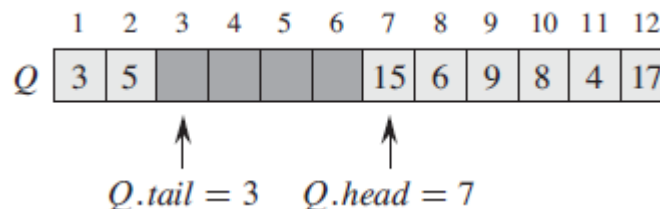


- 作业讲解
 - TC第10.1节练习4、5、6
 - TC第10.2节练习1、2、3、6
 - TC第10.3节练习4、5
 - TC第10.4节练习2、3、4
 - TC第10章问题3

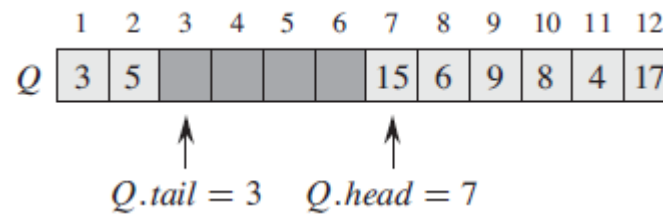
TC第10.1节练习4

- Rewrite ENQUEUE to detect overflow.
 - if ($Q[Q.tail] \neq \text{null}$) ... 对不对?
 - if ($Q.tail == Q.head$) ... 对不对?
 - 不能区分队列是满还是空
- 总是预留一个空位置
 - if ($Q.tail \% Q.length + 1 == Q.head$) overflow
 - if ($(Q.tail + 1) \% Q.length == Q.head$) overflow 对不对?
 - if ($Q.head == Q.tail$) underflow



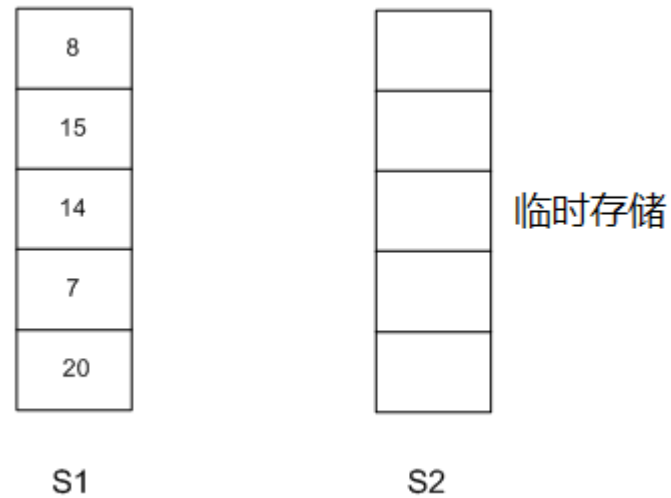
TC第10.1节练习5

- `deque_from_tail`
 - ... `x = Q[Q.tail]`; `Q.tail--`; ... 对不对?

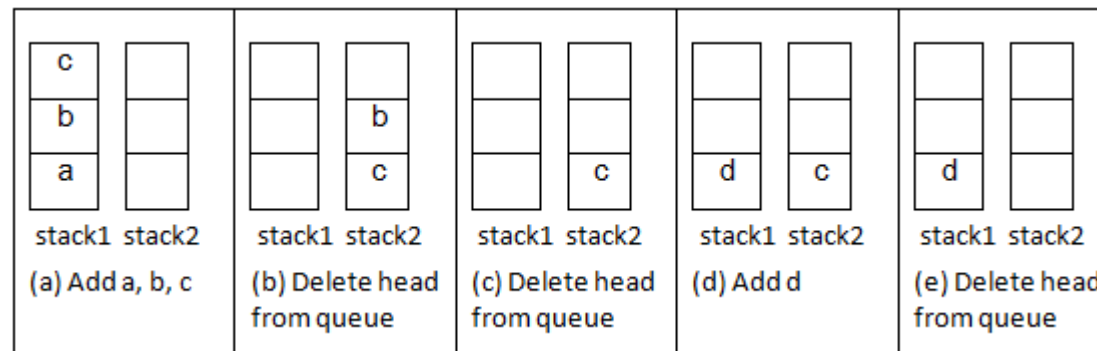


TC第10.1节练习6

- 方法1



- 方法2



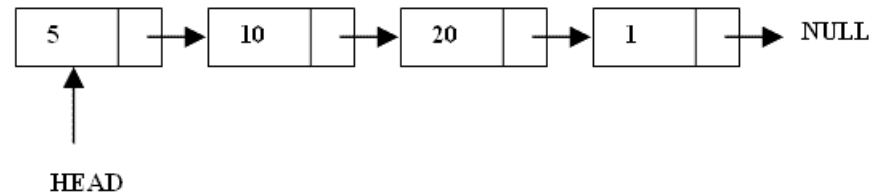
TC第10.2节练习1

- DELETE

```
p = L.head;  
while (p.next != x) {  
    p = p.next;  
}
```

...

