栈、队列

最基本的数据结构

- 栈：LIFO
- 队列：FIFO

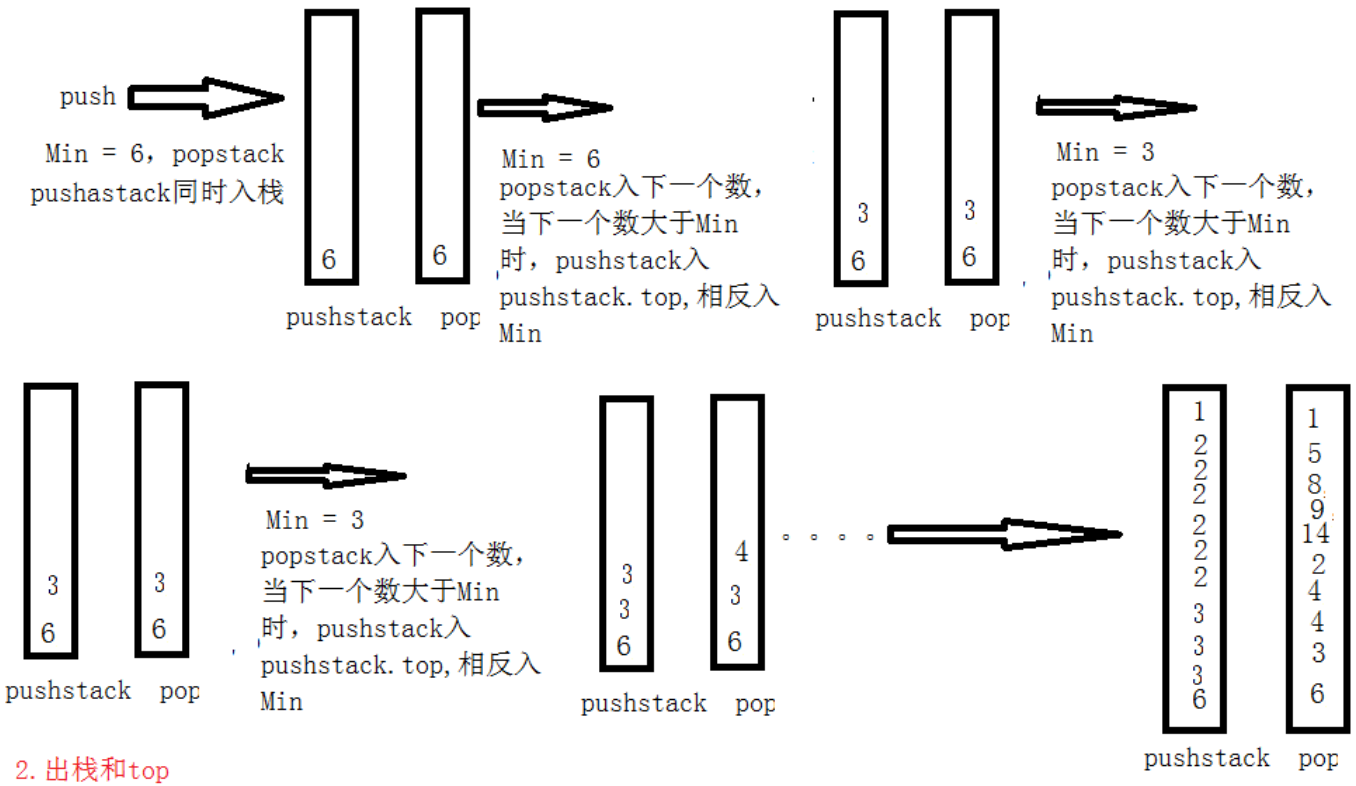
- 都是线性表

栈（Stack）

- 字符（括号）匹配
- 表达式
- 递归/回溯/DFS
- 凸包-Graham扫描法
- 强联通分量-Gabow算法
- 出栈序列（卡特兰数）
- 对顶栈
- ……

实现一个栈Push（出栈）Pop（入栈）Min（返回最小值的操作）的时间复杂度为O(1)

1. push



2. 出栈和top

出栈和top都是pushstack栈顶元素

栈空间

- 编译器自动分配释放，存放函数的参数值，局部变量的值
- 在windows下，栈是向低地址扩展的数据结构，是一块连续的内存区域，栈顶的地址和栈的最大容量是系统预先规定好的，能从栈获得的空间较小。
- 数组局部变量不要开太大
- Windows：-Wl,--stack=SIZE （单位是B）
- Linux：ulimit -s SIZE （单位是KB）

队列

- BFS
- 循环队列
- 双端队列——既可以常数时间内随机访问元素，又可以常数时间内在队列两端插入数据

双栈实现队列——两个栈对导

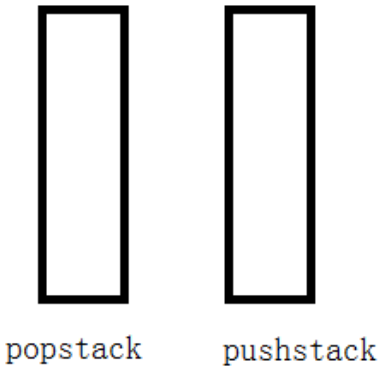
法1：每次插入都得倒回popstack栈的所有元素,最后所有的元素回归popstack

