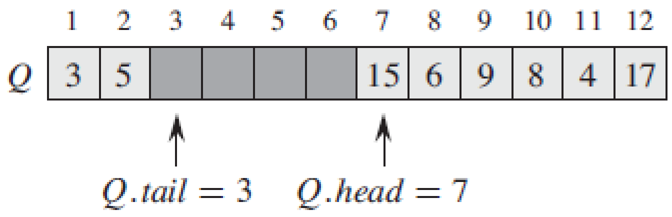
- 书面作业讲解
  - TC第10.1节练习4、5、6
  - TC第10.2节练习1、2、3、6
  - TC第10.3节练习4、5
  - TC第10.4节练习2、3、4
  - TC第10章问题3

TC第10.1节练习4

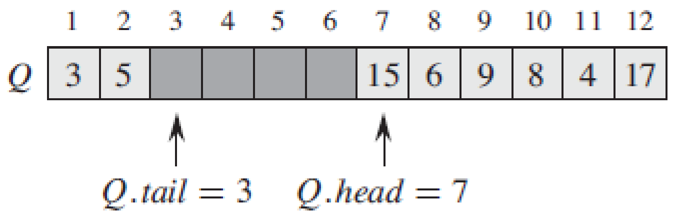
- Rewrite ENQUEUE to detect overflow.
  - if (Q[Q.tail] != null) … 对不对？
  - if (Q.tail == Q.head) … 有没有问题？
  - 不能区分队列是满还是空

- 总是预留一个空位置
  - if (Q.tail%Q.length+1 == Q.head) overflow
  - if (Q.head == Q.tail) underflow
  - if ((Q.tail+1)%Q.length == Q.head) overflow 对不对？



