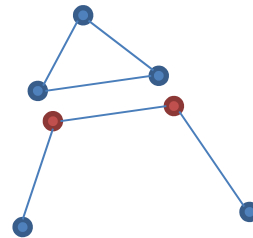
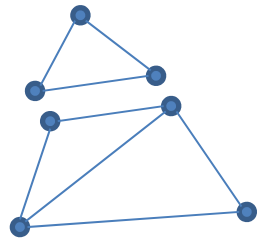


- 作业讲解
  - CS第4.1节问题16、17
  - CS第4.2节问题8、11、17
  - CS第4.3节问题9、13、16
  - CS第4.4节问题1、4、6
  - CS第4.5节问题8、9、10
  
  - CS第5.1节问题6、10、11、12、13
  - CS第5.2节问题2、9、10、14、15
  - CS第5.3节问题3、4、8、11、12、13
  - CS第5.4节问题5、6、8、10、17、20、21

# CS第4.1节问题16

- ... This version does not specify that the ears are nonadjacent. What happens if we try proving this by induction, using the same decomposition that we used in proving the Ear Lemma?



# CS第4.1节问题17

- ... relationship between the number of vertices in a polygon and the number of triangles in any triangulation of that polygon ...
  - 在数学归纳法中，使用top-down而非bottom-up的表述方式，即从要证明的结论开始
  - 例如：将任意 $n$ 边形拆分，而不是从任意 $n-1$ 边形拼接

# CS第4.2节问题8

- $T(n)=2T(n-1)+2000 \quad (n>1)$
- $T(1)=2000$

# CS第4.2节问题11

- 定理4.5

# CS第4.2节问题17

- 定理4.5
- 定理4.6

$$T(n) = r^n + \sum_{i=1}^n r^{n-i} i = r^n + r^n \sum_{i=1}^n i \left(\frac{1}{r}\right)^i = \dots$$

# CS第4.4节问题1

- 定理4.11（主定理的扩展形式）

# CS第4.5节问题8

- 错误1
  - 欲证 $T(n) \leq cn^3$
  - 归纳假设 $T(n/2) \leq c(n/2)^3$
  - 计算 $T(n) \leq cn^3 + n \lg n = O(n^3) + O(n^2) = O(n^3)$ , 得证
- 错误2
  - 欲证 $T(n) \leq cn^3 - d n \lg n$
  - 归纳假设 $T(n/2) \leq c(n/2)^3 - d(n/2) \lg(n/2)$
  - 计算 $T(n) \leq \dots \leq cn^3$ , 得证
- 错误3
  - 欲证 $T(n) \leq c_1 n^3$
  - 归纳假设 $T(n/2) \leq c_1 (n/2)^3$
  - 计算 $T(n) \leq c_1 n^3 + n \lg n \leq c_1 n^3 + c_2 n^3 = c_3 n^3$ , 得证
- 错误4
  - 欲证 $T(n) \leq cn^3 - d$
  - 归纳假设 $T(n/2) \leq c(n/2)^3 - d$
  - 计算 $T(n) \leq cn^3 - d - 7d + n \lg n$ , 只需取 $d \geq n \lg n / 7$ , 得证



# CS第4.5节问题8 (续)

- 一种证法
  - 欲证 $T(n) \leq c(n^3 - n^2) + d$
  - 归纳假设 $T(n/2) \leq c((n/2)^3 - (n/2)^2) + d$
  - 计算 $T(n) \leq cn^3 - 2cn^2 + 8d + n \lg n = [c(n^3 - n^2) + d] + (n \lg n + 7d - cn^2)$   
 $\leq [c(n^3 - n^2) + d] + (n^2 + 7d - cn^2)$
  - 只要 $c \geq 7d + 1$ ,  $T(n) \leq [c(n^3 - n^2) + d] + (n^2 + 7d - (7d + 1)n^2)$   
 $= [c(n^3 - n^2) + d] + 7d(1 - n^2) \leq c(n^3 - n^2) + d$
  - 并且,  $T(1) = d \leq d$ 也成立

# CS第5.1节问题6

- 2 pennies (1 cent), 1 nickel (5 cents), 1 dime (10 cents)
- without replacement

$P_1P_2, P_2P_1$	$p(PP)=1/6$
$P_1D, P_2D$	$p(PD)=1/6$
$DP_1, DP_2$	$p(DP)=1/6$
$P_1N, P_2N$	$p(PN)=1/6$
$NP_1, NP_2$	$p(NP)=1/6$
ND	$p(ND)=1/12$
DN	$p(DN)=1/12$

