

# 作业

## T1

简述你对数据的逻辑结构与存储结构的理解。

- 逻辑结构：对于当前的数据之间的关系进行分析，进而思考应该如何存储。
- 存储结构：设计一定的方法存储到程序中。

## T2

简述数据结构与数据类型的区别与联系。

区别：

- 数据结构：是一种元素之间的关系与组织，类似于通过某种架构将不同的数据单元联系起来的東西。
- 数据类型：就是上述的被联系起来的小单元。

联系：

- 数据结构是组织框架，不同类型的数据是填充组织框架的单元。

## T3

对下列用二元组表示的数据结构，试分别画出对应的逻辑结构图，并指出属于何种结构。

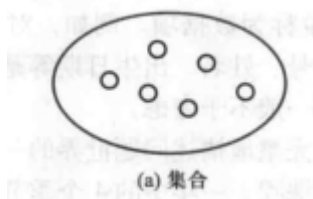
(1)  $A=(D, R)$ , 其中  $D=\{x, y, z, p\}$ ,  $R=\{\}$ ;

(2)  $B=(D, R)$ , 其中  $D=\{a, b, c, d, e\}$ ,  $R=\{(a, b), (b, e), (e, d), (d, c)\}$ ;

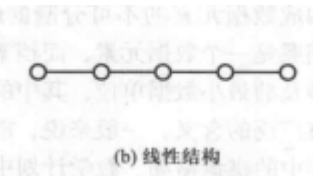
(3)  $G=(D, R)$ , 其中  $D=\{a, b, c, d, e\}$ ,  $R=\{(a, d), (c, e), (b, e), (a, b), (b, c), (d, e)\}$ ;

(4)  $T=(D, R)$ , 其中  $D=\{1, 2, 3, 4, 5, 6\}$ ,  $R=\{(6, 5), (6, 2), (2, 3), (2, 4), (2, 1)\}$ ;

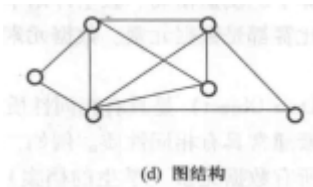
1. 数据之间没有关系，属于集合



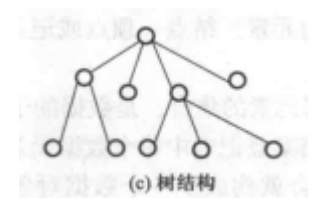
2. 数据之间有一个链式的关系，属于线性结构



3. 数据之间存在一个环，属于图结构



4. 数据之间组成了一个无环图，可以判定为树结构



## T4

何谓抽象数据类型？请你谈谈对它的理解。

所谓抽象数据类型其实就是自定义一个容器用来存储相应的数据，就好比 `int` 这个容器，是用来存储整型数据的。那么抽象数据类型可以自定义为任何一种，假设为 `book`，那么这个数据类型就可以存储每一本书，包括书名、作者、内容等等任何自己想要存储的数据。而 C++ 中的 STL 就是类似于写好的很多中抽象数据类型，比如 `queue`、`stack`、`list` 等等。

## T5

分析以下各程序段，求解它的时间复杂度。

```
1 i = 1; s = 0;
2 while (i < n) {
3     s = s + 10 * i;
4     i++;
5 }
```

只和  $i$  与  $n$  的大小关系相关,  $O(n)$

```
1 while (i + j < n)
2     if (i > j) j++;
3     else i++
```

无论是  $i++$  还是  $j++$  最终都是只会加一次, 而最多只会加  $n$  次,  $O(n)$

```
1 y = 1;
2 while (y * y <= n) y = y + 1;
```

每次  $y$  加一, 最多加到  $\sqrt{n}$  循环就结束了,  $O(\sqrt{n})$

```
1 i = n;
2 while (i > 0) i = i / 2;
```

$n/2^k = 1$  即可算出  $k = \log n$ ,  $O(\log n)$

```
1 for (i = 1; i <= n; i++)
2     for (j = 1; j <= n; j++)
3         for (k = 1; k <= n; k++)
4             s++;
```

三重循环,  $O(n^3)$

```
1 for (i = 1; i <= n; i++)
2     for (j = 1; j <= i; j++)
3         s++;
```

两重循环,  $O(n^2)$

## T6

对一个整型数组  $a[n]$  设计一个排序算法, 用 C++ 作为代码描述, 并分析其时间复杂度。

```

1 void Sort(std::vector<int>& a, int l, int r) {
2     if (l == r) return;
3     int i = l - 1, j = r + 1, m = a[(l + r) >> 1];
4     while (i < j) {
5         while (a[++i] < m);
6         while (a[--j] > m);
7         if (i < j) std::swap(a[i], a[j]);
8     }
9     Sort(a, l, j);
10    Sort(a, j + 1, r);
11 }

```

时间复杂度：  $O(n \log n)$

分析：上述算法使用了快速排序，对于一系列数字；

- 首先取数列中得到任意一个数作为基准，接着扫描一遍数列，将大于基准的数移到其右边，小于基准的数移到其左边，那么一次的时间复杂度就是  $O(n)$
- 接着我们需要考虑重复执行上述遍历几次才可以将数列完全排好序，答案是不确定的，但是可以计算出范围为：  $O(\log n)$  到  $O(n)$ 
  - 对于  $O(\log n)$ ，即假如每一次选取的基准数都是当前数列的中位数，那么一个数列每次都会被均分为一半，那么递归子数列的次数就是  $\log n$
  - 对于  $O(n)$ ，即每一次选取的基准数都是当前数列的最值，那么一个数列在被划分的时候都被划分为一个数和剩余的数列，那么就会被划分  $n$  次，递归子数列的次数就是  $n$
- 综上所述算法的时间复杂度为：  $O(n \log n)$