Week 6. Set & Map

– method to use Set and Map Object

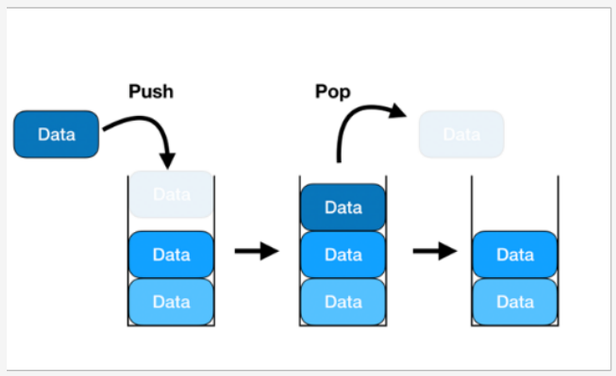
황현석 / 25. 04. 12

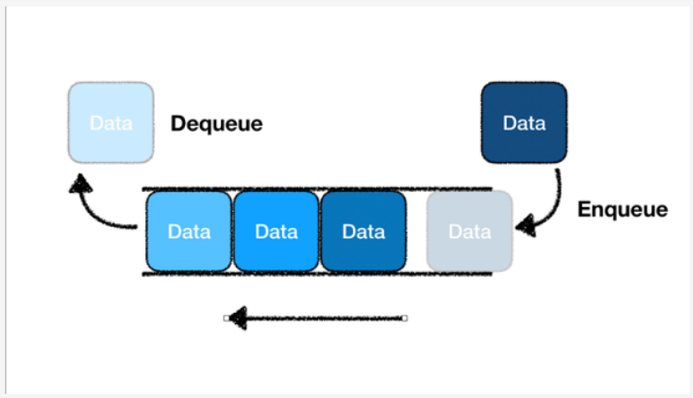
Review.. Very Important

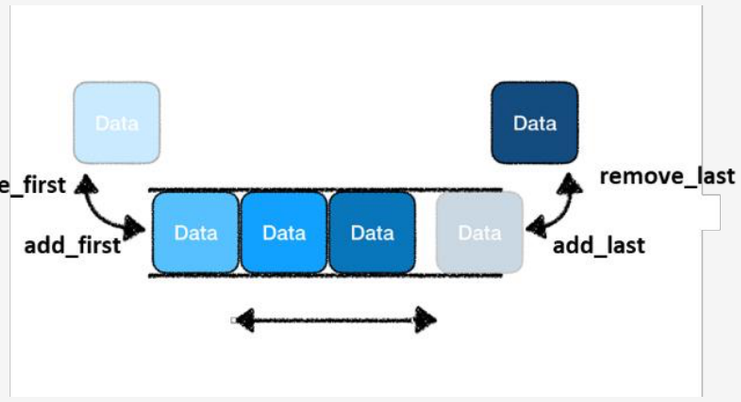
Types of Classes in Java

스택, 큐, 덱

자료를 저장할 수 있고, 도구로 활용할 수 있는 선형 자료 구조의 일종







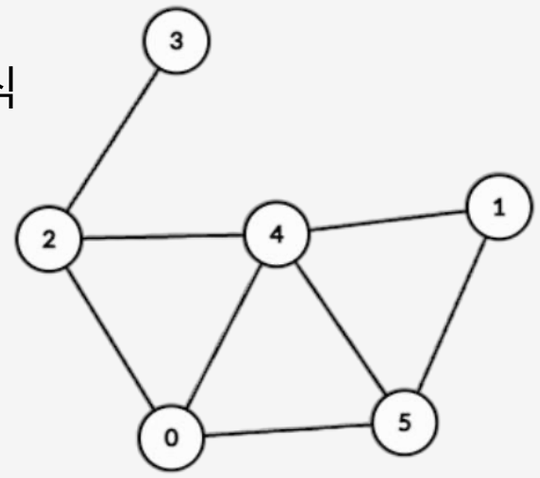
Types of Classes in Java

그래프를 저장 하는 방법

인접리스트 (Adjacency List)

각 정점에 연결된 정점 목록을 리스트로 저장하는 방식
ex) A의 친구는 누구누구.. B의 친구는 누구누구..