# CS第5.1节问题10

- Probability that a five-card hand is straight
  - By using five-element sets as your model

$$\frac{9 \cdot 4^5}{\binom{52}{5}}$$

- By using five-element permutations as your model

$$\frac{9 \cdot (20 \cdot 16 \cdot 12 \cdot 8 \cdot 4)}{52^5} = \frac{9 \cdot 4^5 \cdot 5!}{\binom{52}{5} \cdot 5!} = \frac{9 \cdot 4^5}{\binom{52}{5}}$$

## CS第5.2节问题2

- Selected two from eight kings and queens. What is the probability that the king or queen of spades is among the cards selected?

$$- \frac{\binom{7}{1} + \binom{7}{1} - \binom{2}{2}}{\binom{8}{2}} = \frac{13}{28}$$

$$- 1 - \frac{\binom{6}{2}}{\binom{8}{2}} = \frac{13}{28}$$

# CS第5.2节问题9

- P270, 公式5.10

# CS第5.2节问题10

- P272, Theorem 5.4

# CS第5.2节问题14

- P266, Theorem 5.3

$$1 - \sum_{k=1}^n (-1)^{k+1} \binom{n}{k} \frac{(2n-1-k)! 2^k}{(2n-1)!} = \sum_{k=0}^n (-1)^k \binom{n}{k} \frac{(2n-1-k)! 2^k}{(2n-1)!}$$

# CS第5.2节问题15



- P272, principle of inclusion and exclusion for counting

$$\begin{aligned} N_a(\emptyset) &- \sum_{k=1}^m (-1)^{k+1} \sum_{\substack{i_1, i_2, \dots, i_k \\ 1 \leq i_1 < i_2 < \dots < i_k \leq m}} |E_{i_1} \cap E_{i_2} \cap \dots \cap E_{i_k}| \\ &= N_a(\emptyset) - \sum_{\substack{K \subseteq P \\ K \neq \emptyset}} (-1)^{|K|+1} N_a(K) \\ &= \sum_{K \subseteq P} (-1)^{|K|} N_a(K) \end{aligned}$$

- How this formula could be used to compute the number of onto functions.
  - object  $\rightarrow$  function
  - property  $\rightarrow$  location
  - object has a property  $\rightarrow$  function maps nothing to a location
  - #onto\_function  $\rightarrow N_e(\emptyset)$
  - $N_a(K) \rightarrow (m - |K|)^n$



# CS第5.3节问题3、4

- 怎么证明两个event相互独立?
  - $E \cap F = \emptyset$  
  - $P(E) = P(F)$  
  - 定义:  $P(E|F) = P(E)$
  - 定理5.5:  $P(E)P(F) = P(E \cap F)$

# CS第5.3节问题12

- The probability that the family has two girls, given that **one of the children** is a girl
  - $(1/4)/(3/4)=1/3$

# CS第5.3节问题13

- Monty Hall problem
  - [http://en.wikipedia.org/wiki/Monty\\_Hall\\_problem](http://en.wikipedia.org/wiki/Monty_Hall_problem)
  - $P(\text{switch and win})=2/3$
  - $P(\text{not switch and win})=1/3$

# CS第5.4节问题6

- Expected sum of the tops of n dice
  - $E(X_1 + \dots + X_n) = E(X_1) + \dots + E(X_n) = 3.5n$

# CS第5.4节问题8

- choose 26 cards from 52
- Is the event of having a king on the  $i$ -th draw independent of the event of having a king on the  $j$ -th draw?
  - $P(i)=P(j)=1/13$
  - $\frac{A_4^2}{A_{52}^2} = \frac{1}{13 \cdot 17} \neq P(i) \cdot P(j)$
- How many kings do you expect to see?
  - 思路1:  $E(A)=E(2)=\dots=E(K)$ 且 $\sum E(x)=26 \Rightarrow E(K)=2$
  - 思路2:  $P(x)=26/52=1/2 \Rightarrow E(K)=E(K_{\text{黑}})+E(K_{\text{红}})+E(K_{\text{梅}})+E(K_{\text{方}})=4(1/2)=2$

# CS第5.4节问题10

- $E(c) = \sum X(s)P(s) = \sum cP(s) = c \sum P(s) = c$

# CS第5.4节问题21

- Give an example of a random variable ... with an infinite expected value ...
  - $P(F^i S) = (1-p)^i p \Rightarrow E(X) = \sum (1-p)^i p X(F^i S) = \infty$
  - 例如:  $X(F^i S) = (1-p)^{-i}$

- 教材讨论
  - TC第5章
  - CS第5章第6、7节
  
  - TC第7、8、9章



# 问题1: randomized algorithm

- 什么样的算法可以称作randomized algorithm?
- 什么叫做randomized algorithm的expected running time?
- 它和average-case running time有什么异同?

