

FSE598 前沿计算技术

模块 3 算法设计与分析

单元 1 算法基础

第 1 讲 算法概述

本讲提要

学习内容

- ❑ 算法的概念
- ❑ 算法原语
- ❑ 算法复杂度
- ❑ 算法示例
- ❑ 确定关键步骤： 机器人算法

何谓计算机科学？

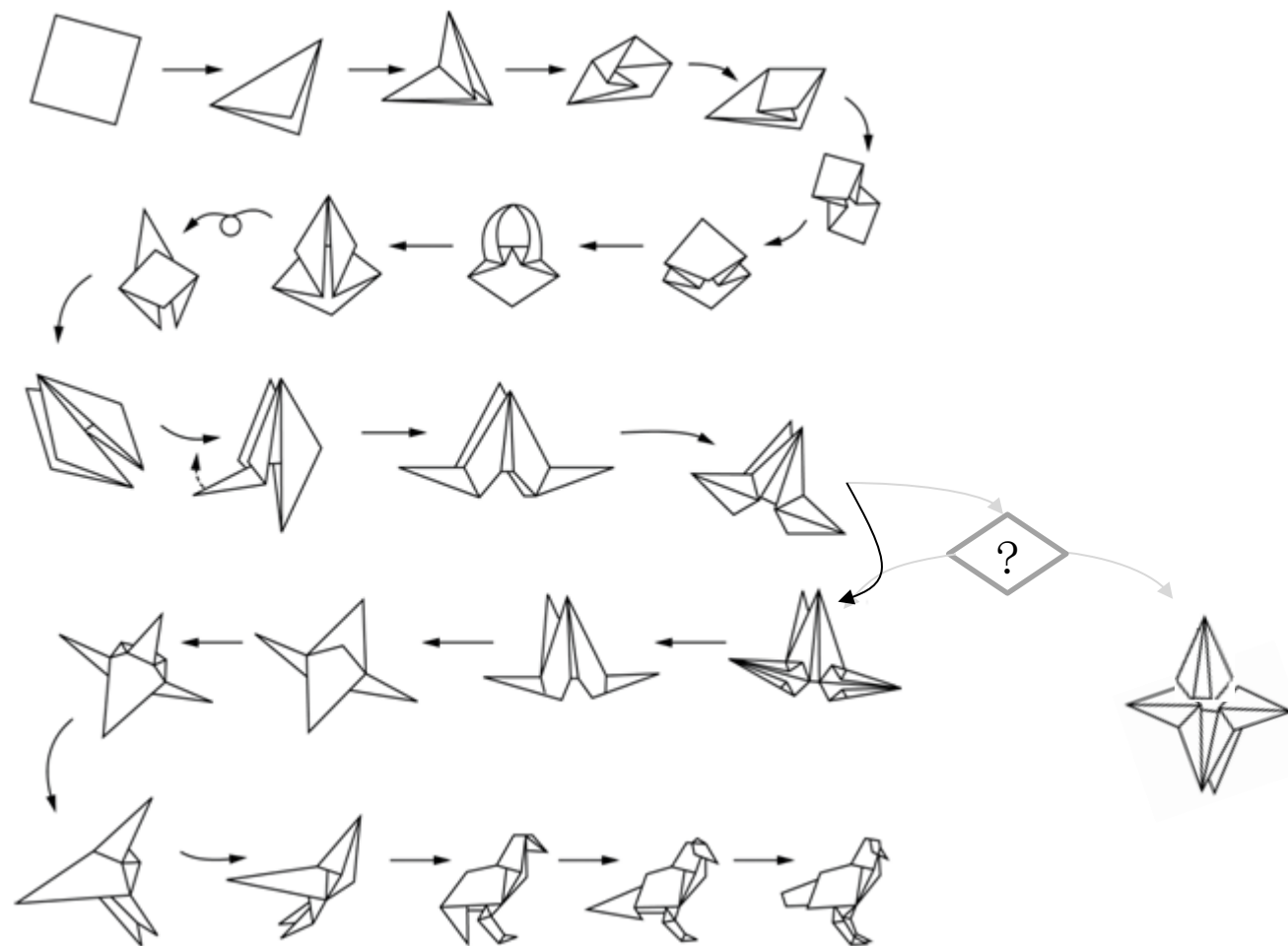
计算机科学

- ❑ 是对**信息（数据）**和计算的理论基础，及其在计算机系统实现和应用的实用技术进行研究的一门学科；
- ❑ 通常被描述为“有关描述和转换信息的算法过程（**算法**）的系统性研究”。
- ❑ 旨在探究以下基本问题：
*哪些问题可以实现（**高效**）自动化？*

