



# 4주차 퀵정렬, 합병정렬

[통계](#) [수정](#) [삭제](#)

iming03 · 방금 전 · 비공개

0

[알고리즘](#) [퀵정렬](#) [합병정렬](#)

## 알고리즘 스터디

[▼ 목록 보기](#)

3/3



### 문제

2차원 평면 위의 점  $N$ 개가 주어진다. 좌표를  $x$ 좌표가 증가하는 순으로,  $x$ 좌표가 같으면  $y$ 좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

### 입력

첫째 줄에 점의 개수  $N$  ( $1 \leq N \leq 100,000$ )이 주어진다. 둘째 줄부터  $N$ 개의 줄에는  $i$ 번점의 위치  $x_i$ 와  $y_i$ 가 주어진다. ( $-100,000 \leq x_i, y_i \leq 100,000$ ) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

### 출력

첫째 줄부터  $N$ 개의 줄에 점을 정렬한 결과를 출력한다.

#### 예제 입력 1 복사

```
5
3 4
1 1
1 -1
2 2
3 3
```

#### 예제 출력 1 복사

```
1 -1
1 1
2 2
3 3
3 4
```

## 백준 문제 11650번

문제

2차원 평면 위의 점  $N$ 개가 주어진다. 좌표를  $x$ 좌표가 증가하는 순으로,  $x$ 좌표가 같으면  $y$ 좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 점의 개수  $N$  ( $1 \leq N \leq 100,000$ )이 주어진다. 둘째 줄부터  $N$ 개의 줄에는  $i$ 번점의 위치  $x_i$ 와  $y_i$ 가 주어진다. ( $-100,000 \leq x_i, y_i \leq 100,000$ ) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

출력

첫째 줄부터  $N$ 개의 줄에 점을 정렬한 결과를 출력한다.

예제 입력 1 복사

```
5
3 4
1 1
1 -1
2 2
3 3
```

예제 출력 1 복사

```
1 -1
1 1
2 2
3 3
3 4
```

이 문제는 단순히 c언어로 간단하게 풀 수 있을 것 같은데 그러면 시간초과가 생기겠지....

이번 주차의 주제인 합병정렬과 퀵정렬에 대해 먼저 알아보자!

## 퀵정렬(quick-sort): 분할통치법에 기초한 정렬 알고리즘

```

Alg. quickSort(L)
  input list  $L$  with  $n$  elements
  output sorted list  $L$ 

1. if ( $L.size()$  > 1)
     $k \leftarrow$  a position in  $L$ 
     $LT, EQ, GT \leftarrow partition(L, k)$ 
    quickSort(LT)
    quickSort(GT)
     $L \leftarrow merge(LT, EQ, GT)$ 
2. return
  
```

### 최악실행시간

quick-sort의 최악은 기준원소가 항상 유일한 최소이거나 최대 원소일 경우

실행시간은  $n+(n-1)+\dots+2+1$ 에 비례

따라서, quick-sort의 최악 실행시간:  $O(n^2)$

## 기대실행시간

크기  $s$ 의 리스트에 대한 quick-sort의 재귀 호출을 고려하면,

좋은 호출: LT와 GT의 크기가 모두  $(3/4)s$ 보다 작다.

나쁜 호출: LT와 GT의 가운데 하나의 크기가  $(3/4)s$ 보다 크다.

## 제자리 퀵 정렬

quick-sort를 제자리에서 수행되도록 구현 가능

분할단계에서, 입력 리스트의 원소들을 재배치하기 위해 대체(replace) 작업을 사용

```

Alg inPlaceQuickSort( $L, l, r$ )
  input list  $L$ , position  $l, r$ 
  output list  $L$  with elements of
    position from  $l$  to  $r$  rearranged in
    increasing order

  1. if ( $l \geq r$ )
    return
  2.  $k \leftarrow$  a position between  $l$  and  $r$ 
  3.  $a, b \leftarrow$  inPlacePartition( $L, l, r, k$ )
  4. inPlaceQuickSort( $L, l, a - 1$ )
  5. inPlaceQuickSort( $L, b + 1, r$ )
  
