정렬

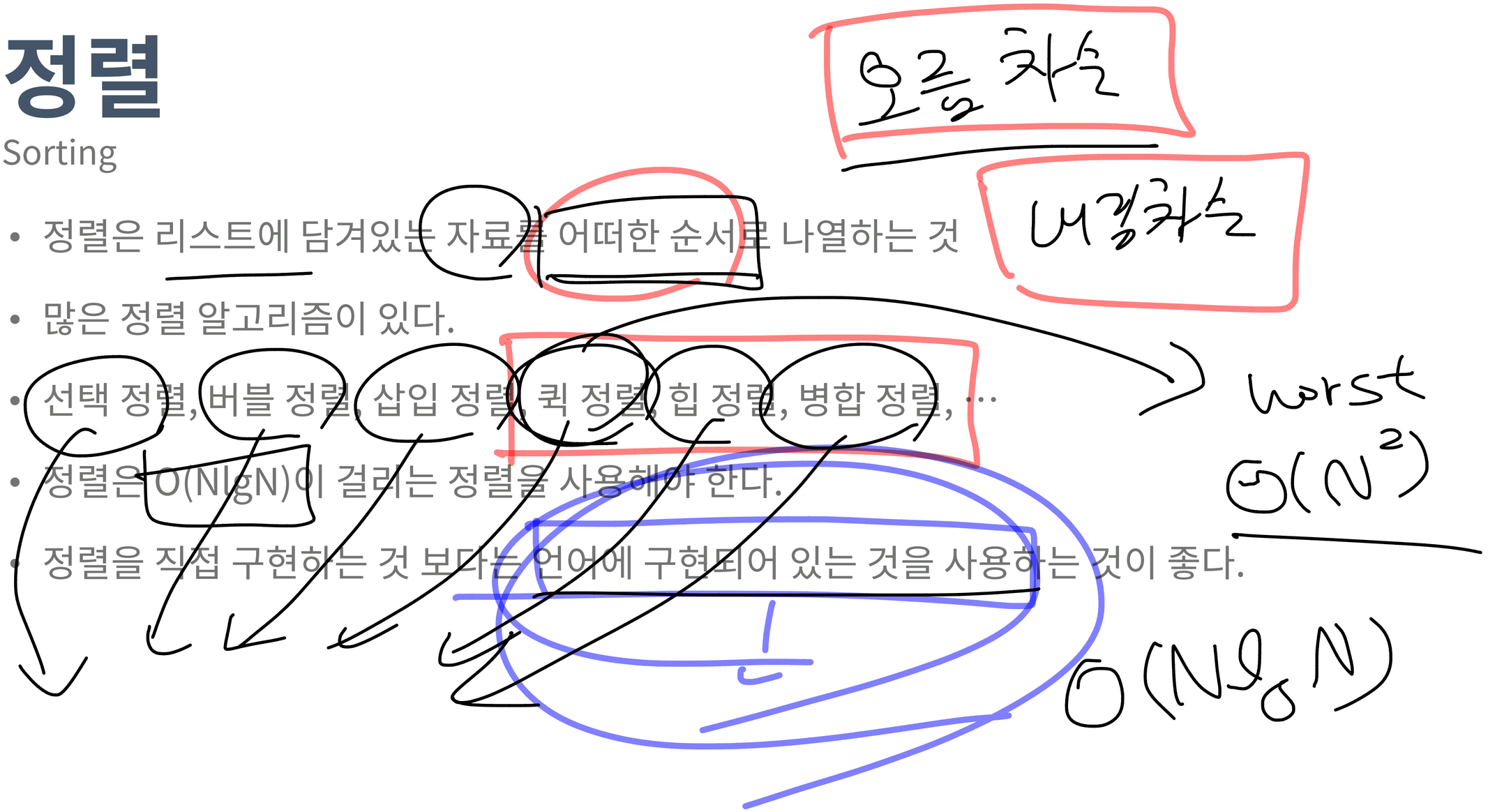
최백준 choi@startlink.io

정렬

정렬

Sorting

- 정렬은 리스트에 담겨있는 자료를 어떠한 순서로 나열하는 것
- 많은 정렬 알고리즘이 있다.
- 선택 정렬, 버블 정렬, 삽입 정렬, 퀵 정렬, 힙 정렬, 병합 정렬, …
- 정렬은 O(NlgN)이 걸리는 정렬을 사용해야 한다.
- 정렬을 직접 구현하는 것 보다는 언어에 구현되어 있는 것을 사용하는 것이 좋다.



C++ 정렬

Sorting

- C++은 algorithm 헤더 파일에 있는 std::sort를 사용한다.
- sort(first, last): [first, last)를 오름차순으로 정렬한다.

```int a[100];
sort(a, a+100);
vector<int> v;
sort(v.begin(), v.end());
```

Java 정렬

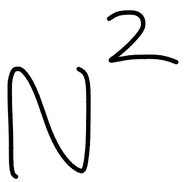
Sorting

- Java는 Arrays.sort나 Collections.sort를 사용한다.
- Arrays.sort(a): a를 오름차순으로 정렬한다.
- Arrays.sort(a, from, to): a의 [from,to)를 오름차순으로 정렬한다.
- Collections.sort(v): v를 오름차순으로 정렬한다.