# 问题1: randomized algorithm

- 什么样的算法可以称作randomized algorithm?
  - Its behavior is determined not only by its input but also by something chosen randomly (e.g. values produced by a random-number generator).
- 什么叫做randomized algorithm的expected running time?
- 它和average-case running time有什么异同?

# 问题1: randomized algorithm

- 什么样的算法可以称作randomized algorithm?
  - Its behavior is determined not only by its input but also by something chosen randomly (e.g. values produced by a random-number generator).
- 什么叫做randomized algorithm的expected running time?
- 它和average-case running time有什么异同?
  - 异: We discuss the average-case running time when the probability distribution is over the inputs to the algorithm, and we discuss the expected running time when the algorithm itself makes random choices.

# 问题1: randomized algorithm (续)

- 你能想到哪些方法生成一个32-bit的（伪）随机数？

# 问题1: randomized algorithm (续)

- 你能想到哪些方法生成一个32-bit的（伪）随机数？
  - Computational methods (pseudo-random number generators)

```
m_w = <choose-initializer>; /* must not be zero */
m_z = <choose-initializer>; /* must not be zero */

uint get_random()
{
    m_z = 36969 * (m_z & 65535) + (m_z >> 16);
    m_w = 18000 * (m_w & 65535) + (m_w >> 16);
    return (m_z << 16) + m_w; /* 32-bit result */
}
```

- Physical methods
  - Coin flipping
  - Dice
  - Variations in the amplitude of atmospheric noise recorded with a normal radio

# 问题1: randomized algorithm (续)

- 你能想到哪些方法对一个数组中的元素随机排序?  
你如何评价这样一个方法的好坏?

# 问题1: randomized algorithm (续)

- 你能想到哪些方法对一个数组中的元素随机排序?  
你如何评价这样一个方法的好坏?

- PERMUTE-BY-SORTING( $A$ )

```
1  $n = A.length$ 
2 let  $P[1..n]$  be a new array
3 for  $i = 1$  to  $n$ 
4      $P[i] = \text{RANDOM}(1, n^3)$ 
5 sort  $A$ , using  $P$  as sort keys
```

- RANDOMIZE-IN-PLACE( $A$ )

```
1  $n = A.length$ 
2 for  $i = 1$  to  $n$ 
3     swap  $A[i]$  with  $A[\text{RANDOM}(i, n)]$ 
```

## 问题2: expected running time

- 目前为止，你掌握了哪些方式计算 $E(X)$ ?
- 你能用其中一种算出这个算法的expected running time吗?

**Exercise 5.6-4** Consider an algorithm that, given a list of  $n$  numbers, prints them all out. Then it picks a random integer between 1 and 3. If the number is 1 or 2, it stops. If the number is 3 it starts again from the beginning. What is the expected running time of this algorithm?



## 问题2: expected running time

- 目前为止，你掌握了哪些方式计算 $E(X)$ ?
  - $E(X) = \sum xP(X=x)$  // 定义
  - $E(X) = \sum E(X_i)$  // indicator random variable
  - $E(aX+bY) = aE(X) + bE(Y)$  // linearity of expectation
  - $E(X) = \sum E(X|F_i)P(F_i)$  // conditional expected value
- 你能用其中一种算出这个算法的expected running time吗?

*Exercise 5.6-4* Consider an algorithm that, given a list of  $n$  numbers, prints them all out. Then it picks a random integer between 1 and 3. If the number is 1 or 2, it stops. If the number is 3 it starts again from the beginning. What is the expected running time of this algorithm?

## 问题2: expected running time

- 目前为止，你掌握了哪些方式计算 $E(X)$ ?
  - $E(X) = \sum xP(X=x)$  // 定义
  - $E(X) = \sum E(X_i)$  // indicator random variable
  - $E(aX+bY) = aE(X) + bE(Y)$  // linearity of expectation
  - $E(X) = \sum E(X|F_i)P(F_i)$  // conditional expected value
- 你能用其中一种算出这个算法的expected running time吗?

