PCT 复习指南

> Author: Haotao (Eric) Lai, Nov 13, 2017

Java Basic APIs

Scanner

不知道 `Scanner` 的话就基本不用考试了，因为会连输入都拿不到。个人建议使用 `nextLine()` 的方法比较好 (如果是处理整形的话，整行取到之后进行分割再从 `String` 转为 `int`)，因为 `nextInt()` 这个方法有一个坑，我曾经被坑过一次。

String

处理输入数据经常需要用到字符串分割的方法，如果是根据空格进行分割，推荐使用如下的方法：

```
// assume variable line is something like "12 23 43 56 43" (seperate by space)
// you want to get all integers into an array

// using regular expression, it can handle the situation
// that more than one space between two integers
String[] strArr = line.split("\\s+);
int      array  = new int[strArr.length];

for (int i = 0; i < array.length; i++) {
  array[i] = Integer.parseInt(strArr[i]);
}
```

System

这个包里有一些比较好用的方法，最常见的当然是 `System.out.println()` 了，比较常用的还有一个拷贝数组的方法：

```
arraycopy(Object src, int srcPos, Object dest, int destPos, int length);
```

Collection

首先必须知道的是如何对一个集合进行遍历，常用的集合有：

- List
  - ArrayList

    - LinkedList
- Stack
- Queue
- Map
- Set

有时候我们想对一个集合进行一些操作，比如我想对一个学生集合，根据每个学生的 GPA 对所有学生进行排序。当然，你可以自己写一个排序的算法来进行操作。但是我们有更加便捷的方法可以实现这个需求：

1. 复写学生类的比较方法；
2. 利用 `Collections` 类的 `sort()` 方法，传入一个比较器作为参数；

```
// method No. 1
class Student implements Comparable<Student>{
  // ...
  public double gpa;
  // ...
  @Override
  public int compareTo(Student student) {
    return (int) (this.gpa - student.gpa);
  }
  // ...
}

class Main {
  public static void main(String[] args) {
    List<Student> stdList;
    // add something to the list
    Collections.sort(stdList);
  }
}
```

```java
// method No. 2
class Student {
  public double gpa;
}

class StudentComparator implements Comparator<Student> {
    @Override
    public int compare(Student s1, Student s2) {
        return (int) (s1.gpa - s2.gpa);
    }
}

class Main {
  public static void main(String[] args) {
    List<Student> stdList;
    // add something to the list
    Collections.sort(stdList, new StudentComparator());
  }
}
```

Tree

主要需要掌握好二叉树 (Binary Tree)，在目前的六次考试当中，第一、二、六次考试都是考的二叉树。具体来讲，需要掌握以下内容：

1. 如何构建二叉树以及如何判定树中各个节点之间的关系；
    1. 建树的时候为了方便自己，每个节点给一个指向 parent 的指针；
    2. 知道一个节点的 `parent`, `children`, `sibling`, `ancestor`, `descendant` 节点分别表示什么；
2. 三种遍历树的方法 (用递归来写，会比用循环写简单很多)；
    1. 前序遍历 (pre-order)；
    2. 中序遍历 (in-order)；
    3. 后序遍历 (post-order)；
3. 知道如何判断两棵树的结果是不是一致 (structural equivalence)，树的形状一样，节点的数值可以不同；
    1. 如果空间复杂度没有要求，可以将其当成完全二叉树，用 boolean 数组来表示当前位置是否存在节点；
    2. 可以同时递归两棵树进行比较；
    3. 可以使用广度优先搜索 (BFS) 进行比较；
4. 知道什么是二叉搜索树 (Binary Search Tree)，以及如何实现搜索树的基本操作；
    1. 添加节点；
    2. 删除节点；

    3. 查找节点；
5. 了解 AVL 树的基本概念，以及平衡树的基本概念 (如果考了什么红黑树，就只能祝你好运了)；
    1. 什么是树高度；
    2. 什么是 AVL 树的平衡定义，什么是平衡因子 (balance-factor)；
    3. 如何检测是否平衡；
    4. 如何重平衡 (rebalance)；

Graph

个人认为不会考很难的图算法，比如 MST, Dijkstra, Bellmen-Flod 等，考到了也就只能祝你好运了。目前的几次考试当中，只有第四次考试考了图 (寻找是否有死锁)，只需要掌握基本的图搜索算法即可：

1. 深度优先搜索 (DFS)，用循环来写简单很多，不要想递归了，需要用一个 `Stack` 来辅助实现；
2. 广度优先搜索 (BFS)，用循环来写简单很多，不要想递归了，需要用一个 `Queue` 来辅助实现；
3. 注意 `Java` 里有现成的 `Stack` 类但是**没有**现成的 `Queue` 类，需要用队列的时候可以考虑 `ArrayDeque` 类；

Special Array

在 pct3 当中有一个 hashing 的题，虽然名字叫 hashing，但是个人认为这是考的数组。需要理解这种特殊的数组，也不知道学名叫什么，就是一个数组的首尾是相连的，有一种 **循环队列** 的感觉。

Filtering

在 pct2 当中有一个特殊的题，这个题的主要难度是边界处理。如果没有“聪明”的做法的话，需要对所有边界上的点做特殊处理，代码会变得非常的混乱而且容易出错。这里的做法是借用 *动态规划* 当中处理边界条件的做法，给整个数据结构 (二维数组) 外面添加多一层空的边界，这样的话就可以用相同的逻辑处理每一个实际数据点。当然，也可以借用 *计算机视觉* 当中的 filter 的概念进行处理。