

# For a While

王慧妍

why@nju.edu.cn

南京大学



计算机科学与技术系



计算机软件研究所



# 回顾

---

- `if`
  - `_Bool`
  - `bool` `stdbool.h`
  
- `switch`
  - 常量表达式
  
- `for`
  - 计数循环的次数

# For循环

- Given a set  $A$  of integers, to compute their minimum.



```
for ( <expression> ; <expression> ; <expression> )  
    <statement>
```

循环开始前的准备

循环结束条件

每轮循环的最后  
一个执行  
惯用法i++

# 输出一个九九乘法表

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

- 注意对齐
- [99.c](#)

# 有的事情需要循环来做，但难以计数

- 怎么判断一个整数的位数

- 353: 3位
- 23518: 5位

- 23518

23518 -----  $x = x \% 10000$

~~23518~~ -----  $x = x \% 1000$

~~23518~~ -----  $x = x \% 100$

~~23518~~ -----  $x = x \% 10$

~~23518~~ -----  $x = x \% 1$

$x == 0$

~~23518~~ -----  $x = x / 10$

~~23518~~ -----  $x = x / 10$

~~23518~~ -----  $x = x / 10$

~~23518~~ -----  $x = x / 10$

~~23518~~ -----  $x = x / 10$

$x == 0$

# while循环

```
while (表达式){  
    语句  
}
```

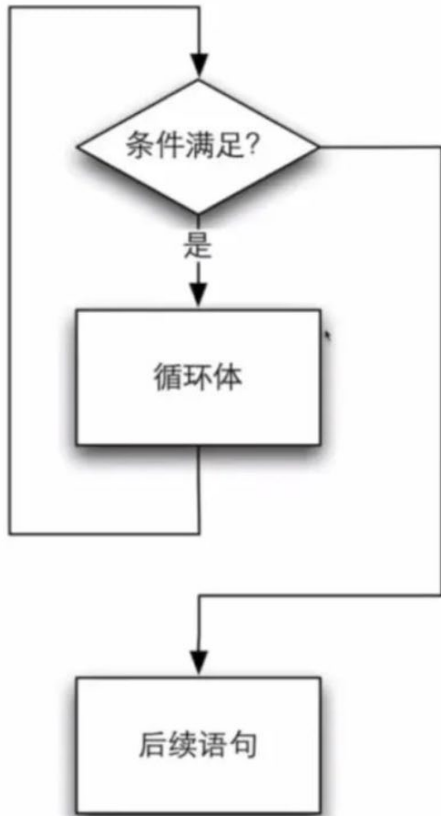
- “当”
  - 当表达式条件满足时，执行循环体内语句



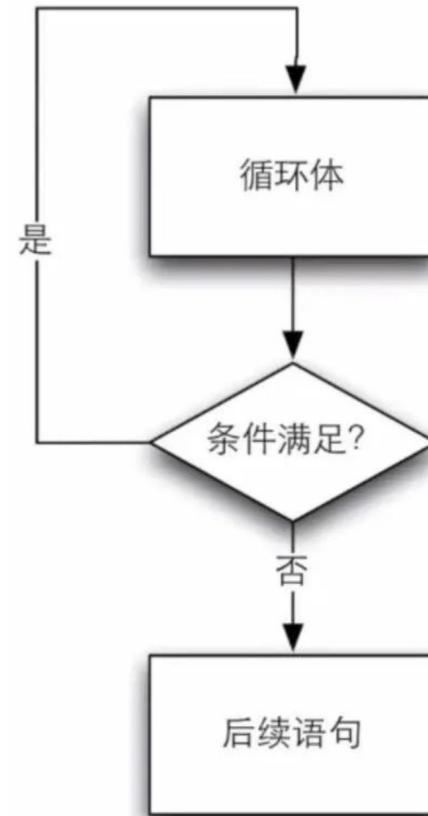
# do-while循环

```
do{  
    语句  
}while (表达式);
```

- 进入循环时不做检查，执行完一轮循环后，检查条件是否满足，满足则进入下一轮，否则结束循环
- “一直做，直到表达式不满足了”



while



do-while

# Num of digits

- 怎么判断一个整数的位数
  - 353: 3位
  - 23518: 5位
  - [digit.c](#)



LESSON

Identify the Place & Value  
of Digits in a  
**Three-Digit Number**

 1 Hundred

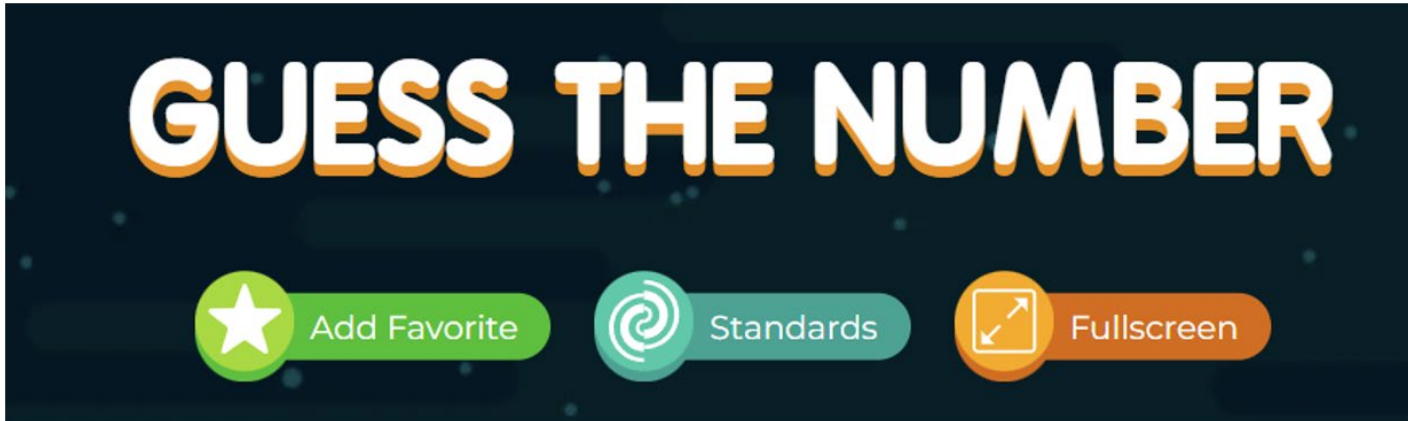
 **127**

 2 Tens

 7 Ones

edumeri NUMBER & OPERATIONS IN BASE TEN 2

- [guess.c](#)
- [guess\\_dowhile.c](#)



# Inverse

- 输入正整数，输出其逆序
  - [inverse.c](#)
- 主要操作：
  - 得到个位数:  $x = x \% 10$
  - 丢弃个位数:  $x = x / 10$



# 循环次数

```
int count = 100;
while(count >= 0){
    printf("%d\n", count);
    count --;
}
```

- 这个循环多少次?
- 循环结束时, 有没有输出0?
- 循环结束后, count的值是多少?
- `for(int cnt = 100; cnt >=0; cnt--)`
- `for(int i = 0; i<=100; i++)`

# 循环次数

```
int count = 100;
while(count > 0){
    count --;
    printf("%d\n", count);
}
```

- 这个循环多少次?
- 循环结束时, 有没有输出0?
- 循环结束后, count的值是多少?
- for(int cnt = 100; cnt >0; cnt--)
- for(int i = 0; i<100; i++)

一大波编程示例即将来袭!



