

## 알고리즘 6 주차

201902694 - 박재우

RANK	TEAM	SCORE	1-MARATHON ○ [1 POINT]	2-CutTREE ● [2 POINTS]	3-TwoMUL ● [3 POINTS]	4-DATASAMPLING ● [4 POINTS]
19	201902694	10 691	46 6 tries	162 1 try	188 1 try	175 2 tries

Submissions				Clarifications
time	problem	lang	result	No clarifications.
10/12/20-17:22	3-TwoMUL	JAVA	CORRECT	Clarification Requests
10/12/20-17:09	4-DATASAMPLING	JAVA	CORRECT	No clarification request.
10/12/20-16:56	2-CutTREE	JAVA	CORRECT	<button>request clarification</button>
10/12/20-16:04	4-DATASAMPLING	JAVA	WRONG-ANSWER	
10/12/20-15:01	1-MARATHON	JAVA	CORRECT	
10/12/20-14:53	1-MARATHON	JAVA	WRONG-ANSWER	
10/12/20-14:49	1-MARATHON	JAVA	WRONG-ANSWER	
10/12/20-14:23	1-MARATHON	JAVA	RUN-ERROR	
10/12/20-14:22	1-MARATHON	JAVA	RUN-ERROR	
10/12/20-14:19	1-MARATHON	JAVA	NO-OUTPUT	

### 문제 1

#### a. 문제 / 목표

마라톤경주에서 완주하지 못한 선수들 이름 사전순 출력

#### b. 해결방법 (소스코드 첨부)

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.HashMap;
import java.util.Scanner;

public class first {

    public static String[] compares(String[] ar1, String[] ar2) {
        String[] result = new String[ar1.length - ar2.length+2];
        HashMap<String, Integer> map = new HashMap<>();
        for (String element : ar1)
            map.put(element, map.getOrDefault(element,0)+1);
        for (String element : ar2)
            map.put(element, map.get(element) - 1);
        int i = 0;
        for (String key : map.keySet()) {
            if(map.get(key) != 0) {
                for (int j=0;j<map.get(key);j++) {
                    i +=1;
                }
            }
        }
    }
}
```

```

        result[i] = key;
    }
}
return result;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String[] ar1 = sc.nextLine().split(" ");
    String[] ar2 = sc.nextLine().split(" ");
    String[] temp = compares(ar1, ar2);
    ArrayList<String> a = new ArrayList<>();
    for(int i=0; i<temp.length; i++) {
        if(temp[i] != null)
            a.add(temp[i]);
    }
    Collections.sort(a);
    for(int i=0; i<a.size(); i++)
        System.out.println(a.get(i));
}
}

```

강의 시간에 배운 마라톤 문제풀이를 응용하여 풀었다.  
 우선 참가한 선수들의 이름을 string 배열로 저장한뒤에  
 이를 key값으로 하여 hashmap을 생성후에 저장하였으며  
 Key값에 해당되는 value는 이름당 1로 주었다.  
 또한 동명이인이 있을경우 추가적으로 1의 값을 더해주었다.  
 이후 도착명단을 통해 위의 방법과 동일하게  
 이름을 key값으로 하여 value값을 1씩 빼주었다.  
 따라서 0의 값이 아닌경우 value의 숫자만큼 이름을 출력하도록 했다.

#### c. 결과

```

t t t t c c c s s a a
a t t c s s
a
c
c
t
t

```

## 시간 복잡도

: collections.sort 의 시간복잡도는  $O(n * \log(n))$ 이지만 이중 for 문을 사용하므로  $O(n^2)$ 이다.

## 자신만의 생각 / 느낀점 :

우선 1 번문제는 이론시간에 배웠던 알고리즘에서 크게 벗어나지 않음으로 크게 문제가 없었다고 느꼈다.

하지만 생각대로 수정되지 않아 4 문제중에서 가장 많은 시간을 잡아 먹는 문제였다. 거의 1 시간 반동안 이문제만 풀었다.

난이도 : 4/5

## 문제 2

### a. 문제 / 목표

입력받은 크기로 입력받은 나무들이 그 크기에 맞게 규칙에 따라 자른 회수를 출력

### b. 해결방법 (소스코드 첨부)

