2023_Autumn_Algorithm_Study_4주차 과제 기록

통계 수정 삭제

jaesung2da · 방금 전 · 비공개

0

> 3주차 과제는
> 11650번: 좌표 정렬하기

좌표 정렬하기

☆

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	123866	59093	45895	47.915%

문제

2차원 평면 위의 점 N개가 주어진다. 좌표를 x좌표가 증가하는 순으로, x좌표가 같으면 y좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 점의 개수 N (1 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N개의 줄에는 i번점의 위치 x_i와 y_i가 주어진다. (-100,000 ≤ x_i, y_i ≤ 100,000) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

출력

첫째 줄부터 N개의 줄에 점을 정렬한 결과를 출력한다.

예제 입력 1 복사

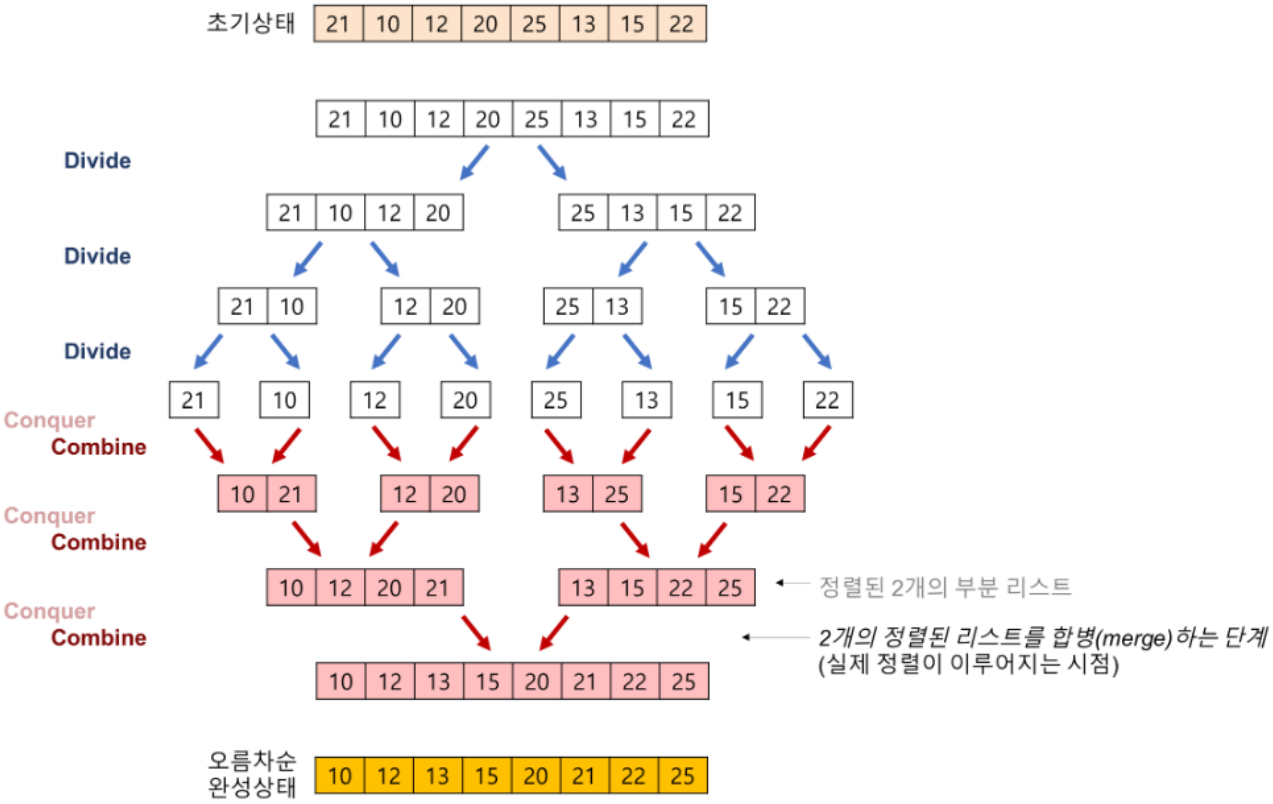
```
5
3 4
1 1
1 -1
2 2
3 3
```

