

第4章 队列

4.0 Bus Stop Queue

4.1 ADT队列

4.2 用指针实现ADT队列

4.3 用循环数组实现ADT队列

4.4 ADT队列的应用——电路布线问题

4.5 ADT队列的其它应用举例

4.0 Bus Stop Queue



4.0 Bus Stop Queue



front



rear



4.0 Bus Stop Queue



front



rear



4.0 Bus Stop Queue



front



rear



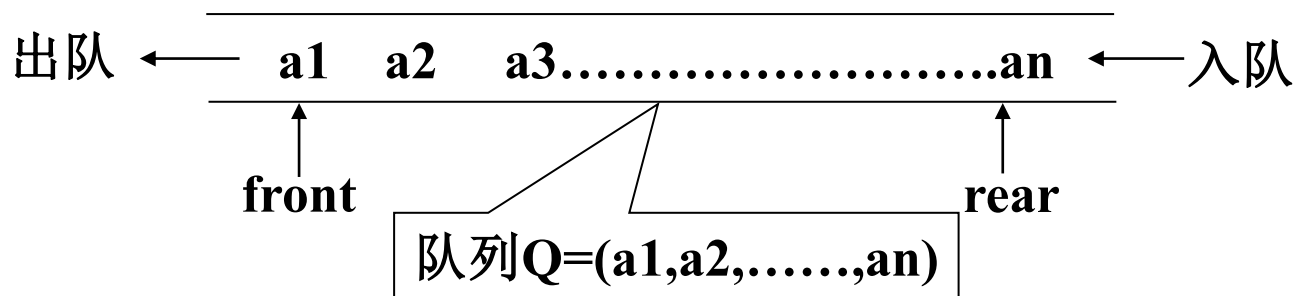
[返回章节目录](#)

4.1 ADT队列(Queue)

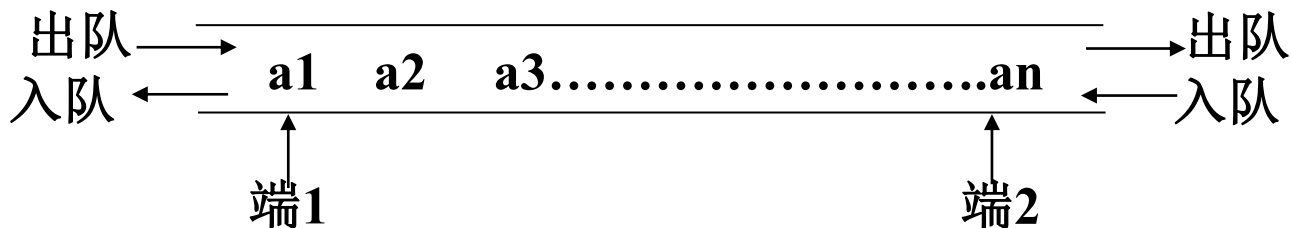
队列是一种特殊的线性表，是**操作受限**的线性表，称其为**限定性数据结构**。

★ 队列的定义及特点

- ⊕ 定义：队列是限定只能在表的一端进行插入，在表的另一端进行删除的线性表
 - ⊕ 队尾(rear)——允许插入的一端
 - ⊕ 队头(front)——允许删除的一端
- ⊕ 队列特点：先进先出(**FIFO**)



• 双端队列



4.1 ADT队列(Queue)

★ ADT队列上定义的常用的基本运算

(1) QueueEmpty(Q):

(2) QueueFull(Q):

(3) QueueFirst(Q):

(4) QueueLast(Q):

(5) EnterQueue(x,Q):

(6) DeleteQueue(Q):

[返回章节目录](#)

4.2 用指针实现ADT队列

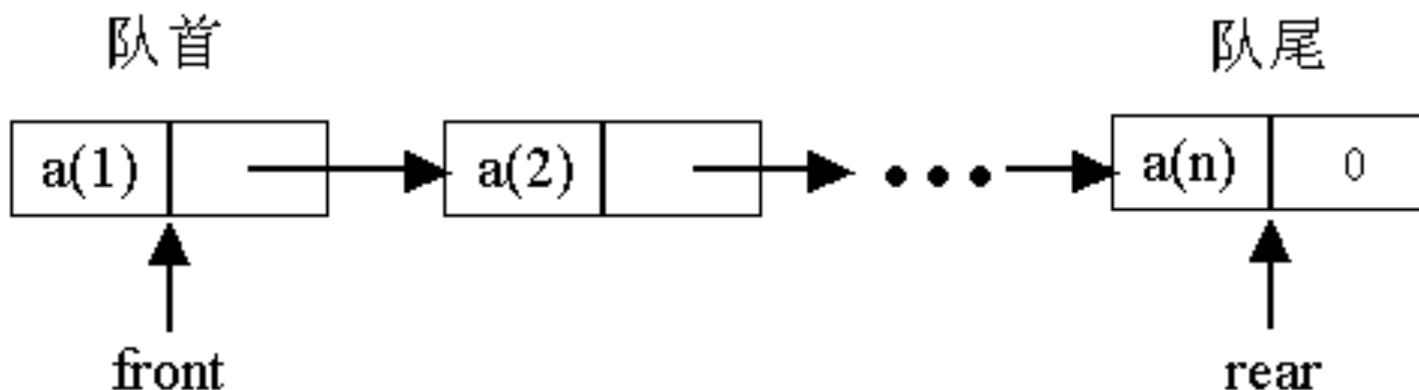
❖ 链队列结点定义

```
typedef struct qnode *qlink;  
typedef struct qnode {  
    QItem element;  
    qlink next;  
} Qnode;
```