```
public class second {

    public static void test (ArrayList<Integer> a, int num, int k) {
        if(num <= k)
            a.add(num);
        else {
            int temp1 = num/3;
            int temp2 = num*2/3;
            test(a, temp1, k);
            test(a,temp2,k);
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int k = sc.nextInt();
        sc.nextLine();
        String[] ar = sc.nextLine().split(" ");
        int[] num = Arrays.stream(ar).mapToInt(Integer::parseInt).toArray();
        ArrayList<Integer> a =new ArrayList<>();
        for(int i=0;i<num.length;i++)
            test(a,num[i],k);
        System.out.print(a.size() - num.length);
    }
}
```

```

}
}

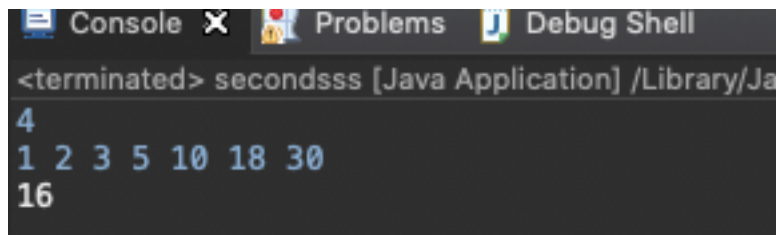
```

우선 목표크기  $k$ 를 입력받고 나무들의 크기를 int배열로 변환하여 저장한다.

그후 arrayList와 재귀를 이용한다. 반복문 속에서 int 배열에 저장된 원소 하나씩 재귀함수를 통해  $k$ 와 같거나 작을 경우 arrayList에 저장하고 아니라면 규칙에 맞게 들어온 int값을 재귀를 돌려  $k$ 와 같거나 클때까지 돌린다. 이 경우 arraylist에는  $k$ 와 같거나 작은 값들만 저장된다.

최종적으로 만들어진 arraylist의 크기에 원래 나무개수를 빼면 나무를 벤 횟수가 나온다.

#### c. 결과



```

<terminated> secondsss [Java Application] /Library/Ja
4
1 2 3 5 10 18 30
16

```

시간 복잡도 :  $T(n) = T(n/3) + T(n*2/3) + c$  (재귀의 사용)

자신만의 생각 / 느낀점 :

2번문제는 재귀를 이용해서 풀었더니 짧은 시도 끝에 간단하게 풀 수 있었다. 또한 구현에도 큰 어려움이 없었고 java의 내부 라이브러리를 이용해서 크게 구현할 코드가 많이 없었다고 생각한다.

난이도 재귀사용시 1/5

질문 : 원래의 의도대로는 문제를 해결을 잘 하지 못하겠다고 느꼈다. 원래 의도의 풀이가 무엇인지 궁금하다.

### 문제 3

#### a. 문제 / 목표

k 와 숫자 리스트가 주어졌을 때 곱하여 k 가 나오는 숫자쌍의 index 를 출력

#### b. 해결방법 (소스코드 첨부)

```
package algorithm.week6;

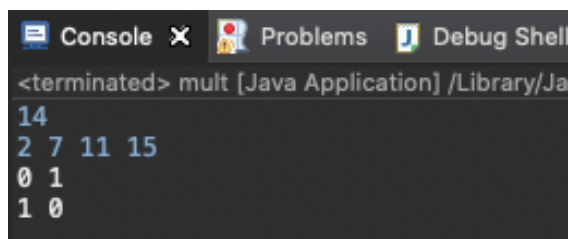
import java.util.Arrays;
import java.util.Scanner;

public class third {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int k = sc.nextInt();
        sc.nextLine();
        String[] ar = sc.nextLine().split(" ");
        int[] num = Arrays.stream(ar).mapToInt(Integer::parseInt).toArray();
        Arrays.sort(num);
        for(int i=0;i<num.length;i++) {
            int les = k%num[i];
            if(les==0) {
                int i2 = Arrays.binarySearch(num,k/num[i]);
                if(i2>=0)
                    System.out.println(i+" "+i2);
            }
        }
    }
}
```

우선 k값과 숫자리스트를 입력받은 뒤에 for문을 통해 가장 첫번째 원소부터 k값을 나눠 나온 값중 나머지가 0인 몫을 binarySearch를 이용해서 숫자리스트에 있는지 검사한다. 만약 존재하여 binarySearch의 반환값이 음수가 아니라면 나눈 숫자의 index와 검색하여 나온 index를 출력한다.

#### c. 결과