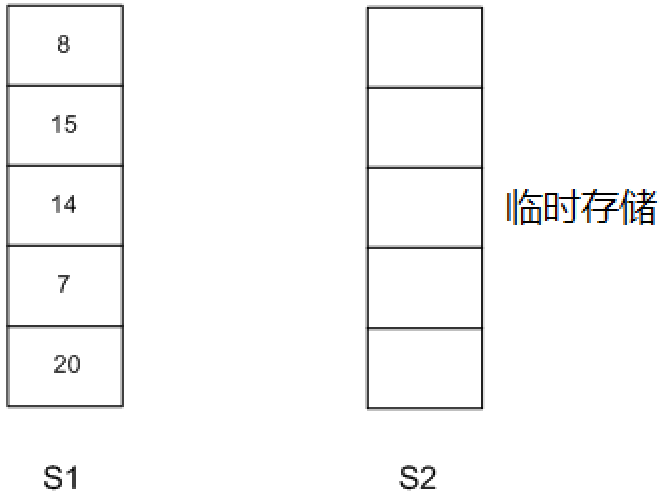
TC第10.1节练习5

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

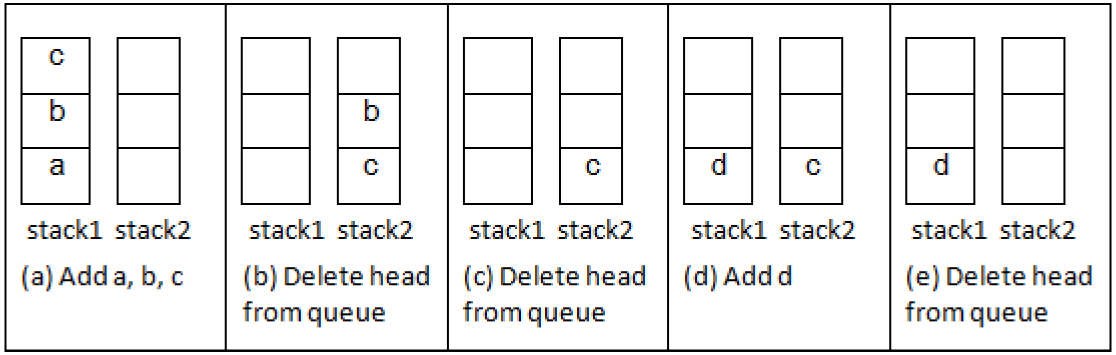


TC第10.1节练习6

- 方法1



- 方法2



(a) Add a, b, c

(b) Delete head from queue

(c) Delete head from queue

(d) Add d

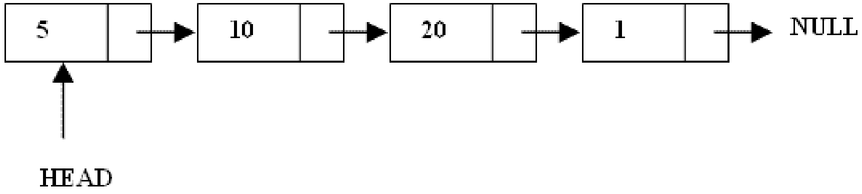
(e) Delete head from queue

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.

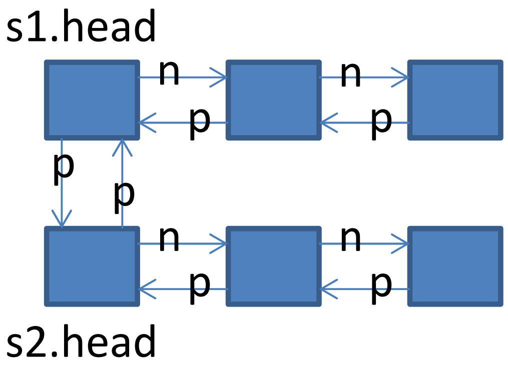
s1.head.prev = s2.head;

s2.head.prev = s1.head;

s = s1.head;

return s;

对不对？



s.head = s1.head;

s1.tail.next = s2.head;

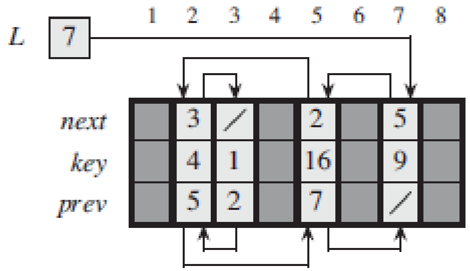
s.tail = s2.tail;

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

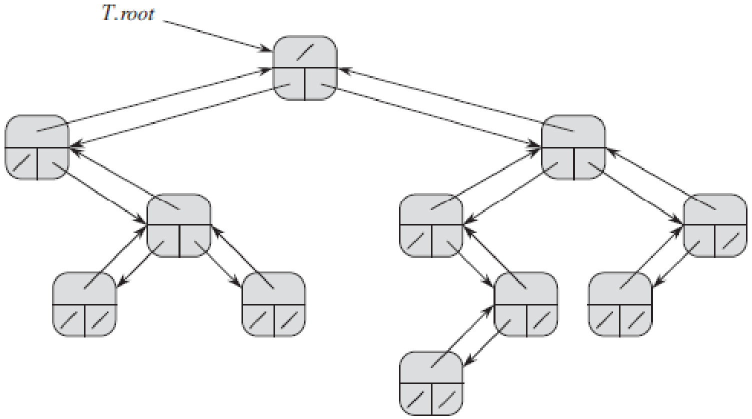
- COMPACTIFY-LIST：搜索和移位



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？

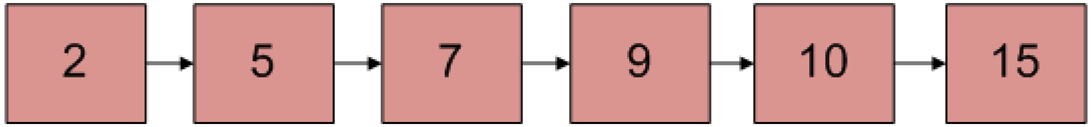
Arbitrary rooted tree, using the left-child, right-sibling representation：

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(≤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、≤t次while
  - CLS走到了 (Line 11)：CLS’执行t次for、≤t次while
  - CLS跳到了 (Line 7)：CLS’执行t次for



TC第10章问题3 (续)

