## 第10章 基本数据结构

#### 练习 10.1-1



**问题分析**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 |  |  |  |  |  |
|  | **↑** |  |  |  |  |  |
| *S*.*top*=1 | | |  |  |  |  |

(a) PUSH(*S*, 4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 | 1 |  |  |  |  |
|  |  | **↑** |  |  |  |  |
| *S*.*top*=2 | | | | |  |  |

(b) PUSH(*S*, 1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 | 1 | 3 |  |  |  |
|  |  |  | **↑** |  |  |  |
| *S*.*top*=3 | | | | | | |

(c) PUSH(*S*, 3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 | 1 | 3 |  |  |  |
|  |  | **↑** |  |  |  |  |
| *S*.*top*=2 | | | | |  |  |

(d) POP(*S*)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 | 1 | 8 |  |  |  |
|  |  |  | **↑** |  |  |  |
| *S*.*top*=3 | | | | | | |

(e) PUSH(*S*, 8)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *S* | 4 | 1 | 8 |  |  |  |
|  |  | **↑** |  |  |  |  |
| *S*.*top*=2 | | | | |  |  |

(f) POP(*S*)

#### 练习 10.1-2



**问题分析**

一个栈从底端开始，向顶端增长；另一个栈从顶端开始，向底端增长。当两个栈的top指针相差为1时，栈就满了。

PUSH\_1(*S*, *x*)

**if** *S*.*top*1 < *S*.*top*2 – 1

*S*.*top*1++

*S*[*S*.*top*1] = *x*

**else**

**error** “overflow”

POP\_1(*S*)

**if** *S*.*top*1 > 0

*S*.*top*1--

**return** *S*[*S*.*top*1+1]

**else**

**error** “underflow”

PUSH\_2(*S*, *x*)

**if** *S*.*top*1 < *S*.*top*2 – 1

*S*.*top*2--

*S*[*S*.*top*2] = *x*

**else**

**error** “overflow”

POP\_2(*S*)

**if** *S*.*top*2 <= *S*.*length*

*S*.*top*2++

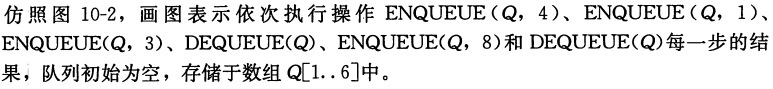
**return** *S*[*S*.*top*2-1]

**else**

**error** “underflow”

注意，在以上代码中，在初始时，栈中还没有一个元素，此时*S*.*top*1 = 0，并且*S*.*top*2 = *S*.*length*+1。

#### 练习 10.1-3



**问题分析**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 |  |  |  |  |  |
|  | **↑** | **↑** |  |  |  |  |
| *head* *tail*  = 1 = 2 | | | |  |  |  |

(a) ENQUEUE(*Q*, 4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 | 1 |  |  |  |  |
|  | **↑** |  | **↑** |  |  |  |
| *head* = 1 *tail* = 3 | | | | |  |  |

(b) ENQUEUE(*Q*, 1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 | 1 | 3 |  |  |  |
|  | **↑** |  |  | **↑** |  |  |
| *head* = 1 *tail* = 4 | | | | | |  |

(c) ENQUEUE(*Q*, 3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 | 1 | 3 |  |  |  |
|  |  | **↑** |  | **↑** |  |  |
|  | *head* = 2 *tail* = 4 | | | | |  |

(d) DEQUEUE(*Q*)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 | 1 | 3 | 8 |  |  |
|  |  | **↑** |  |  | **↑** |  |
|  | *head* = 2 *tail* = 5 | | | | | |

(e) ENQUEUE(*Q*, 8)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *Q* | 4 | | 1 | 3 | 8 |  |  |
|  |  | |  | **↑** |  | **↑** |  |
|  |  | *head* = 3 *tail* = 5 | | | | | |

(e) DEQUEUE(*Q*)

#### 练习 10.1-4



**问题分析**

ENQUEUE(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail* + 1 || *Q*.*head* == *Q*.*tail* + 1 – *Q*.*length*

**error** “overflow”

**else**

*Q*[*Q*.*tail*] = *x*

**if** *Q*.*tail < Q*.*length*

*Q*.*tail*++

**else**

*Q*.*tail* = 1

DEQUEUE(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail*

**error** “underflow”

**else**

*x* = *Q*[*Q*.*head*]