对不对？



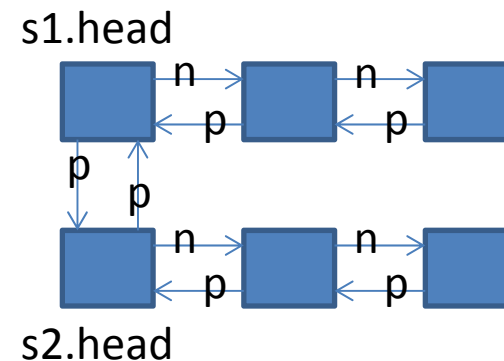
```
p = L.head;  
while (p!=x && p!=null) {  
    p = p.next;  
}
```

...

TC第10.2节练习6

- Support UNION in $O(1)$ time using a suitable list data structure.

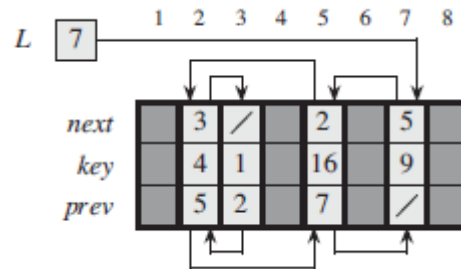
```
s1.head.prev = s2.head;  
s2.head.prev = s1.head;  
s = s1.head;  
return s;  
对不对?
```



```
s.head = s1.head;  
s1.tail.next = s2.head;  
s.tail = s2.tail;
```

TC第10.3节练习4、5

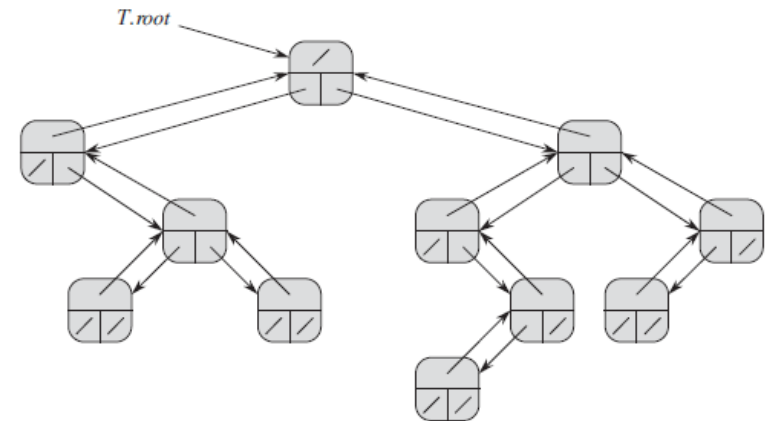
- Using the first m index locations in the multiple-array representation
Hint: Use the array implementation of a stack.
 - 插入：分配第 $m+1$ 个位置
 - 删除：如果删除的不是第 m 个位置，与第 m 个位置交换
- COMPACTIFY-LIST：搜索和移位



TC第10.4节练习3、4

- Nonrecursive traversal, using a stack

```
push(root);  
while(stack is not empty) {  
    curr = pop();  
    print(curr);  
    if (curr.left != null) push(curr.left);  
    if (curr.right != null) push(curr.right);  
}
```



loop invariant是什么？

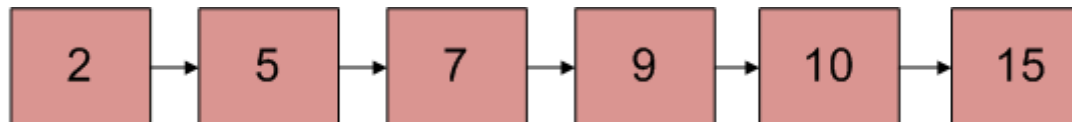
Arbitrary rooted tree, using the left-child, right-sibling representation:

与binary tree一样处理

TC第10章问题3

- CLS: (跳-)走-(跳-)走-.....
- CLS': (跳-)(跳-).....-走-走-.....
- CLS的总里程 = 最后一次成功的跳 + 之后所有的走(\leq while执行次数)
- (a) CLS执行t次while之后, 有三种结果
 - CLS没找到 (Line 10): CLS' 执行t次for、 $\leq t$ 次while
 - CLS走到了 (Line 11): CLS' 执行t次for、 $\leq t$ 次while
 - CLS跳到了 (Line 7): CLS' 执行t次for
- (b) $E = E(\text{for} + \text{while}) = E(\text{for}) + E(\text{while}) = O(t) + E(X_t) = O(t + E(X_t))$
- (c)
$$E[X_t] = \sum_{r=0}^d rP(X_t = r) = \sum_{r=1}^d P(X_t \geq r) \leq \sum_{r=1}^n P(X_t \geq r) \leq \sum_{r=1}^n \left(1 - \frac{r}{n}\right)^t$$

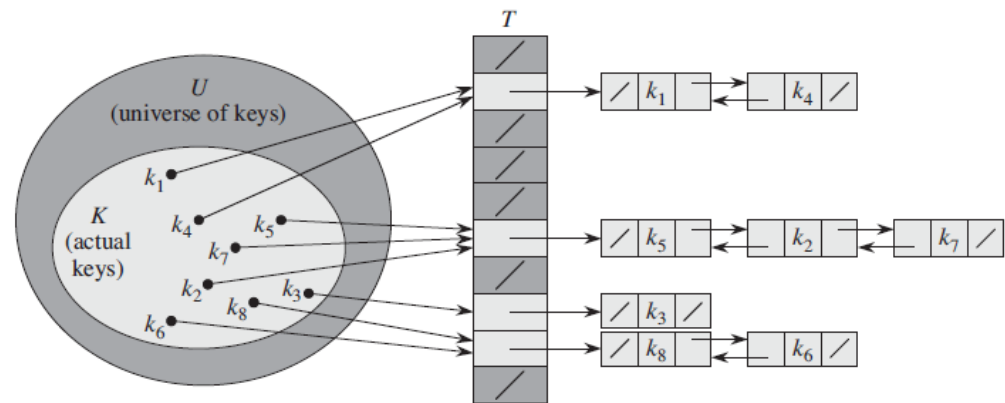
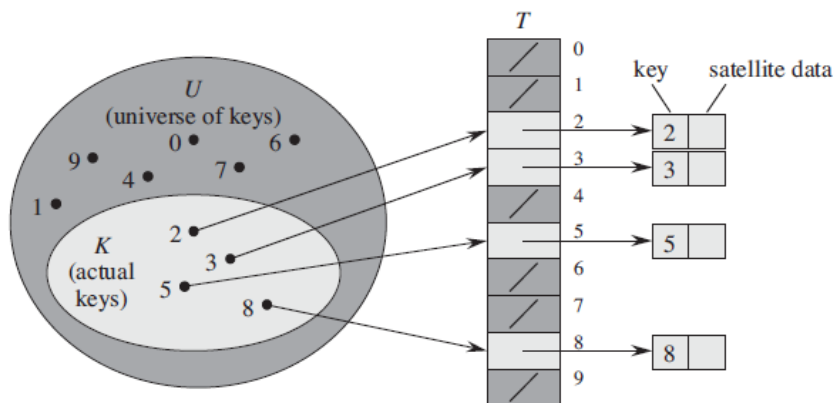
(C.25)



- 教材讨论
 - TC第11章
 - CS第5章第5节

问题1: dictionary

- 什么是dictionary?
- 你如何理解它的两种实现?
 - direct-address table
 - hash table
- 你能分析它们的存储空间和插入/删除/查找时间吗?
- 因此, 你能对比它们的优缺点吗?



问题1: dictionary (续)

- 你理解这段话了吗？

In a hash table in which collisions are resolved by chaining, an unsuccessful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing.

In a hash table in which collisions are resolved by chaining, a successful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing.

What does this analysis mean? If the number of hash-table slots is at least proportional to the number of elements in the table, we have $n = O(m)$ and, consequently, $\alpha = n/m = O(m)/m = O(1)$. Thus, searching takes constant time on average. Since insertion takes $O(1)$ worst-case time and deletion takes $O(1)$ worst-case time when the lists are doubly linked, we can support all dictionary operations in $O(1)$ time on average.