算法的定义


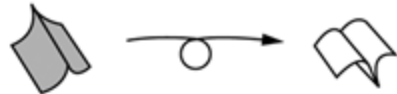
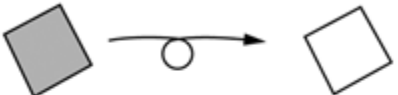

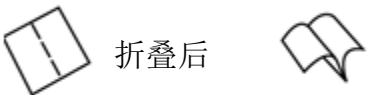

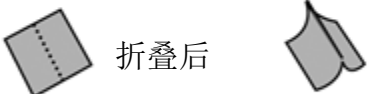

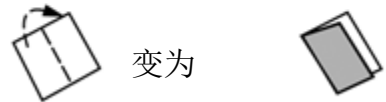

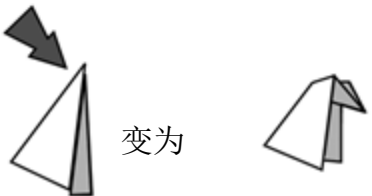
- 算法是指用来定义某个终止进程的一组有序的明确步骤（原语）。
- 算法必须要：
 - 正确：每个有效输入满足说明
 - 终止：在限定时间内交付结果
 - 可通过有限的步骤进行计算
 - 效率高
 - 效率高：计算时间是输入大小 n 的多项式函数；例如： $T(n) = n^3$
 - 效率低：计算时间至少是输入大小 n 的指数函数；例如： $T(n) = 2^n$

用正方形纸折叠小鸟的算法



资料来源: Glenn Brookshear and Dennis Brylow, Computer Science:An Overview (13th Edition), Addison Wesley, 2018

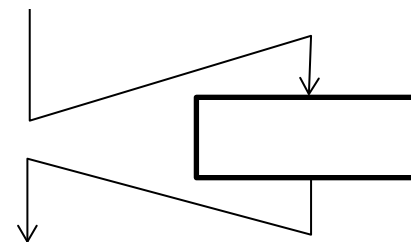
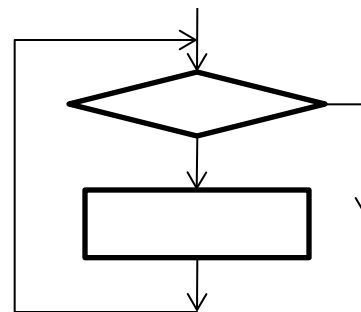
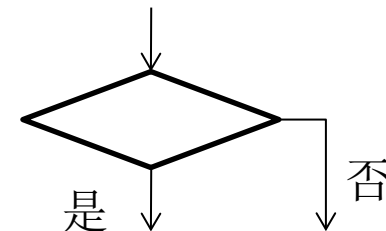
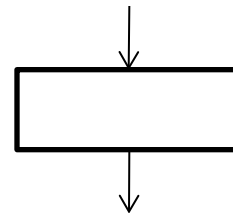
算法原语：易于理解的基本步骤

语法	语义
	按图示把纸张翻过来 
把纸的一面涂上阴影	区分纸张的不同面 如图所示 
	“山谷”折 对折 
	“山峰”折 对折 
	折叠 对折 
	下弯折 对折 

资料来源：Glenn
Brookshear and Dennis Brylow,
Computer Science:An Overview (13th
Edition), Addison Wesley, 2018

伪代码原语

- 赋值 名称 \leftarrow 表达式
- 条件选择
 如果满足某个条件, 则 执行某些操作
 否则 执行某些操作
- 重复执行
 当 满足某个条件, 则一直 执行 某些操作
- 过程
 过程 过程名 (参数)
 函数/操作/VIPLE 活动



过程是一组伪代码

```
Procedure CountTo10    //类似: Python 函数 或 VIPLE 活动
Count  $\leftarrow$  0;
While (Count < 10) do
{
    print ("The number is " and Count);
    Count  $\leftarrow$  Count + 1;
}
```