간선 존재 확인 속도 = 정점에 연결된 간선 수
ex) A랑 B가 친구야? A의 친구목록에서 찾아보자



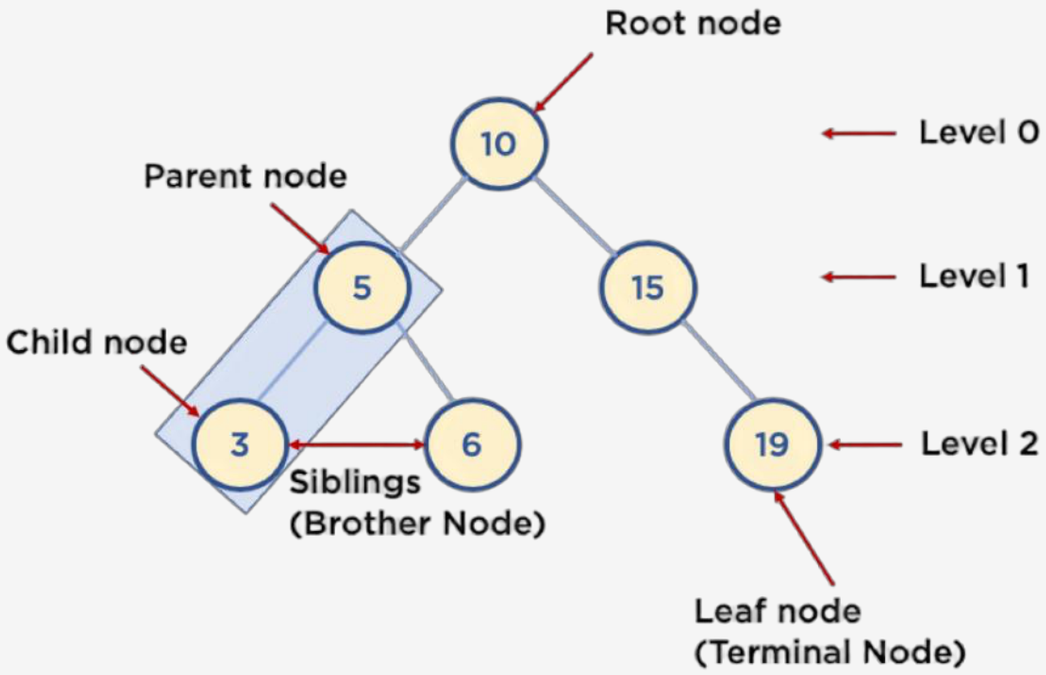
adj[0] = [2, 4, 5]
adj[1] = [4, 5]
adj[2] = [0, 3, 4]
adj[3] = [2]
adj[4] = [0, 1, 2, 5]
adj[5] = [0, 1, 4]

간선 존재 확인속도가 느린대신, 저장공간을 적게 차지한다는 장점이 있다!

Types of Classes in Java

트리란?

트리는 계층적(hierarchical) 구조를 가진 비순환 그래프(사이클이 없는 그래프)입니다!



루트(root) : 트리의 맨 위 노드 (시작점)
부모(parent) : 다른 노드를 가리키는 노드
자식(child) : 부모로부터 연결된 노드
리프(leaf) : 자식이 없는 노드 (끝 노드)
형제(sibling) : 같은 부모를 가진 노드

깊이(depth) : 루트에서 특정 노드까지의 간선 수
높이(height) : 트리의 최대 깊이
서브트리(subtree) : 트리의 일부 (자식 포함 작은 트리)

Table of Contents

1. Set & Map
   A. 집합에 관하여..
   B. HashSet 그리고 HashMap 이 무엇인가요?
   C. 자바에 구현 되어 있는, HashSet, HashMap
2. 우선순위 큐, PriorityQueue
   A. 자바에 구현 되어있는 PriorityQueue
   B. Comparator로 알아보는 정렬 기준 만들기
3. 문제 풀이

What is Set?

자바에서도 그렇고, 대부분의 언어에서 Set은 중복을 허용하지 않는 자료구조 입니다.

ex) A = {1, 2, 3} B = {2, 3, 4}

$$A \cup B = \{1,2,3,4\}$$
$$A \cup B \neq \{1,2,2,3,3,4\}$$

Why use Set?

