<자료구조 – tweeter 분석>

2011130298

심리학과 송은석

1. Manual(설명서)

- 이 프로그램은 한 단어를 tweet 하는 시스템의 내용을 분석하기 위한 용도로 제작되었습니다. 이 시스템에 입력되어 있는 기본적인 정보는 다음과 같습니다.

* 사용자 정보
* 사용자간 친구관계
* 사용자의 tweet한 정보

이 정보를 바탕으로 한 분석 정보를 사용하실 수 있으며 사용방법은 다음과 같습니다.

[1] 기본 설명

(1)처음 실행을 누르면 Select Menu버튼이 보일 것입니다. 이 메뉴에서 원하는 기능을 선택하실 수 있습니다.

(2)기능을 선택하는 방식은 ‘메뉴 번호를 입력하세요’ 옆에 원하는 숫자(1~9)를 입력하고 enter key를 입력하는 것입니다.

(3) 기능에는 총 9가지가 있으며 각각에 해당하는 숫자가 있습니다. 예를 들어 1. Read data files기능을 수행하시려면 1번을 입력한 후 enter key를 누르시면 1번에 해당하는 정보가 인출 될 것입니다.

(4) 메뉴의 종류는 다음과 같습니다.

|  |
| --- |
| **1. Read data files** |
| **2. display statistics** |
| **3. Top 5 most tweeted words** |
| **4. Top 5 most tweeted users** |
| **5. Find users who tweeted a word** |
| **6. Find all people who are friends of the above users** |
| **7. Delete all mentions of a word** |
| **8. Delete all users who mentioned a word** |
| **9. Find strongly connected component** |
| **10. Quit** |

[2] 각 기능 설명

**1. Read data files**

- 메뉴 버튼 선택 상태에서 1을 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* Total user: tweeter를 사용하는 전체 사용자 수를 확인 할 수 있습니다.
* Total tweets: tweeter에서 올라온 전체 tweeter의 숫자를 확인 할 수 있습니다.
* Total friendship: following한 전체 숫자를 확인 할 수 있습니다.

**2. display statistics**

- 메뉴 버튼 선택 상태에서 2를 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* Average number of friends: 전체 사용자들의 평균적인 following 숫자를 확인 할 수 있습니다.
* Minimum number of friends: 전체 상용자들 중 최소의 following 숫자를 확인 할 수 있습니다. (follower숫자가 아닙니다)
* Maximum number of friends: 전체 사용자들 중 최대의 following 숫자를 확인 할 수 있습니다. (follower숫자가 아닙니다)
* Average tweets per users: 전체 사용자의 평균 tweet 숫자를 확인 할 수 있습니다.
* Minimum tweets per users: 전체 사용자중 최소 tweet 숫자를 확인 할 수 있습니다.
* Maximum tweets per users: 전체 사용자중 최대 tweet 숫자를 확인 할 수 있습니다.

**3. Top 5 most tweeted words**

- 메뉴 버튼 선택 상태에서 3을 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 전체 단어 tweet중 가장 많이 사용된 5개의 단어를 표기합니다.
* 1등이 가장 많이 사용한 단어이며 5등 단어가 5개의 단어 중 사용 빈도가 가장 낮은 단어입니다.
* 가장 많이 사용된 단어가 동일한 숫자로 2개 존재할 경우 공동 등수로 표기됩니다.
* 5개의 등수를 표기하므로 공동 등수가 존재하는 경우에도 5개의 등수를 모두 표기합니다.

**4. Top 5 most tweeted users**

- 메뉴 버튼 선택 상태에서 4를 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 가장 활발하게 tweet활동을 하고 있는 5명의 user를 보여줍니다.
* Tweet 숫자가 가장 많은 순서대로 5명을 보여주며 tweet숫자가 동일할 경우 같은 등수로 표기됩니다. (순서로 표현되므로 5명 이상일 수도 있습니다)

**5. Find users who tweeted a word**

- 메뉴 버튼 선택 상태에서 5를 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 특정 단어를 tweet한 사람을 찾습니다. 예를 들어 ‘For’이라는 단어를 사용한 유저를 찾아 모두 표기합니다.

**6. Find all people who are friends of the above users**

- 메뉴 버튼 선택 상태에서 6을 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 특정 단어를 사용한 유저들을 follow하는 유저를 표기합니다. 예를 들어 ‘For’이라는 단어를 ‘A’라는 유저가 사용했을 경우, ‘A’를 follow하는 모든 유저들을 표기합니다.