예제 출력 1 복사

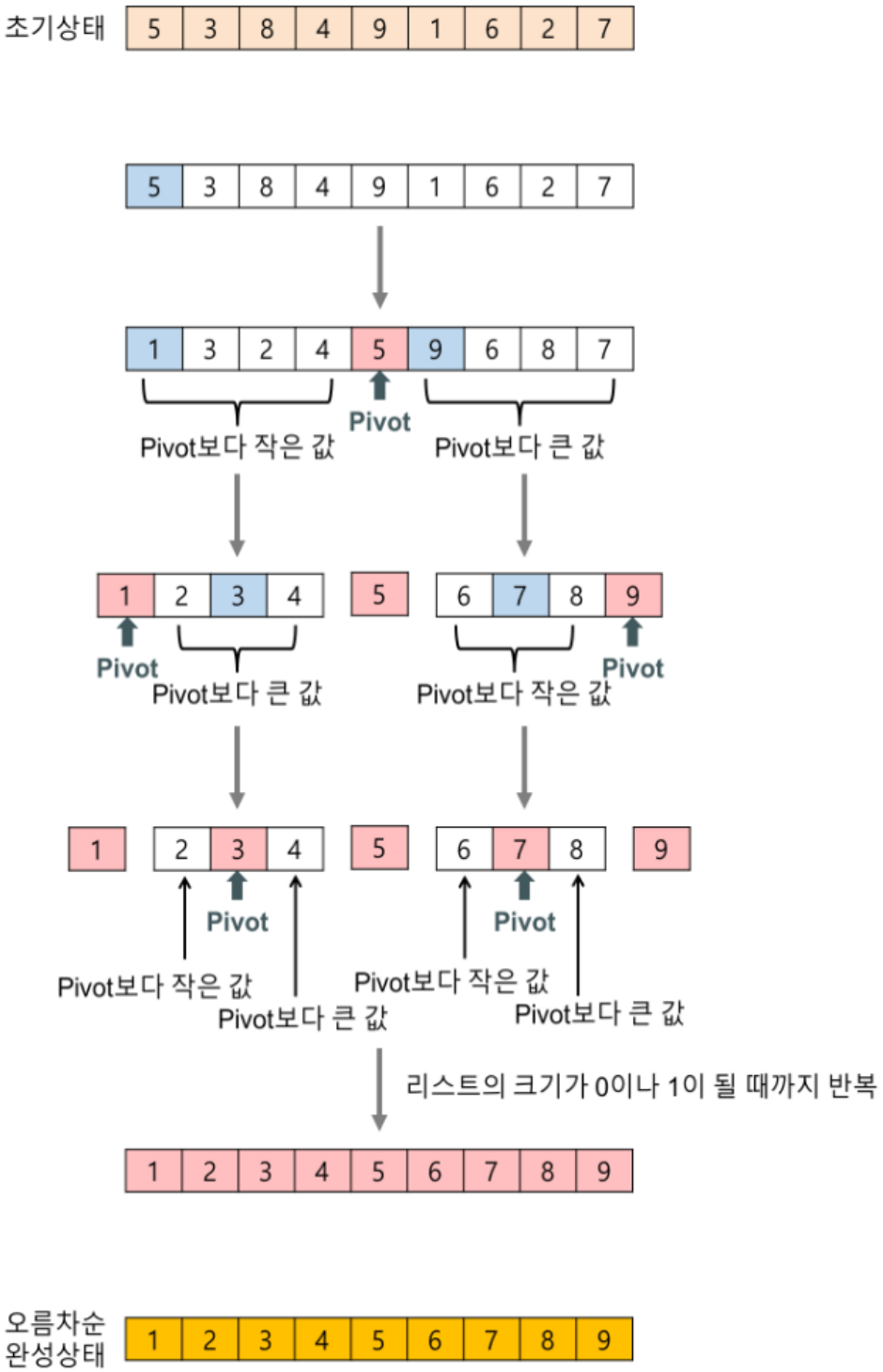
```
1 -1
1 1
2 2
3 3
3 4
```

개념공부

합병정렬 :
분할(Divide): 입력 배열을 같은 크기의 2개의 부분 배열로 분할한다.
정복(Conquer): 부분 배열을 정렬한다. 부분 배열의 크기가 충분히 작지 않으면 순환 호출 을 이용하여 다시 분할 정복 방법을 적용한다.
결합(Combine): 정렬된 부분 배열들을 하나의 배열에 합병한다.
사실 이런 내용보다는 아래 그림으로 이해하는게 더 나은 것 같다.



퀵 정렬 :분할(Divide): 입력 배열을 피벗을 기준으로 비균등하게 2개의 부분 배열(피벗을 중심으로 왼쪽: 피벗보다 작은 요소들, 오른쪽: 피벗보다 큰 요소들)로 분할한다.
정복(Conquer): 부분 배열을 정렬한다. 부분 배열의 크기가 충분히 작지 않으면 순환 호출 을 이용하여 다시 분할 정복 방법을 적용한다.
결합(Combine): 정렬된 부분 배열들을 하나의 배열에 합병한다.
순환 호출이 한번 진행될 때마다 최소한 하나의 원소(피벗)는 최종적으로 위치가 정해지므로, 이 알고리즘은 반드시 끝난다는 것을 보장할 수 있다.
얘도 그림으로 보는게 더 이해하기 좋았다.