```

|   |  |
|---|--|
| <pre> <b>Alg <u>inPlacePartition</u>(<math>A, l, r, k</math>)</b>   <b>input</b> array <math>A[l..r]</math> of <i>distinct</i>     elements, index <math>l, r, k</math>   <b>output</b> final index of the pivot     resulting from partitioning     <math>A[l..r]</math> into <math>LT</math>, pivot, <math>GT</math>    1. <math>p \leftarrow A[k]</math>           {pivot}   2. <math>A[k] \leftrightarrow A[r]</math>   {hide pivot}   3. <math>i \leftarrow l</math>   4. <math>j \leftarrow r - 1</math>         </pre> | <pre>   5. <b>while</b> (<math>i \leq j</math>)     <b>while</b> (<math>i \leq j \ \&amp; \ A[i] \leq p</math>)       <math>i \leftarrow i + 1</math>     <b>while</b> (<math>j \geq i \ \&amp; \ A[j] \geq p</math>)       <math>j \leftarrow j - 1</math>     <b>if</b> (<math>i &lt; j</math>)       <math>A[i] \leftrightarrow A[j]</math>   6. <math>A[i] \leftrightarrow A[r]</math>   {replace pivot}   7. <b>return</b> <math>i</math>           {index of pivot}         </pre> |
|---|--|

## 합병 정렬과 퀵 정렬 비교

|           | 합병 정렬                  | 퀵 정렬                                    |
|-----------|------------------------|---|
| 기법        | 분할통치법                  | 분할통치법                                   |
| 실행시간      | $O(n \log n)$ 최악실행시간   | $O(n^2)$ 최악실행시간<br>$O(n \log n)$ 기대실행시간 |
| 분할 vs. 결합 | 분할은 쉽고,<br>합병은 어렵다     | 분할은 어렵고,<br>합병은 쉽다                      |
| 제자리 구현    | 제자리 합병이 어렵다            | 제자리 분할이 쉽다                              |
| 실제 작업 순서  | 작은 것에서 점점 큰<br>부문제로 진행 | 큰 것에서 점점 작은<br>부문제로 진행                  |

→ 이 비교표를 보고 퀵 정렬, 제자리 구현으로 이 문제를 풀어겠다고 생각했당!

## 코드

```
#include <stdio.h>
#include <stdlib.h>
#pragma warning(disable:4996)

typedef struct Point {
    int x;
    int y;
} Point;

int compare(const void *a, const void *b) {
    Point *pointA = (Point *)a;
    Point *pointB = (Point *)b;

    if (pointA->x < pointB->x || (pointA->x == pointB->x && pointA->y < pointB->y)) {
        return -1;
    } else {
        return 1;
    }
}

void inplaceQuickSort(Point arr[], int l, int r) {
    if (l >= r) {
        return;
    }

    int k = (l + r) / 2;

    int a, b;
    a = b = inplacePartition(arr, l, r, k);

    inplaceQuickSort(arr, l, a - 1);
    inplaceQuickSort(arr, b + 1, r);
}
```

```

int inPlacePartition(Point arr[], int l, int r, int k) {
    Point p = arr[k];
    arr[k] = arr[r];
    arr[r] = p;

    int i = l;
    int j = r - 1;

    while (1) {
        while (i <= j && compare(&arr[i], &p) <= 0) {
            i++;
        }
        while (i <= j && compare(&arr[j], &p) >= 0) {
            j--;
        }
        if (i < j) {
            Point temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        } else {
            break;
        }
    }
    Point temp = arr[i];
    arr[i] = arr[r];
    arr[r] = temp;

    return i;
}

int main() {
    int N;
    scanf("%d", &N);

    Point *points = (Point *)malloc(N * sizeof(Point));

    for (int i = 0; i < N; i++) {
        scanf("%d %d", &points[i].x, &points[i].y);
    }

    inPlaceQuickSort(points, 0, N - 1);

    for (int i = 0; i < N; i++) {
        printf("%d %d\n", points[i].x, points[i].y);
    }

    free(points);
    return 0;
}

```

| 아이디     | 문제    | 결과      | 메모리     | 시간    | 언어       | 코드 길이  |
|---------|-------|---------|---------|-------|----------|--------|
| iming03 | 11650 | 맞았습니다!! | 1900 KB | 64 ms | C99 / 수정 | 2073 B |

이번 문제는 그렇게 어렵지 않았던 것 같다! 알고리즘대로 해서 술술 풀었다.



**강민돌**

민돌이의 공부



이전 포스트

2주차 우선순위 큐

**0개의 댓글**

댓글을 작성하세요

댓글 작성



Powered by  
**Stellate**