**Exercise 5.6-4** Consider an algorithm that, given a list of  $n$  numbers, prints them all out. Then it picks a random integer between 1 and 3. If the number is 1 or 2, it stops. If the number is 3 it starts again from the beginning. What is the expected running time of this algorithm?

$$T(n) = \frac{2}{3}cn + \frac{1}{3}(cn + T(n))$$

## 问题2: expected running time (续)

- 你怎么理解indicator random variable?
- 在这些问题中, indicator random variable分别可以是什么?
  - The expected number of times that we hire a new office assistant.
  - The expected number of pairs of people with the same birthday.

## 问题2: expected running time (续)

- Suppose that you want to output 0 with probability  $1/2$  and 1 with probability  $1/2$ . At your disposal is a procedure `BIASED-RANDOM`, that outputs either 0 or 1. It outputs 1 with some probability  $p$  and 0 with probability  $1 - p$ , where  $0 < p < 1$ , but you do not know what  $p$  is. Give an algorithm that uses `BIASED-RANDOM` as a subroutine, and returns an unbiased answer, returning 0 with probability  $1/2$  and 1 with probability  $1/2$ . What is the expected running time of your algorithm as a function of  $p$ ?

## 问题2: expected running time (续)

- UNBIASED-RANDOM()  
  **Output:** 0 with probability 1/2 and 1 with probability 1/2  
  1 **while true do**  
  2      $a \leftarrow \text{BIASED-RANDOM}()$   
  3      $b \leftarrow \text{BIASED-RANDOM}()$   
  4     **if**  $a < b$  **then return** 0  
  5     **if**  $a > b$  **then return** 1

The algorithm calls BIASED-RANDOM twice to get two random numbers  $A$  and  $B$ . It repeats this until  $A \neq B$ . Then, depending on whether  $A < B$  (that is,  $A = 0$  and  $B = 1$ ) or  $A > B$  (that is,  $A = 1$  and  $B = 0$ ) it returns 0 or 1 respectively.

In any iteration, we have  $\Pr(A < B) = p(1 - p) = \Pr(B < A)$ , that is, the probability that the algorithm returns 0 in that iteration equals to the probability that it returns 1 in that iteration. Since with probability 1 we return something at some point (and not repeat the loop endlessly) and the probabilities of returning 0 and 1 are equal in each iteration, the total probabilities of returning 0 and 1 must be 1/2 and 1/2 respectively.

- 怎么计算expected running time?

## 问题2: expected running time (续)

- UNBIASED-RANDOM()  
  **Output:** 0 with probability  $1/2$  and 1 with probability  $1/2$   
  1 **while** *true* **do**  
  2   |  $a \leftarrow$  BIASED-RANDOM()  
  3   |  $b \leftarrow$  BIASED-RANDOM()  
  4   | **if**  $a < b$  **then** **return** 0  
  5   | **if**  $a > b$  **then** **return** 1
  
- 怎么计算expected running time?

The algorithm stops, if it either returns 0 or 1. In every iteration, the probability of this is  $\Pr(A \neq B) = \Pr(A < B) + \Pr(B < A) = 2p(1 - p)$ . Thus, we have a sequence of independent Bernoulli trials, each with probability  $2p(1 - p)$  of success. Therefore, the number of iterations required before the algorithm stops is geometrically distributed with parameter  $2p(1 - p)$ , and the expected number of iterations is  $1/(2p(1 - p))$ . As each iteration takes constant time (assuming that BIASED-RANDOM takes constant time), the expected running time of the algorithm is  $\Theta(1/(p(1 - p)))$ .

# 问题3: Quicksort

- 你能简述Quicksort的执行过程吗?
- PARTITION中的loop invariant是什么?
- 你能证明PARTITION是totally correct吗?
- 你能证明QUICKSORT是totally correct吗?

QUICKSORT( $A, p, r$ )

```
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )
```

PARTITION( $A, p, r$ )

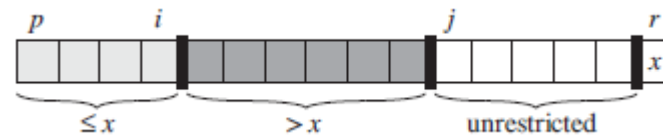
```
1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4      if  $A[j] \leq x$ 
5           $i = i + 1$ 
6          exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

# 问题3: Quicksort

- 你能简述Quicksort的执行过程吗?
- PARTITION中的loop invariant是什么?
- 你能证明PARTITION是totally correct吗?
- 你能证明QUICKSORT是totally correct吗?

