

알고리즘

과제번호	07주차
날 짜	2018.10.25
학 번	201302395
이 름	류경빈

과제 1

Inversion을 count하는 프로그램을 만들어라

```

public static void merge(int arr[], int l, int mid, int r) {
    int i = l;
    int j = mid+1; // merge right 후반부 첫 번째 원소
    int k = l; // temp의 시작점을 뜻
    int temp[] = new int[arr.length];

    while(i<=mid && j<=r) { // merge right와 left 배열이 끝날 때 까지를 의미 i값이 mid 값 즉 m
        if(arr[i] < arr[j]) { // merge left와 right 값을 비교해서 더 작은 값을 가지는 배열의 원소
            temp[k++] = arr[i++];
        } else {
            System.out.println "[" + arr[i] + ", " + arr[j] + "]";
            temp[k++] = arr[j++];
            inversionCount += mid+1-i;
        }
    }
    // merge left, right 둘 중 하나의 배열의 원소를 모두 탐색했을 경우 넘어간다.
    while(i<=mid) // merge left 배열의 원소들이 남았을 경우
        temp[k++] = arr[i++];
    while(j<=r) // merge right 배열의 원소들이 남았을 경우
        temp[k++] = arr[j++];
    for(int index=l; index<k; index++) {
        arr[index] = temp[index]; // temp 배열을 arr에 복사중..
    }
}

```

- 기존 merge sort를 활용한 divide 해주고 inversion 해주는 과정을 진행했다.
- inversionCount를 inversion 진행하는 곳에 count를 진행해준다.
- inversion이 진행 되는 곳에 해당하는 inversion 쌍을 출력해준다.(left 배열의 length인 mid+1까지)

구현 결과 화면

```

TestCountInversion x
/Library/Java/JavaVirtualMach
[5, 3]
[16, 8]
[15, 8]
[5, 4]
[6, 4]
[14, 10]
[26, 10]
[27, 10]
[26, 14]
[27, 14]
[9, 7]
[10, 7]
[14, 7]
[26, 7]
[27, 7]
[10, 9]
[14, 9]
[26, 9]
[27, 9]
[14, 11]
[26, 11]
[27, 11]
[8, 7]
[15, 7]
[16, 7]
[15, 9]
[16, 9]
[15, 10]
[16, 10]
[15, 11]
[16, 11]
[15, 14]
[16, 14]
[29, 12]
[30, 12]
[29, 24]
[30, 24]
[25, 13]
[25, 17]
[25, 20]
[24, 13]
[29, 13]
[30, 13]
[24, 17]
[29, 17]
[30, 17]
[24, 20]
[29, 20]
[30, 20]

```

```

TestCountInversion x
[29, 25]
[30, 25]
[19, 18]
[20, 18]
[24, 18]
[25, 18]
[29, 18]
[30, 18]
[20, 19]
[24, 19]
[25, 19]
[29, 19]
[30, 19]
[24, 21]
[25, 21]
[29, 21]
[30, 21]
[24, 22]
[25, 22]
[29, 22]
[30, 22]
[24, 23]
[25, 23]
[29, 23]
[30, 23]
[14, 12]
[15, 12]
[16, 12]
[26, 12]
[27, 12]
[28, 12]
[14, 13]
[15, 13]
[16, 13]
[26, 13]
[27, 13]
[28, 13]
[26, 17]
[27, 17]
[28, 17]
[26, 18]
[27, 18]
[28, 18]
[26, 19]
[27, 19]
[28, 19]
[26, 20]
[27, 20]
[28, 20]
[26, 21]

```

```

[27, 21]
[28, 21]
[26, 22]
[27, 22]
[28, 22]
[26, 23]
[27, 23]
[28, 23]
[26, 24]
[27, 24]
[28, 24]
[26, 25]
[27, 25]
[28, 25]
Inversion Count : 113

```

```

public int howLongNumber (long number){
    return (int)((Math.log10(number)+1));
}

public int midNumber (long length){
    return (int)(Math.pow(10,(length/2)));
}

public long divisionNumber (long number, int divider){
    return number/divider;
}

public long remainderNumber (long number, int divider){
    return number%divider;
}

```

```

public long karatsubaMult (long numX, long numY){

    int x = howLongNumber(numX);
    int y = howLongNumber(numY);
    int d = x < y ? x : y;

    if (numX < numY){
        return karatsubaMult(numY, numX);
    }
    else if (x == 0 || y == 0){
        return 0;
    }
    else if (d <= THRESHOLD){
        return numX*numY;
    }

    int mid = midNumber(d);

    long x1 = divisionNumber(numX, mid);
    long x0 = remainderNumber(numX, mid);

    long y1 = divisionNumber(numY, mid);
    long y0 = remainderNumber(numY, mid);

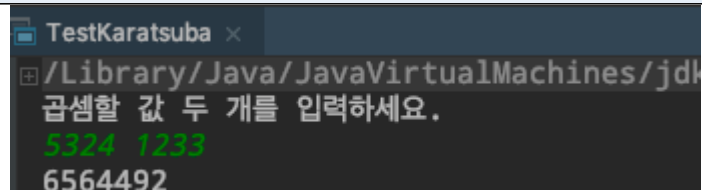
    long z2 = this.karatsubaMult(x1, y1);
    long z0 = this.karatsubaMult(x0, y0);
    long z1 = this.karatsubaMult((x0+x1), (y0+y1))-z2-z0;

    long result = (long) (z2*(Math.pow(mid,2))+z1*mid+z0);
    return result;
}

```

- 입력 된 두 값을 Karatsuba 알고리즘을 통해 divide를 진행하며 한 자리 값이 될 때까지 구해준다.
- Karatsuba 알고리즘을 재귀적으로 계속해서 최종 곱셈 연산 값을 도출한다.
- 최대 자리수는 3자리 THRESHOLD로 진행을 해주어 효율을 높여줌

구현 결과 화면



```

TestKaratsuba x
/Library/Java/JavaVirtualMachines/jdk
곱셈할 값 두 개를 입력하세요.
5324 1233
6564492

```