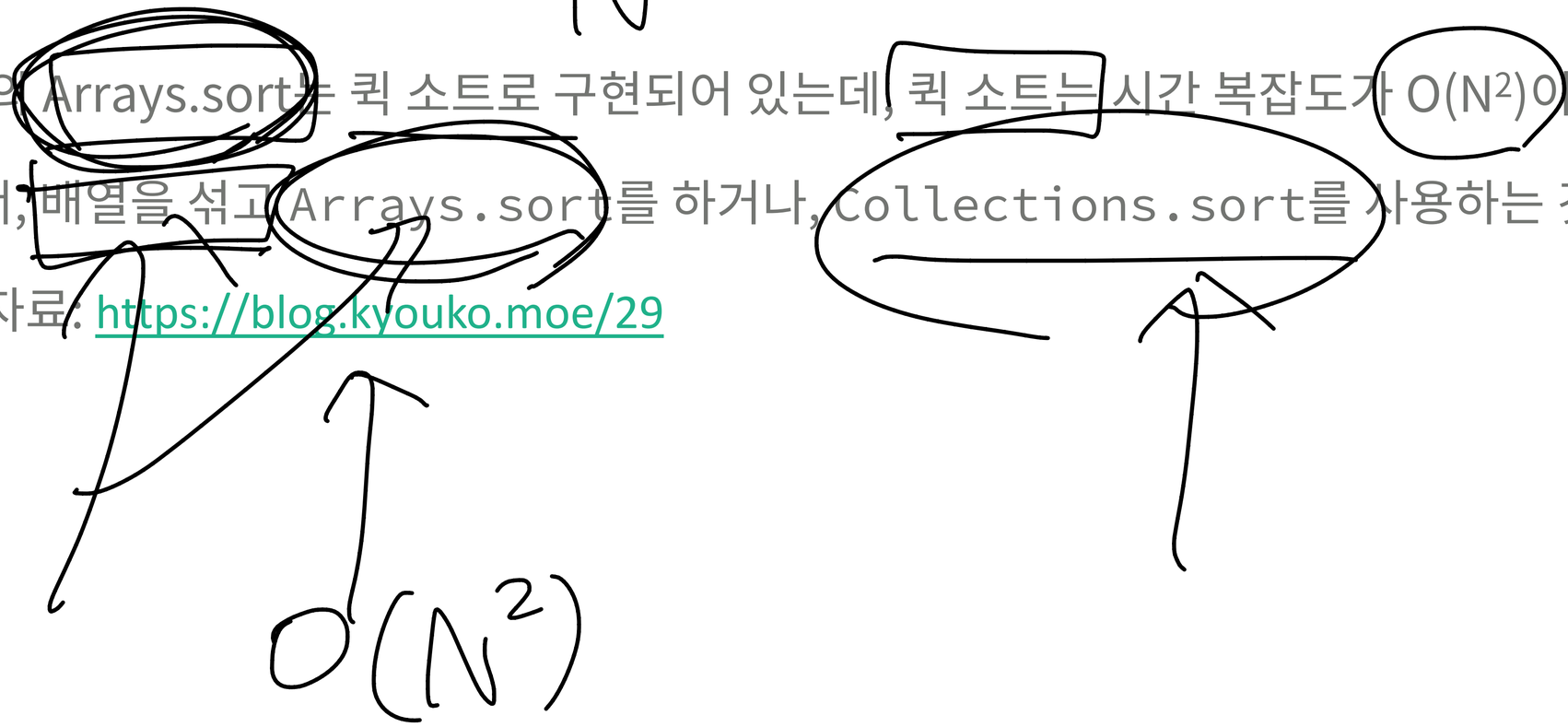
```
int[] a = new int[n];
Arrays.sort(a);
ArrayList<Integer> v = new ArrayList<>();
Collections.sort(v);
```

Java 정렬

Sorting



- Java의 Arrays.sort는 퀵 소트로 구현되어 있는데, 퀵 소트는 시간 복잡도가 O(N²)이다.
- 따라서, 배열을 섞고 Arrays.sort를 하거나, Collections.sort를 사용하는 것이 좋다.
- 관련 자료: https://blog.kyouko.moe/29



Python 정렬

Sorting

- Python 리스트의 sort() 메소드를 사용한다.
- `l.sort()`: l을 오름차순으로 정렬한다.

sorted(l)

```
l.sort()
```

수 정렬하기 2

https://www.acmicpc.net/problem/2751

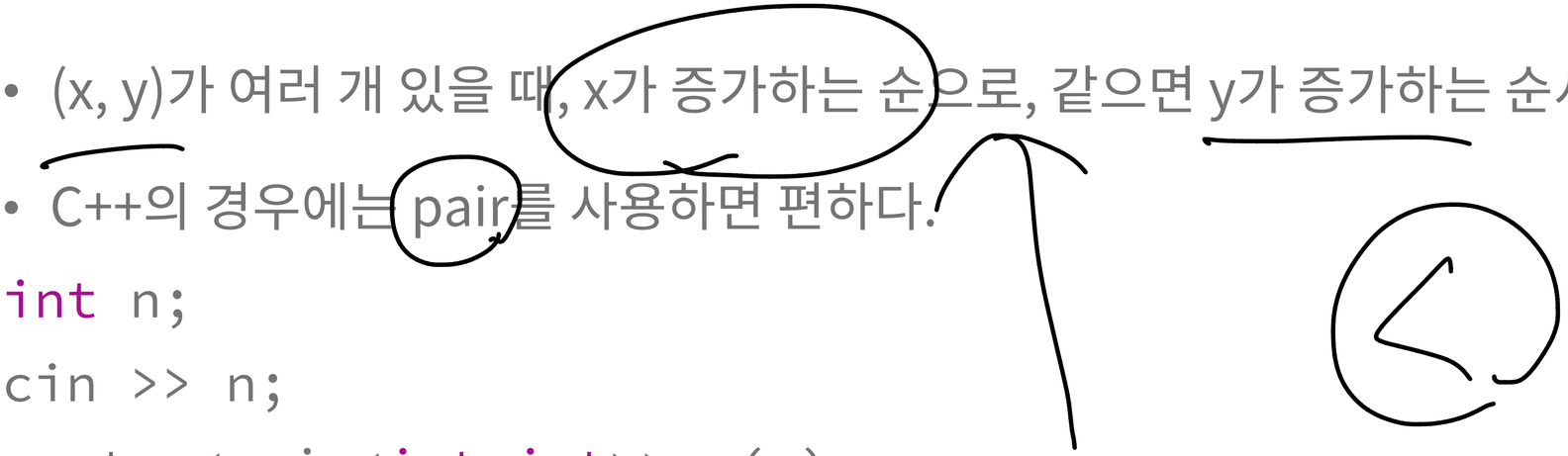
N≤1 백만 NLN

- N개의 수를 정렬하는 문제
- Quick Sort 소스: http://codeplus.codes/a6f2639d87b84713ad69d79186fc89eb
- Merge Sort 소스: http://codeplus.codes/16c69e90effb4eea9841ac0c8ccabb2f
- 라이브러리 사용: http://codeplus.codes/22cd894adfb848de9241c86ffed76395

좌표 정렬하기

https://www.acmicpc.net/problem/11650

- (x, y)가 여러 개 있을 때, x가 증가하는 순으로, 같으면 y가 증가하는 순서로 정렬하는 문제
- C++의 경우에는 pair를 사용하면 편하다.