1. 插入，有俩种情况，popstack为空或者popstack不为空

情况1：popstack为空

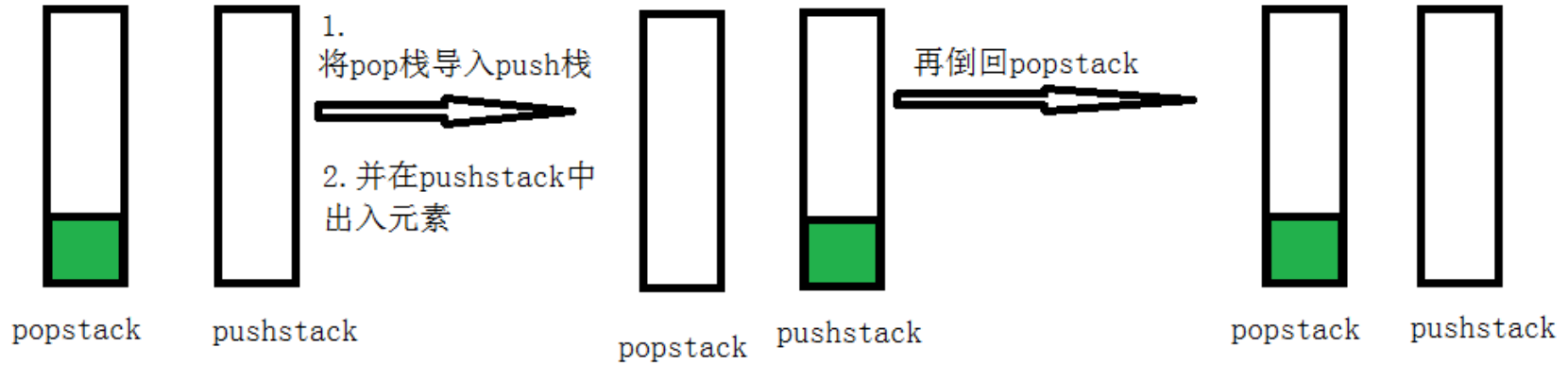
直接在pushstack里插

然后导入popstack



情况2：popstack不为空

1.



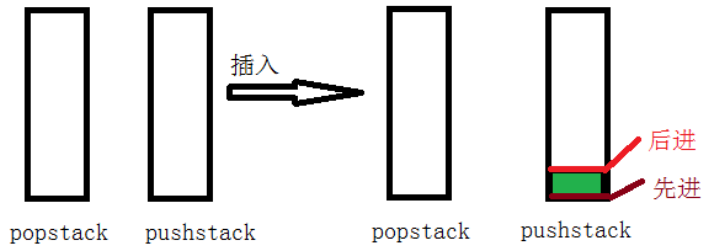
2. 删除　直接删除popstack栈顶元素

3. front
直接返回popstack.top()

