알고리즘

- HW04 : Divide & Conquer 2-

제 출 일	2017년 10월 18일
분 반	02반
담당교수	공은배
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

ㄱ.countInversion

1. 해결 방법

간단히 생각하면 현재 배열 상태에서 작은수가 큰수보다 더뒤에 있는 횟수를 세는 것이다.
MergeSort하면서 오른쪽배열이 왼쪽배열보다 더 작아서 뽑혀서 Merge 될 때마다 현재 왼쪽배열에 남아있는 숫자의 개수를 더하면된다.
counting은 static으로 관리하면서 계산하면되고 mergesort시 남은 배열의 개수를 알기위해 변수를 선언하면된다. 여기선 I로 선언하였다.

```
    static int[] CountInversion(int[] input,int start,int end){
        if(end-start==0) {
            return input;
        }//배열이 하나라면 리턴

        int[] temp1 = new int[(end-start+1)/2];
        System.arraycopy(input, start, temp1, 0, (end-start+1)/2);
        int[] temp2 = new int[(end-start+1)-((end-start+1)/2)];
        System.arraycopy(input, (end-start+1)/2-start, temp2, 0, (end-start+1)-((end-start+1)/2));
        //배열을 반으로 잘라서 저장해준다
        temp1 = CountInversion(temp1,0,(end-start+1)/2-1);
        temp2 = CountInversion(temp2,0,(end-start+1)-((end-start+1)/2)-1);
        //나눈 배열들도 CountInversion 시켜준다.
        input = MergeSort(temp1,(end-start+1)/2,temp2,(end-start+1)-(end-start+1)/2);
        //나눈 배열을 정렬하면서 합쳐준다.
        return input;
    }
    static int[] MergeSort(int[] input1,int input1size,int[] input2,int input2size) {
        int[] result = new int [input1size+input2size];
        int i=input1size;
        int j=input2size;//i와 j는 남은 배열의 수를 관리하기위해 선언하였다.
        int a=0,b=0,k=0;//배열의 접근을 위한 변수 선언
        while(i!=0&&j!=0) {//한쪽에 남은 배열이 없다면 종료한다.
            if(input1[a]>input2[b]) {
                result[k++]=input2[b];
                b++;
                j--;
                counting+=i;
                //만약 input1이 더커서 input2가 결과 배열에 들어갈때 inversion이 생긴다 그값은 현재 input1에 남은 배열의 수만큼이다.
            }
            else {
                result[k++]=input1[a];
                a++;
                i--;//input2가 더크다면 input1값을 뽑아서 결과값에 저장한다 이때는 inversion이 일어나지않는다.
            }
        }
        if(i!=0) {
            System.arraycopy(input1, a, result,b+a, input1size-a);
        }
        if(j!=0) {
            System.arraycopy(input2, b, result,b+a, input2size-b);
        }//0인 부분이 나오면 한쪽은 다썼지만 나머지 한쪽이 있으므로 그대로 저장한다.
        return result;//정렬이 완료된 배열을 리턴한다.

    }
```

2. 실행 결과

```
<terminated> Inversion [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:17:10)
input data : 1 5 4 8 10 2 6 9 12 11 3 7
Sort data : 1 2 3 4 5 6 7 8 9 10 11 12
InversingCount : 22
```

```
<terminated> Inversion [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:35:51)
input data : 1 2 3 4 5 6 7 8 9 10 11 12
Sort data : 1 2 3 4 5 6 7 8 9 10 11 12
InversingCount : 0
```

```
<terminated> Inversion [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:36:18)
input data : 12 11 10 9 8 7 6 5 4 3 2 1
Sort data : 1 2 3 4 5 6 7 8 9 10 11 12
InversingCount : 66
```

```
<terminated> Inversion [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:36:48)
input data : 5 8 11 2 4 1 9 3 12 7 10 6
Sort data : 1 2 3 4 5 6 7 8 9 10 11 12
InversingCount : 29
```

ㄴ.Karatsuba

1. 해결 방법

처음에는 n자리수가 전부다 2제곱인줄알고 구현하였는데 홀수개도 다 해야해서 구현하는데 고민이 많았는데 해결방법은 홀수자리수의 숫자는 *10을 하여서 짝수개로 바꿔서 계산후 return할 때 /100을 해줬더니 해결되었다.

```
while(num>0)
 {
  num = num/10;
  Numcount++;
 }//자리수를 알아내는 과정
if(Numcount%2==1) {
    Numcount++;
    a=a*10;
    b=b*10;
    flag=1;
}//자리수가 홀수개일때 계산이 잘될수있도록 임시로 10을 곱해서 실행한다.
```

그리고 int형으로 입력을 받으니 계산중 자리수를 곱할때나 계산할 때 오버플로우가 수시로 발생하여서 계산에 쓰이는 수는 전부다 Long형으로 바꿔주었더니 실행이 잘 되었다.

```
if(Numcount==2) {//Numcount가 2이면 나눈 반반은 현재 한자리숫자이다 즉 기본 base인 상태이다.
    BigInteger x1 =BigInteger.valueOf(big1);
    BigInteger y1 =BigInteger.valueOf(big2);
    BigInteger x2 =BigInteger.valueOf(small1);
    BigInteger y2 =BigInteger.valueOf(small2);
    z0=x2.multiply(y2);
    z2=x1.multiply(y1);
    z1=x2.add(x1).multiply(y2.add(y1)).subtract(z2).subtract(z0);
    BigInteger digit =BigInteger.valueOf(10);
    digit =digit.pow(Numcount);//10의 제곱을 하기위해 곱할수를 만드는 과정이다.
    result = z2.multiply(digit);
    digit =BigInteger.valueOf(10);
    digit = digit.pow(Numcount/2);
    result = result.add(z1.multiply(digit));
    result = result.add(z0);//만약 4자리수라면 반반
    if(flag==1) {
           result=result.divide(BigInteger.valueOf(100));
    }//자리수가 홀수라면 결과값에서 나누기 100을 해준다.
   return result;}
else {//만약 2자리가 아니라면 더나눠서 karatsub을 해준다.
```

PPT자료를 보고 같이 구현하였다 pow함수를 처음에는 int로 하였더니 10^10일 때 에러가 나서 BigInterger로 바꿔서 구현하였다.

flag는 자리수가 홀수인지 짝수인지 판단한다.

2. 실행 결과

```
<terminated> Karatsuba [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:33:44)
input data : 12345678 65432187
Result : 807804711537786
```

```
<terminated> Karatsuba [Java Application] C:₩Program Files₩Java₩jre1.8.0_144₩bin₩javaw.exe (2017. 10. 18. 오후 8:34:44)
input data : 123456789123 123456789123
Result : 152415787805608911091 29
```

3. 느낀 점

CountInversion은 mergesort에서 count만 하면되서 쉽게 구현하였다.
하지만 karatsuba 알고리즘은 홀수개일 경우와 OverFlow등의 예외가 많아서 처리하는라 힘들었다.
최대한 예외없이 계산이 잘되도록 구현하였다.
처음에는 단순하게 짝수개인줄알아서 쉽게알고 구현하였다가 수정하는라 시간을 더 쓴거같다.