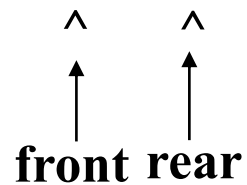


✦ 用指针实现的队列Queue的定义

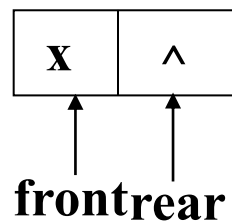
```
typedef struct lque *Queue;  
typedef struct lque{  
    qlink front; //队首结点指针  
    qlink rear;  //队尾结点指针  
}Lqueue;
```



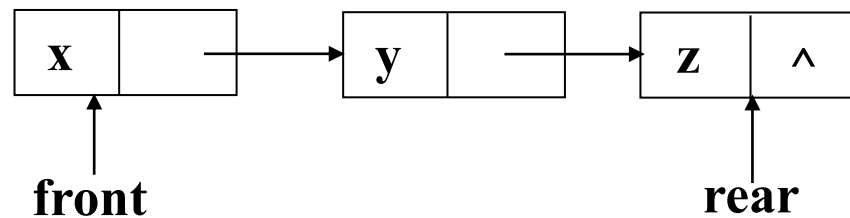
空队列



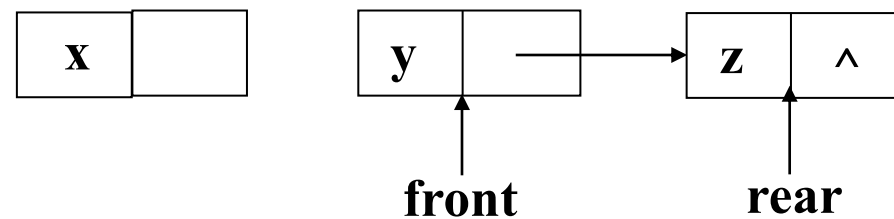
x入队



yz入队



x出队



✿ 入队算法

```
void EnterQueue (QItem x, Queue Q)
{
    qlink p;
    p=NewQNode(); /*创建一个新结点*/
    p->element=x;
    p->next=0;
    /*在队尾插入新结点*/
    if(Q->front) Q->rear->next=p; /*队列非空*/
    else Q->front=p;               /*空队列*/
    Q->rear=p;
}
```

❖ 出队算法

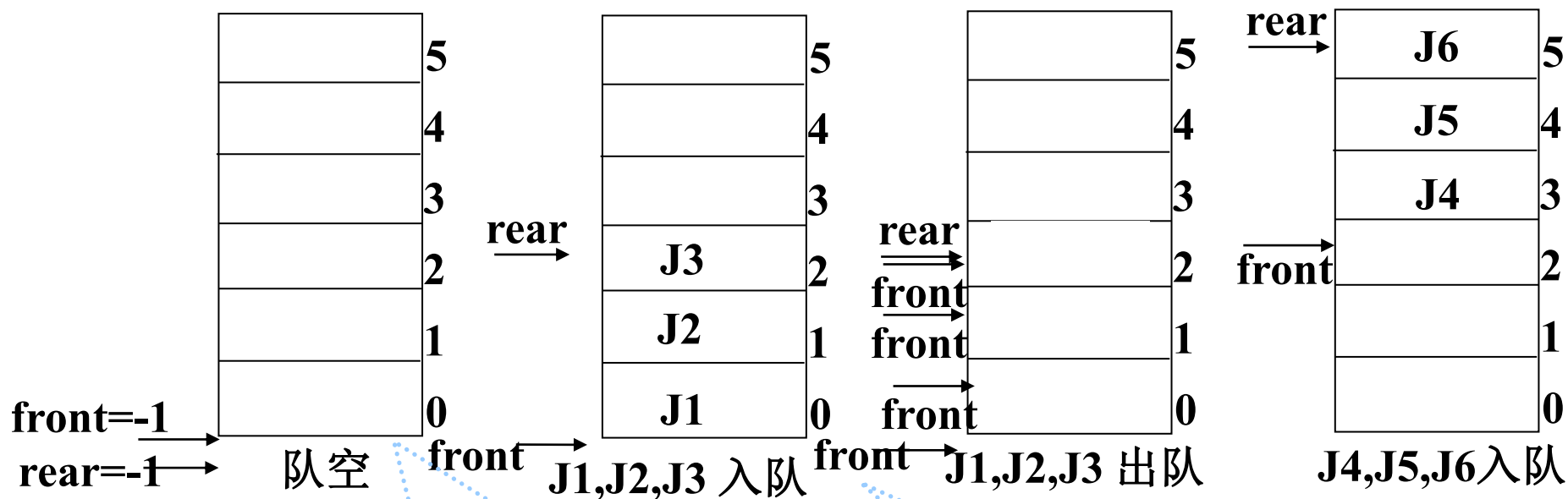
```
QItem DeleteQueue(Queue Q)
{
    qlink p; QItem x;
    if(QueueEmpty(Q)) Error("Queue is empty");
    x= Q->front->element;
    p= Q->front;
    Q->front= Q->front->next;
    free(p);
    return x;
}
```