```
QUICKSORT( $A, p, r$ )  
1  if  $p < r$   
2     $q = \text{PARTITION}(A, p, r)$   
3    QUICKSORT( $A, p, q - 1$ )  
4    QUICKSORT( $A, q + 1, r$ )
```

```
PARTITION( $A, p, r$ )  
1   $x = A[r]$   
2   $i = p - 1$   
3  for  $j = p$  to  $r - 1$   
4    if  $A[j] \leq x$   
5       $i = i + 1$   
6      exchange  $A[i]$  with  $A[j]$   
7  exchange  $A[i + 1]$  with  $A[r]$   
8  return  $i + 1$ 
```



1. If  $p \leq k \leq i$ , then  $A[k] \leq x$ .
2. If  $i + 1 \leq k \leq j - 1$ , then  $A[k] > x$ .
3. If  $k = r$ , then  $A[k] = x$ .



## 问题3: Quicksort (续)

- What is the running time of Quicksort when all elements of array  $A$  have the same value?
- What is the running time of Quicksort when array  $A$  contains distinct elements and is sorted in decreasing order?

QUICKSORT( $A, p, r$ )

```
1  if  $p < r$ 
2       $q = \text{PARTITION}(A, p, r)$ 
3      QUICKSORT( $A, p, q - 1$ )
4      QUICKSORT( $A, q + 1, r$ )
```

PARTITION( $A, p, r$ )

```
1   $x = A[r]$ 
2   $i = p - 1$ 
3  for  $j = p$  to  $r - 1$ 
4      if  $A[j] \leq x$ 
5           $i = i + 1$ 
6          exchange  $A[i]$  with  $A[j]$ 
7  exchange  $A[i + 1]$  with  $A[r]$ 
8  return  $i + 1$ 
```

# 问题3: Quicksort (续)

- RANDOMIZED-QUICKSORT与QUICKSORT有什么不同?
- 这种改变有什么意义?
  
- RANDOMIZED-QUICKSORT的运行时间主要耗费在什么操作?
- 为什么每对元素最多比较1次?

RANDOMIZED-QUICKSORT( $A, p, r$ )

```
1 if  $p < r$ 
2    $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3   RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4   RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```

RANDOMIZED-PARTITION( $A, p, r$ )

```
1  $i = \text{RANDOM}(p, r)$ 
2 exchange  $A[r]$  with  $A[i]$ 
3 return PARTITION( $A, p, r$ )
```

PARTITION( $A, p, r$ )

```
1  $x = A[r]$ 
2  $i = p - 1$ 
3 for  $j = p$  to  $r - 1$ 
4   if  $A[j] \leq x$ 
5      $i = i + 1$ 
6     exchange  $A[i]$  with  $A[j]$ 
7 exchange  $A[i + 1]$  with  $A[r]$ 
8 return  $i + 1$ 
```

# 问题3: Quicksort (续)

- RANDOMIZED-QUICKSORT与QUICKSORT有什么不同?
- 这种改变有什么意义?
  - In exploring the average-case behavior of quicksort, we have made an assumption that all permutations of the input numbers are equally likely. In an engineering situation, however, we cannot always expect this assumption to hold.
- RANDOMIZED-QUICKSORT的运行时间主要耗费在什么操作?
- 为什么每对元素最多比较1次?

RANDOMIZED-QUICKSORT( $A, p, r$ )

```
1 if  $p < r$ 
2    $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3   RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4   RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```

RANDOMIZED-PARTITION( $A, p, r$ )

```
1  $i = \text{RANDOM}(p, r)$ 
2 exchange  $A[r]$  with  $A[i]$ 
3 return PARTITION( $A, p, r$ )
```

PARTITION( $A, p, r$ )