**7. Delete all mentions of a word**

- 메뉴 버튼 선택 상태에서 7을 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 특정 단어를 삭제합니다. 특정 단어에 대한 정보 만을 삭제합니다.
* 이전 7번 기능을 수행한 기록이 있는 경우 이전까지 삭제한 단어들의 목록을 출력합니다.

**8. Delete all users who mentioned a word**

- 메뉴 버튼 선택 상태에서 8을 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 특정 단어를 언급한 모든 사람에 대한 정보를 지웁니다. 그 사람의 기록 및 그 사람과의 친구 관계도 삭제됩니다.
* 이전 8번 기능을 수행한 기록이 있는 경우 이전까지 삭제한 사람들의 목록 또한 출력합니다.

**9. Find strongly connected component**

- 메뉴 버튼 선택 상태에서 9를 누르면 사용하실 수 있습니다.

- 확인 가능한 정보는 다음과 같습니다.

* 사람들간의 친구 관계지도를 보여줍니다. A와 B의 follower이고 B가 C의 follower이며 C가 A의 follower일 경우 A와 B와 C는 서로 연결되어 있는 하나의 그룹으로 판단됩니다

**10. Quit**

- 메뉴 버튼 선택 상태에서 10을 누르면 사용하실 수 있습니다.

* ‘감사합니다’를 출력 한 후 프로그램을 종료합니다.

1. 분석

[1]What data structure you chose and why

- 처음 data는 txt파일로 제공되었기 때문에 이를 list의 형태로 나타내어 분석하기 위해 readline()함수를 사용하였다. 내용의 유무는 readlineTF라는 변수를 지정하여 만약 읽을 수 없는, 즉 비어있는 줄이 생기면 false로 바뀌어 while문이 멈추도록 설계하였다. user정보에서 중요한 것은 처음 고유 번호와 본인의 아이디이기 때문에 이 두 가지를 list의 형태로 저장하여 출력할 수 있게 하였다. Word의 경우도 누가 어떤 내용을 입력한 것이 중요하기 때문에 위와 같은 방식으로 저장하였다. 각각 user()라는 함수와 word()라는 함수로 구현하였다.

기본적인 정보의 형태는 Adjacency list형태로 정보를 저장하여 이를 edge로 저장하

는 것이다. 이 연결된 형태는 linked list이며 vertex각각에는 고유 번호를 name으로 사용자 아이디를 nick으로 저장했다. Name과 nick형태로 정보를 저장 하는 것은 기존 코드로 충분히 표현 가능했으나 edge의 경우 처음에는 for 문을 3중첩으로 사용했다. 이후 교수님께 문의하여 edge파일의 A가 B와 친구라는 정보는 undirected 이므로 A와 B에 해당하는 항목을 각각 list로 만들고 이를 또 다른 vertex로 전달하여 이곳에 앞서서 만든 vertex의 인덱스를 저장하게 만들면 수월하게 add함수를 통해 edge를 만들 수 있게 되었다. 이 분석은 main() 에서 실행했다.

이후 word를 분석했는데 word 정보는 두 가지 방식을 사용하였다. 첫째로 기존에 만들었던 vertex안에 특정 아이디를 가진 사람이 언급한 모든 word를 포함하는 배열을 만들고 이곳에 정보를 저장했다. 이런 방식으로 word를 저장하면 특정 인물이 사용한 tweeter의 수를 분석하는 데에는 용이하나 특정 Word를 찾는 일이 필요한 tweeter분석에서는 용이하지 않은 경우가 발생하여 두 번째 방식인 word라는 구조체를 만들었다. 이 Word에는 전체 tweeter의 내용이 중복되지 않게 저장되었고 각 tweeter 하나를 언급한 모든 사람을 배열로 저장해 놓았다. 즉, @라는 단어를 A ,B ,C가 이용했다면 word를 기준으로 이를 묶어서 정리하는 것이다. 이 분석은 wordlist()를 이용했다. Chain hash를 이용하여 정리하고자 하였으나 계속적으로 실패하여 search 시에 O(l)대신 O(n)을 사용하기 위해 선택한 방법이었다. 또한 단어의 반복을 없애기 위해 hash라는 함수를 만들어서 각 tweet를 이에 해당하는 ASCII 코드로 변환한 다음 동일한 숫자를 제거하는 방식을 사용했으며 이때 multi()라는 함수를 사용했다.