# $\log_2 x$

---

```
#include<stdio.h>

int main(){
    int x;
    int ret = 1;
    //scanf("%d", &x);
    x = 128;
    int t = x;
    while(x > 1){
        x /= 2;
        ret ++;
    }
    printf("log2 of %d is %d.", t, ret);
}
```

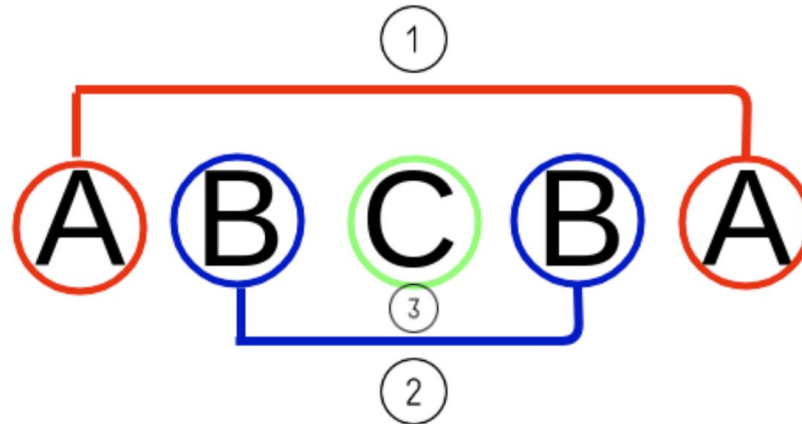
# 水仙花数

---

- 水仙花数是指一个N为正整数 ( $N \geq 3$ )，每个位上的数字的N次幂之和等于本身
  - 如:  $153 = 1^3 + 5^3 + 3^3$
  - 输入N, 输出N位整数的所有水仙花数
  - [narcissus.c](#)

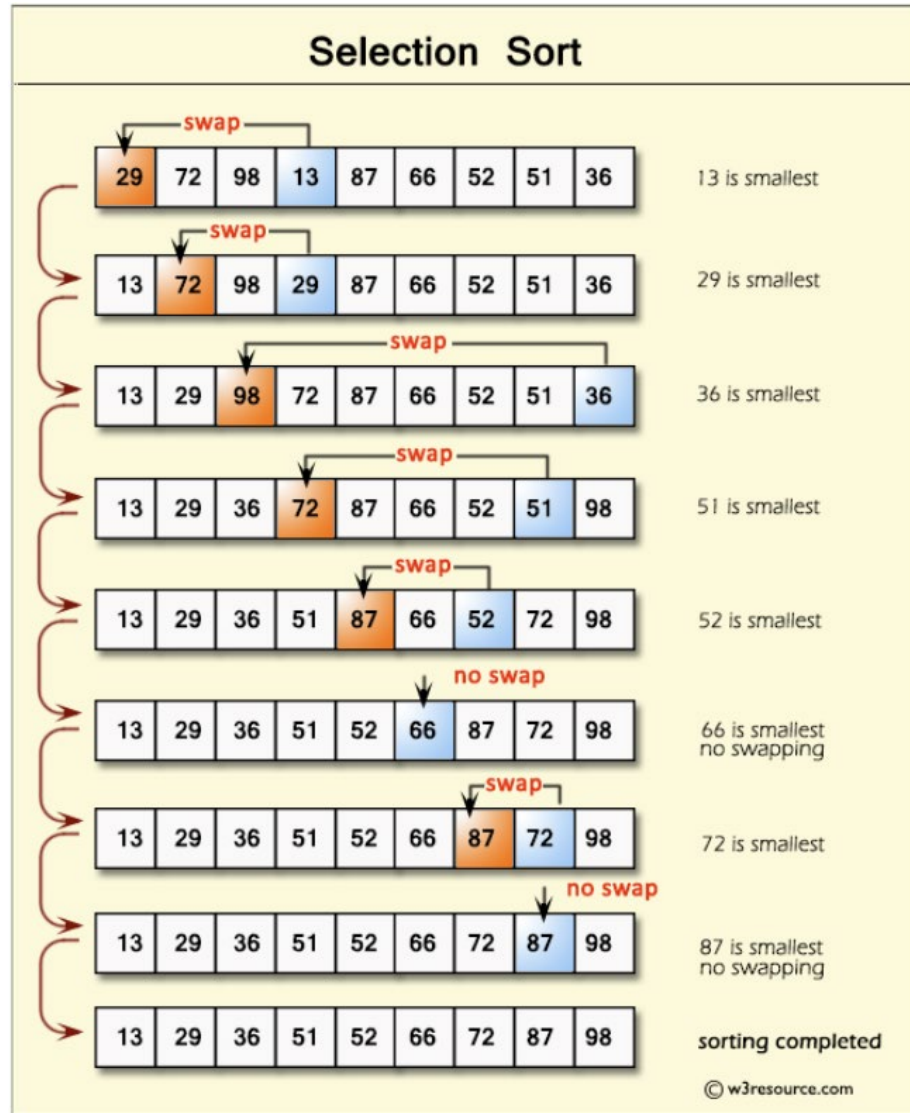
# Palindrome

- “回文”
- [palindrome.c](#)



# Selection Sort

- [selectsort.c](#)



# 有时候需要跳出循环

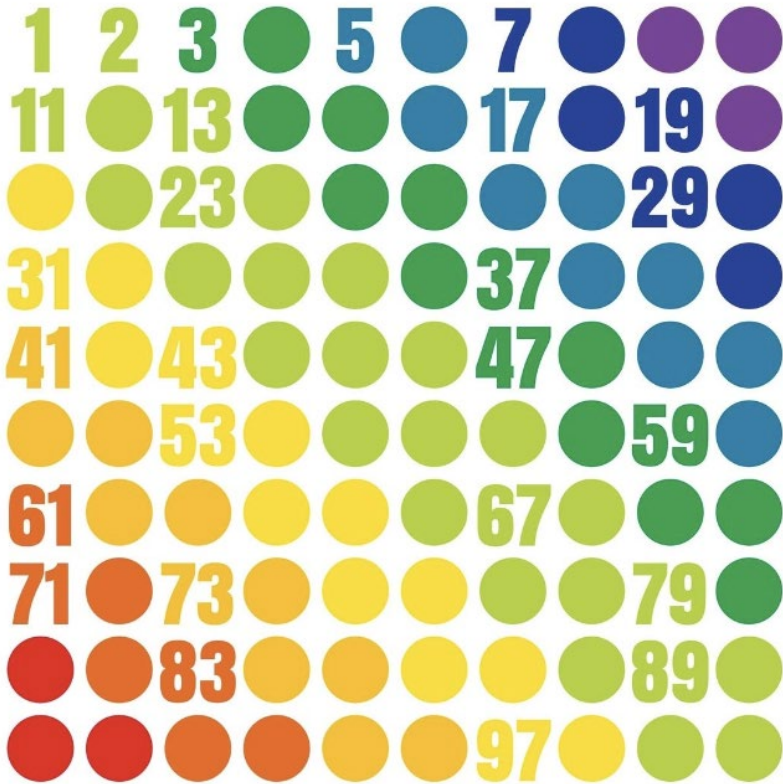
---

- `while(1){.....}`
- `break`
  - 跳出最近的循环
- `continue`
  - 跳出当前这一次循环

# Prime Numbers

- [prime.c](#)

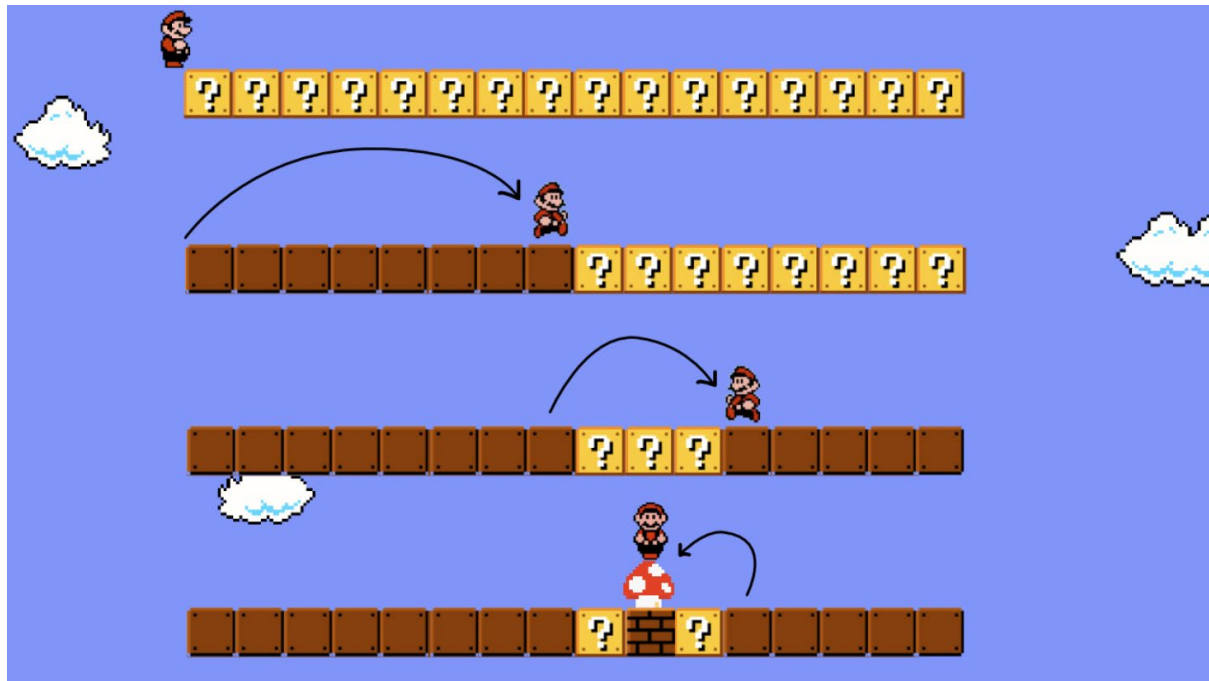
## PRIME NUMBERS



# Binary Search

- 典型二分查找算法

- 斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21, 34, 55, .....
- [binarysearch.c](http://binarysearch.c)



