실습보고서

2020. 12. 06 (일)

[알고리즘 00분반]_14주차
201501513_김성현

조건사항
운영체제 : MAC OS
IDE : IntelliJ
JDK 11 version
라이브러리 : io(입출력), util(코드설계), nio(파일읽기)

Error 사항) 파일을 불러올 때 상대경로로 "./"로 설정하여 같은 패키지 내가 아닌 같은 프로젝트 디렉토리에서 파일을 읽습니다.

1.

<실행 결과>

```
man preference array
1 2 3
1 3 2
3 2 1
woman preference array
2 1 3
1 3 2
2 3 1

process 1
man  : 1 0 0
woman: 1 0 0

process 2
break couple!
man  : 0 1 0
woman: 2 0 0

process 3
man  : 0 1 3
woman: 2 0 3

process 4
man  : 2 1 3
woman: 2 1 3


stable matching algorithm result
man1 is couple with woman2
man2 is couple with woman1
man3 is couple with woman3
```

<코드 설명>

```
String path = "./";
List<String> list = Files.readAllLines(Paths.get( first: path+"data14.txt"));
String[] input = new String[list.size()];
for(int i = 0; i < list.size(); i++){
    input[i] = list.get(i);
}

stable_solution ss = new stable_solution();
ss.solution(input);
```

오류사항에 말씀드린 듯이 path 를 ./로 설정하여 같은 같은 프로젝트 내에 존재하는 data14.txt를 읽어 list 형태로 저장한 다음 input 이라는 문자열 배열을 만들어 해당 값들을 문자열로 저장하도록 하였습니다.

그리고 솔루션이 작동하는 stable_solution 클래스를 불러와 해당 솔루션 메소드를 실행하도록 하였습니다.

```
int[][] man;
int[][] woman;

//save couple after stable matching algorithm
int[] count;
int[] result;
int[] result_convert;
```

먼저 전역변수로 남자들의 여성에 대한 선호도를 저장할 배열과 여성들의 남성에 대한 선호도를 저장할 이차원배열을 형성하였습니다.

그리고 먼저 고백을 하는 남성들의 선호도 순위를 파악하기 위해 count라는 배열을 생성해 몇번째 순위의 상대방한테 고백하는 지 알 수 있도록 하였습니다.

Result 배열은 남성번호에 대해 커플이 된 여성번호를 나타내며 result_convert는 여성번호에 대해 커플이 된 남성번호를 의미합니다.

```java
public void solution(String[] input){
    int length = input.length;
    int people = length/2;
    //System.out.println(people);

    man = new int[people+1][people+1];
    woman = new int[people+1][people+1];

    for(int i = 0; i < length; i++) {
        String[] str = input[i].replaceAll( regex: "[a-zA-Z]",  replacement: "").split( regex: " ");
        int row = Integer.parseInt(str[0]);

        if (i < people) {//man
            //print(str); //for debugging
            int k = 1;
            for (int j = 1; j < str.length; j++) {
                int column = Integer.parseInt(str[j]);
                man[row][column] = k++;
            }
        }
        else {//woman
            //print(str); //for debugging
            int k = 1;
            for (int j = 1; j < str.length; j++) {
                int column = Integer.parseInt(str[j]);
                woman[row][column] = k++;
            }
        }
    }
```

Solution 메소드에서는 입력받은 문자열의 길이를 남여 반으로 나누고 1번부터 시작하는 번호를 맞추기 위해 각 배열을 선언할 때 people(남여의 수를 나타내는 int)+1 로 배열을 선언하였습니다.

