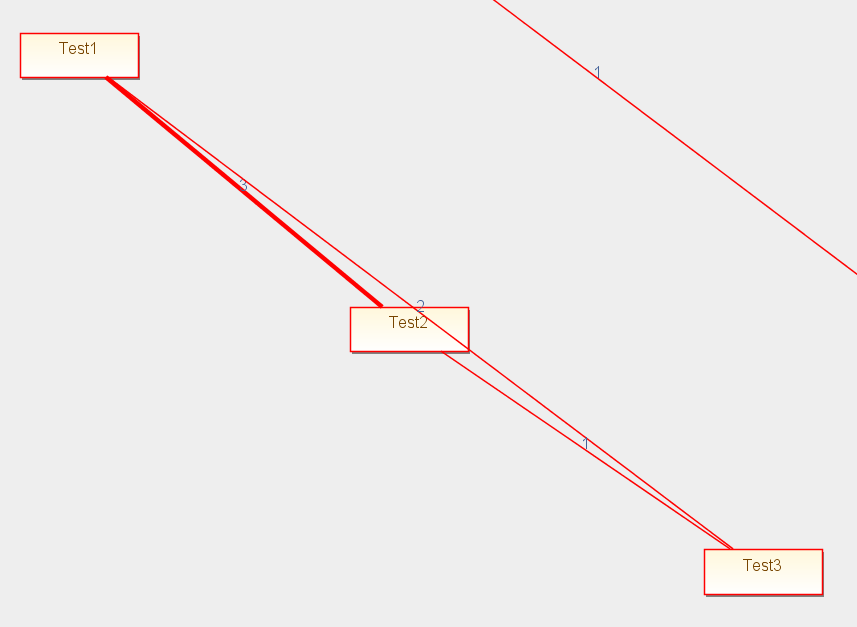
그래서 2차 프로젝트에서는 Thread기능을 수행하는 class를 Streaming package로 분리시켜 system class와 단독적으로 작동하도록 하였고, Call-back 패턴을 이용해서 모듈의 분리와 코드의 재사용성을 향상시켰다. 이를 이용하게 되면 main system에서 일정 시간마다 스트리밍 함수를 호출해야 하는지 확인할 필요 없이 Data streaming 수행 뒤에 이를 system class에 알리기만 하는 과정을 수행하게 된다.

* GUI 개선



좌측은 1차 때 구현한 Co-author Graph이고, 우측은 이번에 개선한 Graph(**CoAuthorShip Graph)**의 출력 결과물이다. 1차 때 구현한 그래프도 보기에는 큰 문제가 없어 보이고, 또한 작동이 잘 되었지만 데이터가 많아지면 많아질수록 버벅거리는 것이 눈에 보였다. GUI 부분을 추가적으로 공부하여 1차때보다 많은 데이터를 입력했음에도 버벅거림이 확실히 줄어들게끔 개선하였다. 이를 통해서 사용자들이 그래프를 열람할 때의 불편함을 확실히 줄였다.

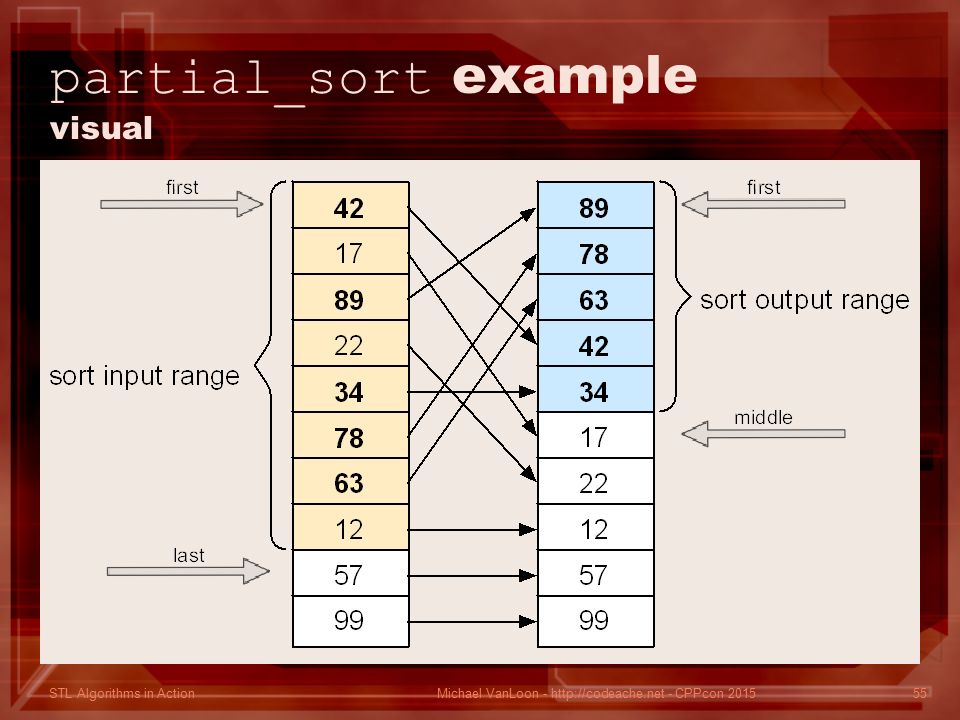
* + **그래프 시각화 개선 - 가중치에 따른 에지표시**



또한 다음과 같이 edge에 가중치에 따라 edge의 굵기를 구별했다. 구현상으로는 가중치가 1~2인경우 기본 edge를 쓰고, 3~4인경우 그림과 같이 굵기가 굵어진다. 그리고 5이상인 경우에는 더욱 굵은 edge를 사용하고 있다. 이를 통해 저자간의 연관성을 쉽게 파악할 수 있을 것이고, 각 Researcher들 간의 공동 작업횟수를 사용자가 더 쉽게 인식할 것이다.

