基础数据结构

湖南师大附中 许力

何为数据结构

- 计算机系统对数据的存储和管理方式

这里的数据是广义上的，所有能被计算机处理的对象都称为“数据”

- 数据结构又可以分为逻辑结构和存储结构

逻辑结构可以天马行空，存储结构则因存储器的特性必须直接或间接线性化

何为数据结构

- 不同的数据结构，在数据的遍历和维护上，效率是不同的

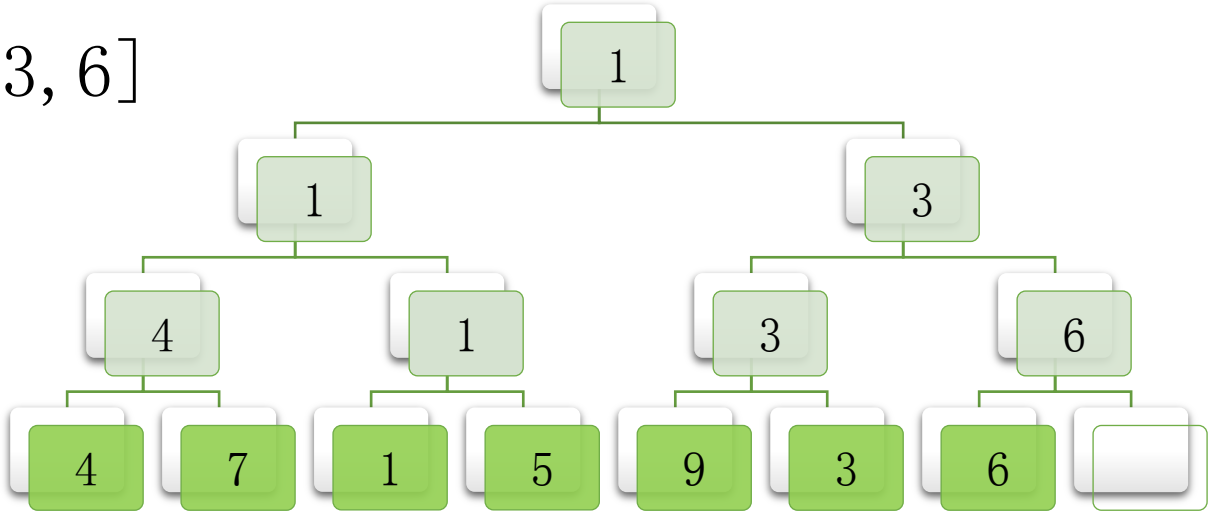
- 比如这样的一类典型问题：

给定长度为n（$n \leqslant 10^6$）的数列，k（$k \leqslant 10^4$）次询问数列中指定区间[L，R]的最小值，注意数列的元素可能被反复更新

何为数据结构

给定长度为n（$n \leqslant 10^6$）的数列，k（$k \leqslant 10^4$）次询问数列中指定区间[L，R]的最小值，注意数列的元素可能被更新

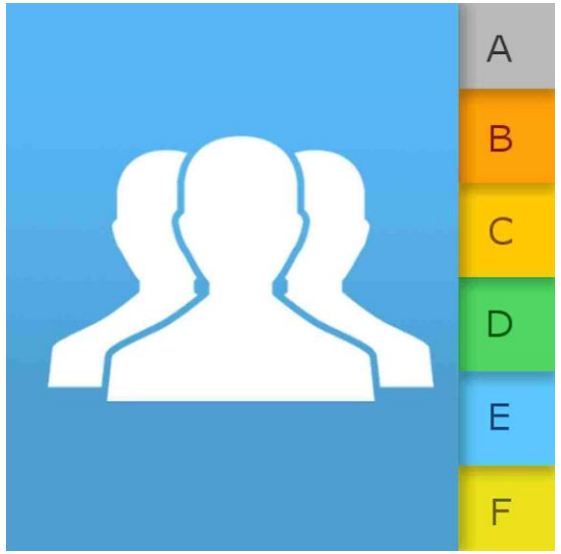
- 假定原始序列为[4,7,1,5,9,3,6]