모든 함수는 menu()라는 함수를 이용하여 구현하였으며 처음 함수를 시작함과 동시에 위의 main()과 wordlist()를 실행하여 이를 a와 b라는 변수에 저장한다. 이후 number라는 변수를 사용하여 이곳에 input으로 정보를 입력 받은 후 if 구문으로 특정 번호를 입력하면 특정 함수를 출력하는 형식으로 만들었다. 이렇게 구현해야 추후 delete를 했을 경우 이 delete된 정보가 저장되기 때문이다.

Number가 1일 경우에는 위에서 언급한 1번 data file읽기를 실행한다. 이 경우 전체 tweet수, 전체 친구 연결 수를 print하는데 전체 숫자는 main()함수를 통해 확인 한 vertex숫자를 사용했다. tweet수와 friendship수를 구하기 위해 각 vertex의 word배열의 len()의 합, 그리고 pointer로 넘어가는 수를 각각 더하면 위의 결과를 얻을 수 있다. 처음부터 O(n)으로 정보를 search하면서 각각의 vertex의 linkedlist를 모두 살펴야 하기 때문에 최악의 경우(모두가 모두와 친구인 경우) O(n^2) 로 정보를 검색하게 된다.

Number가 2일 경우 친구와 tweet수를 분석한다. count라는 변수를 사용하여 전체 친구 수 즉, edge의 전체 수를 구한 다음 모든 사용자의 수로 나누어 전체 edge의 평균 값을 구했다. 가장 큰 친구를 가진 사람, 가장 적은 친구를 가진 사람은 각각 사람의 친구 수를 countfriend 변수를 통해 구한 이후 이를 maxfriend와 minifriend로 저장하여 나타내었다. User당 tweet 수의 분석 또한 위와 동일하게 counttweet변수를 사용하여 각 vertex의 배열 word에 해당하는 len을 측정하여 저장한다. 이 크기를 비교하여 mintweet과 maxtweet을 이용하여 저장했다. 1번과 마찬가지 이유로 O(n^2)이다.

Number가 3일 때는 Top5word()라는 함수를 시행하는데 이는 가장 많이 tweet 된 5개의 단어들을 인출하는 함수이다. Top5word는 앞서서 b로 저장한 Word중심 자료에서 각 word를 말한 사람들인 people의 숫자를 저장하는 배열을 만들어서 이 배열을 sorting하여 가장 큰 수 중심으로 5개의 숫자를 정한 후 이 숫자에 해당하는 word를 인출하는 방식이다. Top5word()는 O(n)으로 search 가능하다

Number가 4일 때는 Top5user()라는 함수를 시행하는데 이는 앞서서 한 것과 동일한 방식을 사용하나 이때 분석하는 것은 한 사람이 말한 word가 저장되어 있는 a(main())이다. Top5user()는 O(n)으로 search 가능하다

Number가 5일 때는 Wordfind()의 경우 b의 word를 찾으면 되므로 O(n)으로 search가 가능하다. Word가 일치한 것을 찾으면 그 list에 해당하는 정보에 people이라는 변수에 이 word를 사용한 사람의 목록이 저장되어 있기 때문에 바로 인출 가능하다.

Number가 6일 때는 위의 search한 특정 word를 언급한 사람을 list의 형태로 저장 받는다(O(n) search) 저장된 리스트를 기반으로 이 사람을 follow하는 사람을 찾아야 한다. 이는 사람의 이름을 바탕으로 search해야 하므로 a라는 리스트에 저장되어 있는vertex의 정보를 비교한다. 1번, 2번과 마찬가지로 모두가 모두와 친구인 경우에는 그리고 특정 word를 모든 사람이 언급했을 경우에는 O(n^2)으로 search하게 되지만 그 외의 경우에는 (test의 경우 10명 내외 이므로 constant) O(n)으로 search 가능하며 이 모든 사람의 이름을 리스트로 저장 받아 출력한다.

Number가 7일 때는 word의 정보를 삭제하면 되므로 word기반으로 저장된 list를 O(n)으로 삭제한다. 이때 혹시 이전에 삭제한 단어가 무엇인지 모를 수 있으므로 deleteword라는 배열을 만들어서 이전에 7번 삭제 기능을 사용하여 특정 단어를 삭제했을 경우 이를 출력하는 것을 if문을 통해 만들었다. 사람을 기반으로 한 vertex의 정보에서 특정 사람이 언급한 모든 word또한 저장되어 있으므로 이를 delete해야 하는데 특정 단어를 모든 사람이 언급했을 경우 O(n^2)으로 search하여 delete하게 된다.