참고한 블로그 주소 합병정렬, 퀵 정렬

문제 풀기 전 전략

아직 합병정렬이나 퀵 정렬에 대해서 배우지 않아서 사실 문제를 읽었을 때 그냥 이중 for문에 조건만 써서 문제를 풀고 싶다고 생각했지만 그래도 이왕 공부하는거 메모리도 줄이고 시간도도 줄이려면 어떻게 풀까 고민을 했는데 **아무래도 퀵 정렬이 합병 정렬보다 빠르니까 퀵 정렬을**

> **쓰는 것이 좋다고 생각했다.** 그리고 사실 그것보다 퀵 정렬 함수 qsort가 <stdlib.h>에 포함되어있다더라 ㅎㅎ 있는건 써야지 뭐 어떡해...

문제점 그리고 해결

> 아 내가 멍청했다 qsort를 라이브러리에서 바로 쓸려했더니 비교함수를 설정해야 한다더라 그건 또 뭔데... 별 수 없이 바로 구글에 c언어 qsort compare 검색 찾았다^__^

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

int compare(const void *first, const void *second);

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

    int *x, *y = NULL;

    x = (int*)malloc(sizeof(int)*N);
    y = (int*)malloc(sizeof(int)*N);

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

    qsort(x, N, sizeof(int), compare);

    for


    return 0;
}

int compare(const void *first, const void *second)
{
    // ** 0. Compare function의 리턴값은 기본적으로 오름차순으로 정렬
    // ** 1. 리턴 값이 0보다 작을 경우 first가 second보다 작다.
    // ** 2. 리턴 값이 0일경우 first와 second의 값이 같다.
    //    ** 3. 리턴 값이 0보다 클 경우first가 second보다 크다.

    // 0. first가 더 크면 참
    if (*(int*)first > *(int*)second)
    {
        return 1; // 1을 반환 : first와 second swap(교환)
    }
    // 1. second가 더 크면 참
    else if (*(int*)first < *(int*)second)
    {
        return -1; // -1을 반환 : 패스
    }
    // 2. 그외에는 0을 반환
```

```
    else
    {
        return 0; // 0을 반환 : 패스
    }
}
```

그리고 다시 문제

> 위 코드가 그 문제에 코드 아 생각해보니 x를 정렬하고 y는 x의 index를 똑같이 따라서 정렬시킨 후 x 값이 같을 경우 y값끼리 비교할려했는데 qsort를 써버리니 x의 index변화를 실시간으로 알수가 없고 과정도 알수가 없더라... 아 망했다 어떡하지 생각하다. 구글링을 해봤더니 새롭게 구조체를 만들어서 문제를 풀더라... 사실 내가 풀었다고 하긴 그렇고 이미 너무 깔끔한 정답코드를 봐버려서 최대한 코드를 안보고 부분부분 개념위주로 코드를 공부해서 써보았다.

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
67446152	tlawotjd	11650	맞았습니다!!	2680 KB	64 ms	C99 / 수정	1006 B	5초 전

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

//좌표-> coordinate라는 구조체를 설정해준다. x좌표, y좌표
typedef struct {
    int x;
    int y;
} coordinate;

// 그전에 공부했던 비교함수를 배열에서 int형 변환이 아닌 설정해둔 구조체 coordinate형으로 비교하고 변환해준다.
int compare(const void *a, const void *b) {
    coordinate *p1 = (coordinate *)a;
    coordinate *p2 = (coordinate *)b;

    //x좌표가 다르면 x로 정렬
    if (p1->x != p2->x) {
        return p1->x - p2->x;
    }

    //x좌표가 같으면 y로 정렬
    return p1->y - p2->y;
}

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

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

    for (int i = 0; i < N; i++) {
```

```
        scanf("%d %d", &points[i].x, &points[i].y);
    }

    // 퀵 정렬을 사용하여 좌표 정렬
    qsort(points, N, sizeof(coordinate), compare);


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

    free(points);

    return 0;
}
```

총평

> 이번에 문제를 풀면서 **합병정렬과 퀵 정렬**에 대해 알게 되었긴 하지만 사실 코드나 개념을 더 많이 고민한 것은 **비교함수 compare함수**를 어떻게 설정하느냐에 대해 알게 되었다. 그리고 이번 문제의 좌표처럼 두가지를 동시에 비교해야하고 비교순위가 있을 때는 **새로운 구조체를 형성**하는 방법이 있다는 것을 알게되었다. 사실 머리 깨지면서 비효율적인 코드도 만들어보고 해야하는데 그렇게 하다가 구글링을하는 과정에서 판도라의 상자처럼 저항없이 봐버렸다.. 그래도 새로운 개념과 접근법을 알게 된 것 같아서 나쁘지 않은 선택..?이었다.

심재성

이전 포스트
2023_Autumn_Algorithm_Study_3주차 과제 기록

0개의 댓글

댓글을 작성하세요

댓글 작성