法2：直接将数据插入pushstack中；需要出队时将popstack的元素出栈，当popstack为空时，将pushstack导入popstack中

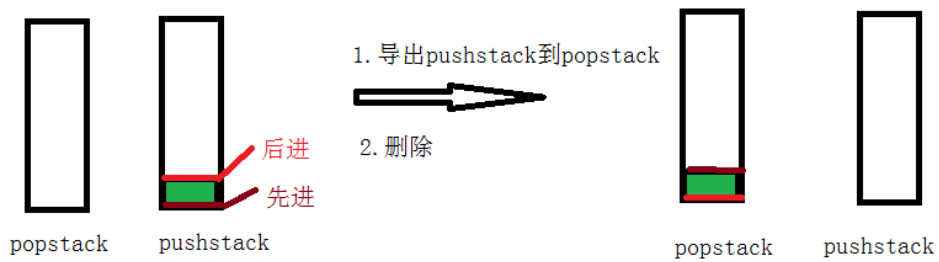
1. 插入

直接在pushstack中出入

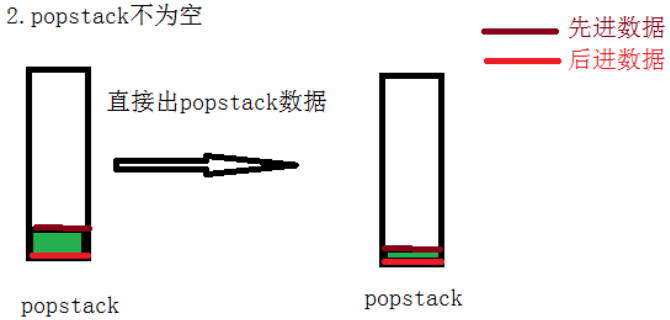


2. 删除

1. popstack为空，将pushstack导入popstack



2. popstack不为空



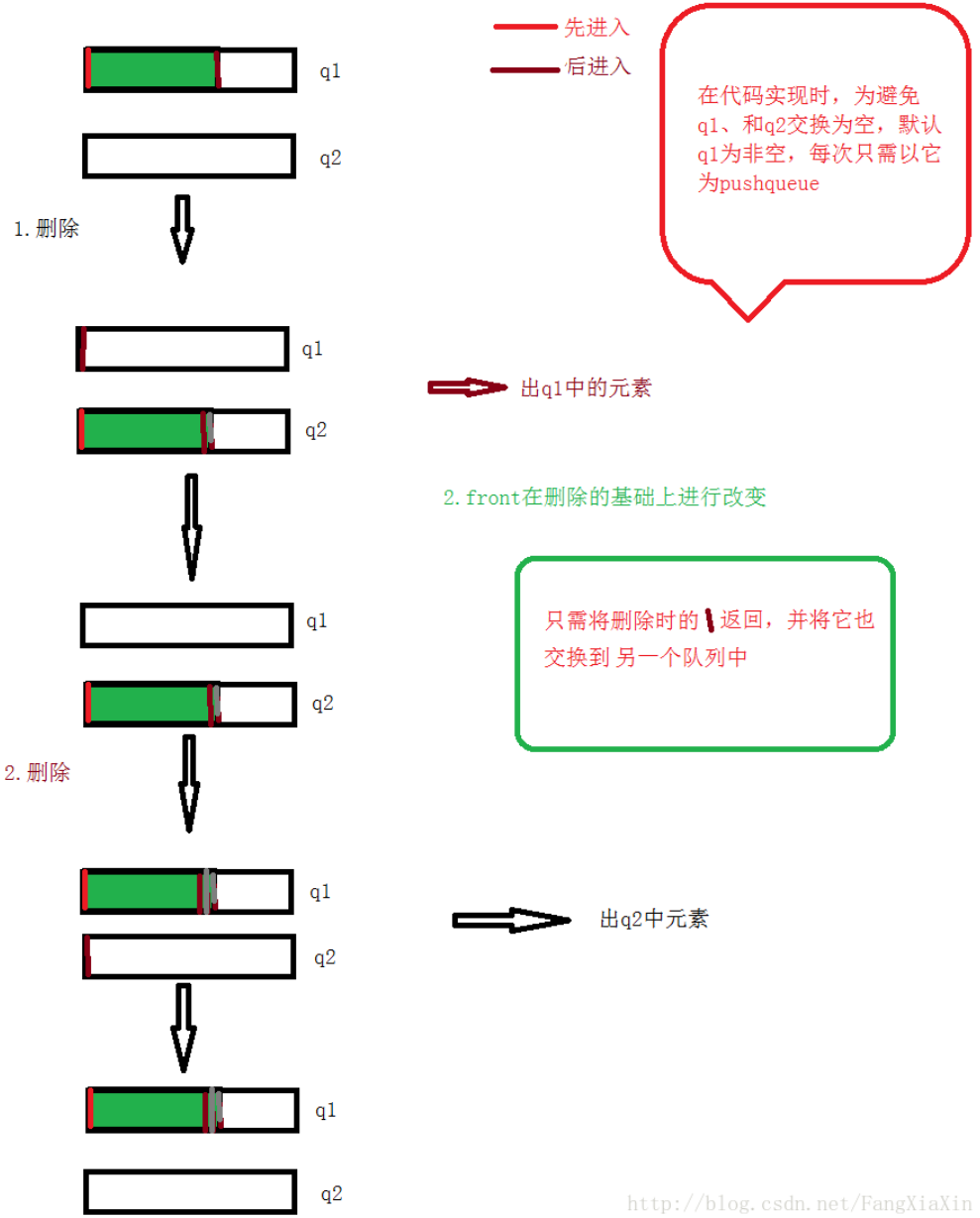
3. front

判断popstack，导入pushstack数据（与上述删除有相似）直接返回popstack数据

双队列实现栈

直接将删除和front

重点：无论任何操作始终要保持有一个队列为空



单调栈

- 数据存在单调性
  - 用于查找下一个较大(小) 的数
  - 确定是否是区间最值
  - 以该元素为最值的最长区间
- O(n^2)到O(n)
- 前缀和优化单调栈：——相应区间和问题

单调队列

- 滑动窗口
- 查询区间最值（不能维护区间k大，因为队列中很有可能没有k个元素）
- 优化DP

单调栈、单调队列区别

- 队列可以从队列头弹出元素，可以方便地根据入队的时间顺序（访问的顺序）删除元素。
- 这样导致了单调队列和单调栈维护的区间不同。当访问到第i个元素时，单调栈维护的区间为[0, i)，而单调队列维护的区间为(lastpop, i)
- 单调队列可以访问“头部”和“尾部”，而单调栈只能访问栈顶（也就是“尾部”）。这导致单调栈无法获取[0, i)的区间最大值/最小值。

~~优先队列~~