[返回章节目录](#)



4.3 用循环数组实现ADT队列

1、先考虑用一维数组sq[M]实现ADT队列



设两个指针 $front, rear$, 约定:
 $rear$ 指示队尾元素;
 $front$ 指示队头元素前一位置
 初值 $front = rear = -1$

空队列条件: $front == rear$
 入队列: $sq[++rear] = x$;
 出队列: $x = sq[++front]$;

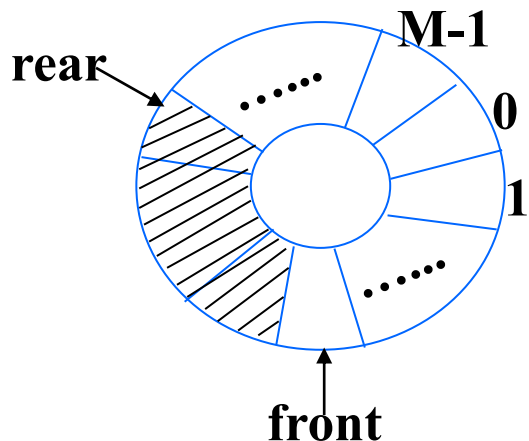


❖ 存在问题

- ❖ 设数组维数为 M ，则：
- ❖ 当 $front=-1, rear=M-1$ 时，再有元素入队发生溢出——**溢出**
- ❖ 当 $front \neq -1, rear=M-1$ 时，再有元素入队发生溢出——**假溢出**

❖ 解决方案

- ❖ 队首固定，每次出队剩余元素向下移动——浪费时间
- ❖ 循环队列
- ❖ 基本思想：把队列**设想成环形**，让 $sq[0]$ 接在 $sq[M-1]$ 之后，若 $rear+1==M$ ，则令 $rear=0$ ；