Set을 사용하는 주요 목적은 중복되지 않는 데이터를 저장하면서, 삽입(add), 삭제(remove), 탐색(contains) 등의 연산을 효율적으로 수행하기 위해서입니다.

ArrayList 등을 이용해서, 원소를 찾고, 하나씩 넣어가면서, Set을 관리하는것은 비효율적입니다.

Set 이라는 add, remove, contains 등의 연산만을 효율적으로 수행하기 위한 자료구조입니다.

흔히 자바에서는, Set을 이용해, ArrayList 안에 중복되는 원소들을 없애는데 사용하기도 합니다.

Why use Set?

흔히 자바에서는, Set을 이용해, ArrayList 안에 중복되는 원소들을 없애는데 사용하기도 합니다.

ArrayList 등을 이용해서, 원소를 찾고, 하나씩 넣어가면서, Set을 관리하는것은 비효율적입니다.

Set 이라는 add, remove, contains 등의 연산만을 효율적으로 수행하기 위한 자료구조입니다.

What is Map?

Map은 Java에서 **키(key)와 값(value)의 쌍으로 데이터를 저장하는 자료구조**입니다. 각 키는 고유(unique)해야 하며, 키를 통해 해당 값을 빠르게 조회할 수 있습니다.

Python 의 Dictionary 문법과 굉장히 유사합니다.

```
my_dict = {
    'hello' : 1,
    'sef' : 2,
    'gsegs' : 3,
    'str' : 'imstring'
}

print(my_dict['str'])
print(my_dict['sef'])
```

```
imstring
2

=== Code Execut
```

Why uses Map?

ArrayList 등을 이용해, (Key, value) 들을 저장한다고 합시다!

Key를 줬을 때, 그에 대응되는 value를 반환하는 자료구조를 만들었습니다!

Key가 어디있는지 모르니까, 전체를 순회 해야합니다.

하지만, 이런식으로 하면, 굉장히 많은 시간이 걸리고 비효율적일 것입니다.

Set에서와 같이 다양한 최적화를 통해, 이것을 줄이고, 효율적인 자료구조를 구현할 수 있습니다.

What is Hash...?

Hash란 데이터를 고정된 크기의 **고유한 숫자 값(해시값)** 으로 변환하는 **함수적 알고리즘**입니다.

EX) “apple” 을 Hash 함수에 넣으면, 19382914 같은 값이 나온다.

```java
int hash = 0;
for (int i = 0; i < s.length(); i++) {
    hash = 31 * hash + s.charAt(i);
}
```

위는 실제, String 데이터에 대한, hash함수를 만드는 과정입니다.

자바 String은 31을 기반으로 하는 다항 해시(polynomial hash) 알고리즘을 사용합니다.

How to use Hash...?

1. 해시 테이블 (Hash Table)

- 가장 핵심 구조는 **배열**입니다.
- HashMap은 내부에 Node<K,V>[] table이라는 배열을 가지고 있습니다.
- HashSet은 이 HashMap을 사용하므로, 결국 해시 기반 자료 구조는 **배열**을 기반으로 동작합니다.

How to use Hash...?

2. HashCode 재보정을 통한, 분산처리
기존에 어떠한 방식으로 만들어진 hashCode나, 사용자가 오버라이딩한 hashCode가 엉망일 수 있습니다.

따라서, h라는 보정된 해쉬값을 다시 만들어서, 테이블의 index 값에 집어넣습니다.

```
1   int h = key.hashCode();
2   int hash = (h >>> 16) ^ h;
3   int index = (table.length - 1) & hash;
```

How to use Hash...?

3. 기본 충돌 처리, Chaining (체이닝)

배열 인덱스 에다가 넣는 건데, LinkedList처럼 넣습니다.

```
1   table[5] → Node("apple") → Node("elppa")
```

그러면, 충돌이 많이 나면, 점점 길어지니까, 탐색도 오래 걸리겠죠…?

How to use Hash...?

4. 충돌이 너무 많아지면? → 트리(Tree)로 전환
자바 8부터는 **성능 저하를 막기 위해**, 충돌 리스트가 일정 길이를 초과하면 Red-Black Tree로 바꿉니다.