* Top-k 추출 성능 보완

기존 프로젝트에서 Top-k 추출 방식은 다음과 같았다. 전체 Researcher들의 연구 횟수를 카운트해서 해당 Researcher의 맴버 변수로 저장해 두고, Researcher들의 연구횟수가 높은 순으로 특정 k개의 Top-k를 추출하는 방식이었다. 이를 구현하기 위해 MinHeap방식을 사용했었다. MinHeap을 사용하게 되면, MinHeap자료구조가 마치 필터처럼 전체 Researcher를 하나하나 탐색하면서 MinHeap의 Top에 있는 Researcher를 pop하였고, 이를 통해서 마지막까지 MinHeap에 남아있는 Researcher들이 전체 Researcher들 중 연구횟수가 많은 Top-k로 볼 수 있었다. 이 방식은 보통의 quick sort보다는 효율이 좋지만 MinHeap이라는 자료구조를 새로 두어야 하는 점, 또한 MinHeap이 Top-k 추출을 위해 설계된 자료구조가 아니기 때문에 효율적이지 못하다는 단점이 있었다. 따라서 우리는 이번 3차 프로젝트 시에 Top-k추출의 성능을 끌어올릴 새로운 방식에 대해서 찾게 되었는데, 이는 바로 partial sort이다.

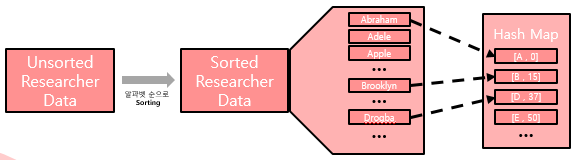


위의 그림과 같이 partial sort를 이용하게 되면 우리가 미리 정해둔 k값 만큼만 앞에서부터 sorting을 하고 나머지 자료들은 더 이상 sorting하지 않는다. 이를 이용해서 더욱 효율적인 프로그램으로 개선하였다고 생각한다.

* 자료구조

1차 때 구현한 인접 다중리스트는 유지하되, 이를 업그레이드 하였다. 현재 수행하고 있는 프로젝트의 전체 data양이 매우 많은 부분에서 착안해서, 자료를 찾는데 걸리는 시간을 줄일 방법을 고민해 보았다.

우리는 현재 자료구조에 알파벳들의 인덱스를 담고있는 HashMap을 추가하기로 했다.



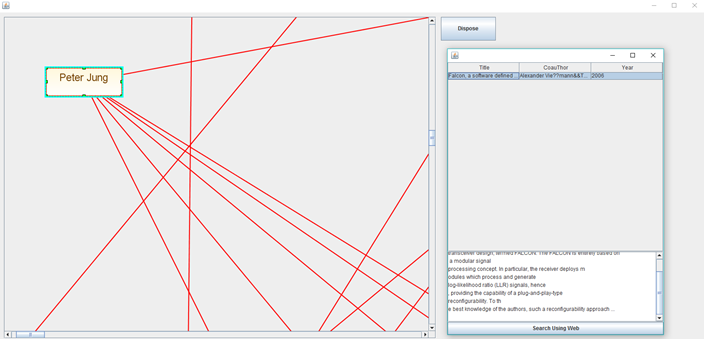
다음은 우리가 이번에 구현한 Hash Map의 구조이다. 먼저 읽은 데이터들(Researcher들이 담겨있는 vertex)은 특정 기준으로 Sorting이 되어있지 않다. 이를 해당 vertex의 researcher name을 기준으로 알파벳순으로 Sorting을 하고, 해당 알파벳으로 시작하는 vertex의 index를 담는 Hash맵을 만들었다. 이 Hash Map을 만드는데 Sorting과 데이터 할당 등의 시간과 메모리 소모가 다소 있기는 하지만, 이전 검색방식에서는 단순 순차탐색을 해야 했던 것에 반해 이제는 저자의 이름에 따라 인덱스로 접근한 뒤, 그 부분에서만 검색하면 되므로 탐색시간을 확실히 줄일 수 있게 되었다.