# goto

- break
  - 只能跳出当前层的循环

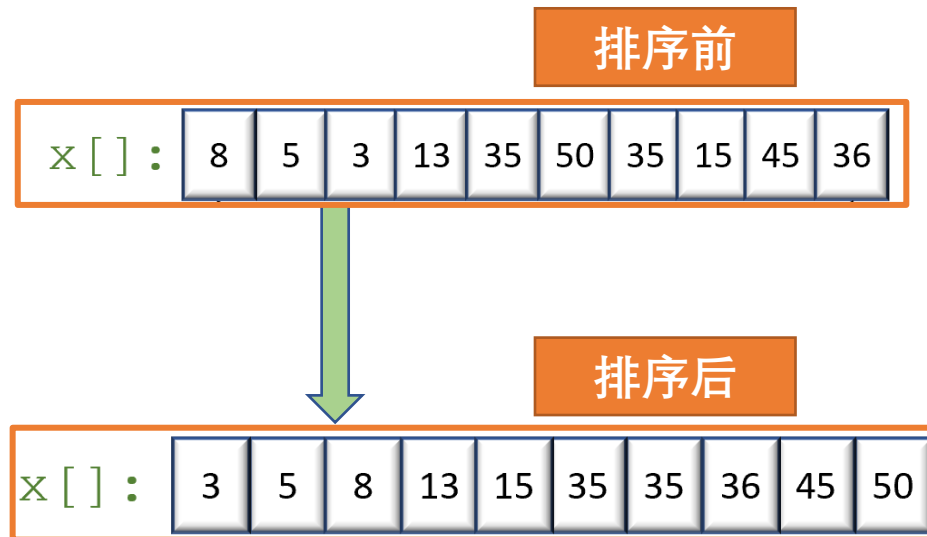
```
goto Label;  
  
.....  
  
Label: .....
```

- 常见用法：跳出深层循环
- 理解，但goto会破坏程序结构性，尽可能不用

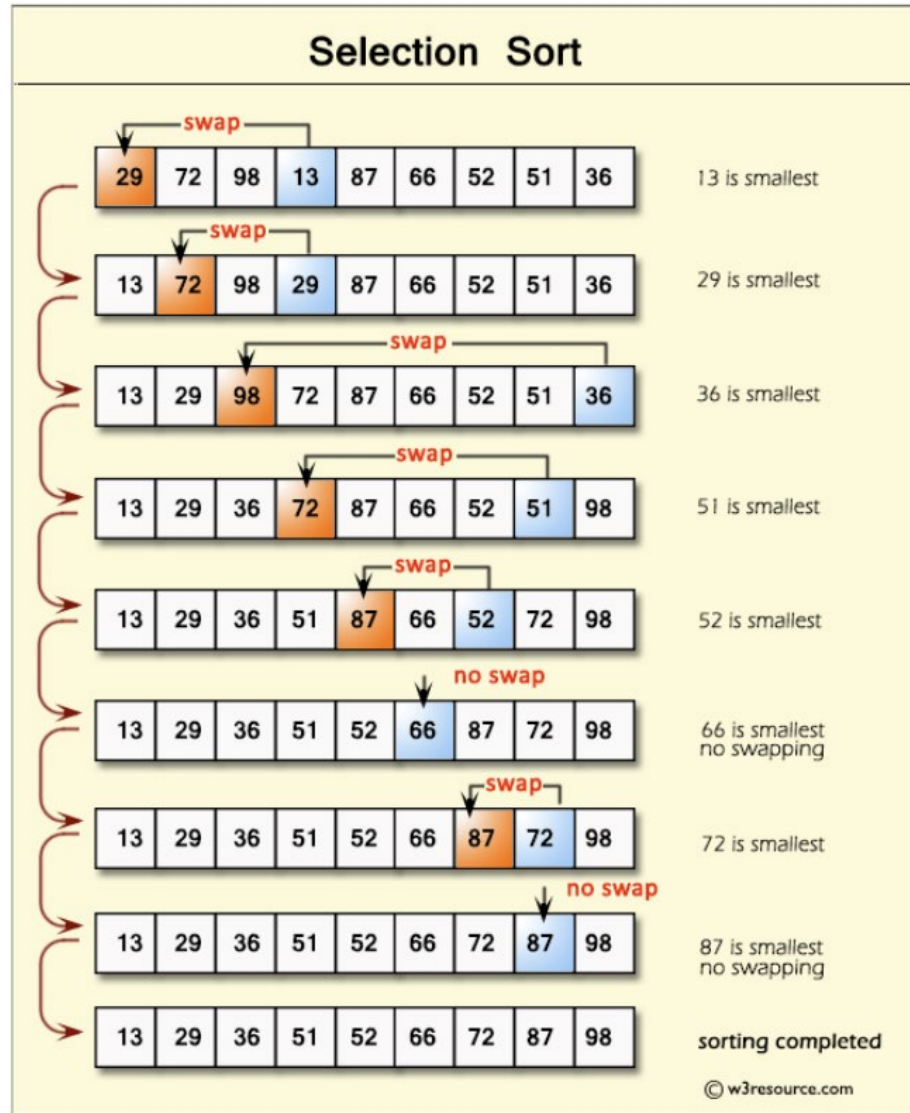
# 冒泡排序

- 基本思想

- 重复地走访过要排序的元素列，依次比较两个相邻的元素并按需交换  
(目标：从小到大排列)
  - 若  $x[i] > x[i+1]$ ，则交换
- 直观表达，每一轮遍历，将一个最大的数移到序列末尾



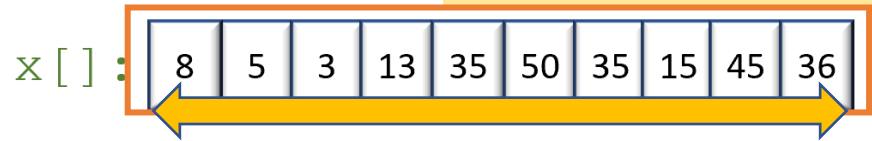
# 选择排序



# 冒泡排序

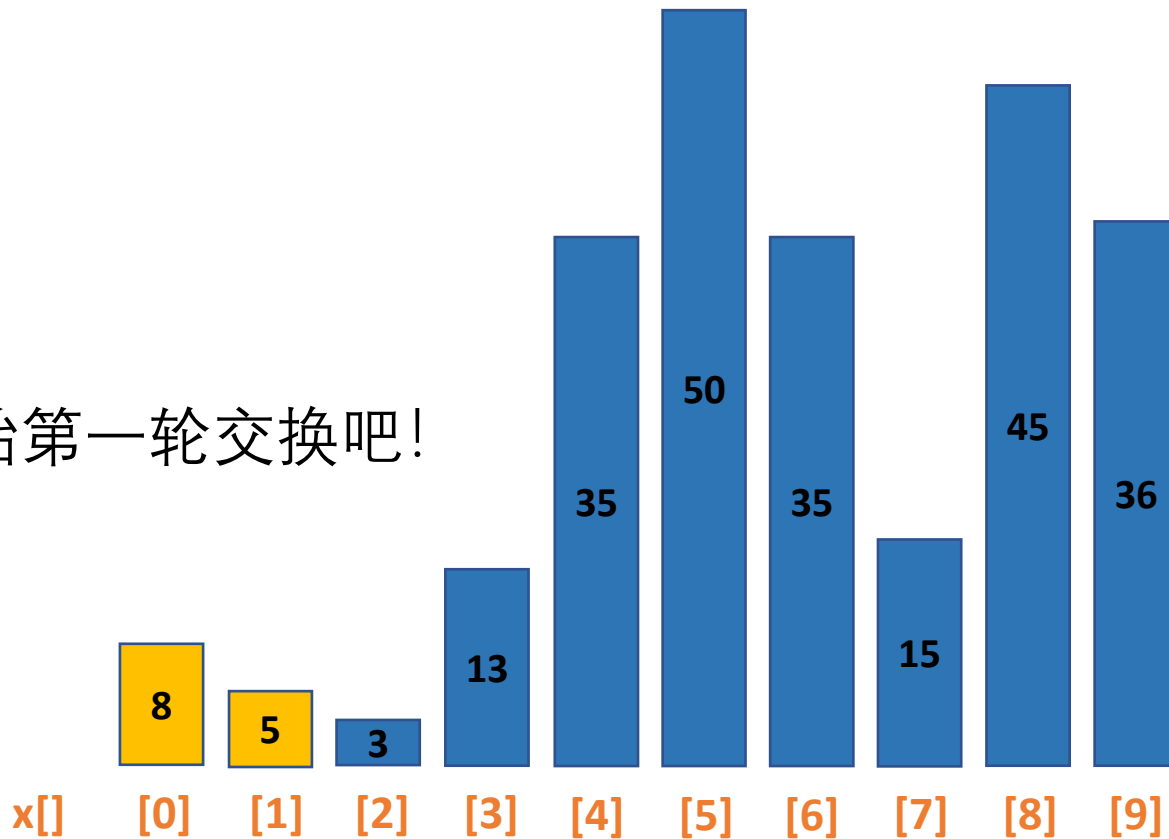
- 基本思想
  - 重复地走访过要排序的元素列，依次比较两个相邻的元素并按需交换（目标：从小到大排列）
    - 若  $x[i] > x[i+1]$ ，则交换
  - 直观表达，每一轮遍历，将一个最大的数移到序列末尾