算法复杂度的度量

最坏情况：（通常）

- $T(n)$ = 算法对任意大小为 n 的输入的最长运行时间。

平均情况：（间或/有时候）

- $T(n)$ = 算法对所有大小为 n 的输入的预期运行时间。
- 须假设输入的统计分布。

最佳情况：（从不）

- 针对较慢的算法，用某些能够快速得出结果的输入进行欺骗。

算法复杂度的考虑因素

- ❑ 实际的执行时间取决于输入：已排序的序列可能更容易排序。因此，算法分析会考虑最坏的情况或平均情况；
- ❑ 实际的执行时间取决于输入的大小：排序 10 个数字比排序 5 个数字花费的时间更长。因此，输入大小 n 被视为一项参数（变量）；
- ❑ 任何输入较小的问题都可以很容易地解决，因此，算法分析侧重于输入较大时的执行时间；
- ❑ 执行时间取决于机器。算法分析会计算所需的步骤（运算），而不是时间。

示例：举重比赛

问题定义

输入： 已知一组数字，数字表示的是运动员举起的重量

输出： 找到最大的重量，以此表示获胜者

运动员 1: $W1$ = 从键盘输入一个数字

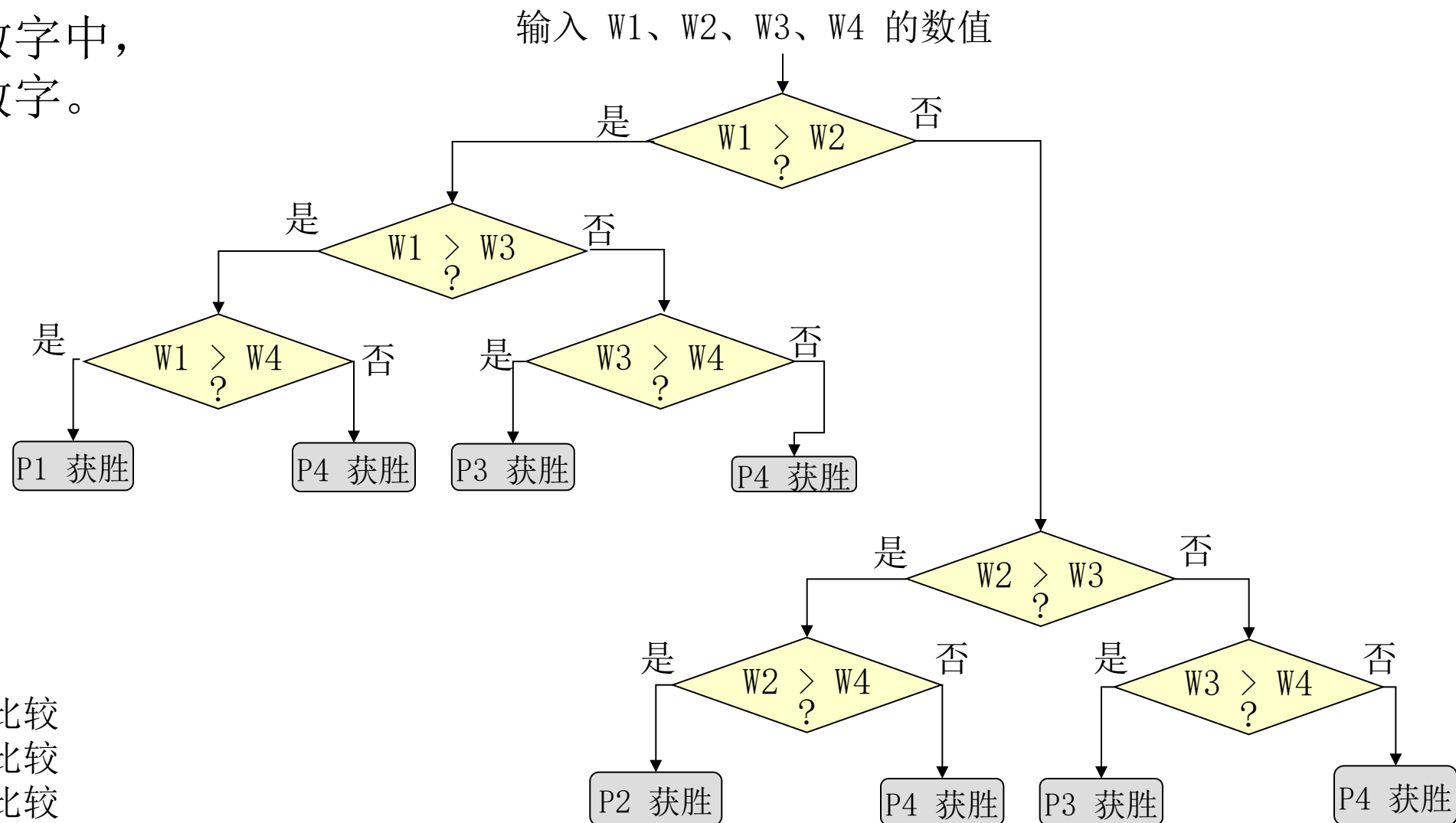