* Show Chain 과정 수정

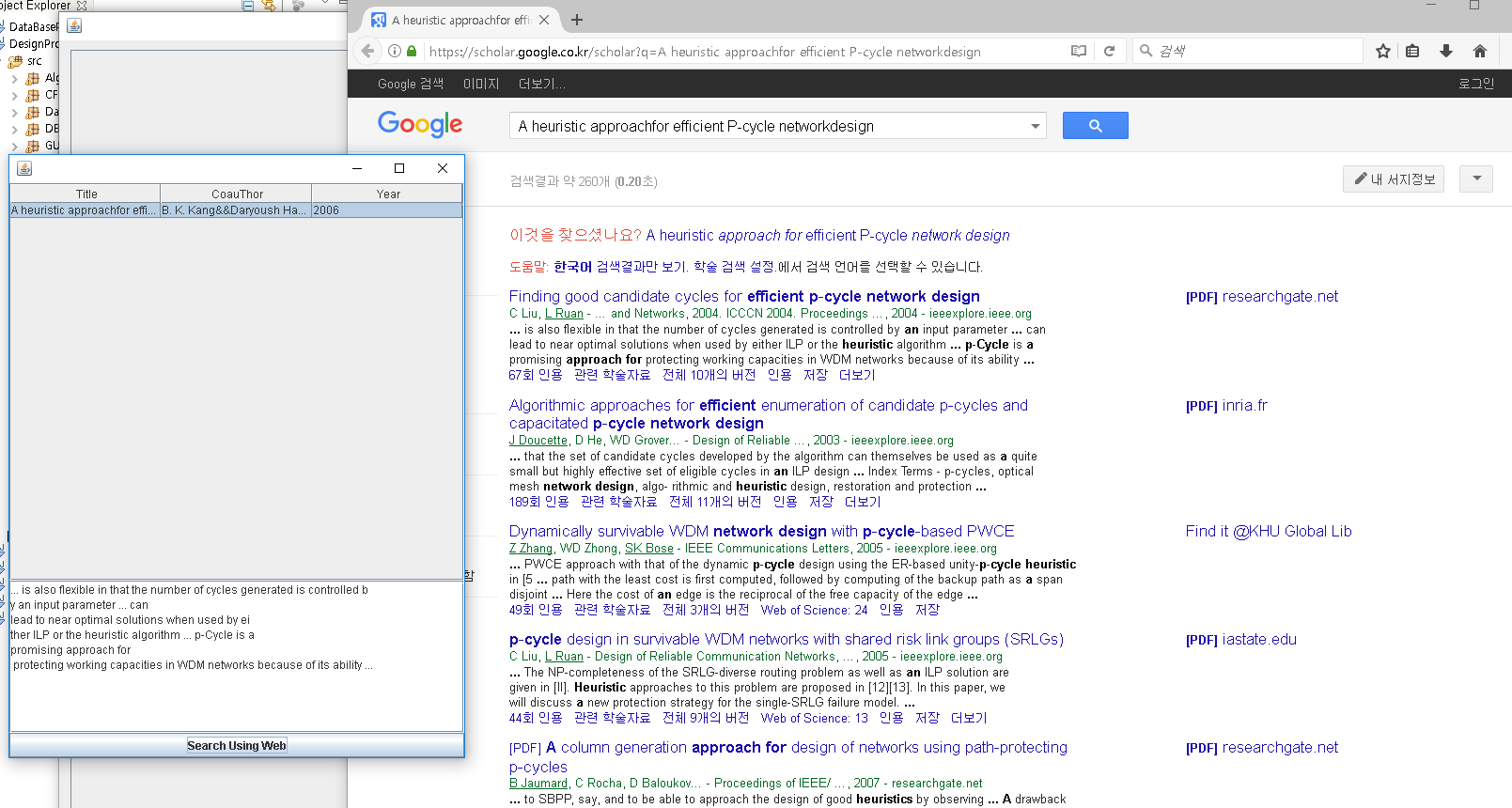


좌측 그림은 1차 프로젝트에서 수행했던 Show Chain과정이고, 우측 그림은 수정된 show Chain Graph의 모습이다. 1차 프로젝트에서도 Show Chain기능은 구현되었지만, 최단거리를 경유하는 vertex만 coloring 되고, edge coloring은 수행하지 않아서 경로를 확실히 알아보기에 어려웠다는 단점이 있었다. 따라서 이번 2차 프로젝트에서는 경로 내의 vertex 뿐 아니라 edge들까지 coloring을 수행해서 확실히 알아보기 쉽게 구현하였다.

또한 경로 탐색 알고리즘을 수정해서 가중치를 고려해서 Chain을 출력하도록 수정하였다. 1차 때는 edge의 weight값(Researcher들 간의 공동연구횟수)는 고려하지 않고 단순히 최소의 edge를 거치게 되는 최단경로를 보여주었다면, 최종결과물은 weight값을 고려해서 만일 특정 vertex와 연결되어있는 edge가 두개 이상이라면, 그 중 weight값이 더 큰 edge와 연결하도록 했다. 다익스트라 알고리즘이 최소의 weight값을 거치는 경로를 찾는 알고리즘인 것을 응용, 이를 활용해서 가중치를 고려한 경로를 찾게 하였다.

* 논문 정보 시각화  
  

Coauthor Graph가 출력되는 부분은 1차 프로젝트와 동일하지만 최종 구현된 프로젝트는 Coauthor Graph에서 해당 Researcher의 vertex를 클릭하면 새로운 Frame이 생성되고 그 Researcher가 작성한 논문 리스트를 열람할 수 있도록 구현하였다. 그 리스트에 기록된 정보로는 논문의 제목, 공저자 목록, 작성 년도 및 논문의 Abstract내용이 있다.

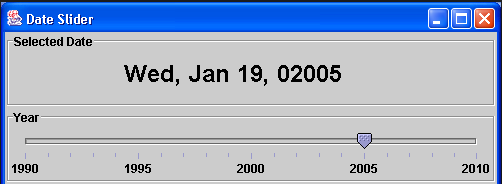


또한 그 논문 리스트 중 특정 논문에 커서를 가져다 대면 앞에서 크롤링기능으로 가져온 해당 논문 요약설명이 팝업창에 뜨게 되고 그 밑에 있는

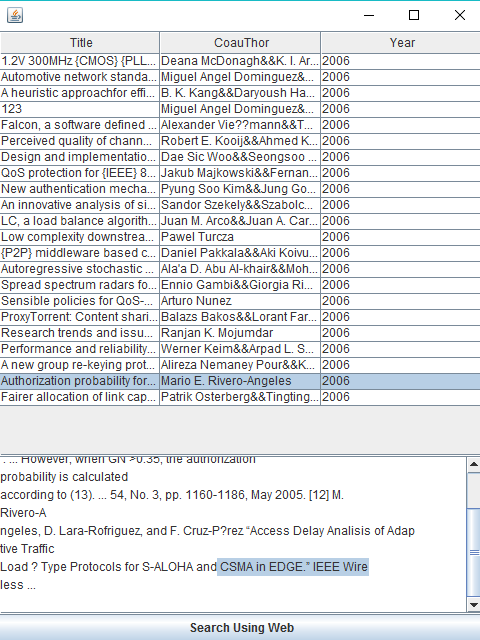
‘Search Using Web’ 버튼을 누르면 구글 학술검색 창으로 가서 해당 논문 제목을 검색한 내용을 볼 수 있도록 했다.