遍历整个数组!

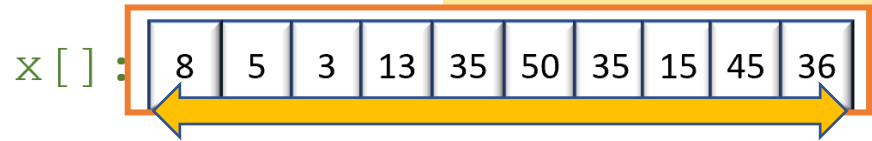


8 > 5, 交换!

开始第一轮交换吧!



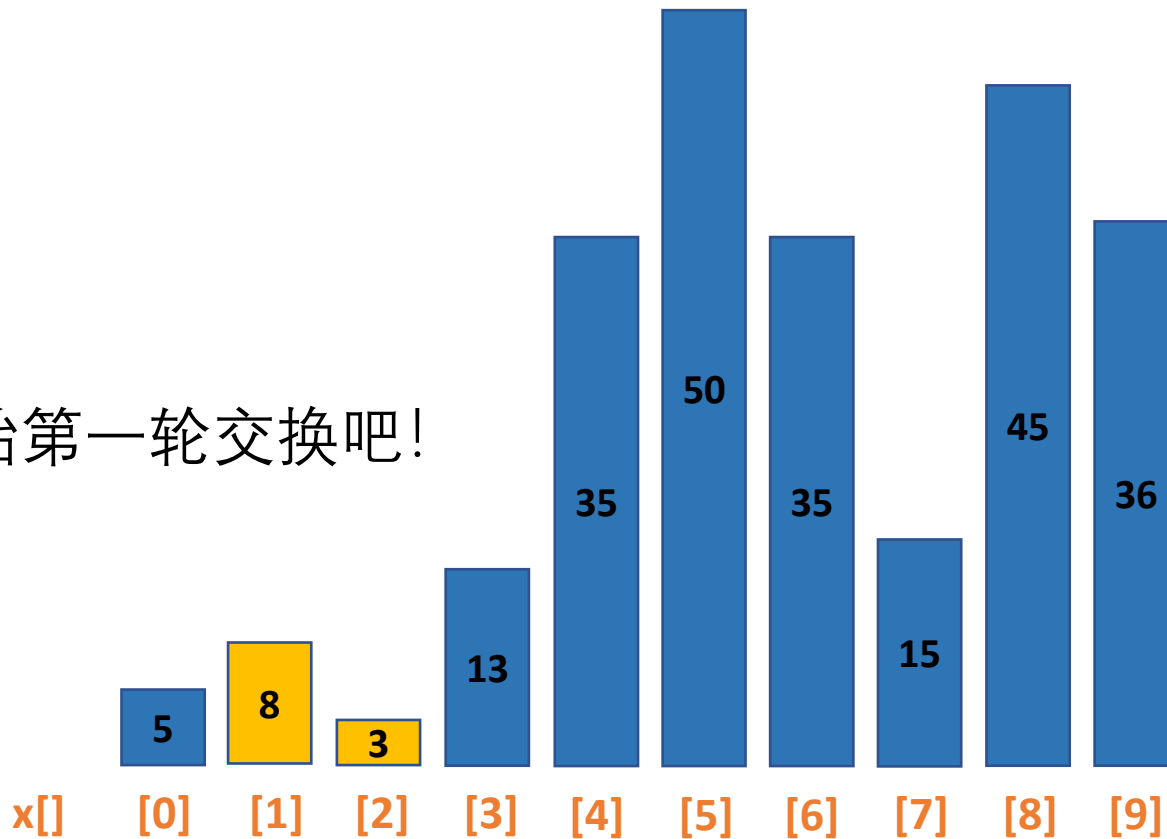
遍历整个数组!



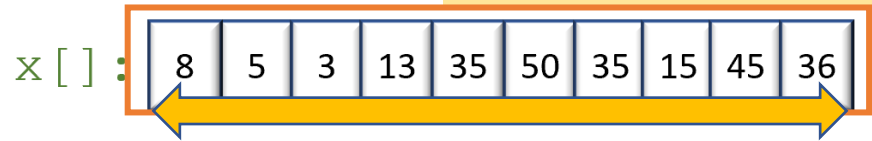
8 > 5, 交换!

8 > 3, 交换!

开始第一轮交换吧!



遍历整个数组!



8 > 5, 交换!

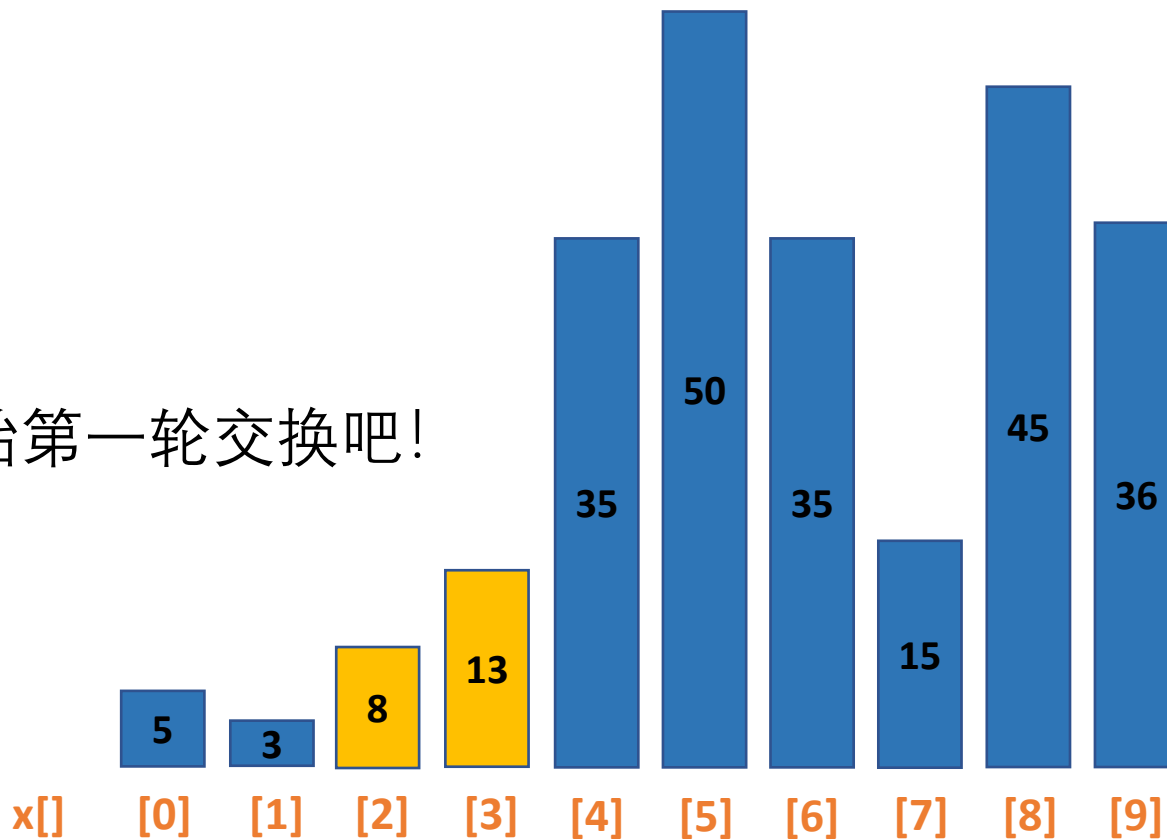
8 > 3, 交换!