```
1  $x = A[r]$ 
2  $i = p - 1$ 
3 for  $j = p$  to  $r - 1$ 
4   if  $A[j] \leq x$ 
5      $i = i + 1$ 
6     exchange  $A[i]$  with  $A[j]$ 
7 exchange  $A[i + 1]$  with  $A[r]$ 
8 return  $i + 1$ 
```

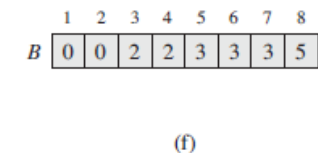
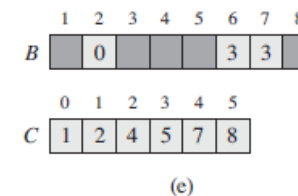
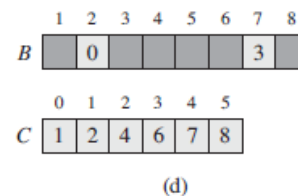
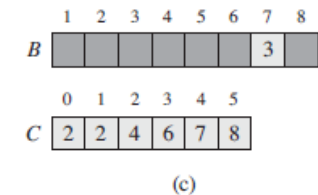
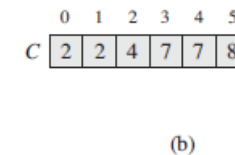
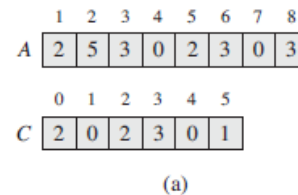
# 问题4: sorting in linear time

- 你能简述counting sort的执行过程吗?
- 为什么它是stable的?  
你能不能将最后一步改为从左往右扫描, 并仍保证stable?
- 它的使用有哪些局限性 (缺点)?

COUNTING-SORT( $A, B, k$ )

```

1  let  $C[0..k]$  be a new array
2  for  $i = 0$  to  $k$ 
3     $C[i] = 0$ 
4  for  $j = 1$  to  $A.length$ 
5     $C[A[j]] = C[A[j]] + 1$ 
6  //  $C[i]$  now contains the number of elements equal to  $i$ .
7  for  $i = 1$  to  $k$ 
8     $C[i] = C[i] + C[i - 1]$ 
9  //  $C[i]$  now contains the number of elements less than or equal to  $i$ .
10 for  $j = A.length$  downto 1
11    $B[C[A[j]]] = A[j]$ 
12    $C[A[j]] = C[A[j]] - 1$ 
    
```



# 问题4: sorting in linear time (续)

- 你能简述radix sort的执行过程吗?
- 为什么它需要调用一个stable sort?  
能不能改为从高位开始排序?
- 你如何理解

We have some flexibility in how to break each key into digits.

RADIX-SORT( $A, d$ )

1 for  $i = 1$  to  $d$

2     use a stable sort to sort array  $A$  on digit  $i$

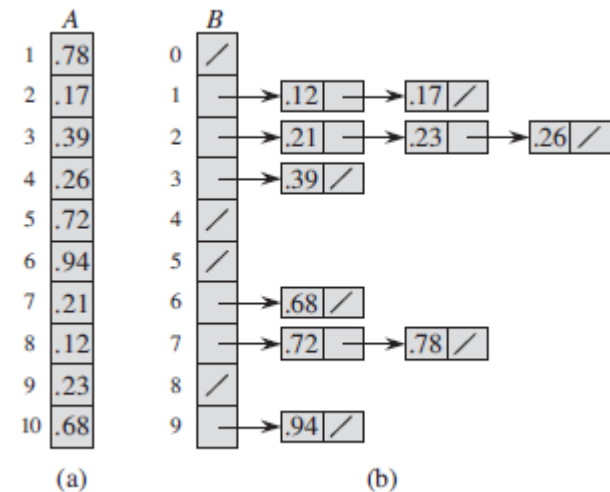
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

# 问题4: sorting in linear time (续)

- 你能简述bucket sort的执行过程吗?

BUCKET-SORT( $A$ )

```
1 let  $B[0..n-1]$  be a new array
2  $n = A.length$ 
3 for  $i = 0$  to  $n-1$ 
4   make  $B[i]$  an empty list
5 for  $i = 1$  to  $n$ 
6   insert  $A[i]$  into list  $B[\lfloor nA[i] \rfloor]$ 
7 for  $i = 0$  to  $n-1$ 
8   sort list  $B[i]$  with insertion sort
9 concatenate the lists  $B[0], B[1], \dots, B[n-1]$  together in order
```



- 它的使用有哪些局限性（缺点）？
- 你能利用类似的思想解决以下问题吗？

We are given  $n$  points in the unit circle,  $p_i = (x_i, y_i)$ , such that  $0 < x_i^2 + y_i^2 \leq 1$  for  $i = 1, 2, \dots, n$ . Suppose that the points are uniformly distributed; that is, the probability of finding a point in any region of the circle is proportional to the area of that region. Design an algorithm with an average-case running time of  $\Theta(n)$  to sort the  $n$  points by their distances  $d_i = \sqrt{x_i^2 + y_i^2}$  from the origin.