* 기간별 논문 시각화

DBLP 웹 사이트에는 수많은 논문 Data가 저장되어 있고 이를 열람할 수도 있다. 하지만 그 수많은 논문 Data를 열람하기에 UI 구성이 미흡한 점을 확인할 수 있었다. 특히 특정 년도에 집필된 논문을 열람하기 위한 검색 필터가 제대로 제공되고 있지 않다는 점을 확인하였고, 이를 해결하기 위한 기간별 논문 열람 기능을 추가하기로 했다.

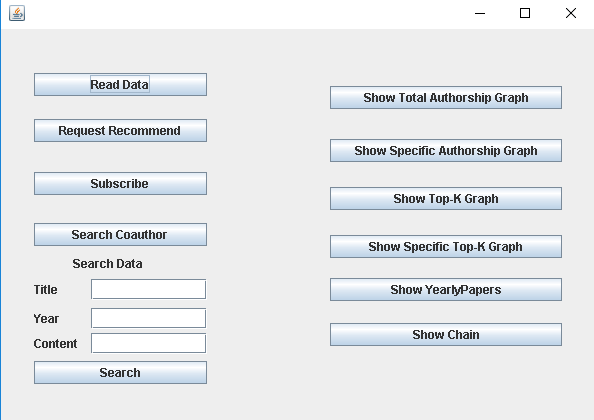


좌측 사진은 우리가 구현하기 한 모델의 프로토타입이다. 이런 방식을 Slider라고 하는데, 이런 JComponent를 이용해서 구현했다. 그리고 우측이 최종적으로 구현한 모델이다. 이를 통해서 검색 범위를 정해주고, 검색하게 되면 다음과 같이 결과가 나온다.

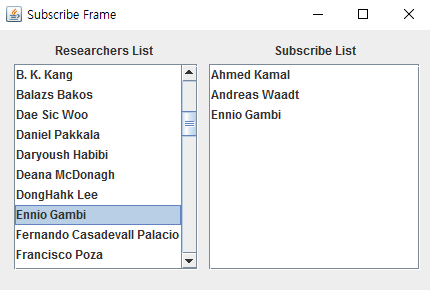


해당 범위에 해당하는 논문이 출력되고, 앞에서 설명한 것과 같이 해당 논문에 커서를 가져다 대면 해당 논문의 요약 내용이 출력되고, 웹 사이트로 가서 자세한 논문 내용을 열람할 수도 있다.

* 구독기능

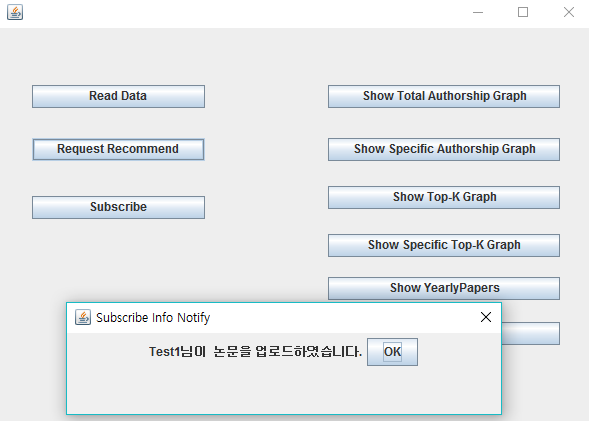


이번에 새롭게 메인화면에 “Subscribe”버튼을 추가하고, 구독 기능을 추가하였다. 이를 통해서 특정 Researcher를 구독할 수 있다.

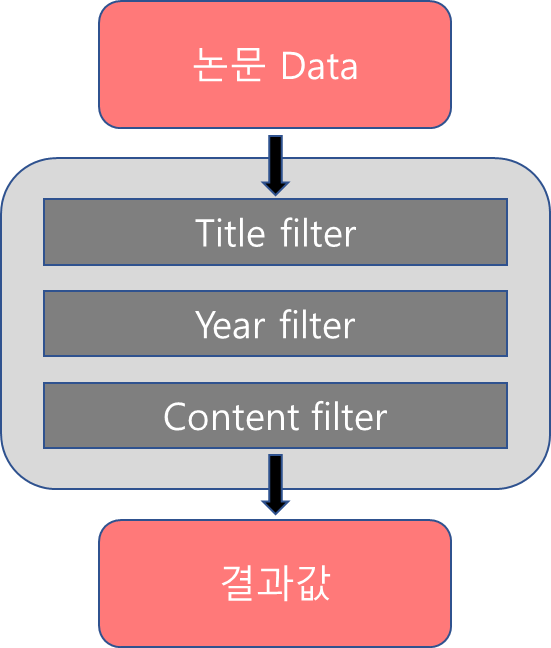


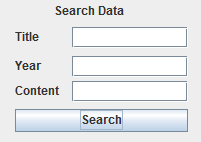
다음과 같은 팝업창으로 Researcher를 구독할 수 있다. 왼쪽 창에서 해당 Researcher를 클릭하면 해당 Researcher와 관련된 논문을 구독할 수 있고, 만약 원하지 않는다면 오른쪽 창에서 클릭해서 구독을 취소할 수 있다.