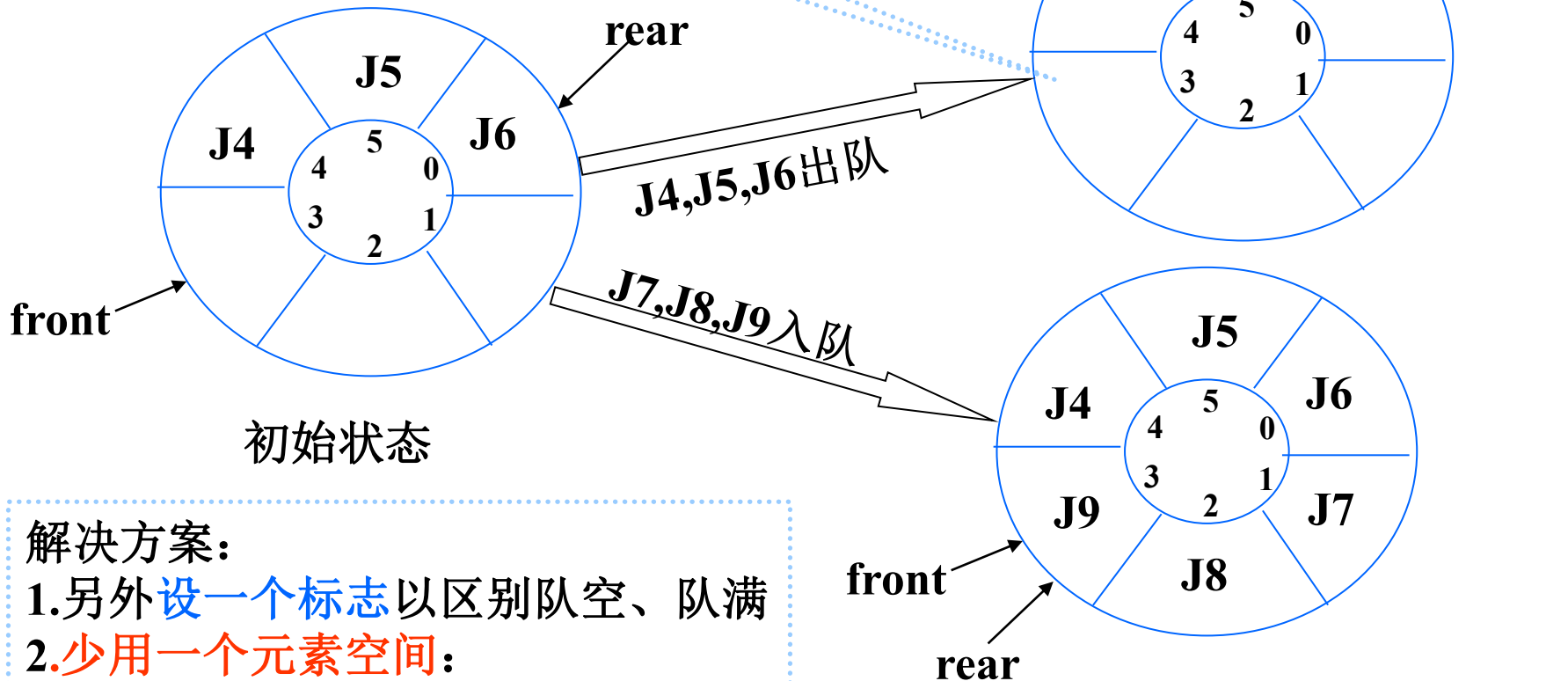


- ❖ 实现：利用“模”运算
- ❖ 入队： $rear=(rear+1)\%M$; $sq[rear]=x$;
- ❖ 出队： $front=(front+1)\%M$; $x=sq[front]$;
- ❖ 队满、队空判定条件



队空: $\text{front} == \text{rear}$

队满: $\text{front} == \text{rear}$



解决方案:

1. 另外设一个标志以区别队空、队满

2. 少用一个元素空间:

队空: $\text{front} == \text{rear}$

队满: $(\text{rear} + 1) \% M == \text{front}$

2、用循环数组实现队列

❖ 用循环数组实现的队列的特征数据及其类型

- 队列元素的类型: **QItem**
- 循环数组的规模: **MaxSize**
- 存放队列的循环数组: **QItem *queue**; 一个分量放一个元素; 约定沿着队列从首到尾的走向是顺时针的。
- 指示队首元素的前一个位置的下标: **front**;
- 指示队尾元素位置的下标: **rear**;
- 约定: **front = rear** 时为空队列,
 $(rear + 1) \% MaxSize = front$ 时为满队列。

❖ 用循环数组实现的队列Queue的定义

```
typedef struct aqueue *Queue;
typedef struct aqueue{
    int maxsize; //循环数组大小
    int front; //队首游标
    int rear; //队尾游标
    QItem queue; //循环数组
}Aqueue;
```

✿ 入队算法:

```
void EnterQueue (QItem x, Queue Q)
{
    if (QueueFull(Q)) Error("Queue is full");
    Q->rear = (Q->rear + 1) % Q->maxSize;
    Q->queue[Q->rear] = x;
}
```

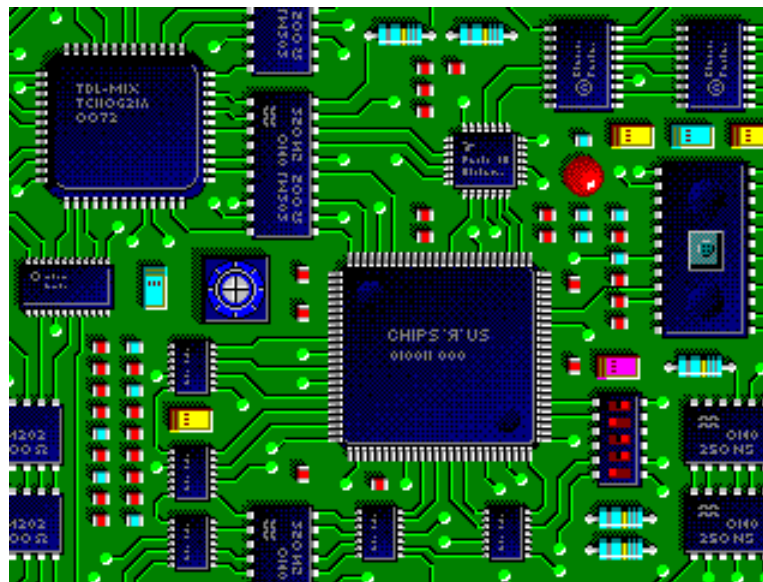
✿ 出队算法:

```
QItem DeleteQueue(Queue Q)
{
    if(QueueEmpty(Q)) Error("Queue is empty");
    Q->front = (Q->front + 1) % Q->maxSize;
    return Q->queue[Q->front];
}
```

[返回目录](#)