```
Console x Problems Debug Shell
<terminated> mult [Java Application] /Library/Ja
14
2 7 11 15
0 1
1 0
```

시간 복잡도 :  $O(n \cdot \log n)$

자신만의 생각 / 느낀점 :

가장 단기간에 해결했던 문제였다고 생각한다.

우선 문제출제의도대로 풀지 않았기 때문일지도 모르지만 원소를 찾는것도 `arrays.binarySearch`로 이진탐색으로 빠르게 원하는 값을 찾을 수 있었으며, 함수 특성상 원하는 값이 없을 경우도 특정 할 수 있기에 빠르게 문제를 풀 수 있었다고 생각한다.

난이도 1/5

#### 문제 4

##### a. 문제 / 목표

데이터 종류별로 수집된 크기가 입력되고, 데이터 종류별 샘플링 가중치가 입력되었을 때 데이터 종류별 가중치 비율을 유지하며 최대로 샘플링한 수를 출력

##### b. 해결방법 (소스코드 첨부)

```
import java.util.Arrays;
import java.util.Scanner;

public class forth {

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String[] ar1 = sc.nextLine().split(" ");
        String[] ar2 = sc.nextLine().split(" ");
        int[] num1 = Arrays.stream(ar1).mapToInt(Integer::parseInt).toArray();
        int[] num2 = Arrays.stream(ar2).mapToInt(Integer::parseInt).toArray();
        int i = 1;
        int [] temp = num2.clone();
        boolean result = true;
        while(result) {
            temp = num2.clone();
            for(int j=0;j<num1.length;j++) {
                temp[j]*=i;
            }
        }
    }
}
```

```

        if(temp[j] > num1[j])
            result = false;
    }
    i++;
}

String test = Integer.toString(num2[0]*(i-2));
for(int j=1;j<num2.length;j++)
    test+=" "+Integer.toString(num2[j]*(i-2));
System.out.println(test);
}
}

```

우선 데이터 총량과 데이터 가중치를 int배열로 변환하여 저장한다.

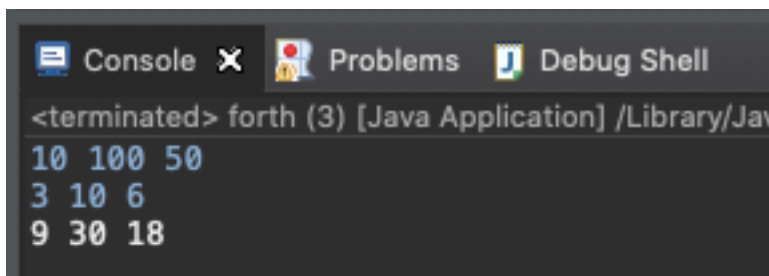
데이터 가중치 배열을 temp배열에 clone 시킨다

그후 while문을 통하여 데이터 가중치에 1~n을 곱해가면서 가중치\*n이 데이터 총량을 넘지 않도록 한다. 매번 반복될때마다 중복해서 곱해지지 않도록 temp를 가중치 배열로 초기화 해준다.

만약 데이터의 총량을 가중치가 뛰어 넘었으면 while문을 돌리고 있던 true를 false로 변경하여 while문을 탈출한다.

또한 반복되면서 증가한 i와 가중치의 곱은 데이터 총량을 뛰어넘어 While문을 탈출 한것임으로 넘기전의 i인 i-2를 가중치에 곱해준다음 출력 형태에 맞게 수정한뒤 출력한다.

### c. 결과



```

<terminated> forth (3) [Java Application] /Library/Jav
10 100 50
3 10 6
9 30 18

```

시간 복잡도 : 특정조건까지의 while 문 반복 :  $O(n)$

자신만의 생각 / 느낀점 :

문제의 해결방법을 떠올리는데에는 시간이 많이 걸리지 않았다.

단 데이터 총량과 데이터 가중치의 개수가 3개인줄 알고 3개만 받고 출력하도록 구현했다가 결과 값이랑 달라 이유를 모른체 계속 수정을

시도했었다. 다행이 20 분이 지난뒤 가능한 한 많이 받을 수 있다는 것을 알게되어 이를 수정하니 정상 작동이 되었다.  
난이도 자체는 어렵지 않았다고 생각한다.

난이도 2/5