만약 내가 구독한 Researcher의 새로운 논문이 실시간 Streaming을 통해서 들어오게되면, 사용자에게 그 정보를 알려준다.

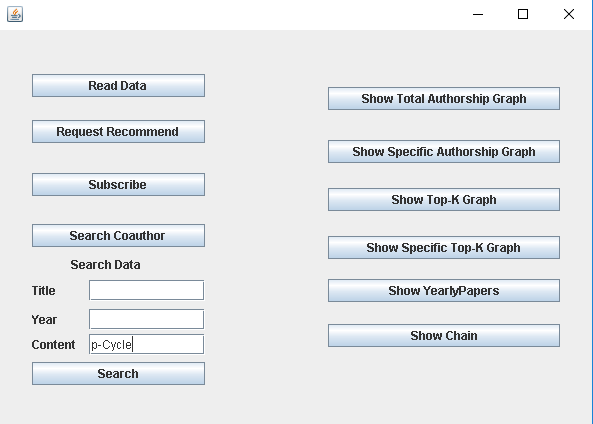


* 속성 기반 논문 검색기능(제목, 년도, 논문 내용)

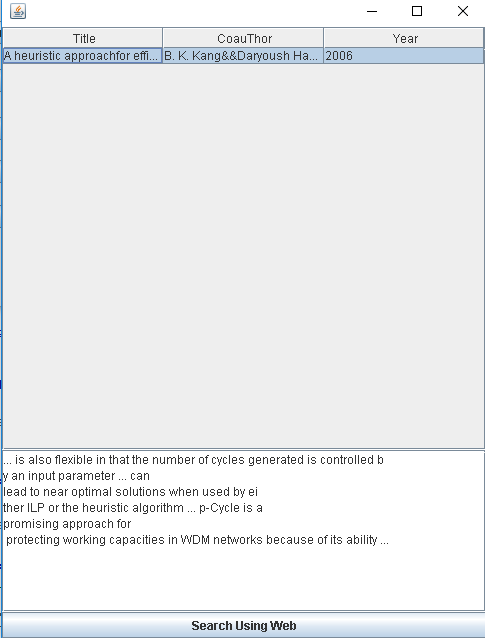
1,2차 프로젝트 구현시 이미 특정 저자를 기준으로 한 그래프 출력, 특정 저자 기준 Top–K 그래프 출력 등의 검색 기능을 구현했지만 추가적인 검색 기능을 더 구현하였다. 위에서 언급한 것 처럼 현재에는 CoAuthorShip Graph에서 저자 노드를 클릭하면 그 저자의 논문 정보들을 볼수 있도록 구현하였지만, 데이터 양이 많아질 수록 사용자가 원하는 정보만을 검색하기엔 힘들 거라 판단하였다. 따라서 전체 저자들을 기준으로 특정 조건을 만족하는 결과물만 보이도록 하게 하기 위해 다양한 검색 필터를 두었다.



위 사진은 구현된 검색 필터의 UI이다. Title, Year, 그리고 논문 내용중에서도 검색하기 위해 Content 필드를 두었다. 이 필드들에 각각 사용자가 원하는 정보를 입력하는 경우 해당 정보와 일치하는 논문들만 필터링 되어 출력되도록 구현하였다. 아래 사진은 p\_Cycle이라는 keyWords가 포함된 논문 리스트가 출력되도록 입력한 것이다.



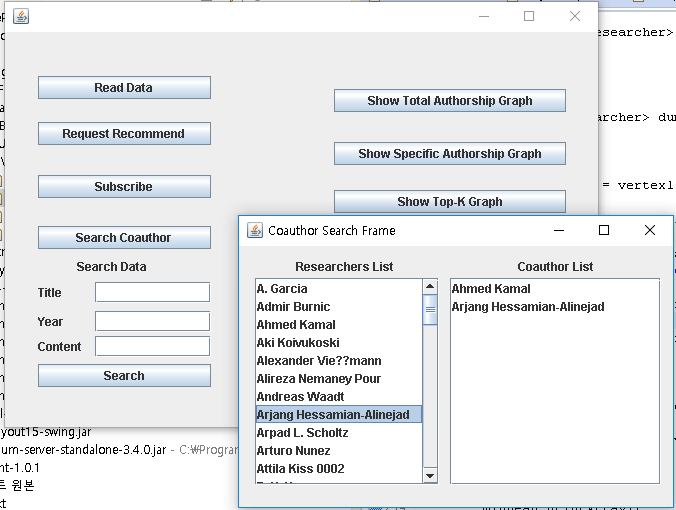
다음 사진은 위 조건으로 Search버튼을 눌러 검색한 결과물이다.



p-Cycle이라는 keyWords가 존재하는것을 확인할 수 있다.

* 공동 저자 기반 논문 검색

저자들 끼리의 논문의 공동작업 여부를 확인 할 수 있도록 하는 기능을 구현하였다. CoAuthorShip Graph에서도 확인할 수 있지만 더 직관적으로 확인할 수 있어야 한다는 필요성을 느껴 구현하였다. 아래 사진은 Search Coauthor 버튼을 눌러 공동 저자 기반 논문 검색을 수행하는 것을 나타낸다.



Coauthor List 필드에 있는 저자들이 같이 논문을 작성한 적이 있는 경우 그 리스트가 새로운 Frame으로 나타나게 된다.

