STAT243 Lecture 2.6 Data Structures

> **Logic**
>
> 选数据结构会影响：内存占用、访问速度、增删是否需要大量复制、后续计算是否方便

1 Standard data structures in Python and R

- 常见：**dataframe**, **list**, **array/vector/matrix/tensor**。
- Python 常用 **numpy array**, **pandas dataframe**（来自额外包）。
- **dict**：按键取值；R 可用 named vector、named list，或更高效的 **environment**。
- R 若不是矩形数据或标准数值对象，常用（嵌套）list。
- 分布式数据结构在 Unit 7 讨论（跨机器的数据）。

1.1 选用建议（按任务）

- 表格数据处理（分组、连接、透视）→ **DataFrame**。
- 高效数值计算、线性代数 → **numpy array / R matrix**。
- 异质、层级数据 → **list**。
- 按键随机访问、映射 → **dict / environment**。
  （以上结合课程描述整理，无额外延展。）

2 Other kinds of data structures

- 本类结构的差异点在于**如何在元素间导航**（访问路径与更新方式）。

2.1 Set

- 定义：不含重复元素的集合；常用于去重与成员测试。

Python

```python
s = {'a', 'b', 'b'}
print(s)
# 输出:
# {'a', 'b'}
print('a' in s)
# 输出:
# True
```

2.2 Linked list

- 结构：每个节点包含值与指向“下一个”的指针；双向链表还指向“上一个”。
- 优点：插入只需改相关指针，无需复制其他元素。
- 缺点：定位任意位置需要从头遍历。

2.3 Trees 与 Graphs

- 共同点：节点与边。
- **Tree**：父子层级结构（也可含父指针）。
- **Graph**：边可无方向，允许出现环。

2.4 Stack 与 Queue

- **Stack**：后进先出，只能直接访问栈顶；嵌套函数调用的行为就是栈，函数调用使用的内存称为 stack。

- **Queue**：先进先出，类似排队。

Python

```python
# stack
stack = []
stack.append('a'); stack.append('b')
print(stack.pop())
# 输出:
# b

# queue (用 deque)
from collections import deque
q = deque(['a','b'])
q.append('c')
print(q.popleft())
# 输出:
# a
```

- 使用方式：可直接实现，或通过 Python/R 的扩展包；R 中并不常用这些结构（但 tree/graph 应用广）。

3 Related concepts

3.1 Types

- 含义：信息如何存储、可做哪些操作。**Primitive types** 与底层存储紧密相关，如 boolean, integer, numeric, character, pointer。

3.2 Pointers

- 指向内存地址的引用；常用于避免不必要的复制。

3.3 Hashes

- 用哈希函数将 key 映射为地址，实现按键快速查找，避免 O(n) 线性扫描。

Python

```python
# 线性扫描 vs dict 查找
items = [{'id': i, 'val': i*i} for i in range(5)]
print([x for x in items if x['id'] == 3][0]['val'])
# 输出:
# 9
d = {x['id']: x['val'] for x in items}
print(d[3])
# 输出:
# 9
```