Number가 8일 때는 word를 언급한 사람의 정보를 삭제해야 하므로 7번과 유사하게 word를 언급한 사람의 list를 받아 이를 통해 a 리스트에 저장되어 있는 사람 중에 이 이름을 가지고 있으면 삭제하는 방식으로 지운 후 남아있는 사람들을 대상으로 linkedlist를 삭제 한다. Word기반으로 저장되어 있는 경우 list에 속한 사람이 다른 word를 사용했을 때 정보가 남아있으므로 이를 비교하면서 삭제한다. 이 때문에 최악의 경우 (모두가 모두의 친구이며 모두가 특정 단어를 언급했다.) O(n^2)으로 구현된다. 또한 혹시 이미 삭제한 사람들에 대한 정보가 필요할 수 있으므로 삭제한 사람들의 id를 보여주는 내용을 if을 통해 추가했다.

Number가 9일 때는 strongly connected component를 출력한다. 그러나 이 기능은 수업시간에 배운 내용을 옮겨서 사용했으므로 top5를 구현하지 못하고 기존에 main()으로 저장했던 edge내용만을 strongly connected component구현하는 항목에 옮겨서 저장하는 기능을 추가하여 구현했다.

Number가 10일 때는 Quit 과 관련된 ‘감사합니다’를 출력 후 프로그램을 종료한다.

[2] What is your expected performance?

- 처음 목표로 했던 것은 내용의 변경이 txt파일에 기록되어 지속적으로 정보가 변경되는 것이었다. 그러나 현실적으로 txt파일에 영향을 미치게 하는 방법을 몰랐기 때문에 처음 프로그램을 실행한 후 종료하기까지 정보가 유지되는 것으로 목표를 바꾸었다. 처음 정보를 저장하는 과정은

이후 교수님께 문의하여 처음의 O(n^3)의 정보 저장 방식에서 O(n^2)으로 변경하여 저장할 수 있었다. 이렇게 list의 형태로 정보를 저장한 이후의 기능이 O(nlogn)으로 실행될 수 있도록 list의 내용을 처음부터 읽어서 정보를 출력할 수 있게 하였다. 리스트는 word기반, id기반으로 두 가지고 존재했으므로 목적에 맞는 리스트를 선택한다면 하나의 내용을 search하는 것 자체는 O(n)으로 가능했다. 그러나 worst case는 모든 사람이 모든 사람과 친구이며 모든 사람이 특정 단어를 언급하는 경우가 될 수 있고 이 경우 리스트 정보의 비교는 O(n^2)으로 실행된다.

현실적으로 그런 경우가 드물고 평균적인 트위터 숫자나 친구의 숫자가 전체 다가 될 가능성이 적기 때문에 한 사람이 말한 단어의 숫자나 한 사람이 follow하는 숫자(edge)는 n보다 적은 수가 될 것이며 O(n^2)이 worst case지만 실행은 이보다는 빠를 것으로 기대된다.

[3] How would you improve the system in the future?

- Shortest path 문제를 풀지 못했다. Dijkstra algorithm을 사용해야 할 것으로 예상되지만 이를 실제로 구현하는 단계에서 실패했다. 이 부분에 대한 보완이 필요하다.

- 계속적으로 컴퓨터가 멈추는 현상이 발생하여 strongly connected component는 제대로 작동하는지 확인이 되지 않았기 때문에 더 보완이 필요할 것으로 생각된다.

- linked list삭제가 원하는 대로 이루어지지 않았다. 때문에 하나의 함수로 깔끔하게 삭제하지 못하고 여러 가지 방법을 이용하여 나누어 정리하였다.

- 사람 기반, word기반으로 나누어 정리하였는데 이를 동시에, 그리고 효과적으로 저장할 수 있는 방식을 찾아야 할 것 같다.

- nlogn으로 알고리즘을 만들고 싶었으나 모두가 모두와 친구인 경우, 모두가 특정 단어를 언급한 경우에는 search가 n^2으로 수행되는 단점이 있다. 이 부분을 해결해야 한다.

[4] Self evaluation

* Submit a github account = 10
* Commit source code displaying menu = 10
* Commit the first draft of manual = 10
* Read data files = 20
* Statistics = 20
* Top 5 most tweeted words = 10
* Top 5 most tweeted users = 5
* Find all users who mentioned a word = 10
* Find all users who are friend of the above user = 5
* Top 5 strongly connected components = 5
* Find shortest path from a user = 0