* 공동 작업할 저자 추천 기능

우리는 저자들을 서로 추천해 주는 기준으로 각 저자들의 전공(주 분야)의 유사성을 선정하였다. 기본적인 paper Key를 사용해 제공되는 데이터에서는 각각 저자가 작성한 논문 제목이외의 정보를 얻기에는 힘들다. 그래서 제목 만을 가지고 KeyWords 및 주 분야를 추출해 내야 하는데 이는 정확도를 보장해 주기 어렵다. 따라서 우리는 2차 프로젝트 때 미리 크롤링 과정을 통해 논문들의 요약 내용(Abstract)를 가져와 DB에 저장하는 작업을 수행하고 이 요약 내용들까지 사용하여 KeyWords 및 저자들의 주 분야를 추출하였다.

추출하는데 사용되는 알고리즘은 Tf-idf이다. 이 알고리즘은 해당 용어(Term)가 현 Document에 출현한 빈도수를 산출하는 것인데 이를 위해 TF값을 산출하여야 한다. 이를 위한 방법으로는 아래와 같다.

* 불린 빈도: tf(t,d)

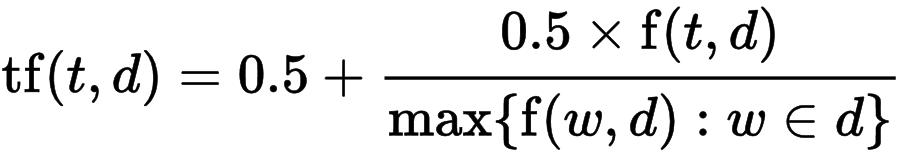
Term이 Document에 한 번이라도 출현한다면 1(True)로 설정하고 아니면 0(False)으로 설정함

* 로그 스케일 빈도: tf(t,d) = log(f(t,d) + 1)

위 불린 빈도를 로그함수를 통해 표현함

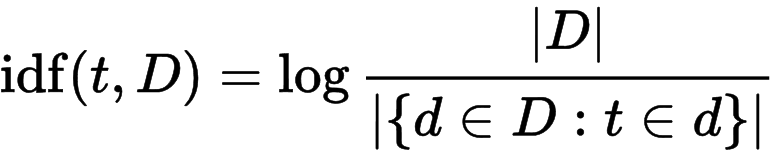
* 증가 빈도: 문서의 길이에 따라 단어의 빈도값 조정

아래와 같은 공식을 통해 구함

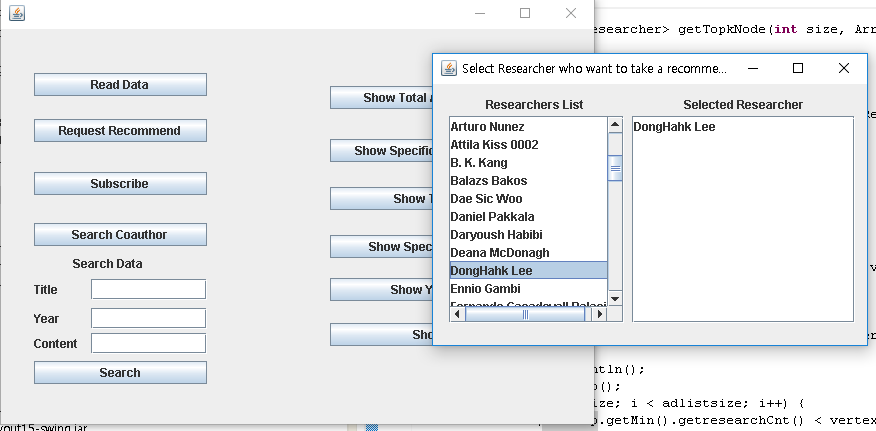


* 역문서 빈도: 한 단어가 문서 전체 집합에서 공통적으로 얼마나 나왔는지를 나타내 주는 값

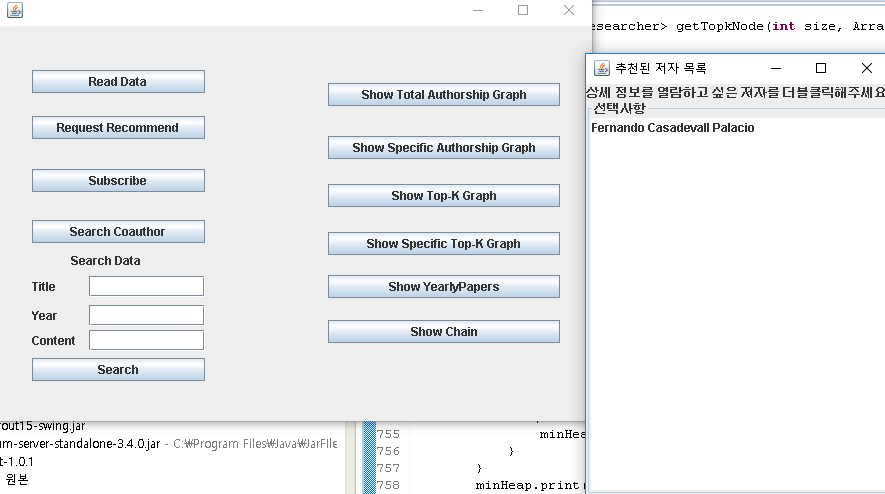
전체 문서의 수를 해당 단어를 포함한 문서의 수로 나눈 뒤 로그를 취해서 얻는다. 공식은 아래와 같다.