한 인덱스에 8개 이상 충돌이 생기고,

- 전체 해시 테이블 크기가 **64 이상**이면

→ 연결 리스트 → **트리로 변환** Treeify

How to use Hash...?

4. …? RB – Tree

용도와 장점 [편집]

레드-블랙 트리는 자료의 삽입과 삭제, 검색에서 최악의 경우에도 일정한 실행 시간을 보장한다(worst-case guarantees). 이는 실시간 처리와 같은 실행 시간이 중요한 경우에 유용하게 쓰일 뿐만 아니라, 일정한 실행 시간을 보장하는 또 다른 자료구조를 만드는 데에도 쓸모가 있다. 예를 들면, 각종 기하학 계산에 쓰이는 많은 자료 구조들이 레드-블랙 트리를 기반으로 만들어져 있다.

AVL 트리는 레드-블랙 트리보다 더 엄격하게 균형이 잡혀 있기 때문에, 삽입과 삭제를 할 때 최악의 경우에는 더 많은 회전(rotations)이 필요하다.

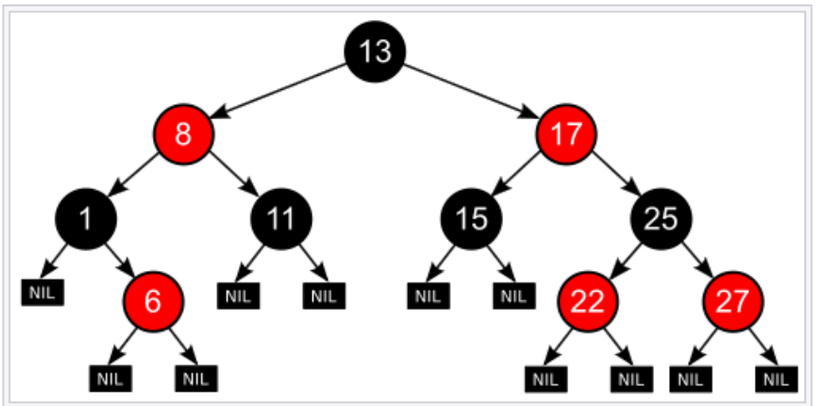
레드-블랙 트리는 함수형 프로그래밍에서 특히 유용한데, 함수형 프로그래밍에서 쓰이는 연관 배열이나 집합(set)등을 내부적으로 레드-블랙 트리로 구현해 놓은 경우가 많다. 이런 구현에는 삽입, 삭제시 O(log n)만큼의 시간이 필요하다.

레드-블랙 트리는 2-3-4 트리와 등장변환이 가능하다(isometry). 다시 말해서, 모든 2-3-4 트리에는 구성 원소와 그 순서(order)가 같은 레드-블랙 트리가 최소한 하나 이상 존재한다는 말이다. 2-3-4 트리에서의 삽입, 삭제 과정은 레드-블랙 트리에서의 색 전환(color-flipping)과 회전(rotation)과 같은 개념이다. 그러므로 실제로는 잘 쓰이지 않지만 2-3-4 트리는 레드-블랙 트리의 동작 과정(logic)을 이해하는 데 많은 도움을 주기 때문에 많은 알고리즘 교과서들이 레드-블랙 트리가 나오기 바로 전에 2-3-4 트리를 소개하고 있다.

특성(Properties) [편집]

레드-블랙 트리는 각각의 노드가 *레드* 나 *블랙* 인 *색상* 속성을 가지고 있는 이진 탐색 트리이다. 이진 탐색 트리가 가지고 있는 일반적인 조건에 다음과 같은 추가적인 조건을 만족해야 유효한(valid) 레드-블랙 트리가 된다:[1]

1. 노드는 레드 혹은 블랙 중의 하나이다.
2. 루트 노드는 블랙이다.
3. 모든 리프 노드들(NIL)은 블랙이다.
4. 레드 노드의 자식노드 양쪽은 언제나 모두 블랙이다. (즉, 레드 노드는 연달아 나타날 수 없으며, 블랙 노드만이 레드 노드의 부모 노드가 될 수 있다)



레드-블랙 트리의 예

Ok.. Hash is using to Optimizing

네!! Hash 라는 숫자 값을 가지고 뭔가 최적화를 하고 있었군요…

이런게 자바에 내장되어 있으면, 그냥 막 사용해도 되는건가요..?
-〉 OK.

