# 题目

请你设计一个队列，支持在前，中，后三个位置的 push 和 pop 操作。

请你完成 FrontMiddleBack 类：

FrontMiddleBack() 初始化队列。

void pushFront(int val) 将 val 添加到队列的 最前面 。

void pushMiddle(int val) 将 val 添加到队列的 正中间 。

void pushBack(int val) 将 val 添加到队里的 最后面 。

int popFront() 将最前面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回-1 。

int popMiddle()将正中间的元素从队列中删除并返回值，如果删除之前队列为空，那么返回-1 。

int popBack()将最后面的元素从队列中删除并返回值，如果删除之前队列为空，那么返回-1 。

请注意当有 两个 中间位置的时候，选择靠前面的位置进行操作。比方说：

将6 添加到 [1, 2, 3, 4, 5] 的中间位置，结果数组为 [1, 2, 6, 3, 4, 5] 。

从 [1, 2, 3, 4, 5, 6] 的中间位置弹出元素，返回 3 ，数组变为 [1, 2, 4, 5, 6] 。

示例 1：

输入：

["FrontMiddleBackQueue", "pushFront", "pushBack", "pushMiddle", "pushMiddle", "popFront", "popMiddle", "popMiddle", "popBack", "popFront"]

[[], [1], [2], [3], [4], [], [], [], [], []]

输出：

[null, null, null, null, null, 1, 3, 4, 2, -1]

解释：

FrontMiddleBackQueue q = new FrontMiddleBackQueue();

q.pushFront(1); // [1]

q.pushBack(2); // [1, 2]

q.pushMiddle(3); // [1, 3, 2]

q.pushMiddle(4); // [1, 4, 3, 2]

q.popFront(); // 返回 1 -> [4, 3, 2]

q.popMiddle(); // 返回 3 -> [4, 2]

q.popMiddle(); // 返回 4 -> [2]

q.popBack(); // 返回 2 -> []

q.popFront(); // 返回 -1 -> [] （队列为空）

提示：

1 <= val <= 109

最多调用1000次 pushFront，pushMiddle，pushBack，popFront，popMiddle和popBack。

# 分析

**代码：**

class FrontMiddleBackQueue {

private:

vector<int> q;

public:

FrontMiddleBackQueue() {}

void pushFront(int val) {

q.insert(q.begin(), val);

}

void pushMiddle(int val) {

// 注意正确计算位置

int pos = q.size() / 2;

q.insert(q.begin() + pos, val);

}

void pushBack(int val) {

q.push\_back(val);

}

int popFront() {

if (q.empty()) {

return -1;

}

int ret = q[0];

q.erase(q.begin());

return ret;

}

int popMiddle() {

if (q.empty()) {

return -1;

}

// 注意正确计算位置

int pos = (q.size() - 1) / 2;

int ret = q[pos];

q.erase(q.begin() + pos);

return ret;

}

int popBack() {

if (q.empty()) {

return -1;

}

int ret = q.back();

q.pop\_back();

return ret;

}

};

**复杂度分析：**

时间复杂度：O(n^2)，其中n是操作次数。在最坏情况下，我们每次都调用pushFront()或者pushMiddle()，插入元素的时间复杂度与数组的长度成正比，为O(n)，因此总时间复杂度为O(n^2)。

空间复杂度：O(n)。