- (b) E=E(for+while)=E(for)+E(while)=O(t)+E(X_t)=O(t+E(X_t))
- (c) $E\left[X_t\right]=\sum_{r=0}^{d} rP\left(X_t = r\right)=\sum_{r=1}^{d} P\left(X_t \geq r\right) \leq \sum_{r=1}^{n} P\left(X_t \geq r\right) \leq \sum_{r=1}^{n}\left(1-\frac{r}{n}\right)^t$

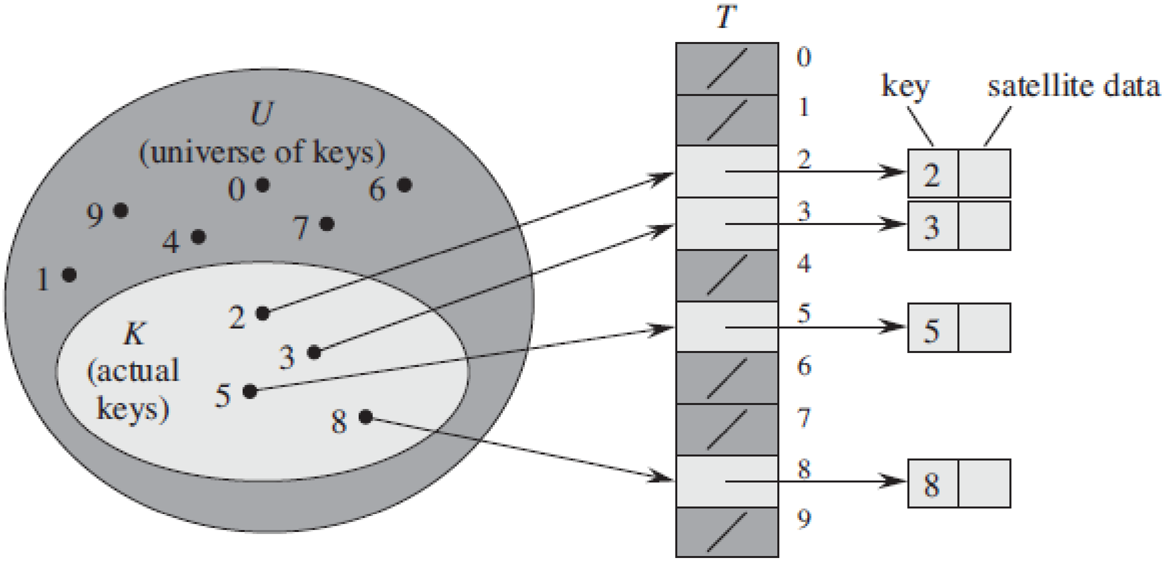
- 教材答疑和讨论
  - TC第11章
  - CS第5章第5节

问题1：dictionary

- dictionary是什么？它要求具备哪些操作？
  - Insert
  - Search
  - Delete
- 它有哪些用途？你能举出一些实际例子吗？

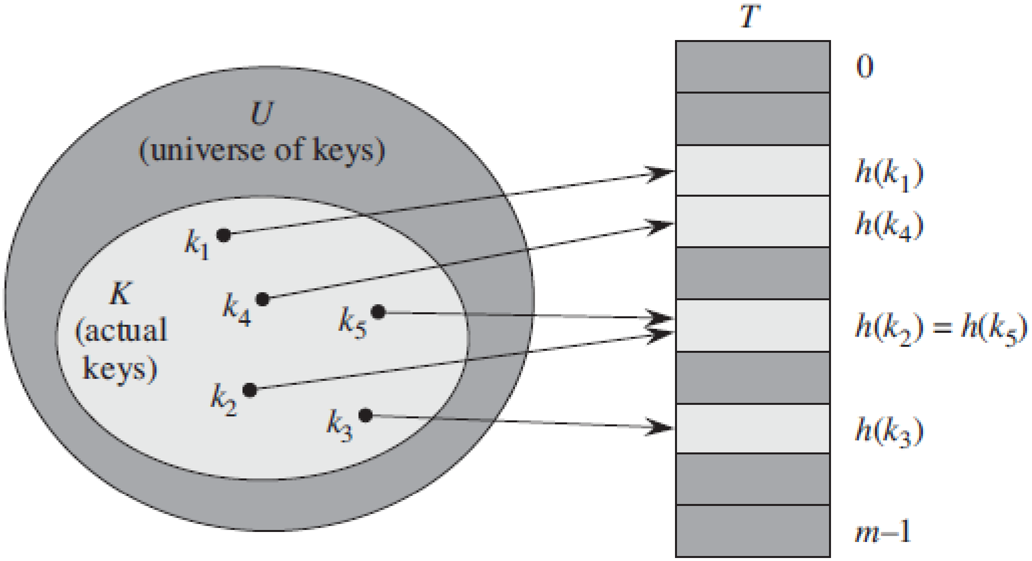
问题1：dictionary (续)