근데, 남용하면, 그래도 너무 느려지는거 아닌가요?
-〉 YES

실력있는 개발자가 되기 위해서는, 내부 구조를 다 공부해야하는건가요?
-〉No, 하지만 교양이다.

How to use Hash Set, Map?

```java
public static void main(String[] args) throws IOException{
    HashSet<Integer> set = new HashSet<>();
    HashMap<Integer, String> map = new HashMap<>();

    for (int i=0;i<5;i++) {
        set.add(i);
        map.put(i, "나는 키가 %d 로 저장되어있는 문자열이에요.".formatted(i));
    }

    boolean isContain = set.contains(1);
    boolean isContain2 = map.containsKey(1);
    boolean isContain3 = map.containsValue("나는 키가 1 로 저장되어있는 문자열이에요.");

    System.out.println(isContain);
    System.out.println(isContain2);
    System.out.println(isContain3);
}
```

```
true
true
true
```

Method of HashSet

메서드	설명
`add(E e)`	요소 추가 (`true` 반환: 새로 추가됨)
`remove(Object o)`	요소 제거
`contains(Object o)`	요소 존재 여부 확인
`isEmpty()`	비어 있는지 확인
`size()`	요소 개수 반환
`clear()`	전체 요소 삭제
`iterator()`	요소 순회용 반복자 반환
`toArray()`	배열로 변환
`stream()`	Stream API 사용 가능
`retainAll(Collection<?>)`	주어진 컬렉션과의 **교집합만 유지**

Method of HashMap

메서드	설명
`put(K key, V value)`	key에 value 저장 (기존 값 덮어쓰기)
`get(Object key)`	key에 해당하는 value 반환 (없으면 null)
`remove(Object key)`	key-value 쌍 삭제
`containsKey(Object key)`	특정 key 존재 여부
`containsValue(Object value)`	특정 value 존재 여부
`isEmpty()`	비어 있는지 확인
`size()`	전체 key 개수 반환
`clear()`	모든 entry 삭제
`keySet()`	모든 key 집합 반환 (`Set<K>`)
`values()`	모든 value 모음 반환 (`Collection<V>`)
`entrySet()`	모든 entry (key-value 쌍) 반환 (`Set<Map.Entry<K, V>>`)
`putIfAbsent(K key, V value)`	key가 없을 때만 추가
`compute()`, `computeIfAbsent()`	key 기반 연산으로 값 추가/수정

Practice..

https://www.acmicpc.net/problem/10815

문제

숫자 카드는 정수 하나가 적혀져 있는 카드이다. 상근이는 숫자 카드 N개를 가지고 있다. 정수 M개가 주어졌을 때, 이 수가 적혀있는 숫자 카드를 상근이가 가지고 있는지 아닌지를 구하는 프로그램을 작성하시오.

상근이는, 숫자 카드를 N개 가지고 있다.
다음 M개의 숫자 카드를 상근이가 가지고 있으면, 1 그렇지 않으면, 0을 출력한다.

Practice..

```java
public class Main {
    public static void main(String[] args) throws IOException{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int N = Integer.parseInt(br.readLine());

        StringTokenizer st = new StringTokenizer(br.readLine());

        HashSet<Integer> set = new HashSet<>();

        for (int i=0;i<N;i++) {
            set.add(Integer.parseInt(st.nextToken()));
        }

        int M = Integer.parseInt(br.readLine());

        st = new StringTokenizer(br.readLine());

        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
        for (int i=0;i<M;i++) {
            boolean isContain = set.contains(Integer.parseInt(st.nextToken()));
            bw.write(isContain ? "1 " : "0 ");
        }

        bw.flush();
    }
}
```

What is Priority Queue..?

Priority Queue는 일반적인 큐(Queue)와는 다르게, 우선순위(priority)에 따라 요소를 꺼내는 자료구조입니다.

즉, 먼저 들어온 순서가 아닌, 우선순위가 높은 요소가 먼저 나가는 큐입니다.

여러분이 우선순위를 명시해놓으면, 그 기준을 보고 자료를 정렬해주는 클래스가 내장되어 있습니다.

