알고리즘 과제(07)

201404377 진승언

1. 과제 목표 및 해결 방법

➔ 이번과제는 트리 Union-Find를 구현하는게 목표였다. 상호 배타적 집합(완전히 독립된 집합들(교집합 X))을 관리하는 자료구조 중 트리를 활용한 방법이다. 이것을 노드를 직접 구현해서 사용하는데 Node에는 본인ID(식별자), 부모, 번호, Next(입력상 다음 원소)가 필요했다. 결과물 출력파일은 2개가 나와야하는데 1차 출력(Output1.txt)은 각 노드가 가진 번호를 기준으로 자신과 부모의 ID출력, 2차 출력(Output2.txt)은 모든 Tree를 번호 순서대로 Union후 모든 노드의 자신과 부모의 ID출력이였다.

unionFind를 구현할려면 크게 3가지 과정이 필요한데 Make-Set과 Find-Set 그리고 Union이다. 각각 설명하자면 Make-Set은 원소 x로만 구성된 집합을 만드는 것이고 Find-Set은 원소 x를 가진 집합을 알아내는 것이고 Union은 원소 x를 가진 집합과 원소 y를 가진 집합을 하나로 합치는것이다.

먼저 데이터 1, 2, 3, 4, 5인 데이터를 묶어줄 tmp노드들을 생성하고 읽어온 데이터를 id와 데이터로 노드 생성을 한 후 makeSet을 해주었다. 이 읽어온 데이터들은 next를 이용해서 연결해 줌으로서 읽어온 노드를 순서대로 불러올 수 있게끔 해주었다. 그리고 이 읽어온 노드들을 같은 데이터끼리 union이 도되록 해줘서 1차출력을 완료하였다. 그리고 같은 데이터끼리 묶여진 상태인 트리를 서로 합쳐주도록 union을 하여 2차출력을 완료하였다

..

2. 주요 부분 코드 설명(알고리즘 부분 코드 캡쳐)

```
Node tmp1 = new Node(1);     //1~5까지 같은 데이터를 묶어줄 때 사용할 노드
Node tmp2 = new Node(2);
Node tmp3 = new Node(3);
Node tmp4 = new Node(4);
Node tmp5 = new Node(5);
union.makeSet(tmp1);
union.makeSet(tmp2);
union.makeSet(tmp3);
union.makeSet(tmp4);
union.makeSet(tmp5);
```

먼저 위와 같이 1, 2, 3, 4, 5인 데이터들을 묶어줄 임시노드를 만들었다. 그리고 makeSet을 해주었다.

```
public class Node {

    Node parent;
    Node next;
    String id;
    int data;
    Node(){

    }
    Node(int data){
        this.data= data;
    }

    Node(String id) {
        this.id = id;
    }
    Node(String id, int data){
        this.id = id;
        this.data = data;
    }

    public void setData(int data) {
        this.data = data;
    }

}
```

기본 노드의 구조이다. 노드는 id와 data값을 갖고있으며 parent노드와 next로 연결된 노드를 갖고있다.

```java

public class UnionFind {

    public void makeSet(Node x) {    //노드의 부모를 자신으로 함
        x.parent = x;
    }


    public Node findSet(Node x) {    //해당 노드의 최상위 부모를 찾음
        if(x.parent == x) {
            return x;
        }
        else {
            return findSet(x.parent);
        }
    }

    public void unions(Node x, Node y) {  //한노드의 최상위노드의 부모를 다른 노드의 최상위노드로 이어줌
        findSet(y).parent = findSet(x);
    }
```

위는 UnionFind 코드이다. makeSet은 처음 노드를 만들 때 사용하는데 해당노드의 부모를 자기자신으로 한다.

findSet은 재귀적으로 계속 해당노드의 부모를 타고올라가서 최상위 부모노드를(루트노드) 반환하는 함수이다.

unions는 노드 2개를 서로 합쳐주는데 findSet으로 최상위 노드를(루트노드) 찾은 후 그 노드 두개를 부모와 자식관계로 이어준다.

```java
Node totalNode = new Node();     //읽어온 노드를 저장하는데 사용할 노드(LinkedList 자료구조처럼 사용할거
Node head = totalNode;  //totalNode를 사용한 후 초기화할때 사용할 노드 (처음을 가리키는 포인터역할)

//읽어온 id와 데이터를 가진 노드를 생성후 makeSet 후 totalNode에 한꺼번에 저장
for(i=0; i< size; i++) {
    Node tmp = new Node(data1[i], data2[i]);
    union.makeSet(tmp);
    totalNode.next = tmp;
    totalNode = totalNode.next;
}
```

입력파일로부터 저장된 id와 데이터로 노드를 생성한 후 totalNode라는 노드를 만들어서 데이터로 읽어온 노드들을 관리하고 next로 이어지게끔 만들어주었다.