- direct-address table是如何实现dictionary的？
- 它有哪些优缺点？（时间、空间、实现难度）



问题1：dictionary (续)

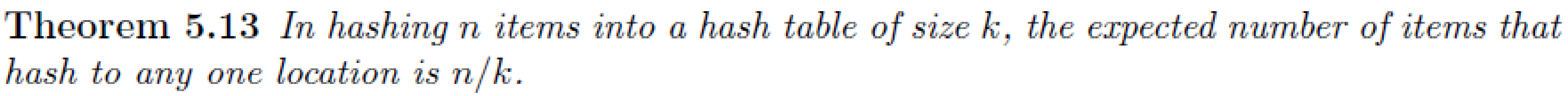
- hash table与direct-address table的本质区别是什么？
- 因此，它有哪些相对的优缺点？（时间、空间、实现难度）



- 以下我们只讨论simple uniform hashing

问题2：collision

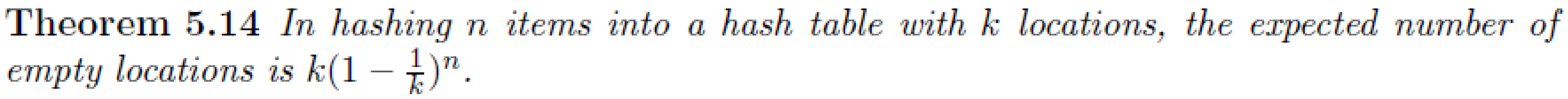
- expected number of items per location



Theorem 5.13 *In hashing n items into a hash table of size k, the expected number of items that hash to any one location is n/k.*

问题2： collision (续)

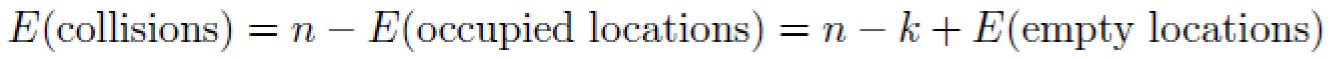
- expected number of empty locations



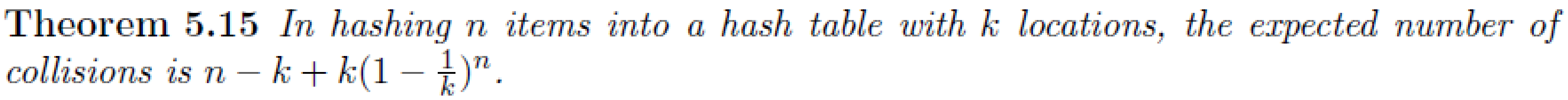
Theorem 5.14 *In hashing n items into a hash table with k locations, the expected number of empty locations is $k(1-\frac{1}{k})^n$.*

问题2：collision (续)