OI= Programming+Algorithm+Data struct

算法对数据结构的要求

- 举例来说，要写一个手机通讯录的程序，对数据结构能支持的功能有哪些要求？



- 能插入
- 能删除
- 能修改
- 能合并
- 能查询

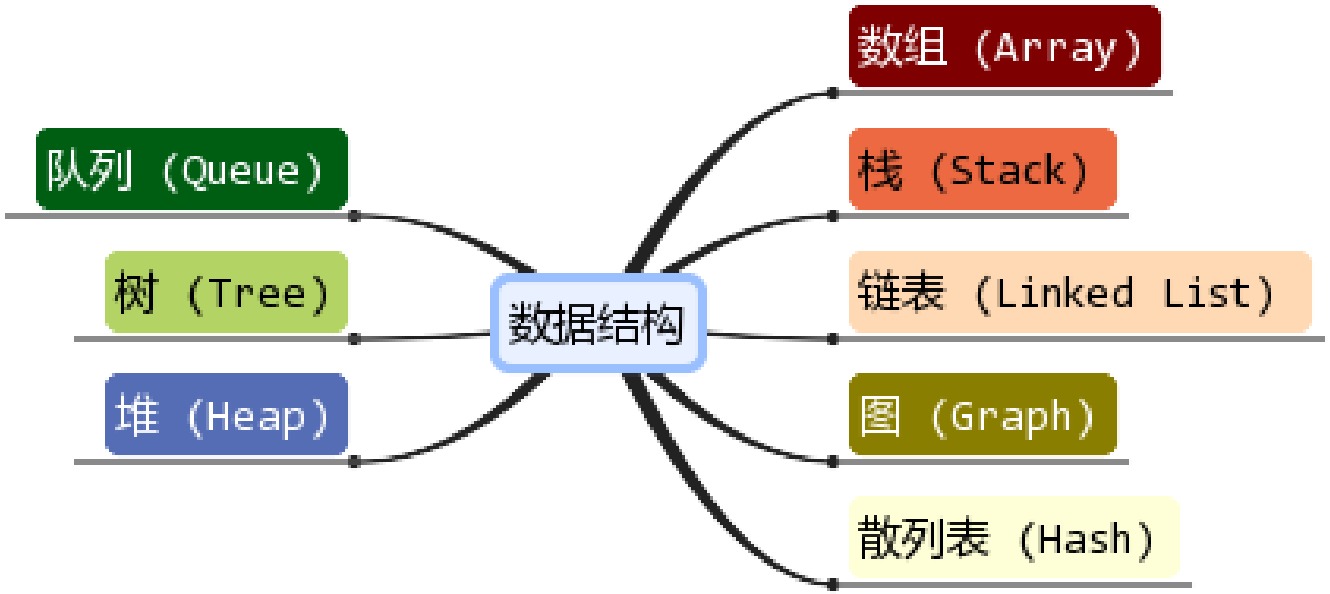
基础数据结构类型

集合类

线性类

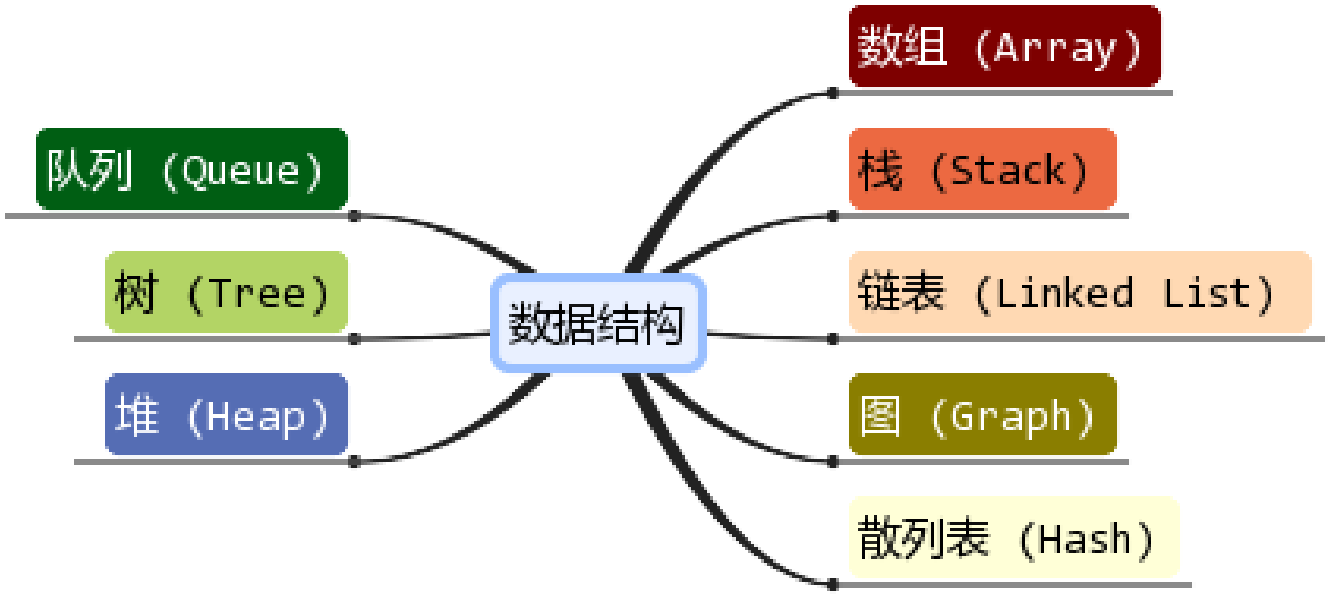
树

图



基础数据结构类型

但其实也可以直接分为两大类：线性的，和非线性的



目录

- 数组
- 栈
- 队列
- 链表
- 树/二叉树
- 堆
- 并查集
- 线段树
- 树状数组
- LCA
- 哈夫曼树

线性数据结构

数组/栈/队列/链表

Vector

- STL中自带的Vector（向量）
- 可以理解为动态数组，即弹性长度数组。具体用法：

1. 定义：vector<int> vec
2. 插入元素x：vec.push_back(x)
3. 删除最后一个元素：vec.pop_back()
4. 返回第一个 / 最后一个 / 任意元素：vec.front() / vec.back() / vec[i]
5. 返回大小：vec.size()

用vector代替普通数组

- 求数列最小值

Sample input	Sample output
4 //n 4 3 9 9	3

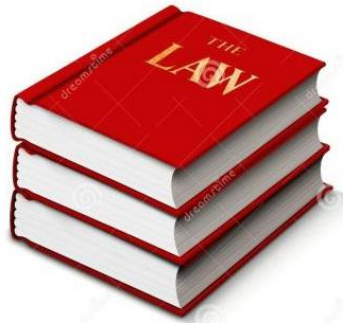
Vector

- Vector还有很多丰富的用法，比如insert、erase、clear

- 需要了解的是：vector不仅仅是弹性长度，而且它还是开辟的连续内存空间，所以在访问速度上也比普通数组快一些
- 但这同时带来了负面效应：使用vector时内存开销只增不减，往往超出预计，所以题目卡内存的时候注意慎用vector，或者及时释放内存

```
vector<int>().swap(vec);
```

