2012120151 정상덕

**자료구조 Final Project Report**

사용자 Profile의 경우 key값으로 빠르게 Id를 찾아와야 하므로 Hash방식을 사용하였습니다. 과제 요구사항에 있어서 Word Data의 경우 검색해야 하는 경우가 많았으므로 처음에는 Binary Search Tree를 생각하였습니다. 그러나 중간고사 이전의 Binary Search Tree를 구현해 보았지만 중복 값을 잘 처리하지 못하였습니다. 그래서 좀 더 Advanced된 Binary Search Tree와 Balanced된 Tree로 구현해 보고 싶다라는 생각을 하게 되었습니다. 그 결과 Word Data에 대하여서는 Red-Black Tree를 적용하게 되었습니다. 확실히 중복된 값이 있어도 문제가 없었고 검색이나 삽입, 삭제에 있어서도 더 나은 결과를 얻을 수 있었습니다. 다만 Tree의 key값을 Word의 아스키 값으로 바꾸어 주는 과정에서 한글은 2bytes고 영어는 1byte인 관계로 한글의 첫 글자는 음수로 나와서 key값이 고루 분배되지 못하는 문제가 발생하였습니다. 이를 해결하기 위해 모든 글자의 아스키 값에 대하여 절대값처리를 하고 그 외에도 단순히 더하기 연산보다는 곱하기와 덧셈을 골고루 조합하였습니다.

Strongly Connected Component를 구하려면 Friend관계에 해당하는 Data는 Vertex와 인접리스트로 나타내야 한다고 생각했습니다. 그래서 Friend Data를 위해 일단 DFS Vertex노드를 정의하였습니다. 그러나 user의 총 숫자를 알 수 없었으므로 정적 할당이 아니라 동적 할당이 가능하도록 Vertex 구조체 내에도 Next포인터를 하나 만들어서 List의 형태로 만들게 되었습니다. 각각의 Vertex에 대하여서는 Adj 구조체를 통하여 인접리스트를 만들어 주었습니다. 그리고 DFSVertex와 Adj list에 각각 해당하는 값들을 넣었고, Transpose와 DFS함수까지 모두 구현했습니다. 그렇지만 f값에 따라 정렬하는 방법을 구현하지 못하여 Strongly Connected Component는 구해내지 못하였습니다. 만약 f값에 따라 정렬하는 방법을 구현하여 DFS함수를 통해 f값을 찾아내고.. Transpose된 것을 가장 큰 f값부터 다시 DFS한다면 구할 수 있을 것이라 예상합니다.

Statistic에서 요구하는 조건을 찾기 위해 Word Data와 Friend Data를 한 줄씩 읽어올 때 이전에 이미 읽었던 값인지를 확인하는 작업이 필요했습니다. 이미 나왔던 data이면 Count만 증가시켜주는 작업을 위해 불필요하게 Binary Search Tree를 각각 만들어서.. 매번 검색해주는 과정을 거쳤습니다. 그나마 Binary Search라서 빠르게 찾아올 것이라고 생각했지만 이러한 방식을 새로운 값을 넣을 때마다 정렬되게 만든다면 새로운 값인지 확인해야 될 때 일부분과 비교하기만 하면 되기 때문에 더욱 향상된 결과를 나타낼 것이라고 기대합니다.

또한 기존 word의 아스키 값을 기준으로 만들어진 Red-Black Tree에서 Tree를 순회하면서 Count를 기준으로 한 임시적인 Binary Search Tree를 만들어주는 과정을 거쳐서 Top 5 tweet과 Top 5 user를 산출하게 되었습니다. 이런 부분에 있어서 공간할당을 불필요하게 너무 많이 해서 따로 Tree를 만들지 않고도 Count 순으로 정렬하는 방법을 사용했더라면 더 좋은 성능을 낼 수 있었을 것이라 예상합니다.

단어를 통해 그 단어를 사용하는 모든 사용자들을 찾는 과정에서는 기존에 중복된 값을 처리하기 위해 별도의 Binary Search Tree를 만들 때 이미 중복된 값이면 Count와 함께 해당 노드에 list로 사용자의 key를 넣어주었습니다. 그래서 단어를 입력 받고 그 단어를 다시 아스키값으로 변환하는 과정을 거쳐서 해당 노드를 찾은 뒤 list를 읽어오는 방식으로 찾아내었습니다. 다만 아스키값 변환 함수를 최대한 고루 분산되도록 만들었지만 다른 단어임에도 같은 아스키값을 반환하는 경우가 많이 발생하였습니다. 그래서 노드를 검색할 때 단순히 아스키값으로만 찾을 수 없었고 단어 아스키값과 word를 같이 입력 받아서 아스키 값이 같을 뿐만 아니라 strcmp함수를 통해 해당 노드의 word와 입력된 word가 같을 경우에 검색이 완료되도록 하였습니다.