- expected number of collisions



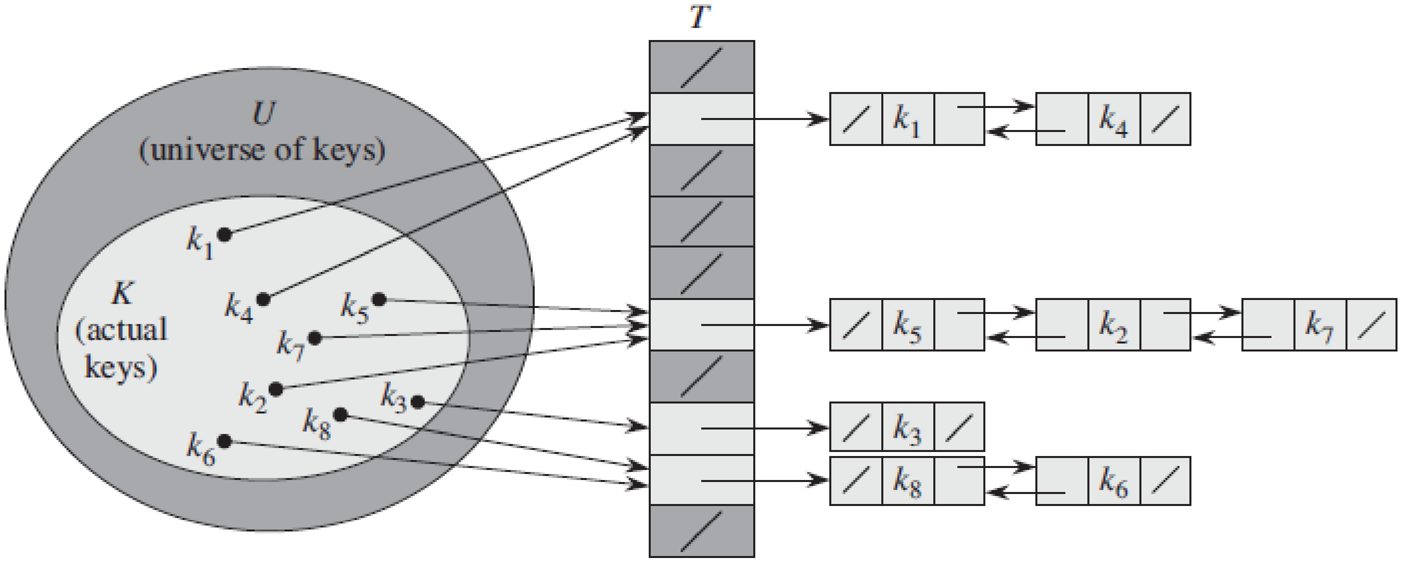
$$E(\text{collisions}) = n - E(\text{occupied locations}) = n - k + E(\text{empty locations})$$



Theorem 5.15 *In hashing n items into a hash table with k locations, the expected number of collisions is $n - k + k(1 - \frac{1}{k})^n$.*

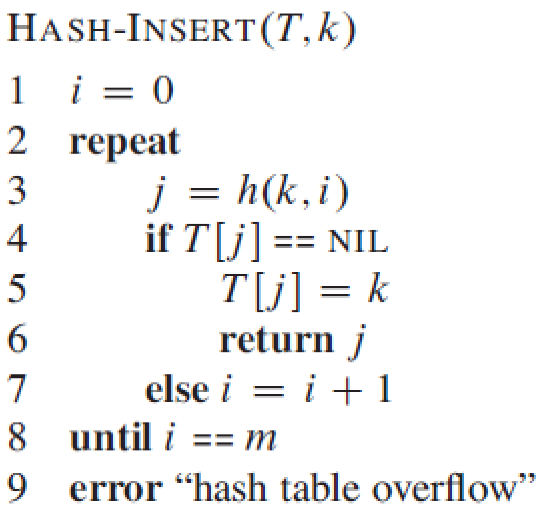
问题3：collision resolution

- chaining是如何解决collision的？
- insert、search、delect的运行时间分别是多少？
- 因此，它有哪些优缺点？（时间、空间、实现难度）

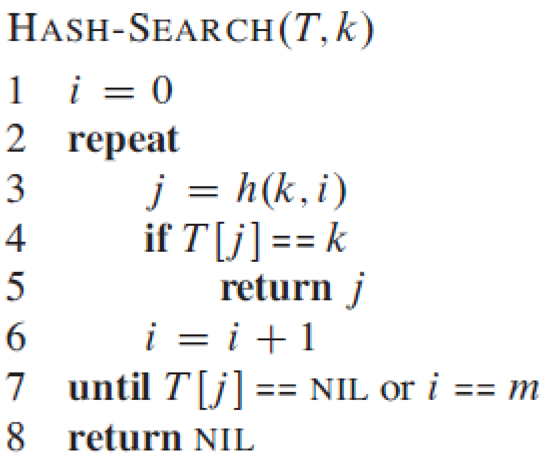


问题3：collision resolution (续)