8 < 13, 不交换!

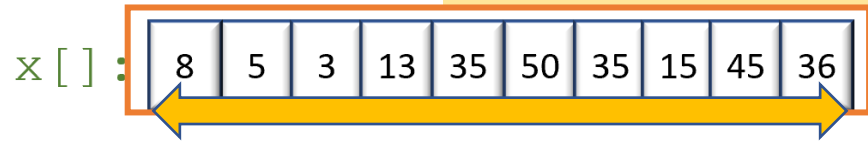
13 < 35, 不交换!

35 < 50, 不交换!

开始第一轮交换吧!



遍历整个数组!



8 > 5, 交换!

8 > 3, 交换!

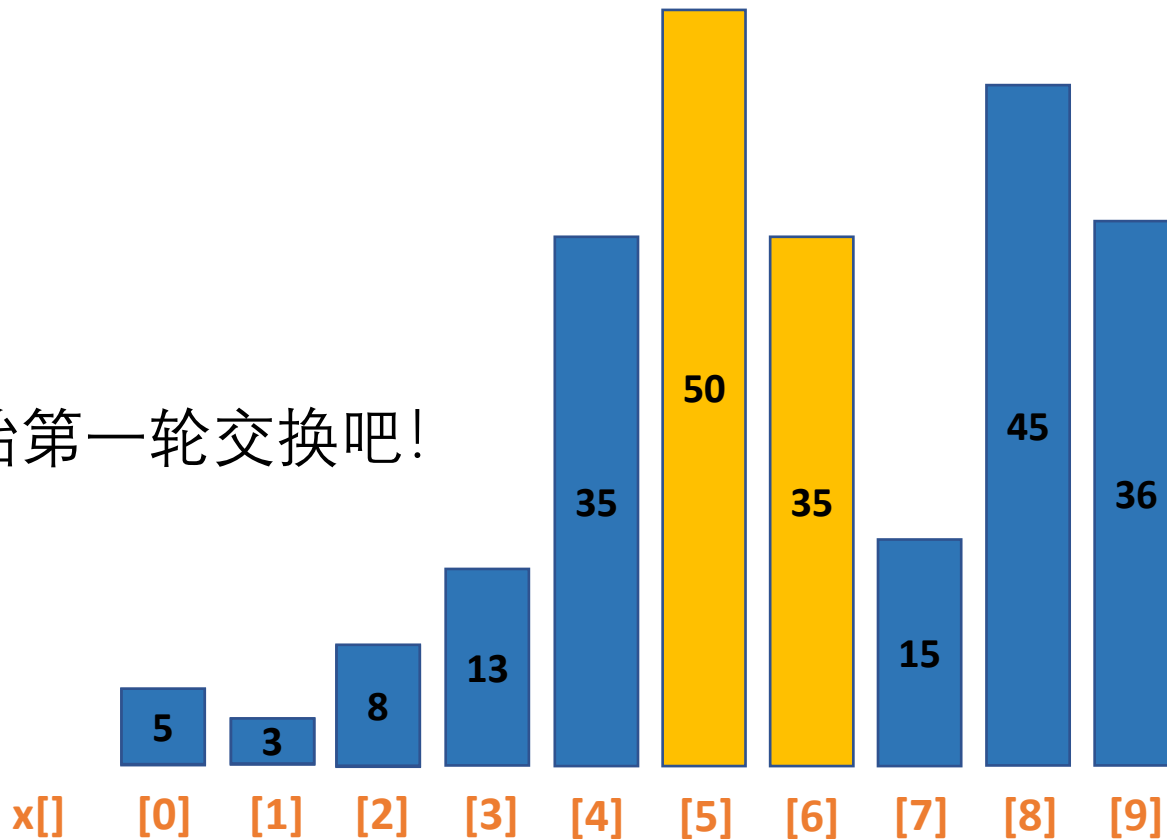
8 < 13, 不交换!

13 < 35, 不交换!

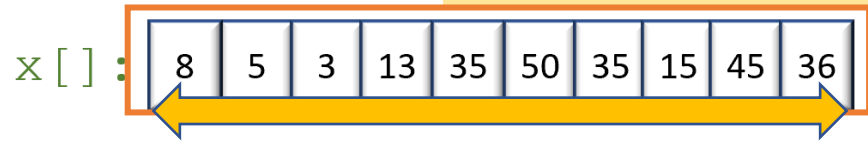
35 < 50, 不交换!

50 > 35, 交换!

开始第一轮交换吧!



遍历整个数组!



8 > 5, 交换!

8 > 3, 交换!

8 < 13, 不交换!

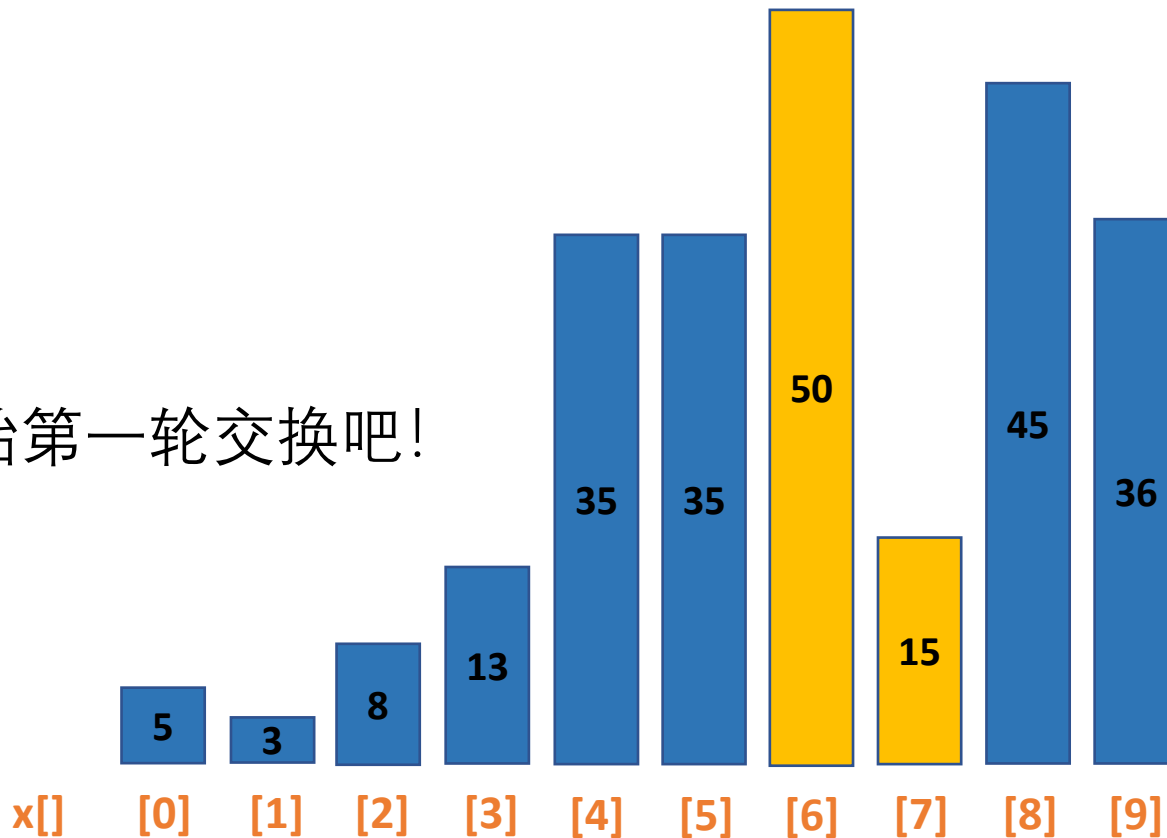
13 < 35, 不交换!

35 < 50, 不交换!

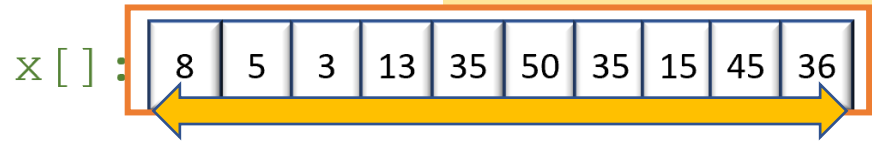
50 > 35, 交换!