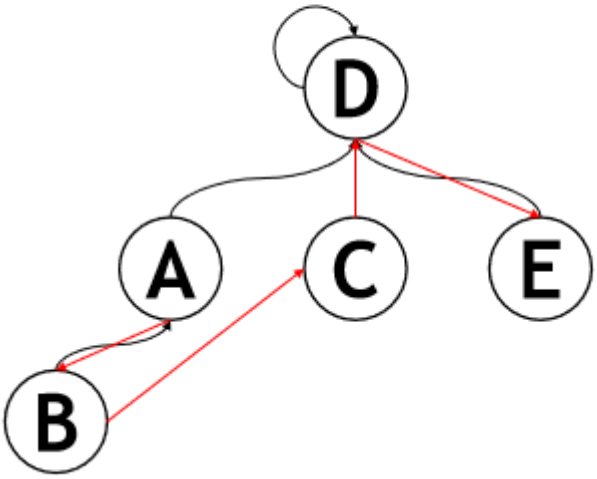
```
//과제 Union1
while(totalNode.next != null) {
    if (totalNode.next.data == 1) {
        union.unions(totalNode.next, tmp1);
    }
    else if(totalNode.next.data == 2) {
        union.unions(totalNode.next, tmp2);
    }
    else if(totalNode.next.data == 3) {
        union.unions(totalNode.next, tmp3);
    }
    else if(totalNode.next.data == 4) {
        union.unions(totalNode.next, tmp4);
    }
    else if(totalNode.next.data == 5) {
        union.unions(totalNode.next, tmp5);
    }
    totalNode = totalNode.next;
}

totalNode =head;
```

next로 이어진 노드들을 불러오면서 각각의 같은 데이터끼리 묶어지게끔 union해주었다. while문으로 처음 노드부터 끝 노드까지 순차적으로 불러오고 if문으로 같은 데이터끼리 묶어질 수 있게끔 하였다. 그러므로 각 노드가 가지고 있는 숫자를 기준으로 Union 되었다.

```
//모든 Tree를 유니온 수행 (데이터 1,2,3,4,5 가진 트리들을 차례로대로 union) <union2>
union.unions(tmp2, tmp1);
union.unions(tmp3, tmp2);
union.unions(tmp4, tmp3);
union.unions(tmp5, tmp4);
```

트리 Union-Find

모든 Tree를 유니온 수행
두번째 출력(Output2.txt) 모든 노드의 자기 자신과 자신의 부모 출력



같은 데이터들끼리 묶어진 데이터 트리들을 위 ppt화면과 같은 순서로 union시켜주었다. 그러므로 (A,B)와 (D,C,E)노드를 union할 때 D가 루트노드가 되게끔 하였다.

결과(시간 복잡도 포함)

1. output1

```
Below is  Output1

CFXGD   KSSOJ
KUMPN   ADHXE
KSSOJ   EHCHF
EMGNT   DCLSL
DCLSL   LYETM
PFVNH   FSSVX
NJJJK   SDIDY
LYETM   UALBL
EHCHF   WMMMB
WMMMB   WSIQS
SDIDY   IDJRQ
IDJRQ   KCJPN
KCJPN   MDAXY
UALBL   UALBL
FSSVX   VCPUC
MDAXY   MDAXY
VCPUC   GJAFJ
GJAFJ   GJAFJ
WSIQS   WSIQS
ADHXE   ADHXE

```

각 노드가 가지고 있는 숫자를 기준으로 Union 수행 첫번째 출력한 결과로서 모든 노드의 자기 자신과 자신의 부모가 출력됨을 볼 수 있었다.

2.output2

```
Below is  Output2

CFXGD   KSSOJ
KUMPN   ADHXE
KSSOJ   EHCHF
EMGNT   DCLSL
DCLSL   LYETM
PFVNH   FSSVX
NJJJK   SDIDY
LYETM   UALBL
EHCHF   WMMMB
WMMMB   WSIQS
SDIDY   IDJRQ
IDJRQ   KCJPN
KCJPN   MDAXY
UALBL   MDAXY
FSSVX   VCPUC
MDAXY   WSIQS
VCPUC   GJAFJ
GJAFJ   UALBL
WSIQS   WSIQS
ADHXE   GJAFJ

```

모든 Tree를 유니온 수행 두번째 출력한 결과로서 모든 노드의 자기 자신과 자신의 부모가 출력됨을 볼 수 있었다.

Make-Set은 하나의 원소로 된 집합을 초기화하는 것이므로 상수 시간이 든다. 단순히 노드의 부모를 자기 자신으로 설정하는 것이기 때문이다. FindSet과 Union은 노드의 트리 구조에 따라 시간복잡도가 달라지는데 트리 구조이므로 평균 logn의 시간복잡도를 지닌다고 볼 수 있고 노드의 개수만큼 반복을 해야한다. 따라서 총 시간복잡도는 상수를 무시한 O(nlogn) 이라고 볼 수 있다.