- open addressing与chaining的本质区别是什么？
- 因此，它有哪些相对的优缺点？（时间、空间、实现难度）



```
HASH-INSERT$(T, k)$
$i = 0$
repeat
    $j = h(k, i)$
    if $T[j]$ == NIL
        $T[j] = k$
        return $j$
    else $i = i + 1$
until $i == m$
error "hash table overflow"
```



```
HASH-SEARCH$(T, k)$
$i = 0$
repeat
    $j = h(k, i)$
    if $T[j] == k$
        return $j$
    $i = i + 1$
until $T[j]$ == NIL or $i == m$
return NIL
```

问题3： collision resolution (续)

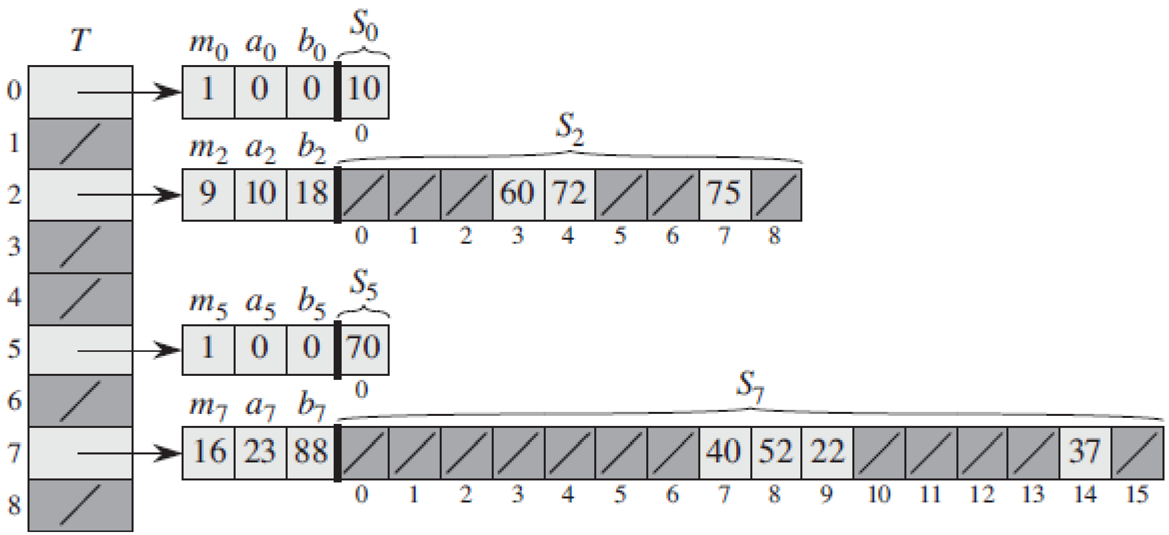
- open addressing的三种方法的基本思路分别是什么？
  - linear probing $h(k, i) = (h'(k) + i) \bmod m$
  - quadratic probing $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$
  - double hashing $h(k, i) = (h_1(k) + i h_2(k)) \bmod m$
- 它们在效果上有什么区别?

问题3：collision resolution (续)

- chaining和open addressing的运行时间主要取决于什么？
- 因此，当速度变得很慢时，你有什么对策？

问题3：collision resolution (续)

- perfect hashing与chaining的本质区别是什么？
- search的运行时间是多少？
- 因此，它有哪些相对的优缺点？（时间、空间、实现难度）



- 如果resolution是对collision的*治疗*,
- 那么如何尽可能*预防*collision呢？

问题4： hash function

- 你觉得一个好的hash function应该具有哪些特点？
  - Satisfies (approximately) the assumption of simple uniform hashing.
  - Depends on all the bits of the key.
  - Runs fast.
  - …

问题4：hash function (续)

- 如果有人跟你捣乱，构造出的key总是引发collision，你准备怎么应对？
  - universal hashing: to ***choose the hash function randomly*** in a way that is independent of the keys that are actually going to be stored