4.4 ADT队列的应用

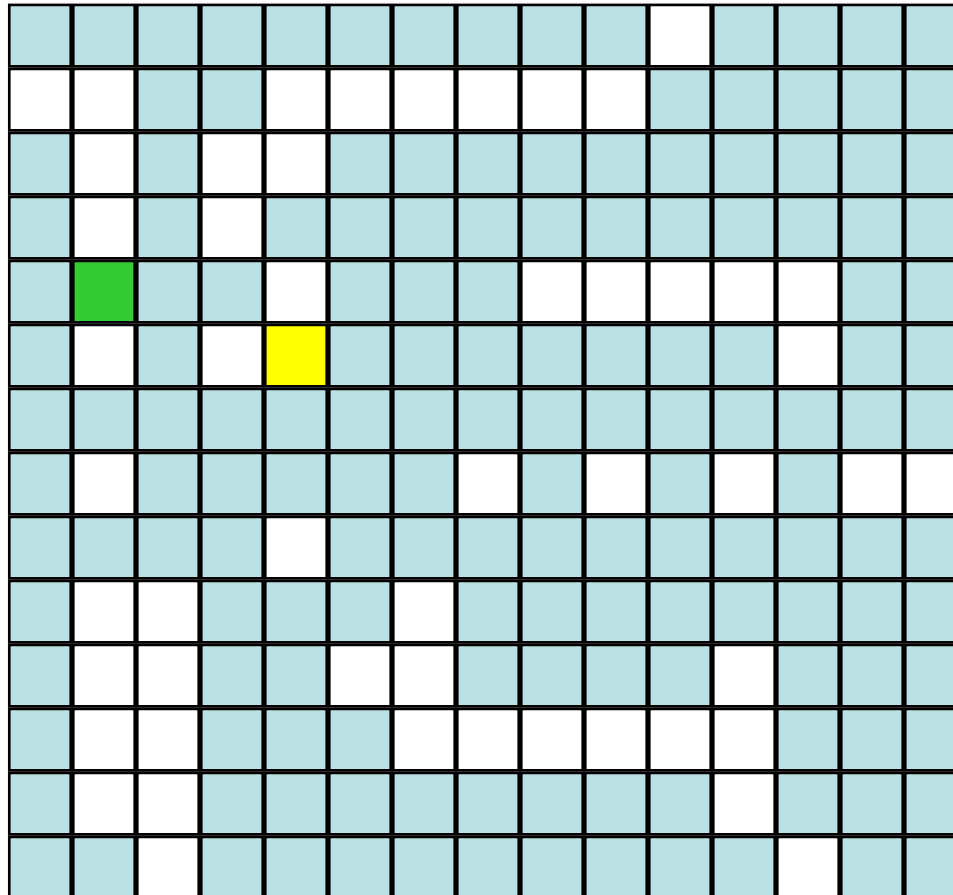
—电路布线问题



电路布线问题


 start pin

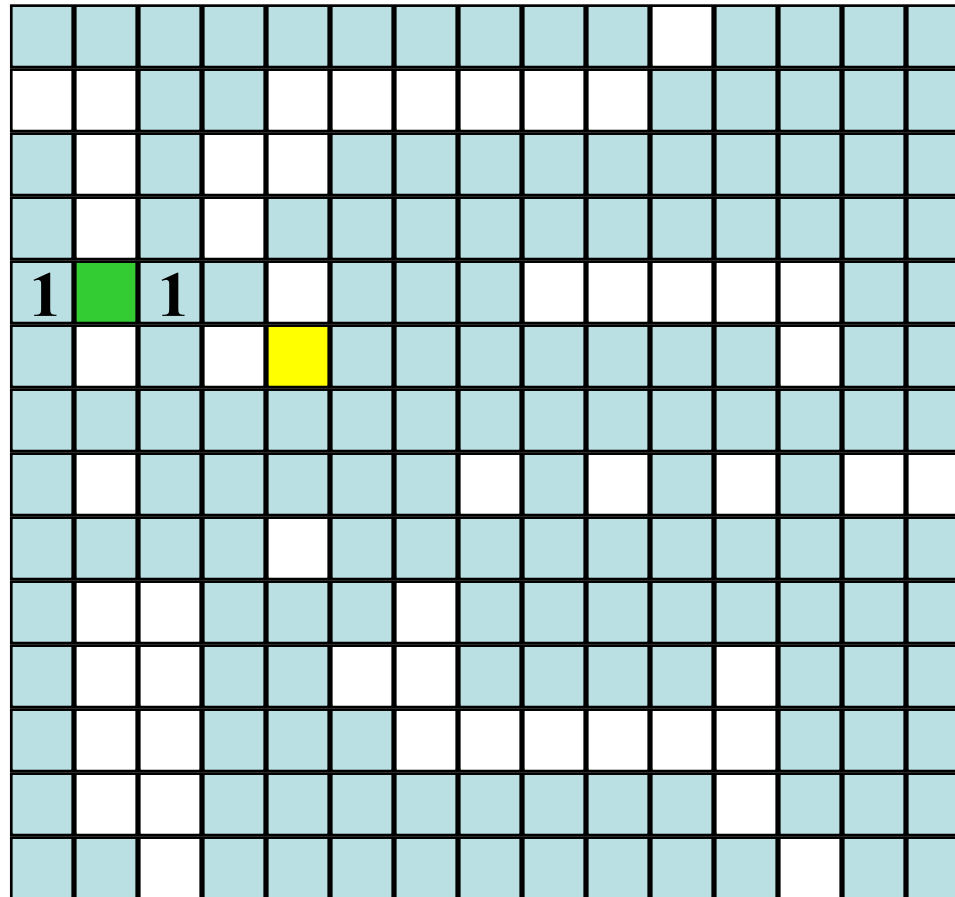
 end pin