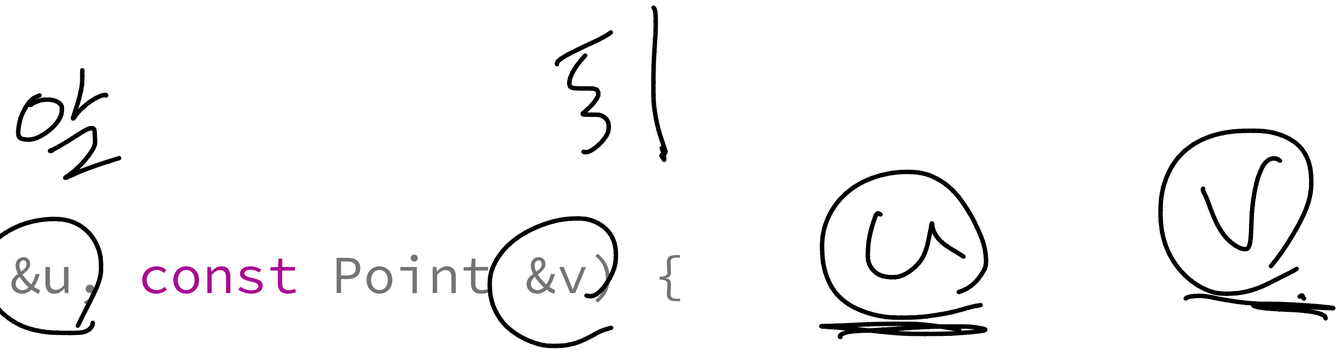

```cpp
int n;
cin >> n;
vector<pair<int,int>> a(n);
for (int i=0; i<n; i++) {
    cin >> a[i].first >> a[i].second;
}
sort(a.begin(), a.end());
for (int i=0; i<n; i++) {
    cout << a[i].first << ' ' << a[i].second << '\n';
}
```

좌표 정렬하기

https://www.acmicpc.net/problem/11650

- 직접 struct를 구현하는 경우에는 비교 함수를 만들어 줘야 한다.

```
struct Point {
    int x, y;
};
bool cmp(const Point &u, const Point &v) {
    if (u.x < v.x) {
        return true;
    } else if (u.x == v.x) {
        return u.y < v.y;
    } else {
        return false;
    }
}
```



좌표 정렬하기

https://www.acmicpc.net/problem/11650

- cmp 함수는 u가 v의 앞에 오는 것이면 true, 아니면 false 이다.
- const와 &는 꼭 있어야 한다.

```
bool cmp(const Point &u, const Point &v) {
    if (u.x < v.x) {
        return true;
    } else if (u.x == v.x) {
        return u.y < v.y;
    } else {
        return false;
    }
}
```

좌표 정렬하기

https://www.acmicpc.net/problem/11650

- 비교 함수를 만드는 경우에는 3번째 인자로 함수 이름을 넘겨줘야 한다.

```
bool cmp(const Point &u, const Point &v) {
    if (u.x < v.x) {
        return true;
    } else if (u.x == v.x) {
        return u.y < v.y;
    } else {
        return false;
    }
}
sort(a.begin(), a.end(), cmp);
```

좌표 정렬하기

https://www.acmicpc.net/problem/11650

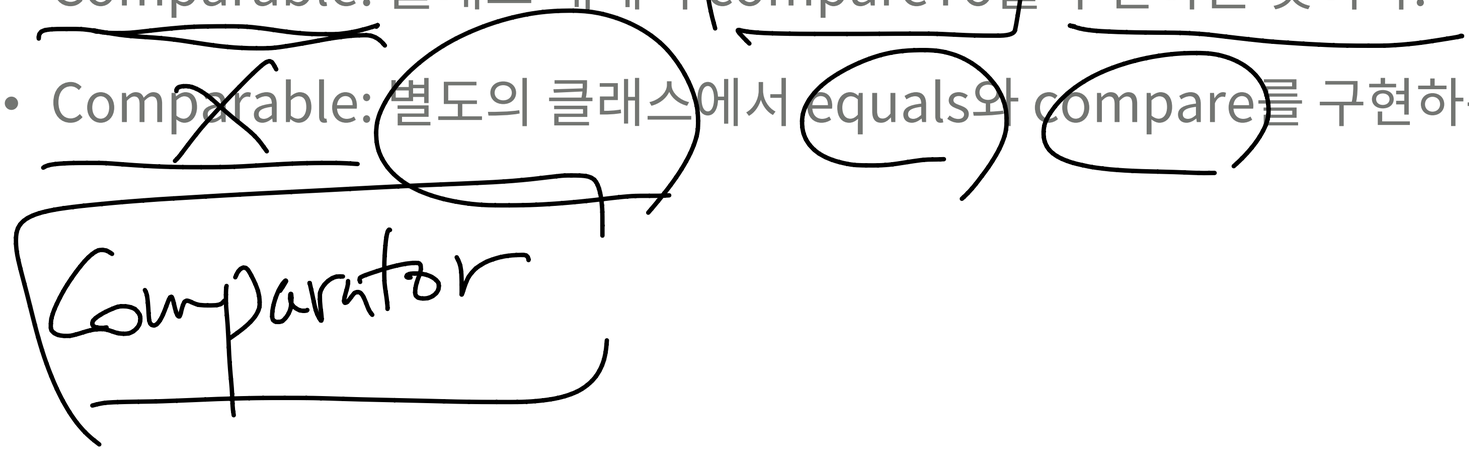
- < 연산자를 오버로딩 할 수도 있다. 이 경우에는 3번째 인자가 필요 없다.

```
struct Point {
    int x, y;
    bool operator < (const Point &v) const {
        if (x < v.x) {
            return true;
        } else if (x == v.x) {
            return y < v.y;
        } else {
            return false;
        }
    }
}
```

좌표 정렬하기

https://www.acmicpc.net/problem/11650

- Java의 경우에는 두 가지 방법이 있다.
- Comparable: 클래스 내에서 compareTo를 구현하는 것이다.
- Comparable: 별도의 클래스에서 equals와 compare를 구현하는 것이다.