공동 작업할 저자 추천 기능은 위 과정을 통해 추출한 KeyWords들을 주 분야라고 판단하여 비슷한 주 분야를 가진 저자들끼리 서로 추천해주도록 구현하였다.



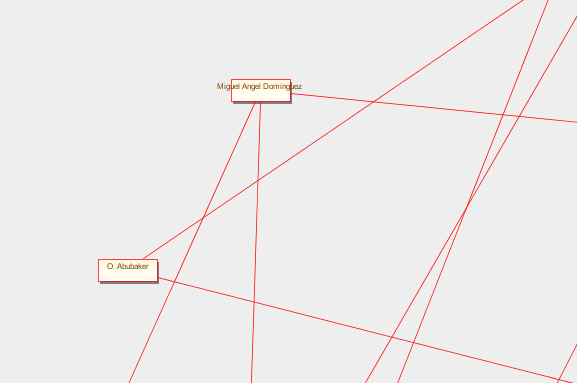
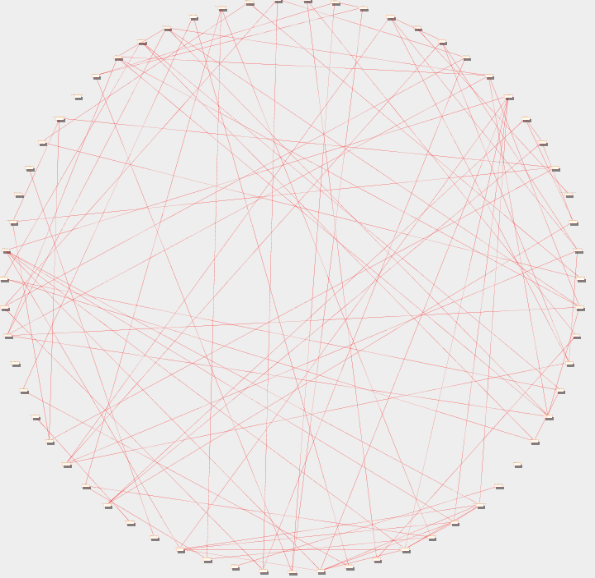
위 사진은 Request Recommend 버튼을 눌렀을 때 나타나는 화면이다. 시스템의 메모리에 올라와있는 저자들 중에서 추천의 기준이 될 저자를 선택한 뒤 창의 화면을 닫으면 아래와 같은 새로운 Frame이 나타난다

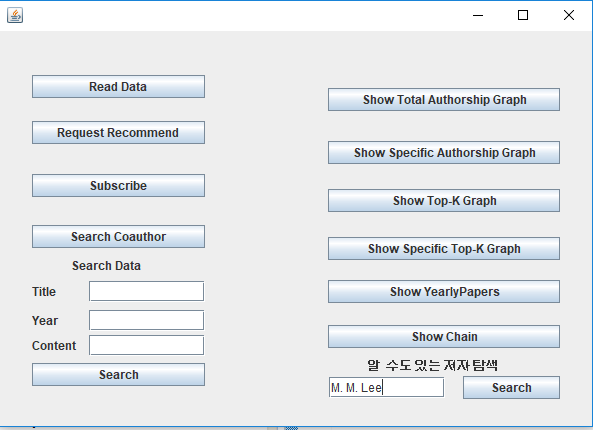
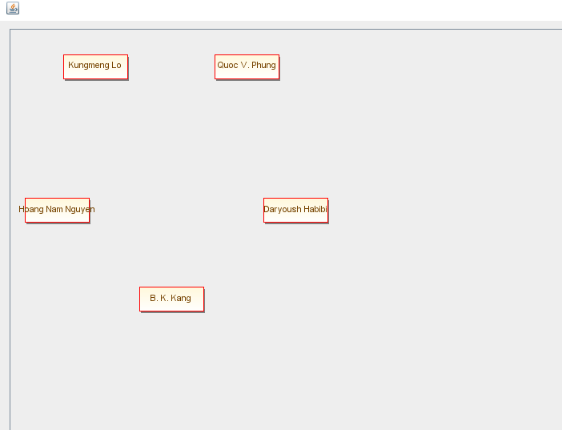


이 화면은 선택된 저자를 기준으로, 그 저자와 유사한 전공 분야를 연구하고 있는 저자의 리스트를 나타낸 것이다. 다른 여타 기능들과 마찬가지로 추천된 저자의 이름을 더블클릭하면 그 저자의 상세 정보를 열람할 수 있도록 구현하였다.

* 알 수도 있는 저자 추천

현재 많은 Researcher들은 거미줄처럼 얽혀있고, 서로 직.간접적인 관련성이 있다.

