题目描述

公司某部门软件教导团正在组织新员工每日打卡学习活动，他们开展这项学习活动已经一个月了，所以想统计下这个月优秀的打卡员工。每个员工会对应一个id，每天的打卡记录记录当天打卡员工的id集合，一共30天。

请你实现代码帮助统计出打卡次数top5的员工。加入打卡次数相同，将较早参与打卡的员工排在前面，如果开始参与打卡的时间还是一样，将id较小的员工排在前面。

注：不考虑并列的情况，按规则返回前5名员工的id即可，如果当月打卡的员工少于5个，按规则排序返回所有有打卡记录的员工id。

输入描述

第一行输入为新员工数量N，表示新员工编号id为0到N-1，N的范围为[1,100]

第二行输入为30个整数，表示每天打卡的员工数量，每天至少有1名员工打卡。

之后30行为每天打卡的员工id集合，id不会重复。

输出描述

按顺序输出打卡top5员工的id，用空格隔开。

用例

输入	11 4 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 1 7 10 0 1 6 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 6 10 7 10
输出	10 0 1 7 6
说明	员工编号范围为0~10，id为10的员工连续打卡30天，排第一，id为0,1,6,7的员工打卡都是两天，id为0,1,7的员工在第一天就打卡，比id为6的员工早，排在前面，0,1,7按id升序排列，所以输出[10,0,1,7,6]

	7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 1 2 3 4 5 0 1 2 3 4 5

输入	7 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 1 2 3 4 5
输出	0 1 2 3 4
说明	员工编号范围为0-6，id为0，1，2，3，4，5的员工打卡次数相同，最早开始打卡的时间也一样，所以按id升序返回前5个id

输入	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1
输出	1 0
说明	只有两名员工参与打卡，按规则排序输出两名员工的id

题目解析

简单排序题。需要注意的是，排序要素需要记录每个员工第一次打卡日期，作为第二优先级排序。

JavaScript算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
```

题目解析

简单排序题。需要注意的是，排序要素需要记录每个员工第一次打卡日期，作为第二优先级排序。

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 32) {
    const n = lines[0] - 0;
    const dayCount = lines[1].split(" ").map(Number);
    const dayIds = lines.slice(2).map((line) => line.split(" ").map(Number));
    console.log(getResult(dayIds));

    lines.length = 0;
  }
});

function getResult(dayIds) {
  const employees = {};

  for (let i = 0; i < dayIds.length; i++) {
    const ids = dayIds[i];
    for (let id of ids) {
      if (employees[id]) {
        employees[id].count++;
      } else {
        employees[id] = {
          count: 1,
          firstDay: i,
        };
      }
    }
  }

  let arr = [];
  for (let id in employees) {
    const { count, firstDay } = employees[id];
    arr.push([id, count, firstDay]);
  }

  arr.sort((a, b) =>
    b[1] !== a[1] ? b[1] - a[1] : b[2] !== a[2] ? a[2] - b[2] : a[0] - b[0]
  );

  return arr
    .slice(0, 5)
    .map(([id]) => id)
    .join(" ");
}
```

Java算法源码

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int[] dayCount = new int[30];
    for (int i = 0; i < 30; i++) {
      dayCount[i] = sc.nextInt();
    }

    int[][] dayIds = new int[30][];
    for (int i = 0; i < 30; i++) {
      int m = dayCount[i];
      dayIds[i] = new int[m];
      for (int j = 0; j < m; j++) {
        dayIds[i][j] = sc.nextInt();
      }
    }

    System.out.println(getResult(dayIds));
  }

  public static String getResult(int[][] dayIds) {
```

Java算法源码

```java
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int[] dayCount = new int[30];
    for (int i = 0; i < 30; i++) {
      dayCount[i] = sc.nextInt();
    }

    int[][] dayIds = new int[30][];
    for (int i = 0; i < 30; i++) {
      int m = dayCount[i];
      dayIds[i] = new int[m];
      for (int j = 0; j < m; j++) {
        dayIds[i][j] = sc.nextInt();
      }
    }

    System.out.println(getResult(dayIds));
  }

  public static String getResult(int[][] dayIds) {
    HashMap<Integer, Integer[]> employees = new HashMap<>();

    for (int i = 0; i < dayIds.length; i++) {
      int[] ids = dayIds[i];

      for (int id : ids) {
        if (employees.containsKey(id)) {
          employees.get(id)[0]++;
        } else {
          // 加入数组含义是：该id员工的 [打卡次数, 第一天打卡日期]
          employees.put(id, new Integer[] {1, i});
        }
      }
    }

    ArrayList<Integer[]> list = new ArrayList<>();
    for (Integer id : employees.keySet()) {
      Integer[] employee = employees.get(id);
      int count = employee[0];
      int firstDay = employee[1];
      list.add(new Integer[] {id, count, firstDay});
    }

    list.sort(
        (a, b) ->
            a[1].equals(b[1]) ? (a[2].equals(b[2]) ? a[0] - b[0] : a[2] - b[2]) : b[1] - a[1]);

    StringJoiner sj = new StringJoiner(" ");
    // 不考虑并列的情况，按规则返回前5名员工的id即可，如果当月打卡的员工少于5个，按规则排序返回所有有打卡记录的员工id
    for (int i = 0; i < Math.min(5, list.size()); i++) {
      sj.add(list.get(i)[0] + "");
    }
    return sj.toString();
  }
}
```

Python算法源码

```python
# 输入获取
n = int(input())
dayCount = list(map(int, input().split()))

dayIds = []
for i in range(30):
    dayIds.append(list(map(int, input().split())))


# 算法入口
def getResult(dayIds):
    employees = {}

    for i in range(len(dayIds)):
        ids = dayIds[i]
        for id in ids:
            if employees.get(id) is not None:
                employees[id]["count"] += 1
            else:
                employees[id] = {
                    'count': 1,
                    'firstDay': i
                }
```

Python算法源码

复制

```python
# 输入获取
n = int(input())
dayCount = list(map(int, input().split()))

dayIds = []
for i in range(30):
    dayIds.append(list(map(int, input().split())))


# 算法入口
def getResult(dayIds):
    employees = {}

    for i in range(len(dayIds)):
        ids = dayIds[i]
        for id in ids:
            if employees.get(id) is not None:
                employees[id]["count"] += 1
            else:
                employees[id] = {
                    'count': 1,
                    'firstDay': i
                }

    arr = []
    for id in employees.keys():
        arr.append((id, employees[id]["count"], employees[id]["firstDay"]))

    arr.sort(key=lambda x: (-x[1], x[2], x[0]))

    return " ".join(list(map(lambda x: str(x[0]), arr[:5])))


print(getResult(dayIds))
```