Label all reachable squares **1** unit from start.

电路布线问题

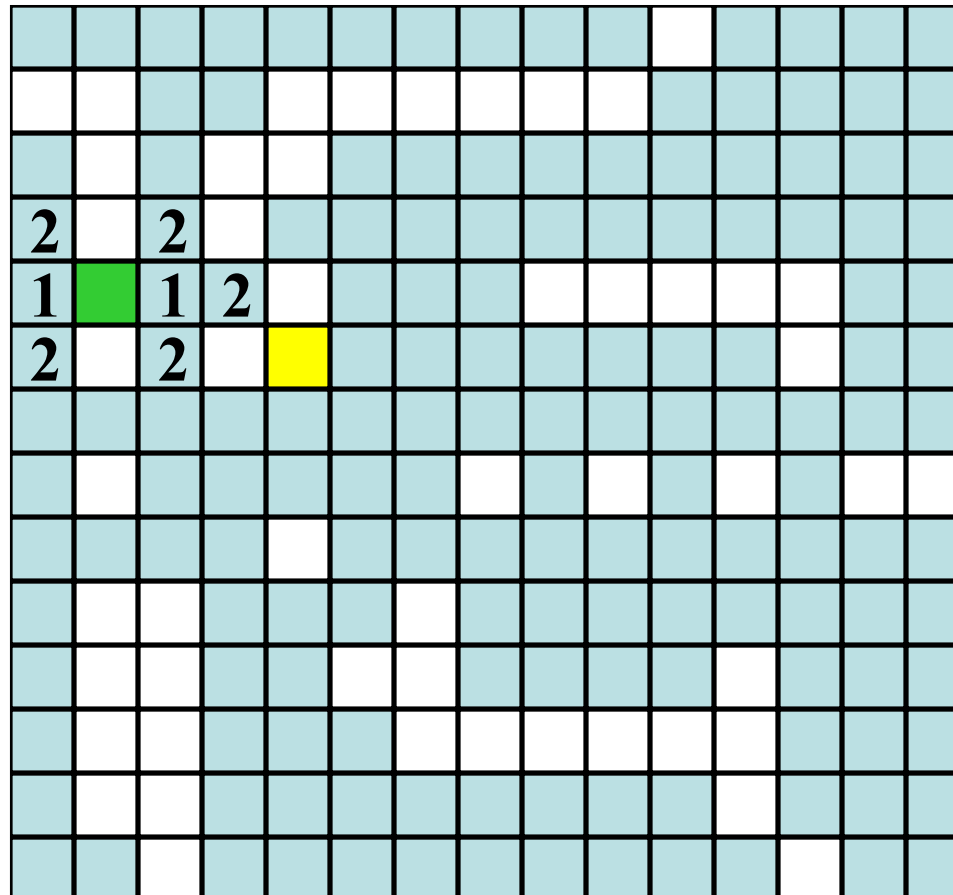
 start pin
 end pin



Label all reachable unlabeled squares **2** units from start.