Comparable

java.lang.Comparable<T>

```
public int compareTo(Point that) {
    if (this.x < that.x) {
        return -1;
    } else if (this.x == that.x) {
        if (this.y < that.y) {
            return -1;
        } else if (this.y == that.y) {
            return 0;
        } else {
            return 1;
        }
    } else {
        return 1;
    }
}
```

this that

=

>

Comparable

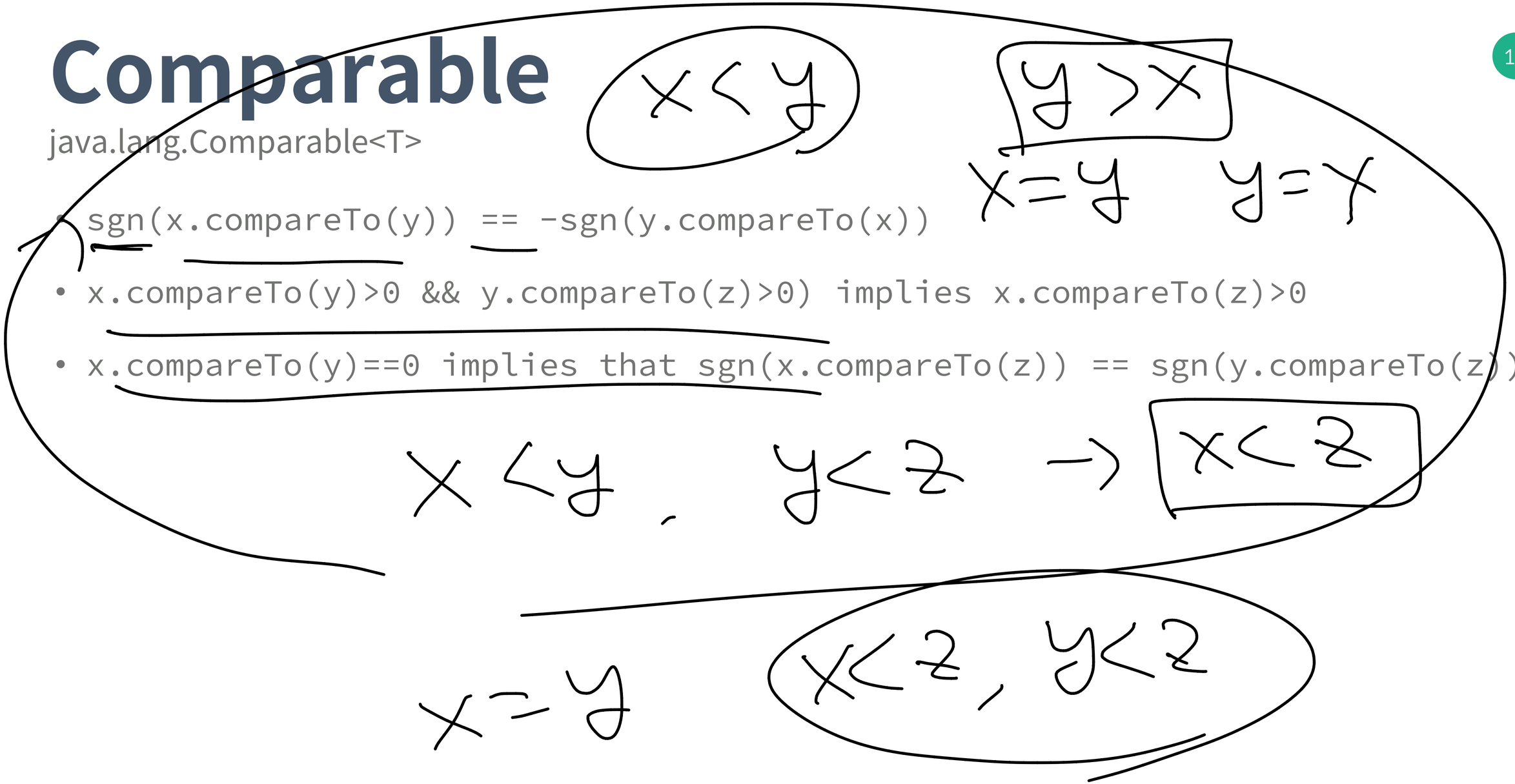
java.lang.Comparable<T>

- this < that 이면 음수
- this == that 이면 0
- this > that 이면 양수

Comparable

java.lang.Comparable<T>

- sgn(x.compareTo(y)) == -sgn(y.compareTo(x))
- x.compareTo(y)>0 && y.compareTo(z)>0) implies x.compareTo(z)>0
- x.compareTo(y)==0 implies that sgn(x.compareTo(z)) == sgn(y.compareTo(z))



Comparator

java.util.Comparator<T>

```
class PointComparator implements Comparator<Point> {
    public int compare(Point u, Point v) {
        if (u.x < v.x) {
            return -1;
        } else if (u.x == v.x) {
            if (u.y < v.y) { return -1; }
            else if (u.y == v.y) { return 0; }
            else { return 1; }
        } else {
            return 1;
        }
    }
}
Arrays.sort(a, new PointComparator());
```

Comparator

java.util.Comparator<T>

```
Arrays.sort(a, new Comparator<Point>() {
    public int compare(Point u, Point v) {
        if (u.x < v.x) {
            return -1;
        } else if (u.x == v.x) {
            if (u.y < v.y) { return -1; }
            else if (u.y == v.y) { return 0; }
            else { return 1; }
        } else {
            return 1;
        }
    }
});
```