그리고 반복문을 통해 입력받은 문자열들을 split을 통하여 띄어쓰기 단위로 배열을 형성하는데에 있어서 영어로 된 부분들을 모두 정규표현식을 사용해 삭제하도록 하였습니다. 그리고 먼저 입력받은 가장 첫번째 배열이 현재 입력될 남자의 번호이므로 행으로 설정하고 people의 값보다 작으면 남자 크면 여자로 구분하여 다시 반복문을 통해 열들에 대하여 선호도를 저장하도록 하였습니다.

```
System.out.println("man preference array");
print(man);
System.out.println("woman preference array");
print(woman);
System.out.println();

//for stable matching
Queue<Integer> que = new LinkedList<>();
result = new int[people+1];//initialization to zero
result_convert = new int[people+1];
count = new int[people+1];
for(int i = 0; i < count.length; i++){
    count[i] = 1;//initialization to one
}

for(int i = 1; i <= people; i++){
    que.offer(i);
}
```

그리고 각 선호도를 나타내는 배열을 출력해준 다음 Queue를 선언해 순서대로 남성들의 번호를 저장해주록 하면서 각 배열들을 선언해주고 count의 경우 여성 역시 번호가 1번부터 시작하므로 1로 모두 초기화를 해주었습니다.

```java
int plus = 1;
while(!que.isEmpty()){
    System.out.println("process "+(plus++));
    int _man = que.poll();//1
    int count_temp = count[_man];//해당 남자의 순위 -> 기각 당한 적이 있는지 확인

    int wantedWoman = man[_man][count_temp];//남성이 고백하는 여성(1순위부터 순차적으로) //1

    if(result_convert[wantedWoman] == 0){//만약 좋아하는 여성이 커플이 아니라면
        result_convert[wantedWoman] = _man;
        result[_man] = wantedWoman;
        count[_man] += 1;
    }
    else{//만약 선호하는 여성이 커플이라면
        int competition = result_convert[wantedWoman];//경쟁하는 남자 번호 //1

        int cpt_score = woman[wantedWoman][competition];
        int man_score = woman[wantedWoman][_man];

        if(cpt_score > man_score){//순위가 같을 일이 없으니까 단순히 구분
            System.out.println("break couple!");
            result[competition] = 0;//disconnect couple
            //기존 커플 깨짐ㅠ

            result_convert[wantedWoman] = _man;
            result[_man] = wantedWoman;

            que.offer(competition);
        }
        else{
            count[_man] += 1;
            que.offer(_man);
        }

    }
```

Plus의 경우 process 과정을 출력하는데에 있어서 표시하기 위해 선언을 하였고 while 반복문을 통해 앞서 선언한 que가 빌 때까지 실행하도록 하였습니다.

_man은 현재 남성의 번호를 나타내며 count_temp는 남성이 고백할 대상(선호도 순)을 나타냅니다. 그리고 wantedWoman의 경우는 실제로 고백하는 여성의 번호를 나타냅니다.

그리고 wantedWoman이 해당하는 result_convert의 배열 값이 0이라면 고백을 받은 적이 없음을 나타내므로 바로 커플이 성사되게 result와 result_convert 배열에 남여번호를 지정해두고 커플이 깨지는 경우는 고려하여 count배열에서 해당 남성의 다음 순위를 나타내도록 하였습니다.

그리고 만약 wantedWoman이 이미 커플이라면 현재 커플인 남성의 번호를 competition이라는 변수에 저장하여 해당 여성의 커플인 남성의 순위와 현재 남성의 순위를 비교하여 각 cpt_score 변수에는 현재 고백한 여성의 현재 커플인 남성의 순위, man_score변수에는 현재 고백한 여성의 현재 고백한 남성의 순위를 저장하고 비교를 하도록 하였습니다.

그래서 만약 현재 남성의 score 값이 더 낮은 경우 커플이 깨진 것을 표시하고 커플이었던 남자에 해당하는 result 배열의 값을 다시 0으로 초기화 해준 다음 result_convert에서 고백한 여성은 고백한 남성과 연결시키고 result에서 고백한 남성은 고백한 여성과 이어주고 커플이었던 남자는 다시 que에 넣어 추후에 커플이 다시 되도록 작성하였습니다.

그리고 만약 현재 커플인 남성의 score(cpt_score) 가 더 높다면 고백한 남성의 다음 고백할 사람을 체크하기위해 count 배열에서 고백한 남성에 해당하는 값에 +1을 해주고 다시 que에 넣어 결과적으로 모든 사람이 될 수 있는 한 stable matching이 되도록 구현하였습니다.