50 > 15, 交换!

开始第一轮交换吧!



遍历整个数组!



8 > 5, 交换!

8 > 3, 交换!

8 < 13, 不交换!

13 < 35, 不交换!

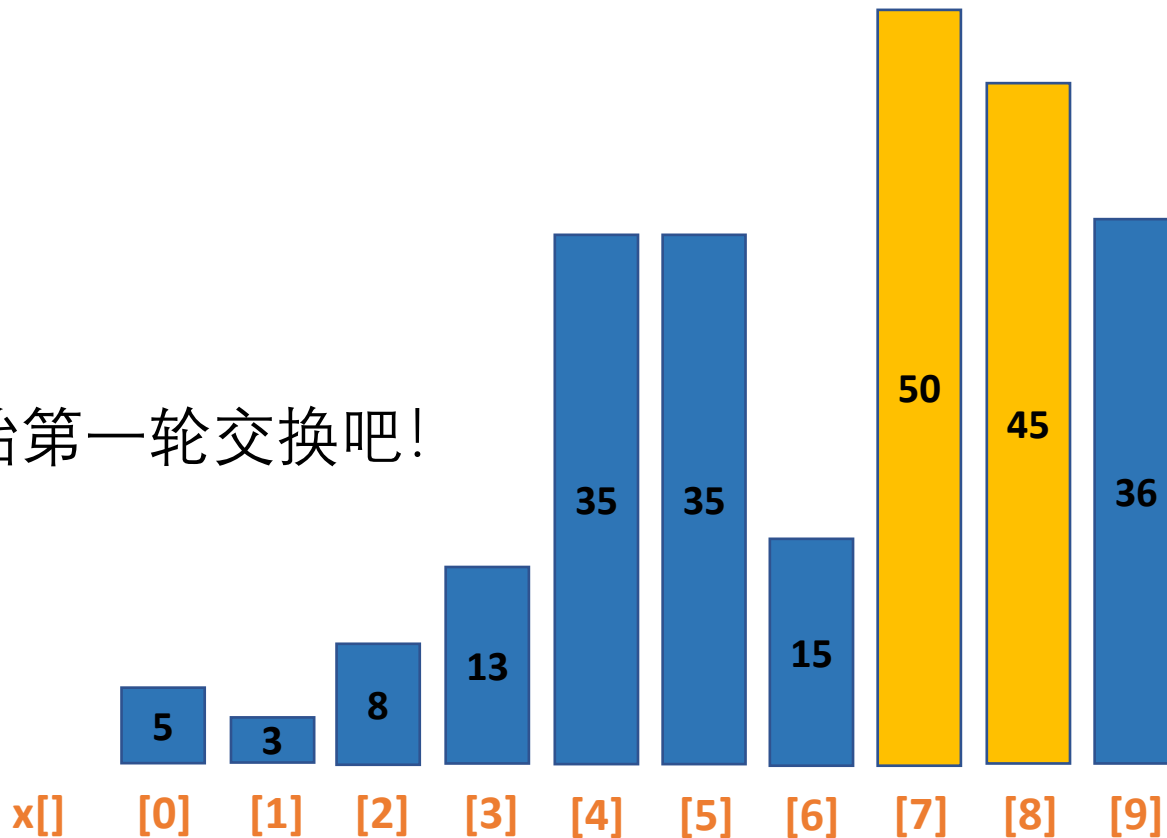
35 < 50, 不交换!

50 > 35, 交换!

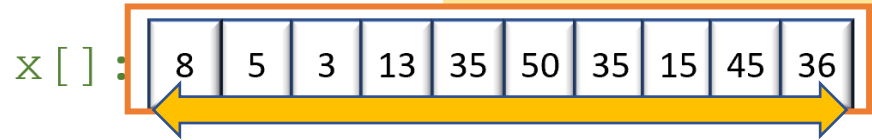
50 > 15, 交换!

50 > 45, 交换!

开始第一轮交换吧!



## 遍历整个数组!



8 > 5, 交换!

8 > 3, 交换!

8 < 13, 不交换!

13 < 35, 不交换!

35 < 50, 不交换!

50 > 35, 交换!

50 > 15, 交换!

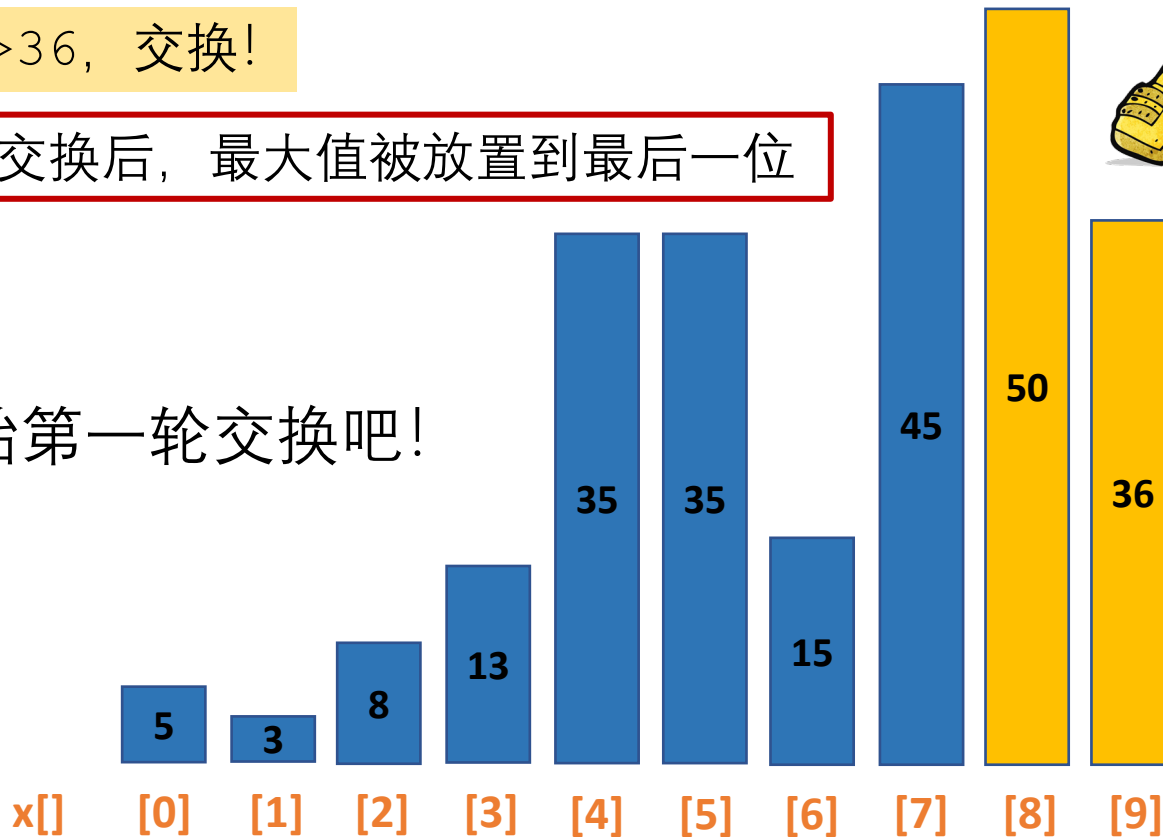
50 > 45, 交换!

50 > 36, 交换!

第一轮交换后, 最大值被放置到最后一位



开始第一轮交换吧!