电路布线问题

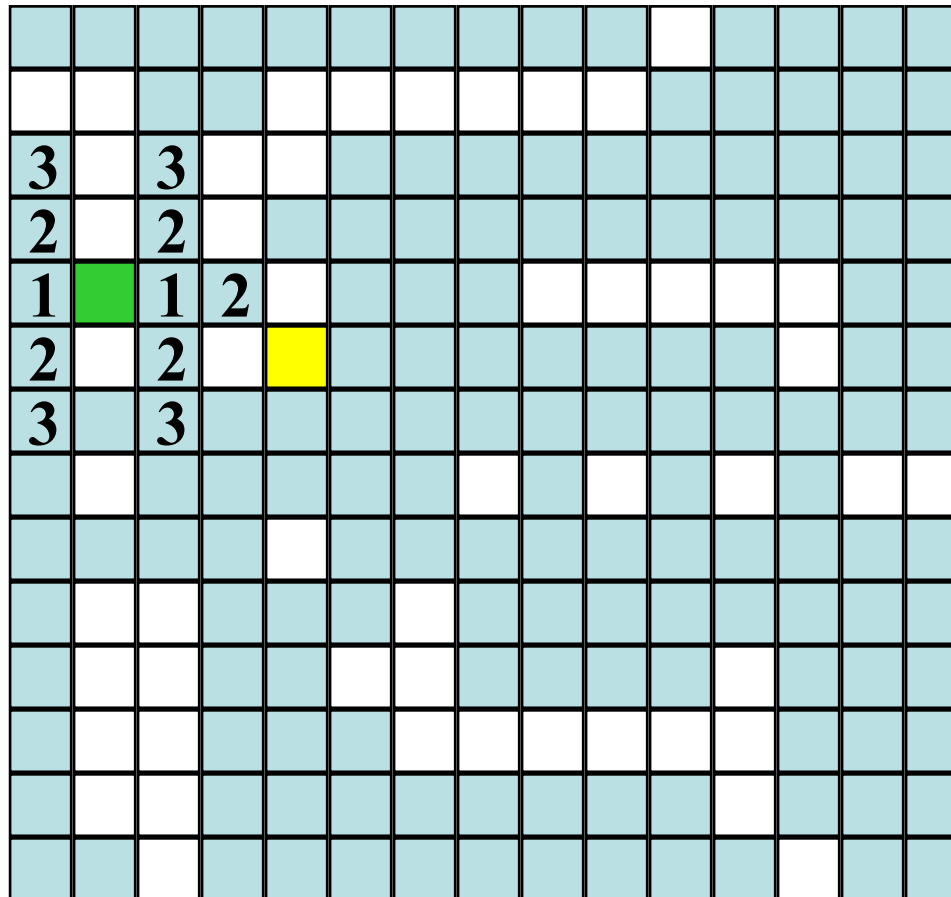
 start pin
 end pin



Label all reachable unlabeled squares **3** units from start.