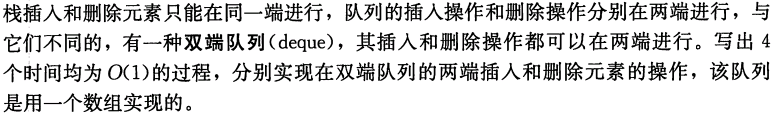
**if** *Q*.*head* == *Q*.*length*

*Q*.*head* = 1

**else**

*Q*.*head*++

#### 练习 10.1-5



**问题分析**

DEQUE\_PUSH\_BACK(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail* + 1 || *Q*.*head* == *Q*.*tail* + 1 – *Q*.*length*

**error** “overflow”

**else**

*Q*[*Q*.*tail*] = *x*

**if** *Q*.*tail* < *Q*.*length*

*Q*.*tail*++

**else**

*Q*.*tail* = 1

DEQUE\_POP\_FRONT(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail*

**error** “underflow”

**else**

*x* = *Q*[*Q*.*head*]

**if** *Q*.*head* < *Q*.*length*

*Q*.*head*++

**else**

*Q*.*head* = 1

DEQUE\_PUSH\_FRONT(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail* + 1 || *Q*.*head* == *Q*.*tail* + 1 – *Q*.*length*

**error** “overflow”

**else**

**if** *Q*.*head* > 1

*Q*.*head*--

**else**

*Q*.*head* = *Q*.*length*

*Q*[*Q*.*head* ] = *x*

DEQUE\_POP\_BACK(*Q*, *x*)

**if** *Q*.*head* == *Q*.*tail*

**error** “underflow”

**else**

**if** *Q*.*tail* > 1

*Q*.*tail*--

**else**

*Q*.*tail* = *Q*.*length*

*x* = *Q*[*Q*.*tail*]

#### 练习 10.1-6



**问题分析**

假设有两个栈*S*1和*S*2。如果要ENQUEUE，就在*S*1上进行PUSH。如果要DEQUEUE，就在*S*2上进行POP。在DEQUEUE时，如果*S*2为空，则从*S*1中逐个弹出元素并插入*S*2中，直到*S*1为空，再从*S*2中弹出一个元素。

这一算法的时间复杂度最坏情况下为O(*n*)。但是平均来看，该算法的摊还时间为O(1)，因为每个元素只会经历常量次的移动操作。

#### 练习 10.1-7



**问题分析**

假设有两个队列*Q*1和*Q*2。如果要PUSH，就在*Q*1上进行ENQUEUE。如果要POP，就在*Q*2上进行DEQUEUE。在POP时，如果*Q*2为空，则从*Q*1中逐个弹出元素并插入*Q*2中，直到*Q*1为空，再从*Q*2中弹出一个元素。

这一算法的时间复杂度最坏情况下为O(*n*)。但是平均来看，该算法的摊还时间为O(1)，因为每个元素只会经历常量次的移动操作。

#### 练习 10.2-1



**问题分析**

INSERT操作在表头插入一个元素，可以在O(1)时间内实现。

SINGLY\_LIST\_INSERT(*L*, *x*)

*x*.*next* = *L*.*head*

*L*.*head* = *x*

DELETE操作需要修改被删除元素的前一个元素的*next*指针。由于单链表没有*prev*指针，所以不能直接找到被删除元素的前一个元素。而需要从链表头部开始遍历，直到找到被删除元素的前一个元素为止，所花费的时间为O(*n*)。

SINGLY\_LIST\_DELETE(*L*, *x*)

**if** *L*.*head* == *x*

*L*.*head* = *x*.*next*

**else**

*prev* = *L*.*head*

**while** *prev*.*next* != *x*

*prev* = *prev*.*next*

*prev*.*next* = *x*.*next*

#### 练习 10.2-2



**问题分析**

SINGLY\_LIST\_PUSH(*L*, *x*)

SINGLY\_LIST\_INSERT(*L*, *x*)

SINGLY\_LIST\_POP(*L*)

**if** *L*.*head* == NIL

**error** “underflow”

**else**

*x* = *L*.*head*.*key*

SINGLY\_LIST\_DELETE(*L*, *L*.*head*)

**return** *x*

在SINGLY\_LIST\_POP中，调用了SINGLY\_LIST\_DELETE。由于删除的是头部元素，所以SINGLY\_LIST\_DELETE所花费的时间为O(1)。