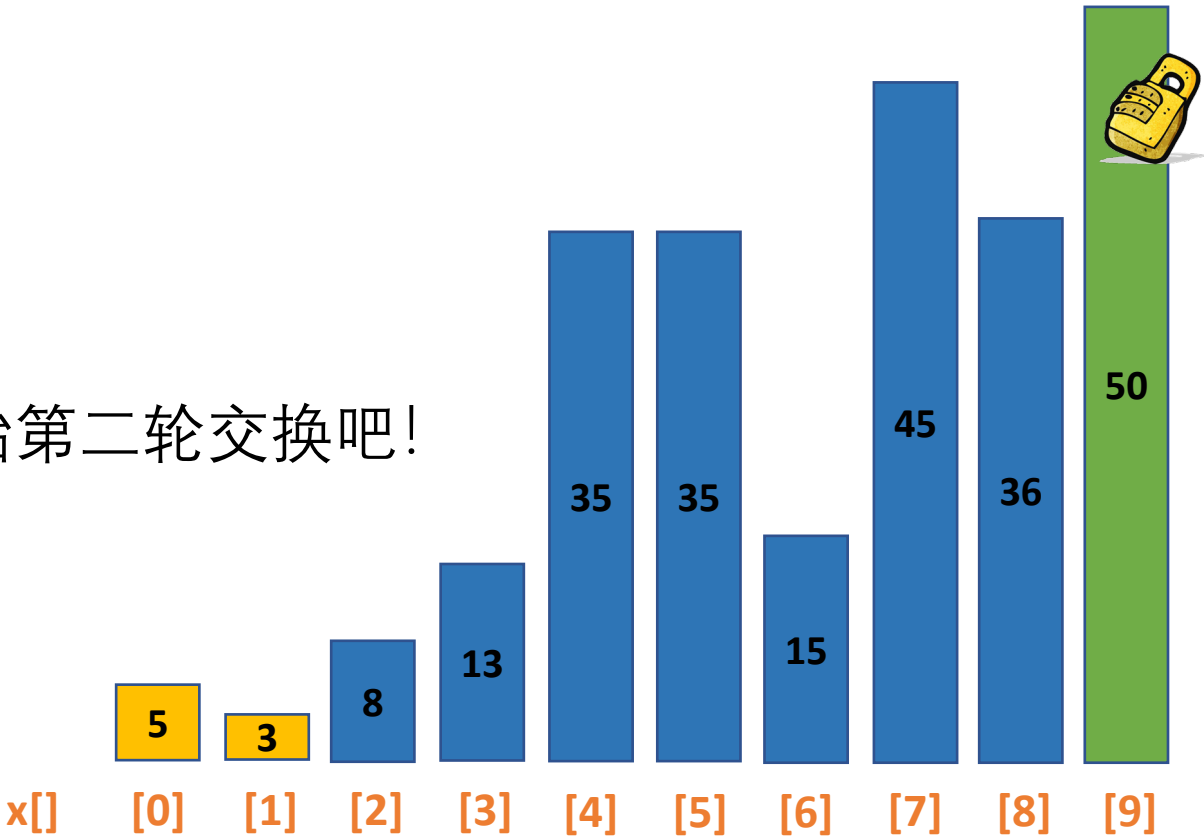
第一轮交换后，最大值被放置到最后一位

遍历长度减一!



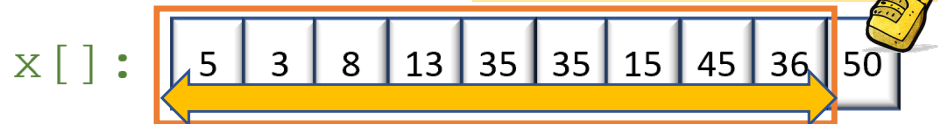
5 > 3, 交换!

开始第二轮交换吧!



第一轮交换后，最大值被放置到最后一位

遍历长度减一！



5 > 3, 交换!

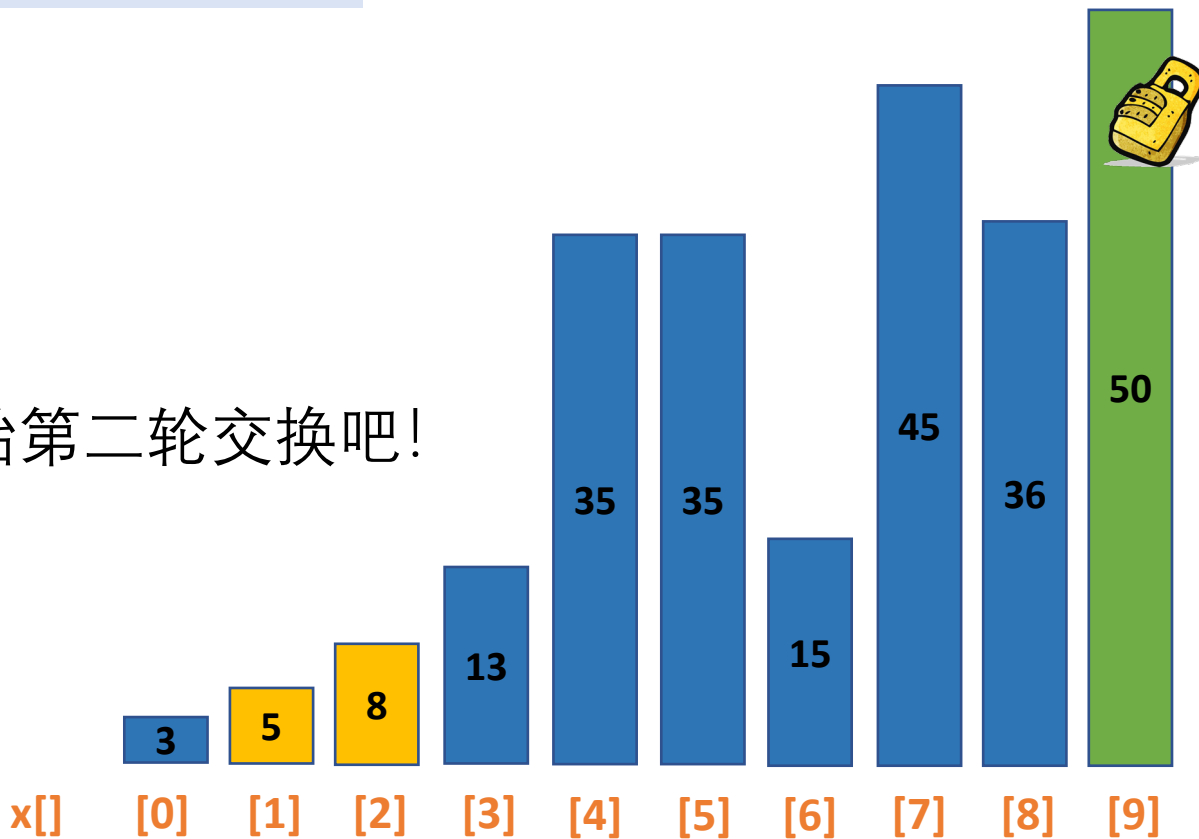
5 < 8, 不交换!

8 < 13, 不交换!

13 < 35, 不交换!

35 == 35, 不交换!

开始第二轮交换吧!



第一轮交换后，最大值被放置到最后一位

遍历长度减一!



5 > 3, 交换!

5 < 8, 不交换!

8 < 13, 不交换!

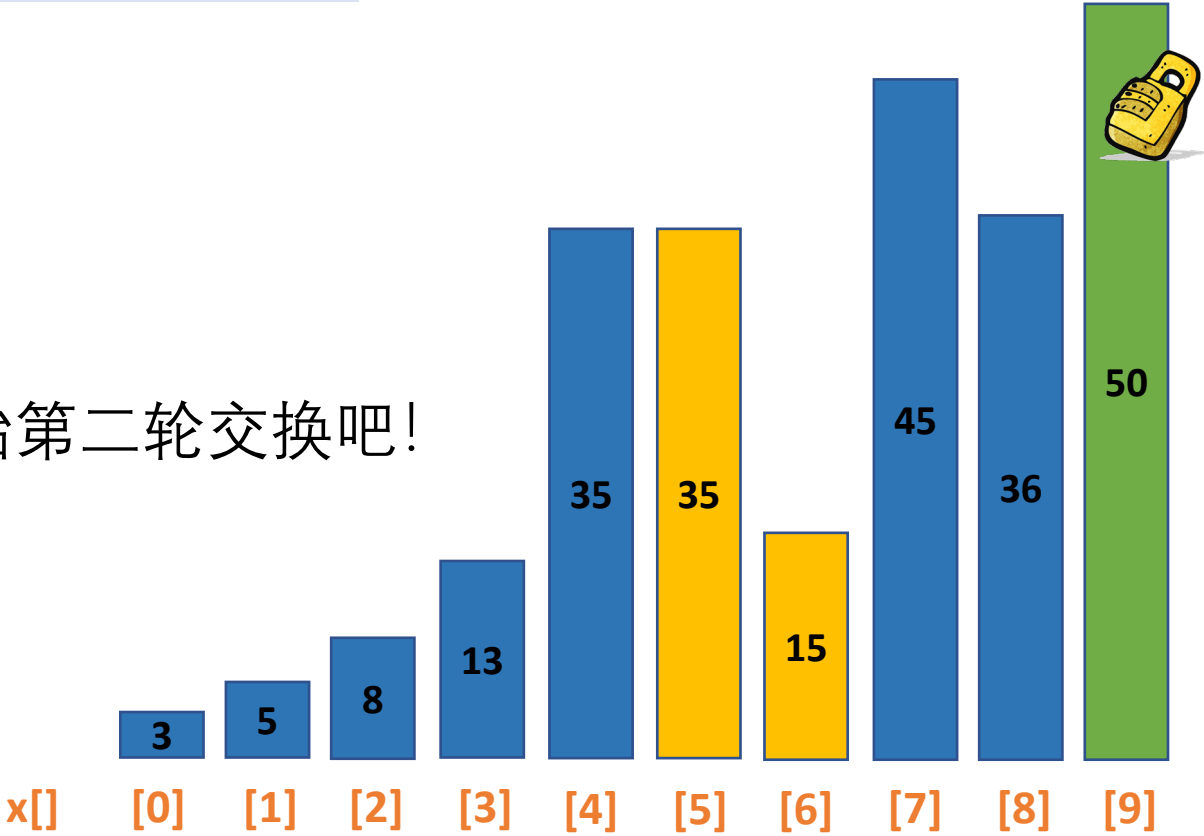
13 < 35, 不交换!

