ACE Tutorial 4

Question 1: Stacks and Queues

Consider an array-based queue, where the underlying array of size N is used in a circular fashion. We keep track of two variables: f referring to the index of the front element and sz referring to the number of stored elements. When the queue has *fewer than N* elements, the array index $r = (f + sz) \bmod N$ is the first empty slot past the rear of the queue.

Consider a queue that has an underlying array *A* of size 5. Fill in the following f, sz and r values, and show the state of the array A after each operation.

- f (front)：指向队列的前端（即最先插入但尚未删除的元素）。
- sz (size)：存储的元素数量。
- r (rear)：指向队列的下一个空槽，计算方式为 r = (f + sz) mod N，其中 N = 5（数组的大小）。

- Initial State of *A*

Index	0	1	2	3	4
Element					

	value
f	0
sz	0
r	0

- Enqueue 4

Index	0	1	2	3	4
Element	4				

	value
f	0

sz	1
r	1

- Dequeue

Index	0	1	2	3	4
Element					

	value
f	1
sz	0
r	1

- Enqueue 7

Index	0	1	2	3	4
Element		7			

	value
f	1
sz	1
r	2

- Enqueue 10

Index	0	1	2	3	4
Element		7	10		

	value
f	1
sz	2
r	3

- Enqueue 13

Index	0	1	2	3	4
Element		7	10	13	

	value
f	1
sz	3
r	4

- Enqueue 16

Index	0	1	2	3	4
Element		7	10	13	16

	value
f	1
sz	4
r	0

- Dequeue

Index	0	1	2	3	4
Element			10	13	16

	value
f	2
sz	3
r	0

- Dequeue

Index	0	1	2	3	4
Element				13	16

	value
f	3
sz	2
r	0

- Enqueue 19

Index	0	1	2	3	4
Element	19			13	16

	value
f	3
sz	3
r	1

- Enqueue 22

Index	0	1	2	3	4
Element	19	22		13	16

	value
f	3
sz	4
r	2

- Enqueue 25

Index	0	1	2	3	4
Element	19	22	25	13	16

	value
f	3
sz	5
r	3

What happens here? Is *r* referring to an empty cell? Can we add more elements to the array?

- **当 sz == N 时，队列已满，无法继续 enqueue 操作，必须 dequeue 才能腾出空间。**
- **r 的值指向 f，意味着 r 不再指向空单元格，而是循环回到 f。**
- **这是 基于数组的循环队列 在存满后的行为。**

Question 2: Lists

Consider a growable array-based array list. Let *push(o)* be the operation that adds an element *o* at the end of the list. For the pseudocode of the *push(o)* algorithm, see Slide 13 in Lists.pdf. When the array is full, we replace the array with a larger one. There are two commonly used strategies which determine the size of the new array.

Incremental strategy: when an array of size *n* is full, we replace it with a new array of size *(n+c)*, where *c* is a constant.

Doubling strategy: when an array of size *n* is full, we replace it with a new array of size *2n*.

Assume that when the array is not full, adding an element into it takes a constant time 1. Fill in the two tables below, which illustrate the process of performing a series of n *push(o)* operations over an initial array which is empty and of size 1, using the incremental strategy and the doubling strategy, respectively. For the incremental strategy, we set *c=3*.

规则:

- 初始数组大小 1
- 当数组满时，新数组大小 = 旧大小 + 3
- 插入元素的时间:
  - **直接插入**：时间 1
  - **扩容（复制元素）**：需要 旧数组大小 的时间

Incremental strategy, *c=3*

Array size	Push *i*-th element	Time for adding elements	Time for copying elements
1	1	1	0
1+3=4	2	1	1=c-2
4	3	1	0
4	4	1	0
4+3=7	5	1	4=2c-2
7	6	1	0
7	7	1	0
7+3=10	8	1	7=3c-2
10	9	1	0
10	10	1	0

10+3=13	11	1	10=4c-2
13	12	1	0

Let *m* denote the total number of push operations in the series, *k* denote the number of times of increasing the array size. Can you express the relationship between *m* and *k* using *c*?

$$m = ck$$

k=m/c

Let *T(m)* denote the total time for performing these *m* push operations. How to express *T(m)* using *m, k* and *c*? Which big-Oh class does *T(m)* belong to? Which big-Oh class does *T(m)/m* belong to?

T(m)=m+(c+2c+...+kc)-2k=m+[(1+k)k/2]c-2k

T(m)属于 O(m^2)

T(m)/m 属于 O(m)

规则：

- 初始数组大小 1
- 当数组满时，新数组大小 = 旧大小 × 2
- 插入元素的时间：
  - **直接插入**：时间 1
  - **扩容（复制元素）**：需要 旧数组大小 的时间

Doubling strategy

Array size	Push *i*-th element	Time for adding elements	Time for copying elements
1	1	1	0
1*2=2	2	1	1
2*2=4	3	1	2
4	4	1	0
4*2=8	5	1	4
8	6	1	0
8	7	1	0
8	8	1	0
8*2=16	9	1	8
16	10	1	0
16	11	1	0
16	12	1	0
16	13	1	0
16	14	1	0
16	15	1	0
16	16	1	0

Let *m* denote the total number of push operations in the series, *k* denote the number of times of increasing the array size. Can you express the relationship between *m* and *k*?

m=2^k

k=log(2)m

Let *T(m)* denote the total time for performing these *m* push operations. How to express *T(m)* using *m* and *k*? Which big-Oh class does *T(m)* belong to? Which big-Oh class does *T(m)/m* belong to?

T(m)=m+1+2+...+2^(k-1)=m+2^k-1=2m-1 or T(m)=m+1+2+...+2^k=m+2^(k+1)-1=3m-1

$T(m)$ is in $O(m)$

$T(m)/m$ is in $O(1)$