#### 练习 10.2-3



**问题分析**

用单链表实现队列，需要对单链表做一个小小的修改，即记录指向链表尾部元素的指针。ENQUEUE操作只需要在链表尾部插入元素即可，而DEQUEUE只需要在删除链表头部元素即可。由于我们记录下了指向链表尾部元素的指针，所以在链表尾部插入元素只需要花费O(1)时间。

SINGLY\_LIST\_ENQUEUE(*L*, *x*)

*x*.*next* = NIL

**if** *L*.*tail* == NIL

*L*.*head* = *L*.*tail* = *x*

**else**

*L*.*tail*.*next* = *x*

*L*.*tail* = *x*

SINGLY\_LIST\_DEQUEUE(*L*)

**if** *L*.*head* == NIL

**error** “underflow”

**else**

*x* = *L*.*head*.*key*

SINGLY\_LIST\_DELETE(*L*, *L*.*head*)

**if** *L*.*head* == NIL

*L*.*tail* = NIL

**return** *x*

#### 练习 10.2-4



**问题分析**

开始搜索前，让*L*.*nil*.*key* = *k*。

#### 练习 10.2-5



**问题分析**

SINGLY\_CIRCULAR\_LIST\_SEARCH(*L*, *k*)

**if** *L*.*head* == NIL

**return** NIL

**elseif** *L*.*head*.*key* == *k*

**return** *L*.*head*

**else**

*x* = *L*.*head*.*next*

**while** *x*.*key* != *k*

**if** *x* == *L*.*head*

**return** NIL

*x* = *x*.*next*

**return** *x*

SINGLY\_CIRCULAR\_LIST\_INSERT(*L*, *k*)

create a list element *x*

*x*.*key* = *k*

**if** *L*.*head* == NIL

*x*.*next* = *x*

**else**

*x*.*next* = *L*.*head*

*L*.*head* = *x*

SINGLY\_CIRCULAR\_LIST\_DELETE(*L*, *k*)

**if** *L*.*head* == NIL

**error** “underflow”

**elseif** *L*.*head*.*key* == *k*

**if** *L*.*head*.*next* == *L*.*head*

*L*.*head* = NIL

**else**

*L*.*head* = *L*.*head*.*next*

**else**

*x* = *L*.*head*.*next*

*prev* = *L*.*head*

**while** *x*.*key* != *k*

**if** *x* == *L*.*head*

**error** “no element”

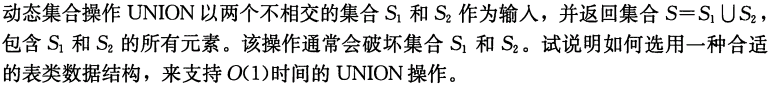
*prev* = *x*

*x* = *x*.*next*

*prev*.*next* = *x*.*next*

INSERT的时间为O(1)，SEARCH和DELETE的时间为O(*n*)。

#### 练习 10.2-6



**问题分析**

采用不带哨兵的双向循环链表。

UNION(*S*1, *S*2)

create a new list *S*

**if** *S*1.*head* == NIL

*S*.*head* = *S*2.*head*

**else**

*S*.*head* = *S*1.*head*

**if** *S*2.*head* != NIL

*S*1.*head*.*prev*.*next* = *S*2.*head*

*S*2.*head*.*prev*.*next* = *S*1.*head*

*tail*1 = *S*1.*head*.*prev*

*S*1.*head*.*prev* = *S*2.*head*.*prev*

*S*2.*head*.*prev* = *tail*1

**return** *S*

#### 练习 10.2-7



**问题分析**

SINGLY\_LIST\_REVERSE(*L*)

*x* = *L*.*head*

*prev* = NIL

**while** *x* != NIL

*y* = *x*.*next*

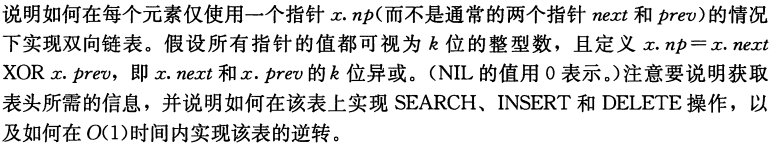
*x*.*next* = *prev*

*prev* = *x*

*x* = *y*

*L*.*head* = *prev*

#### 练习 10.2-8



**问题分析**