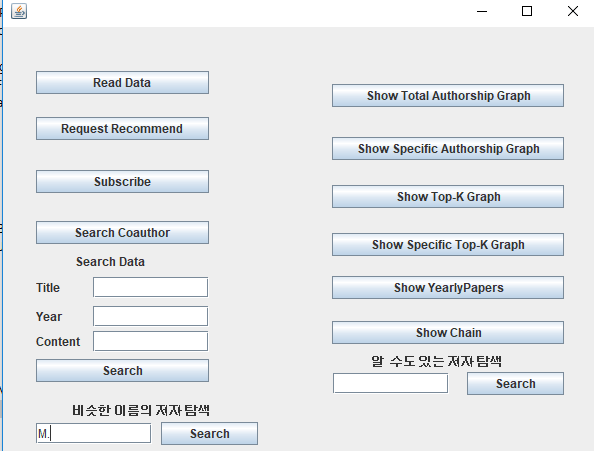
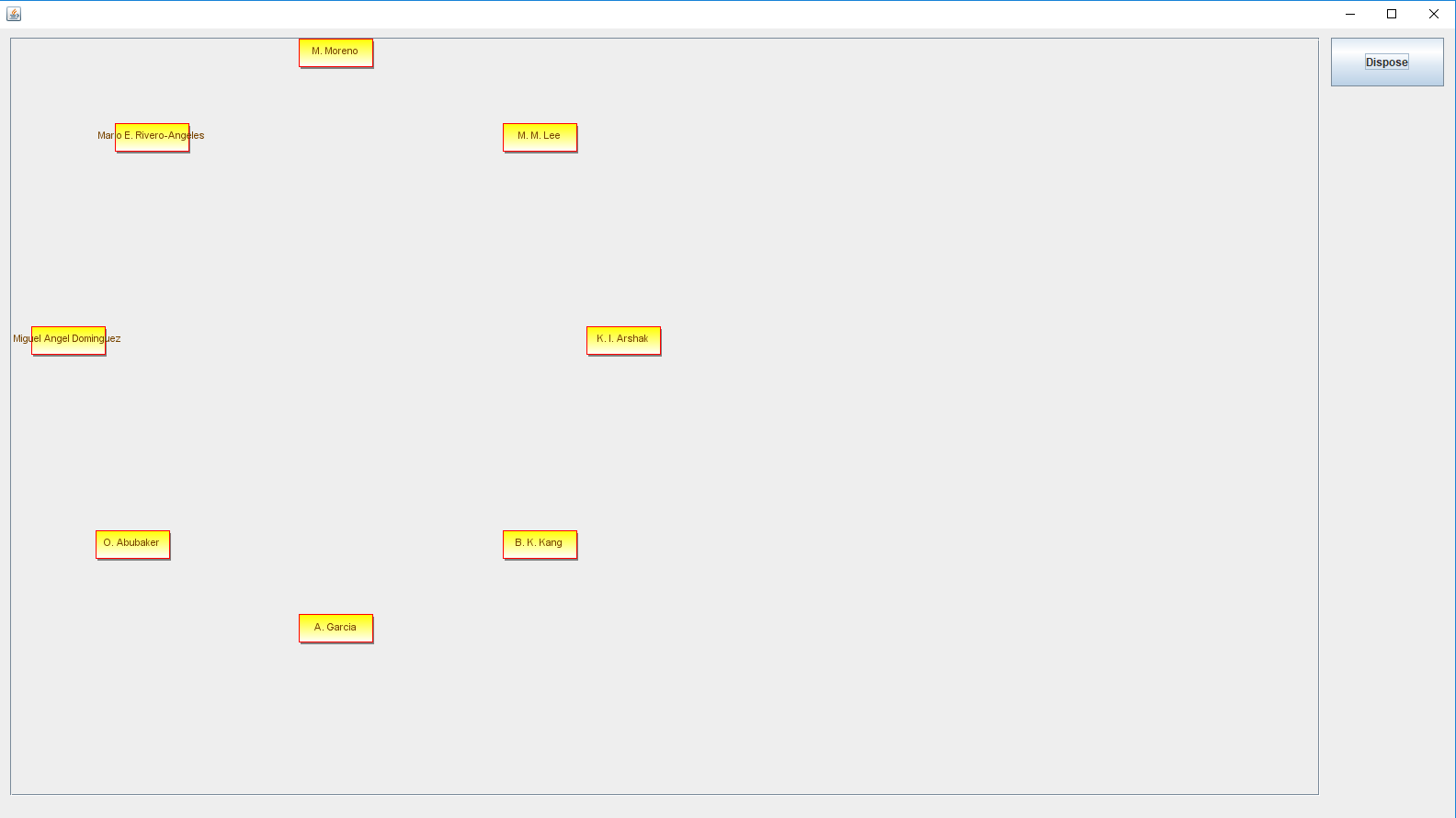


Researcher들은 직접적으로 연결(서로 연구를 진행)되어 있는 이들도 있을 거지만 대부분은 서로 직접적으로 연구를 진행한 적은 없을 것이다. 하지만 직접적으로 연결은 되어 있지 않더라도 한 다리만 건너면 관련이 있는 Researcher들이 많다. 따라서 우리는 이에 착안해서 ‘알 수도 있는 저자’기능을 추가하기로 했다. 또한 이 기능은 지금 우리가 진행하고 있는 프로젝트인 ‘DBLP시스템을 기반으로 한 SNS 시스템’설계에도 적합하다고 판단하였다. 현재 많은 SNS들은 이미 ‘알 수도 있는 친구’를 찾아주는 기능을 지원해주고 있다. 그리고 이 기능의 세부 내용은 자세히는 모르지만 특정 유저의 친구 목록의 친구 목록을 서치하는 방식인 것으로 알고있다. 따라서 이 기능을 구현하였다.

위 사진과 같이 알 수도 있는 저자를 입력하면 해당 저자 vertex에서 2차 Depth까지 탐색하여 이에 해당하는 저자들을 불러와 알 수도 있는 researcher로 추천을 해준다.

* 비슷한 이름의 저자 찾기

Secondary String Library를 사용하여 사용자로 부터 입력받은 이름과 유사한 이름을 가진 저자들을 그래프 형식으로 출력되게끔 구현하였다.

위의 사진과 같이 사용자가 입력한 M.과 유사한 이름을 가진 저자들의 vertex가 그래프 형식으로 출력되고, vertex를 클릭 시 해당 저자의 정보를 확인할 수 있도록 하였다.