좌표 정렬하기

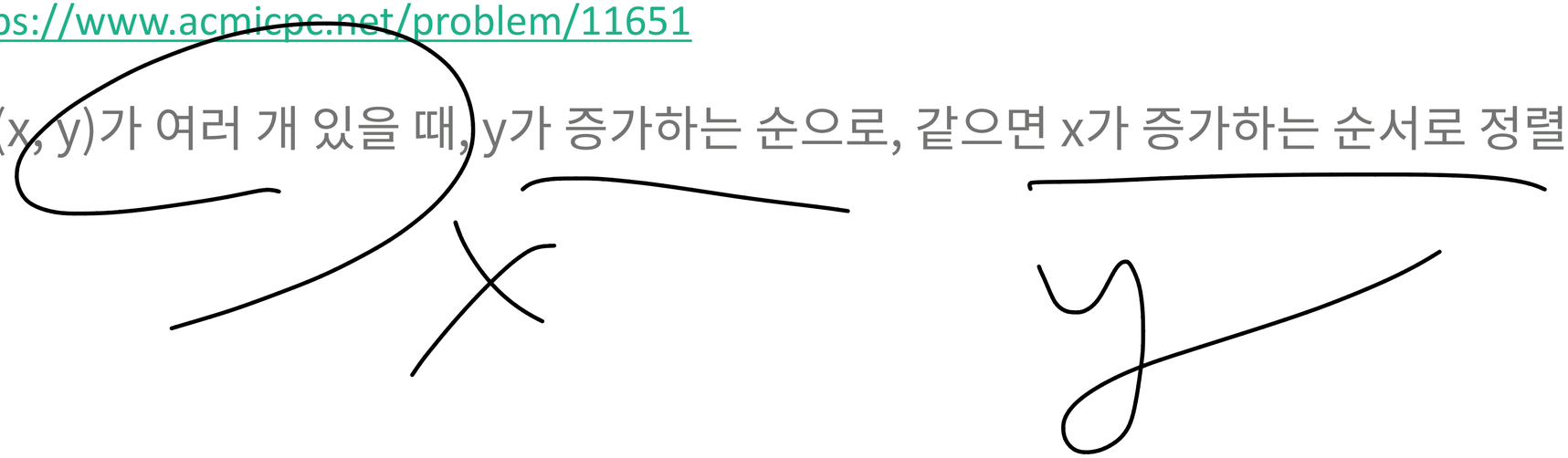
https://www.acmicpc.net/problem/11650

- C++ (pair): http://codeplus.codes/ff3cb84e5929410e90a31a35f92768bf
- C++ (struct, cmp): http://codeplus.codes/6655bff5944f421595fa9e223593cff9
- C++ (struct, 연산자 오버로딩): http://codeplus.codes/ee569248c20f42c88f28ea0d8033a7c6
- Java (Comparable): http://codeplus.codes/b2b6cb734820404eb40795b1a6e92884
- Java (Comparator): http://codeplus.codes/d41f4c785b694b94bc7383301a24579c
- Java (Comparator): http://codeplus.codes/45d95bfebb6f470cb384ce78474956f5

좌표 정렬하기 2

https://www.acmicpc.net/problem/11651

- (x, y)가 여러 개 있을 때, y가 증가하는 순으로, 같으면 x가 증가하는 순서로 정렬하는 문제



좌표 정렬하기 2

https://www.acmicpc.net/problem/11651

- 비교 함수를 y를 우선해서 비교하게 변경하면 된다.
- 또는 x와 y를 반대로 저장한 다음, 정렬하고, 다시 반대로 저장하는 방법도 있다.

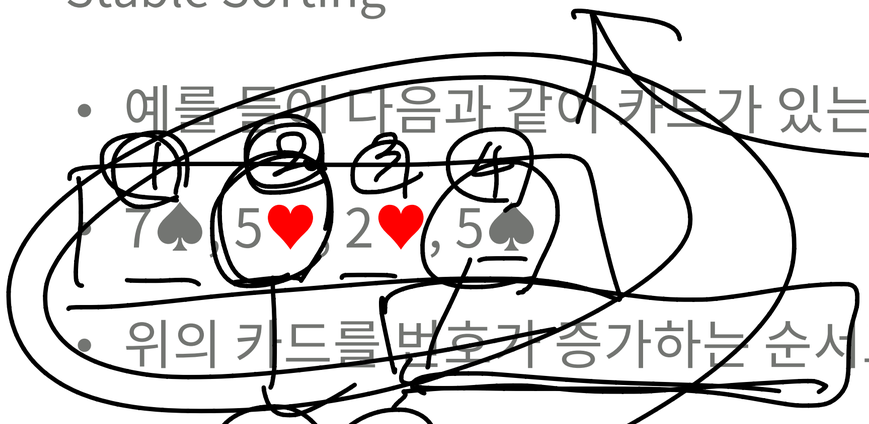
좌표 정렬하기 2

https://www.acmicpc.net/problem/11651

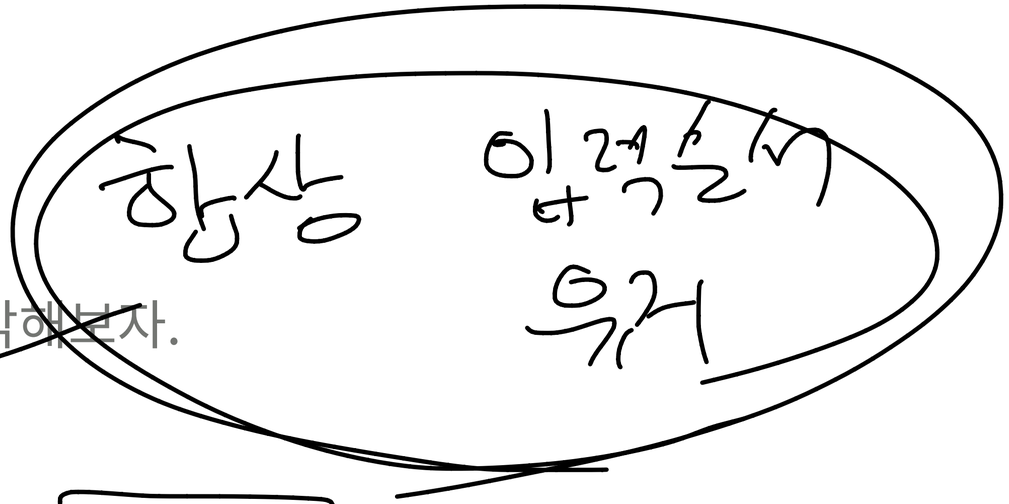
- C++ (pair, cmp): http://codeplus.codes/fc4e1bd27a0c4ed9904046498305d1f7
- C++ (pair, 뒤집어서 저장): http://codeplus.codes/d950837b114a42229e2e9269c4691b9f
- Java: http://codeplus.codes/c157a74561f0446d89698a14697e3894