- 如何做到那个 “if” ？

问题1: dictionary (续)

```
void addEntry(int hash, K key, V value, int bucketIndex) {  
    if ((size >= threshold) && (null != table[bucketIndex])) {  
        resize(2 * table.length);  
        hash = (null != key) ? hash(key) : 0;  
        bucketIndex = indexFor(hash, table.length);  
    }  
  
    createEntry(hash, key, value, bucketIndex);  
}
```

问题2: hash function

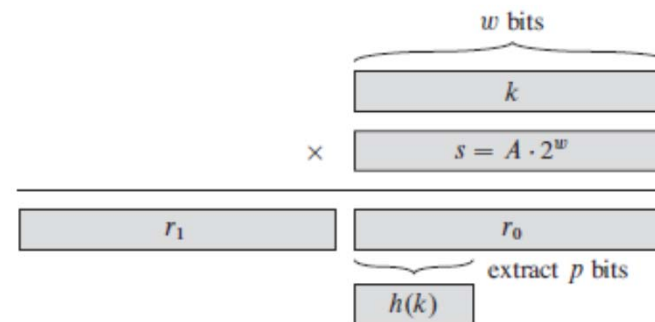
- 你如何理解一个好的hash function应有的这些要素？
 - Satisfies (approximately) the assumption of simple uniform hashing.
 - Derives the hash value in a way that we expect to be independent of any patterns that might exist in the data.
- 你如何理解simple uniform hashing？
它对hash table为什么至关重要？

问题2: hash function (续)

- 你理解这两种hash function了吗?

$$h(k) = k \bmod m$$

$$h(k) = \lfloor m (kA \bmod 1) \rfloor$$



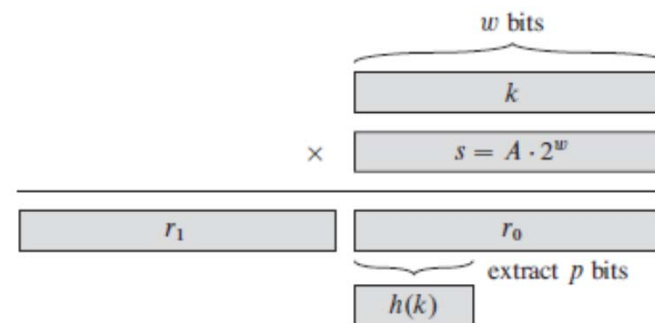
- 这些hash function在实际中能确保是simple uniform hashing吗?
如果不能, 怎么办?

问题2: hash function (续)

- 你理解这两种hash function了吗?

$$h(k) = k \bmod m$$

$$h(k) = \lfloor m (kA \bmod 1) \rfloor$$



- 这些hash function在实际中能确保是simple uniform hashing吗?
如果不能, 怎么办?
 - universal hashing: to choose the hash function randomly in a way that is independent of the keys that are actually going to be stored

问题3: probability calculations in hashing

- 你会计算这些期望值吗?

- expected number of items per location n/k

- expected number of empty locations $k(1 - \frac{1}{k})^n$

- expected number of collisions $n - k + k(1 - \frac{1}{k})^n$

- expected time until all locations have at least one item

$$\sum_{j=1}^k \frac{k}{k-j+1}$$

问题4: collision resolution

- 你理解open addressing了吗？
它与chaining的本质区别是什么？
因此，它有哪些相对的优缺点？

HASH-INSERT(T, k)

```
1   $i = 0$ 
2  repeat
3       $j = h(k, i)$ 
4      if  $T[j] == \text{NIL}$ 
5           $T[j] = k$ 
6          return  $j$ 
7      else  $i = i + 1$ 
8  until  $i == m$ 
9  error "hash table overflow"
```

HASH-SEARCH(T, k)

```
1   $i = 0$ 
2  repeat
3       $j = h(k, i)$ 
4      if  $T[j] == k$ 
5          return  $j$ 
6       $i = i + 1$ 
7  until  $T[j] == \text{NIL}$  or  $i == m$ 
8  return NIL
```

问题4: collision resolution (续)

- 一个好的h函数应该具有哪些特点?
- 你理解这些h函数了吗? 它们为什么不是最好的h函数?
 - linear probing $h(k, i) = (h'(k) + i) \bmod m$
 - quadratic probing $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$
 - double hashing $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$
- 你理解这些具体原因了吗?
 - linear probing: primary clustering
 - quadratic probing: secondary clustering

问题4: collision resolution (续)

- 一个好的h函数应该具有哪些特点?
 - The probe sequence is a permutation of $\langle 0, 1, \dots, m-1 \rangle$.
 - uniform hashing: The probe sequence of each key is equally likely to be any of the $m!$ permutations of $\langle 0, 1, \dots, m-1 \rangle$.
- 你理解这些h函数了吗? 它们为什么不是最好的h函数?
 - linear probing $h(k, i) = (h'(k) + i) \bmod m$
 - quadratic probing $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$
 - double hashing $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$
- 你理解这些具体原因了吗?
 - linear probing: primary clustering
 - quadratic probing: secondary clustering

问题4: collision resolution (续)

- 你理解perfect hashing了吗？
它与chaining的本质区别是什么？
因此，它有哪些相对的优缺点？