What is Priority Queue..?

Heap이란?

Heap(힙)은 **완전 이진 트리** 형태로 구성된 **특수한 자료구조**로,
항상 **최댓값 또는 최솟값을 빠르게 찾아야 하는 경우**에 매우 유용합니다.

힙의 특징

- **완전 이진 트리**: 마지막 레벨을 제외하고는 모두 채워져 있으며, 왼쪽부터 차례대로 채워짐
- **우선순위 기반 구조**: 부모 노드와 자식 노드 간에 크기 관계를 유지함
- **배열 기반 구현**: 트리처럼 생겼지만 실제로는 배열로 구현되어 효율적임

What is Priority Queue..?

```java
public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        pq.add(1);
        pq.add(5);
        System.out.println(pq.poll());
        pq.add(10);
        pq.add(2);
        System.out.println(pq.poll());
        pq.add(5);
        pq.add(-14);
        System.out.println(pq.poll());
    }
}
```

```
/Users/hye
1
2
-14
```

Priority Queue는 작은 것 부터 우선 내뱉는다..

What is Priority Queue..?

```java
class Point {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>();

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }
    }
}
```

다음과 같이 나는 Point 객체를 PriorityQueue 에 넣고 관리 하고 싶다.

잘 작동 할까…?

What is Priority Queue..?

```java
class Point {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>();

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }
    }
}
```

```
/Users/hyeonseok/Library/Java/JavaVirtualMachines/openjdk-23.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/
Exception in thread "main" java.lang.ClassCastException Create breakpoint : class Point cannot be cast to class java.lang.Comparable (P
    at java.base/java.util.PriorityQueue.siftUpComparable(PriorityQueue.java:643)
    at java.base/java.util.PriorityQueue.siftUp(PriorityQueue.java:639)
    at java.base/java.util.PriorityQueue.offer(PriorityQueue.java:330)
    at java.base/java.util.PriorityQueue.add(PriorityQueue.java:311)
    at Main.main(Main.java:18)
```

What is Priority Queue..?

```java
PriorityQueue<Point> pq = new PriorityQueue<>(new Comparator<Point>() {
    @Override
    public int compare(Point o1, Point o2) {
        return Integer.compare(o1.x, o2.x);
    }
});
```

```java
class Point implements Comparable<Point> {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public int compareTo(Point o) {
        return Integer.compare(this.x, o.x);
    }
}
```

What is Priority Queue..?

```java
class Point implements Comparable<Point> {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public int compareTo(Point o) {
        return Integer.compare(this.x, o.x);
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>();

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }

        while(!pq.isEmpty()) {
            Point p = pq.poll();
            System.out.println(p.x + " " + p.y);
        }
    }
}
```

```java
import java.io.*;
import java.util.*;

class Point  {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>(new Comparator<Point>() {
            @Override
            public int compare(Point o1, Point o2) {
                return Integer.compare(o1.x, o2.x);
            }
        });

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }

        while(!pq.isEmpty()) {
            Point p = pq.poll();
            System.out.println(p.x + " " + p.y);
        }
    }
}
```

What is Priority Queue..?

```java
class Point implements Comparable<Point> {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public int compareTo(Point o) {
        return Integer.compare(this.x, o.x);
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>();

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }

        while(!pq.isEmpty()) {
            Point p = pq.poll();
            System.out.println(p.x + " " + p.y);
        }
    }
}
```

```
/Users/
18 21
34 81
38 52
56 30
67 14
72 27
72 28
73 59
78 70
83 97
```

```java
import java.io.*;
import java.util.*;

class Point  {
    int x, y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}

public class Main {
    public static void main(String[] args) throws IOException{
        PriorityQueue<Point> pq = new PriorityQueue<>(new Comparator<Point>() {
            @Override
            public int compare(Point o1, Point o2) {
                return Integer.compare(o1.x, o2.x);
            }
        });

        Random rand = new Random();
        for (int i=0;i<10;i++) {
            pq.add(new Point(rand.nextInt(100),rand.nextInt(100)));
        }

        while(!pq.isEmpty()) {
            Point p = pq.poll();
            System.out.println(p.x + " " + p.y);
        }
    }
}
```

We’ve completed Week 6