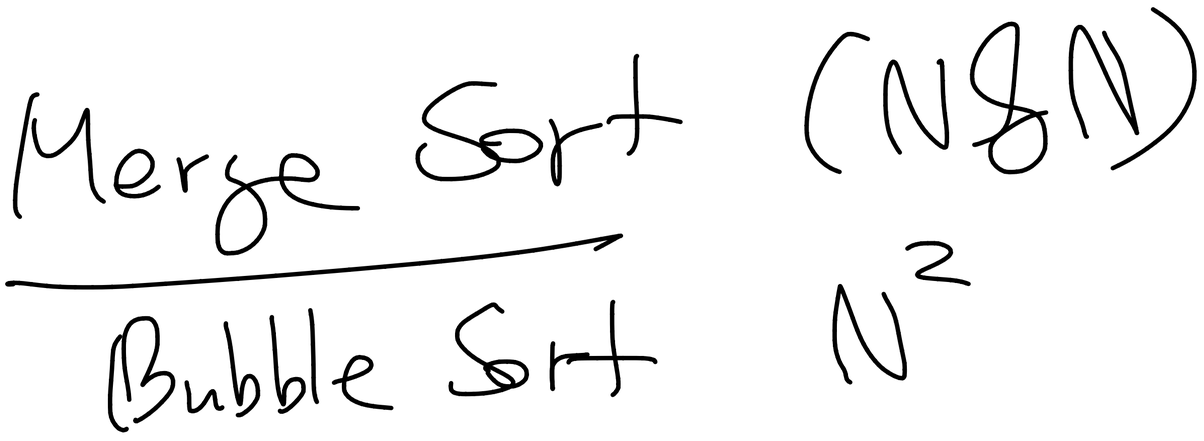
Stable Sorting

Stable Sorting



- 예를 들어 다음과 같이 카드가 있는 경우를 생각해보자.
- 7♠, 5♥, 2♥, 5♠
- 위의 카드를 번호가 증가하는 순서로 정렬했을 때, 5♥ 와 5♠의 순서에 대해 생각해본다.
- 2♥, 5♥, 5♠, 7♠
- 와 같이 정렬이 될 수도 있고
- 2♥, 5♠, 5♥, 7♠
- 와 같이 정렬이 될 수도 있다.





- 같은 것이 있는 경우에 정렬하기 전의 순서가 유지되는 정렬 알고리즘을 Stable Sorting 알고리즘이라고 한다.

Stable Sorting

Stable Sorting

- 시간복잡도가 NlgN 인 정렬 알고리즘 중에는 병합 정렬(Merge Sort)가 있다.
- STL에는 stable_sort 알고리즘이 있다.
- Stable Sorting이 아닌 정렬 알고리즘은, 원래 순서를 의미하는 변수를 하나 더 저장해서 Stable Sort의 효과를 만들 수 있다.

나이순 정렬

https://www.acmicpc.net/problem/10814

- 온라인 저지에 가입한 사람들의 나이와 이름이 가입한 순서대로 주어진다.
- 회원들을 나이가 증가하는 순으로, 나이가 같으면 먼저 가입한 사람이 앞에 오는 순서로 정렬하는 문제
- 가입한 순서는 입력으로 들어오지 않기 때문에, 따로 저장해줘야 한다.

```
struct Person {
    int age;
    string name;
    int join;
};
```

나이순 정렬

https://www.acmicpc.net/problem/10814

- C++ (sort): http://codeplus.codes/18e373e530374e778883a90f35b327f4
- C++ (stable_sort): http://codeplus.codes/7d76f70bc0864873b2a5eebbfcfbcd51
- Java: http://codeplus.codes/0976c36759064a31a18c16d5630f504f

국영수

https://www.acmicpc.net/problem/10825

- 도현이네 반 학생 N명의 이름과 국어, 영어, 수학 점수가 주어진다.
- 다음과 같은 조건으로 학생의 성적을 정렬하는 문제

1. 국어 점수가 감소하는 순서로
2. 국어 점수가 같으면 영어 점수가 증가하는 순서로
3. 국어 점수와 영어 점수가 같으면 수학 점수가 감소하는 순서로
4. 모든 점수가 같으면 이름이 사전 순으로 증가하는 순서로

국영수

https://www.acmicpc.net/problem/10825

- C++: http://codeplus.codes/69ce4659c3cd4f64b37b4710fc6e2dca
- C++ (tuple 사용): http://codeplus.codes/357a52d3b97b43cca8b9b73da4856a05
- Java: http://codeplus.codes/59df880f2ac74bb0a70c327bffec1128

수 정렬하기 3

https://www.acmicpc.net/problem/10989

- N개의 수를 정렬하는 문제 ($1 \le N \le 10,000,000$)
- 입력으로 주어지는 수는 10,000보다 작거나 같은 자연수이다.

- O(N + 10,000) 만에 풀 수 있다.

- cnt[i] = 입력으로 들어온 i의 개수로 풀 수 있다.