35 == 35, 不交换!

35 > 15, 交换!

35 < 45, 不交换!

开始第二轮交换吧!



第一轮交换后，最大值被放置到最后一位

遍历长度减一！



5 > 3, 交换!

5 < 8, 不交换!

8 < 13, 不交换!

13 < 35, 不交换!

35 == 35, 不交换!

35 > 13, 交换!

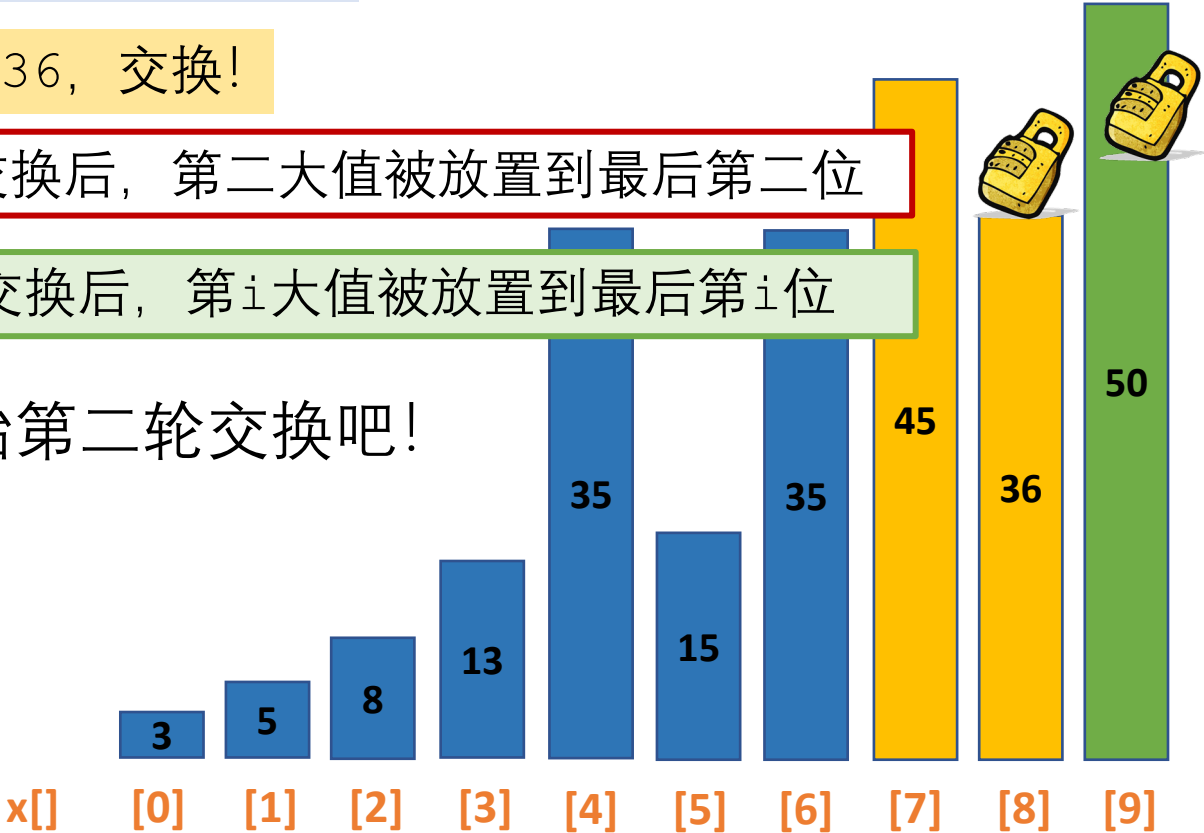
35 < 45, 不交换!

45 > 36, 交换!

第二轮交换后，第二大值被放置到最后第二位

第 i 轮交换后，第 i 大值被放置到最后第 i 位

开始第二轮交换吧!



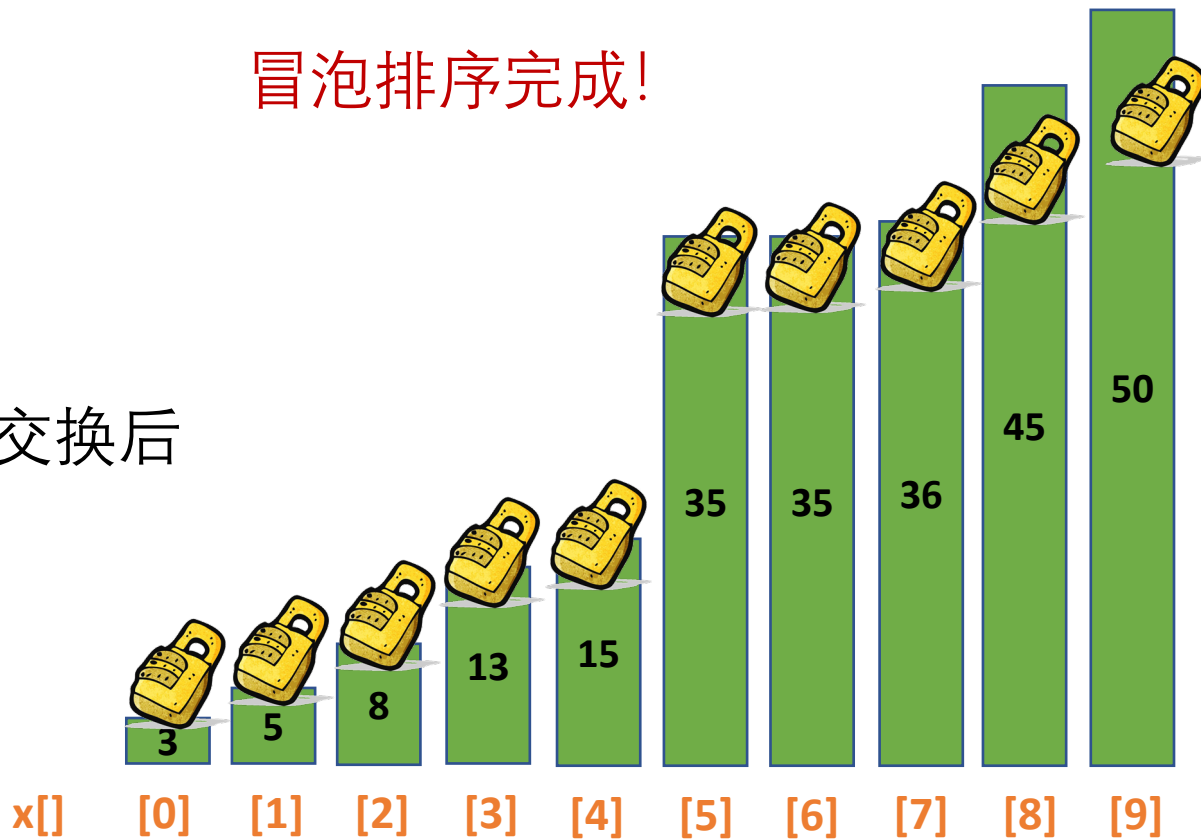
第*i*轮交换后，第*i*大值被放置到最后第*i*位

遍历*n*轮后：

$x[]$  : 3 5 8 13 15 35 35 36 45 50

冒泡排序完成!

*n*轮交换后



# 冒泡排序

- 基本思想

- 重复地走访过要排序的元素列，依次比较两个相邻的元素并按需交换（目标：从小到大排列）
  - 若  $x[i] > x[i+1]$ ，则交换
- 直观表达，**每一轮遍历，将一个最大的数移到序列末尾**
- [bubblesort.c](#)



End.

---

Keep coding!  
孰能生巧!