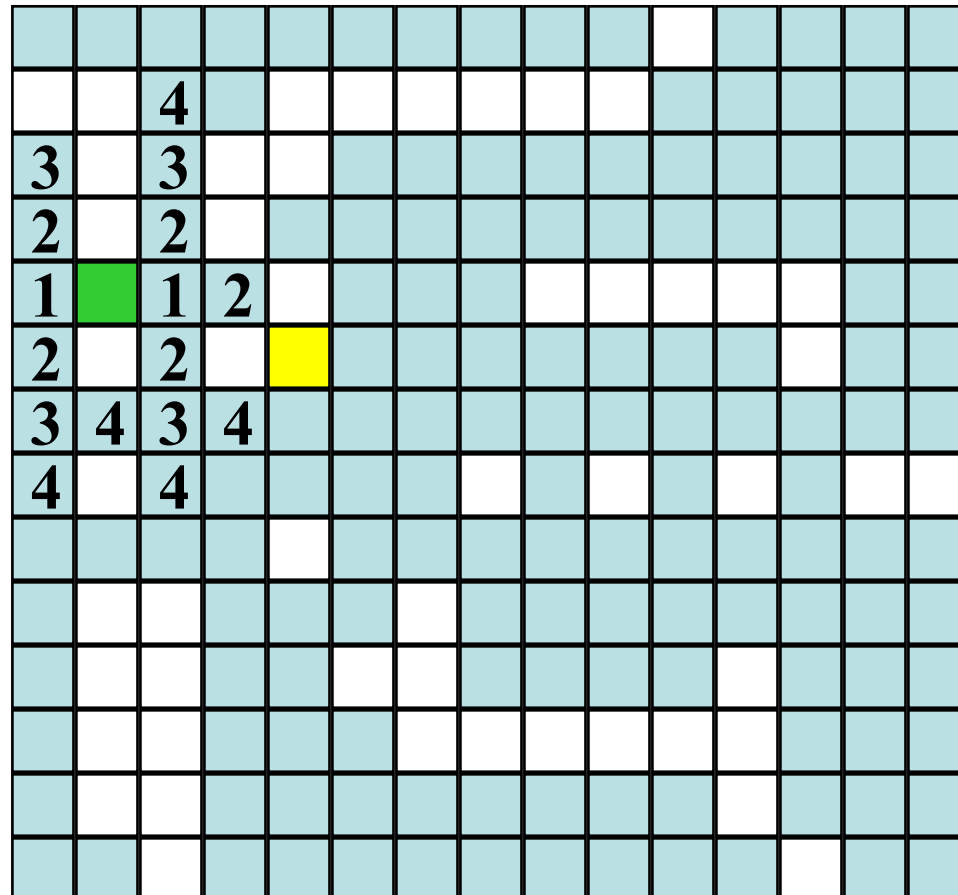
电路布线问题

 start pin
 end pin



电路布线问题

 start pin
 end pin

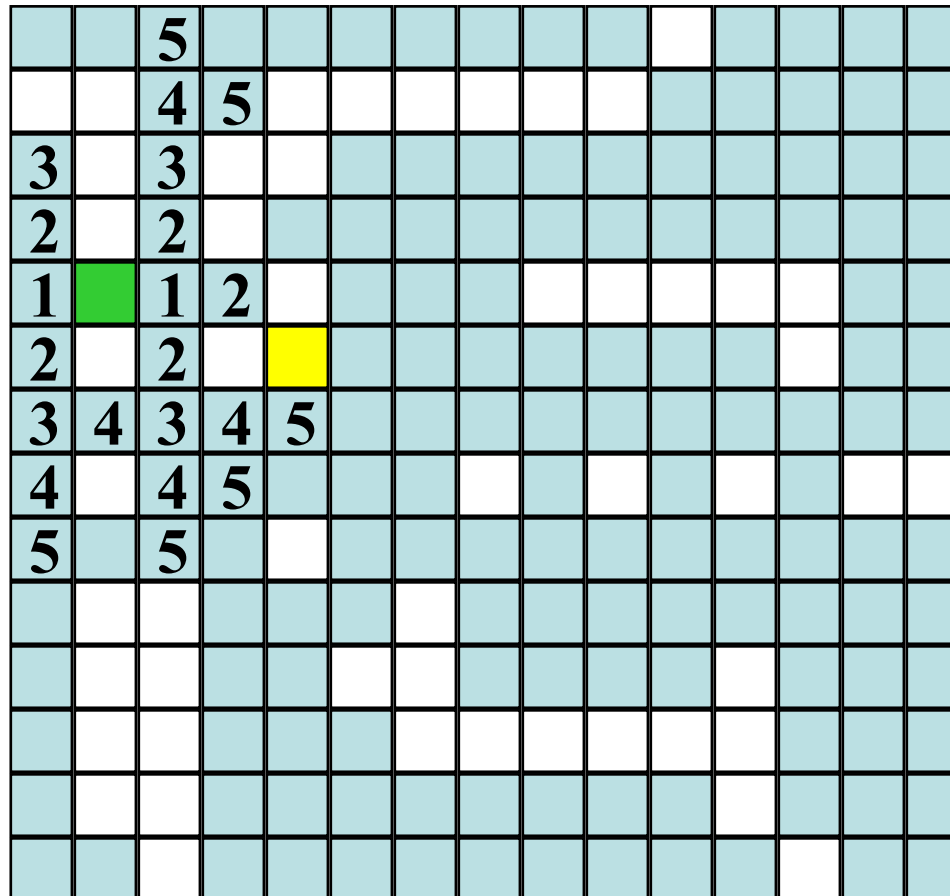


Label all reachable unlabeled squares 5 units from start.

电路布线问题

 start pin



 end pin



Label all reachable unlabeled squares **6** units from start.

电路布线问题

 start pin
 end pin

	6	5	6											
		4	5											
3		3												
2		2												
1		1	2											
2		2			6									
3	4	3	4	5	6									
4		4	5	6										
5	6	5	6											
6														

End pin reached. Traceback.

电路布线问题

 start pin
 end pin

	6	5	6										
		4	5										
3		3											
2		2											
1		1	2										
2		2		6									
3	4	3	4	5	6								
4		4	5	6									
5	6	5	6										
6													

End pin reached. Traceback.

[返回章节目录](#)



4.5 队列其它应用举例

划分子集问题

问题描述：已知集合 $A=\{a_1, a_2, \dots, a_n\}$ ，及集合上的关系

$R=\{ (a_i, a_j) \mid a_i, a_j \in A, i \neq j\}$ ，其中 (a_i, a_j) 表示 a_i 与 a_j 间存在冲突关系。

要求将 A 划分成互不相交的子集 $A_1, A_2, \dots, A_k, (k \leq n)$ ，使任何子集中的元素均无冲突关系，同时要求分子集个数尽可能少

例 $A=\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$R=\{ (2, 8), (9, 4), (2, 9), (2, 1), (2, 5), (6, 2), (5, 9),$
 $(5, 6), (5, 4), (7, 5), (7, 6), (3, 7), (6, 3) \}$

可行的子集划分为：

$A_1=\{ 1, 3, 4, 8 \}$

$A_2=\{ 2, 7 \}$

$A_3=\{ 5 \}$

$A_4=\{ 6, 9 \}$

	1							
1				1	1		1	1
					1	1		
				1				
	1		1		1	1		
	1	1		1		1		1
		1		1				1
	1							
	1				1	1		

❖ 队列其它应用举例（续）

- ❖ 图元识别问题
- ❖ 离散时间模拟
- ❖ 车皮排序问题
- ❖ 图的广度优先遍历
- ❖ 基数排序

[返回章节目录](#)

THE END