사용할 일 없음

수 정렬하기 3

https://www.acmicpc.net/problem/10989

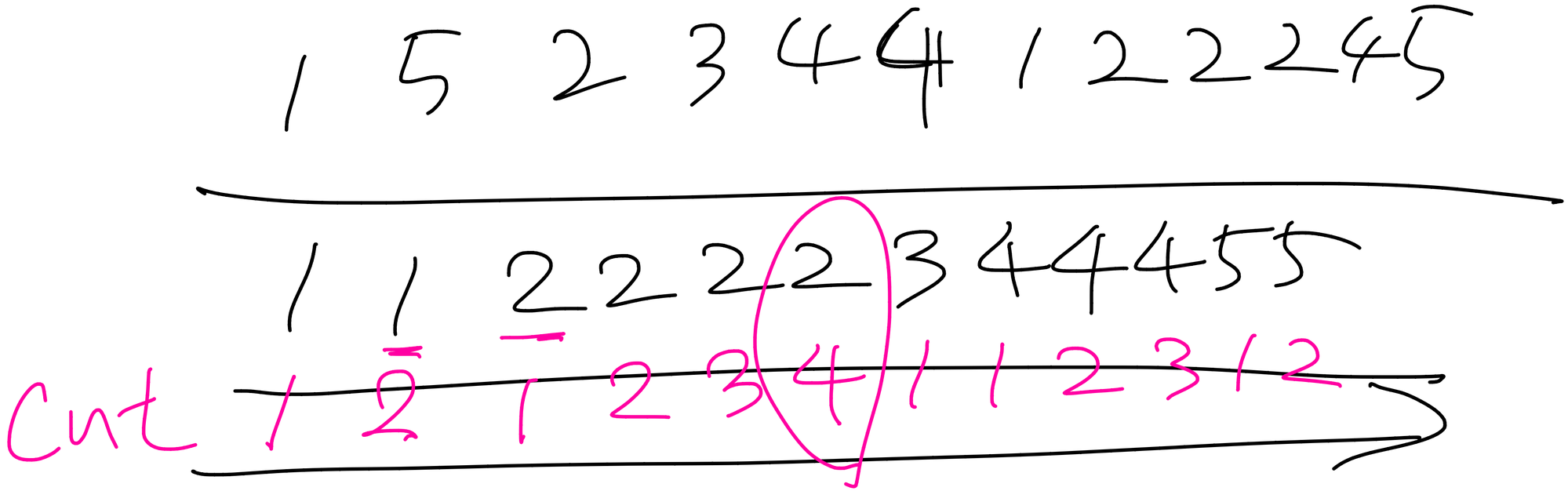
- 소스: http://codeplus.codes/922280f9be9f4966b66c5841666ac8c1

정렬 응용하기

카드

https://www.acmicpc.net/problem/11652

- 준규가 가지고 있는 카드가 주어졌을 때, 가장 많이 가지고 있는 정수를 구하는 문제
- 정렬한 이후에 문제를 풀면 된다.
- 정렬을 한 이후에는 같은 수가 인접해 있기 때문에, O(N)만에 문제를 풀 수 있다.



카드

https://www.acmicpc.net/problem/11652

- 소스: http://codeplus.codes/ab021e6ff9934af4828a92585ff8c5c8

버블 소트

https://www.acmicpc.net/problem/1377

- 크기가 N인 배열 A가 주어진다 (1 ≤ N ≤ 500,000)
- 이 때, A를 버블 소트를 이용해서 정렬했을 때, 몇 번만에 정렬되는지 구하는 문제
- 다음 페이지에 나와있는 소스를 실행시켰을 때, 출력되는 값을 구하는 문제

버블 소트

https://www.acmicpc.net/problem/1377

버블소트

$O(N^2)$

```
bool change = false;
for (int i=1; i<=n+1; i++) {
    change = false;
    for (int j=1; j<=n-i; j++) {
        if (a[j] > a[j+1]) {
            change = true;
            swap(a[j], a[j+1]);
        }
    }
    if (change == false) {
        break;
    }
}
cout << i << '\n';
```