栈



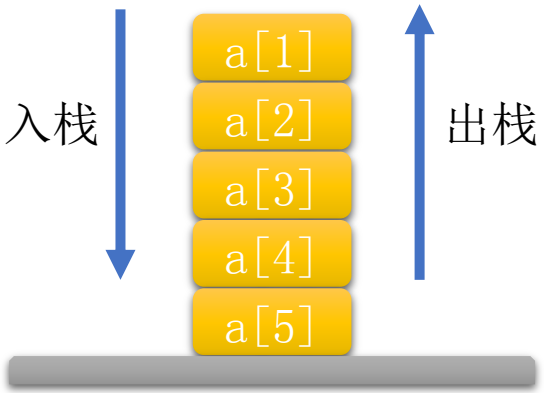
- 后进先出（LIFO）的线性数据结构

- 其模型类似于摆放在桌上的一叠书，后放上的书处于最上方，可被直接取走；先放的书被压在下方，需要先拿走放于其上的书，才能被取走

问

- 元素入栈的顺序为a_1、a_2、a_3、a_4、a_5。如果第1个出栈的是a_3，那么第5个出栈的可能是？

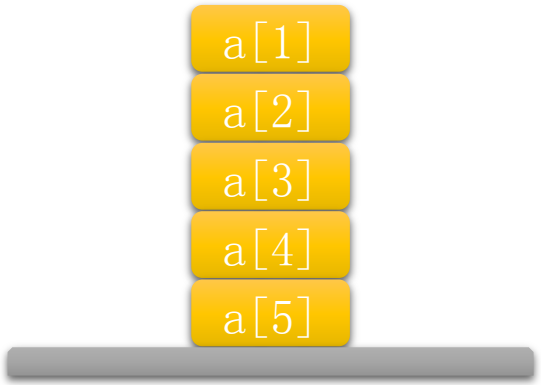
a_1、a_4、a_5

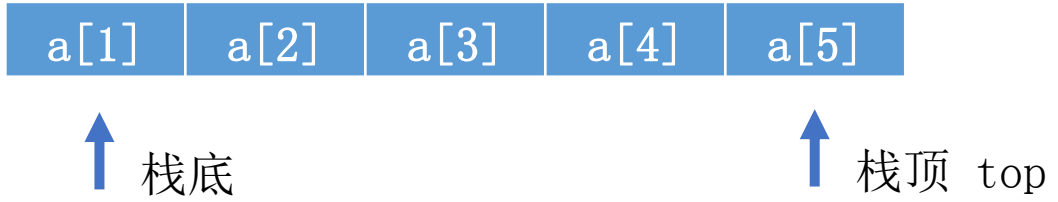


栈的代码实现

1. 数组模拟实现

代码要求：读入数据，模拟实现栈的日常操作：入栈、出栈



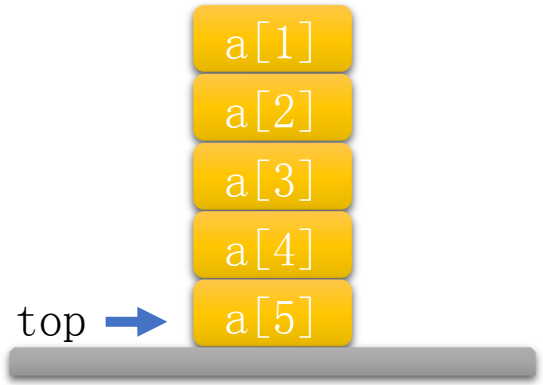


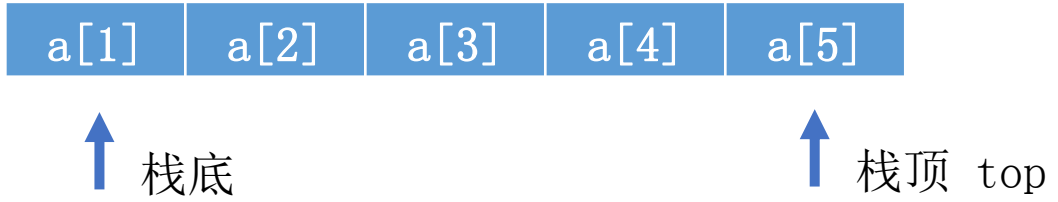
栈的代码实现

1. 数组模拟实现

只需要记录栈顶元素top即可

入栈：++top，出栈：top--





栈的代码实现

2. STL实现

STL中自带标准栈：stack。具体用法：

1. 定义：stack<int> s
2. 元素x入栈：s.push(x)
3. 返回栈顶元素：s.top()
4. 弹出栈顶元素：s.pop()
5. 返回栈大小：s.size()
6. 栈空：s.empty()

计算表达式

- 我们熟知的算术表达式：

8 - (3 + 2 * 6) / 5 + 4

其实是中缀表达式，因为运算符在两个运算数的中间

- 但是对于程序而言，处理后缀表达式效率更高：

8326 * + 5 / - 4 +

后缀表达式无须考虑运算优先级，自然也无须括号

计算表达式

- 但是对于程序而言，处理后缀表达式形态的算术式效率更高：

8326 * + 5 / - 4 +

程序可以从左至右遍历表达式，借助于栈，发现是数字就入栈，发现是运算符就从栈顶连续取出两个数字进行运算，运算结果再入栈

后缀表达式求值

- 计算后缀表达式的值。算术表达式由个位正整数以及“+”、“-”、“*”、“/”运算符构成

Sample input	Sample output
8326 * + 5 / - 4 +	9

中缀表达式转后缀表达式

- 转换规则：

1. 遇到数字直接输出
2. 遇到运算符或者左括号都压入栈