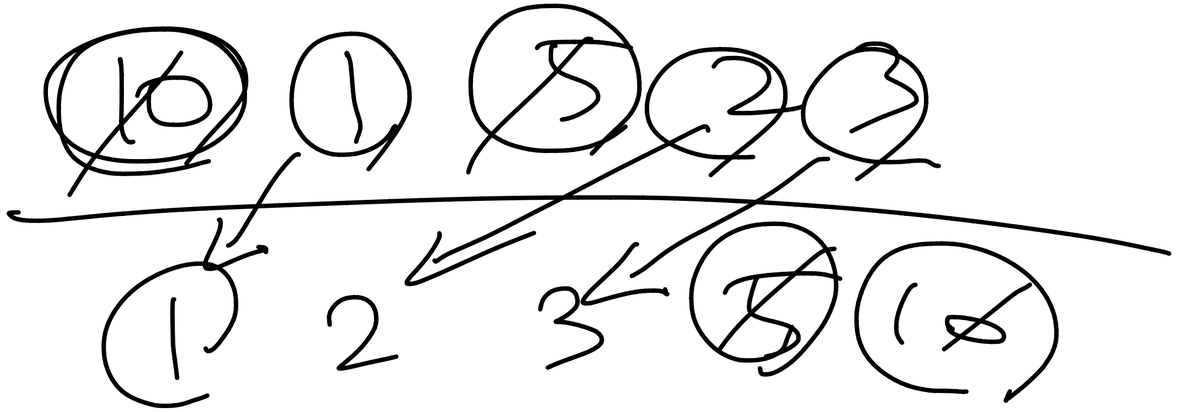
버블 소트

https://www.acmicpc.net/problem/1377

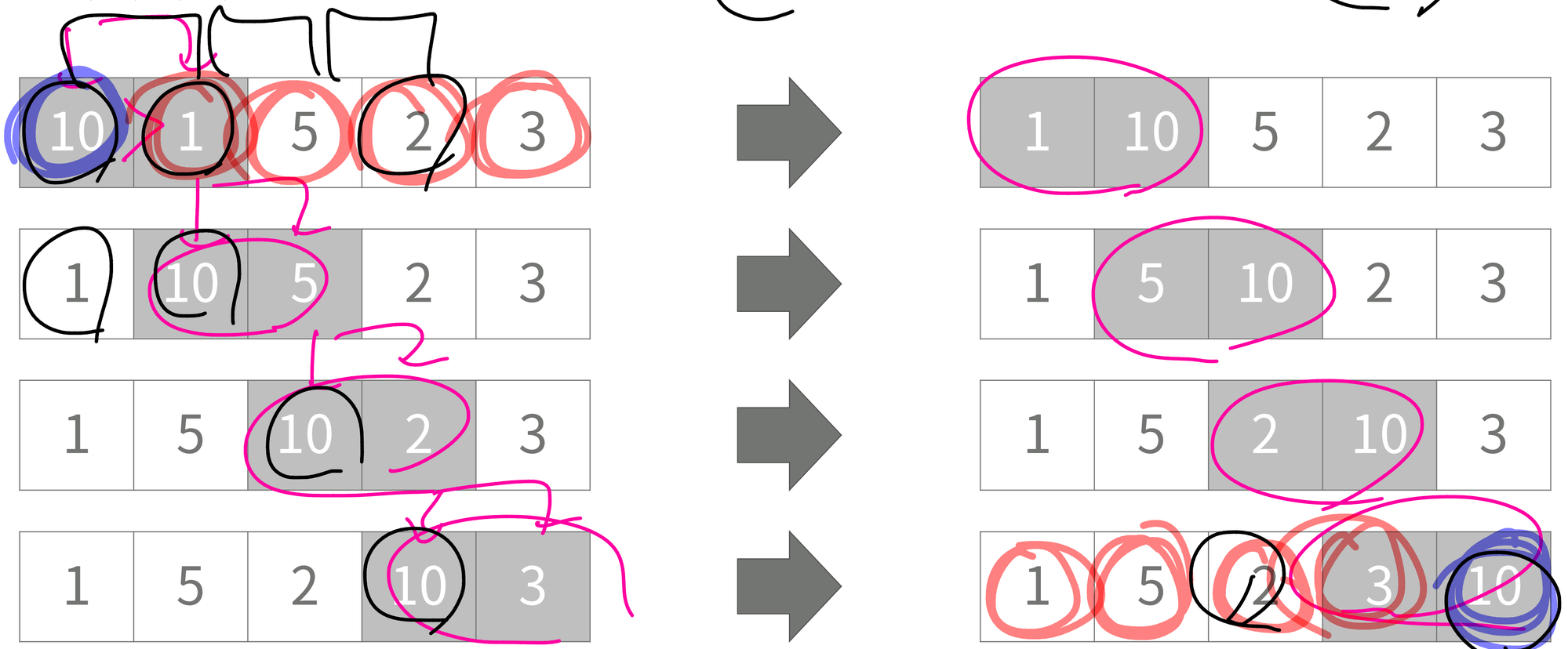
- 배열의 크기가 최대 500,000이기 때문에, 버블 소트를 실제로 수행하는 것은 시간이 너무 오래 걸린다

버블 소트

https://www.acmicpc.net/problem/1377



- A = [10, 1, 5, 2, 3]

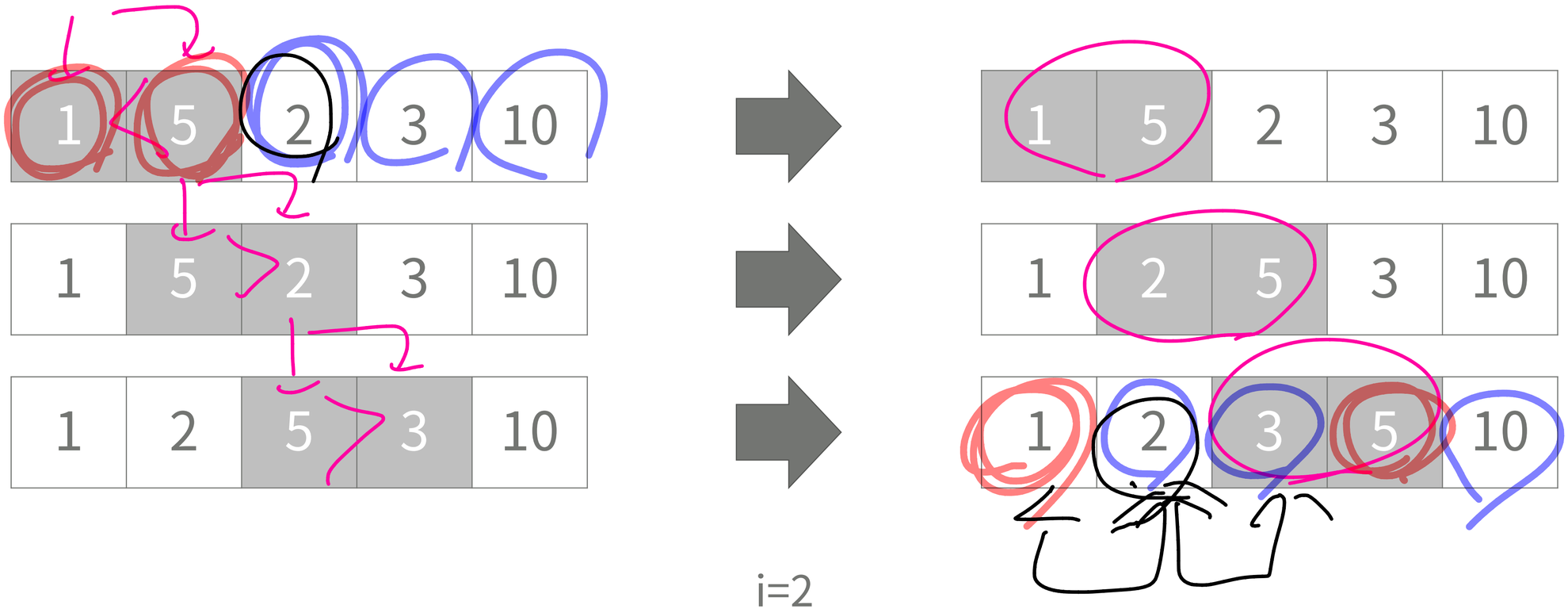


i=1

버블 소트

https://www.acmicpc.net/problem/1377

- A = [10, 1, 5, 2, 3]



버블 소트

https://www.acmicpc.net/problem/1377

- 뒤에 있는 수가 앞으로 오는 것은 한 번에 한 칸만 가능하다.
- 원래 상태와 정렬된 상태를 비교해서, 뒤에 있는 수가 앞으로 오는 경우에 몇 칸 오는지를 조사해서
- 최대값을 구해야 한다

버블 소트

https://www.acmicpc.net/problem/1377

- 소스: http://codeplus.codes/e254d8ab8d634b7988ef6245b0957385