3. 遇到右括号则一直弹栈输出直到遇到左括号
4. 压入运算符时，如果栈顶符号不为括号且运算符优先级不小于当前运算符，则弹出
5. 弹出栈顶运算符并输出。直到栈空或者遇到左括号或优先级低的运算符时停止弹栈，压入当前的运算符
6. 读入结束后弹出栈内所有运算符

中缀表达式转后缀表达式

- 试一试

8 + 7 - 6 + 5 + 4 * 3 / 2 * (1 + 9)

8 7 + 6 - 5 + 4 3 * 2 / 1 9 + * +

中缀表达式转后缀表达式

- 试一试

(8 + (7 - 6) + 5) + 4 * 3 / 2 * (1 + 9)

8 7 6 - + 5 + 4 3 * 2 / 1 9 + * +

中缀表达式转后缀表达式

- 转换的另一种方法：

1. 先按照运算符的优先级对中缀表达式加括号

8 + 7 - 6 + 5 + 4 * 3 / 2 * (1 + 9)

((((8 + 7) - 6) + 5) + (((4 * 3) / 2) * (1 + 9)))

中缀表达式转后缀表达式

- 转换的另一种方法：

2. 将运算符移到括号的后面

(((((8 + 7) - 6) + 5) + (((4 * 3) / 2) * (1 + 9)))

((((8 7) + 6) - 5) + (((4 3) * 2) / (1 9) +) *) +

中缀表达式转后缀表达式

- 转换的另一种方法：

3. 去掉括号

(((8 7) + 6) - 5) + (((4 3) * 2) / (1 9) +) *) +

8 7 + 6 - 5 + 4 3 * 2 / 1 9 + * +

回文串

- 一段长度未知的字符串（最长10^6个字符），判断其是否为回文串，若是则输出“yes”，否则输出“no”

Sample input	Sample output	Sample input2	Sample output2
Ahaha	no	To be eb oT	yes

栈的拓展：单调栈

- 栈中元素保持单调不增/不减的，即为单调栈

最小元素

- 给定一个长为 n （$n \leqslant 10^6$）的整数序列，求出序列中每个元素右边第一个比该元素小的元素。如果没有则输出“0”

Sample input	Sample output
8 //n 7 2 1 4 5 1 3 2	2 1 0 1 1 0 2 0