运动员 2: $W2$ = 从键盘输入一个数字

运动员 3: $W3$ = 从键盘输入一个数字

运动员 4: $W4$ = 从键盘输入一个数字

算法 1（流程图）

从四个输入数字中，
找到最大的数字。



复杂度分析

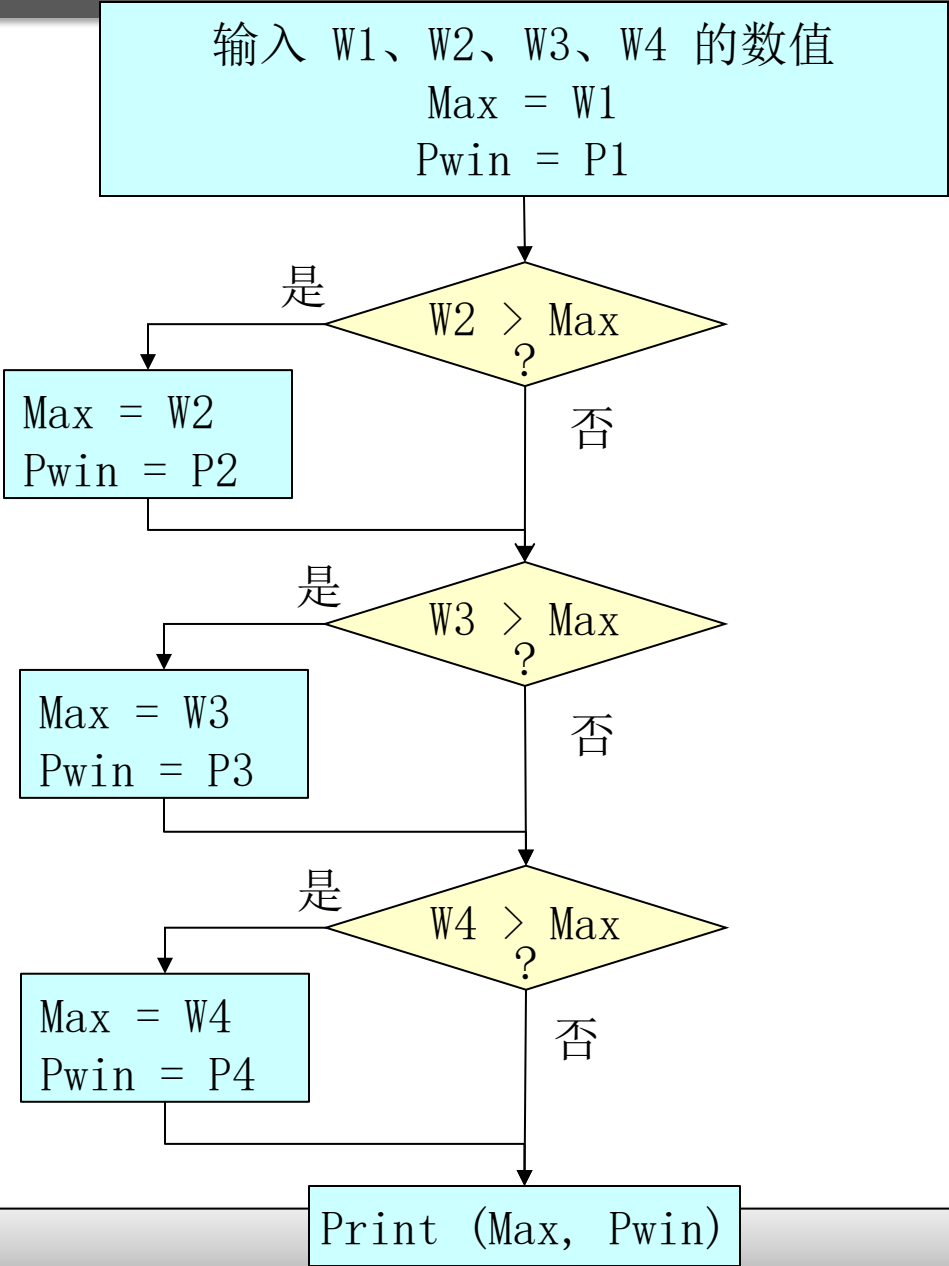
最坏情况：3 次比较

平均情况：3 次比较

最佳情况：3 次比较

算法 2（流程图）

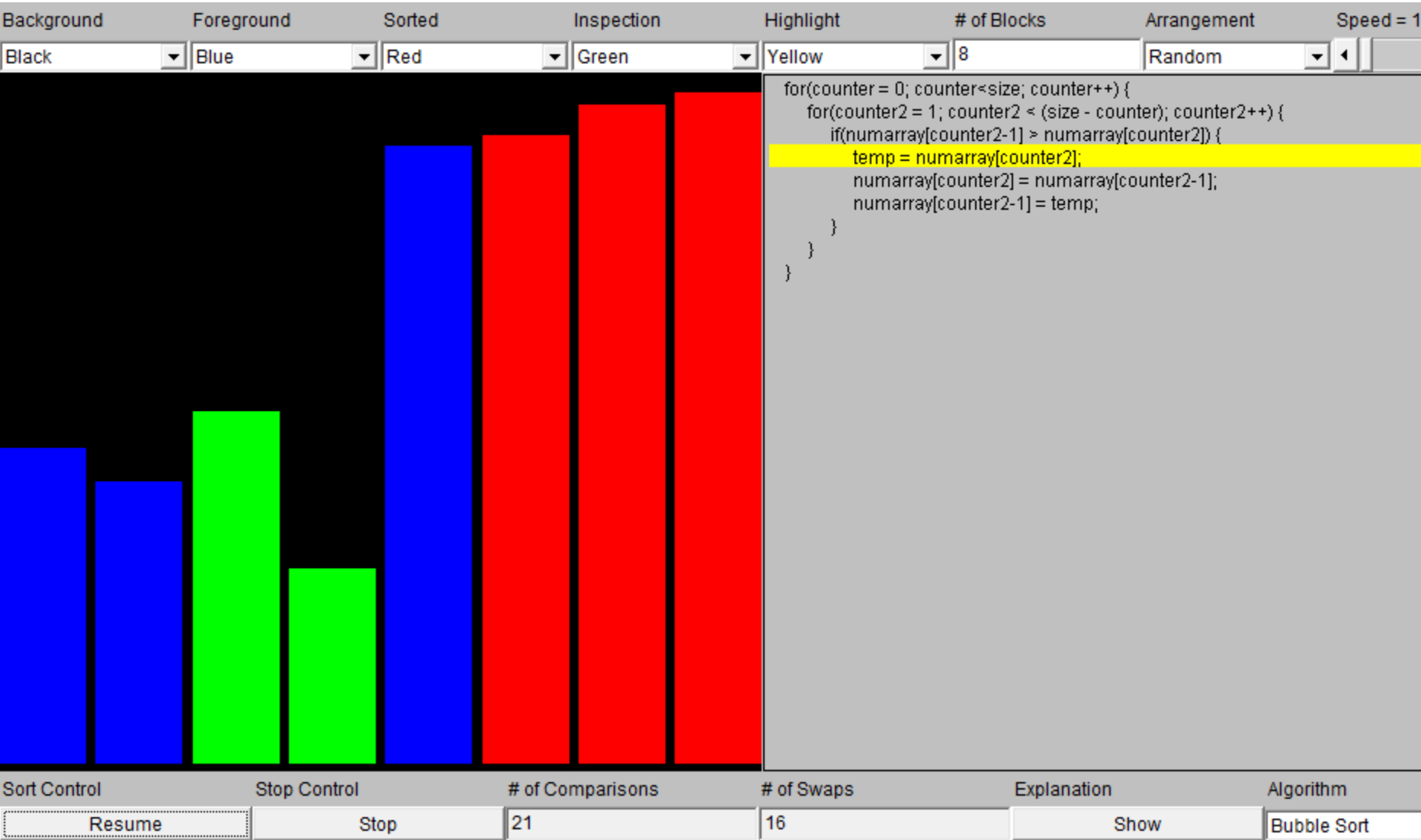
将最大重量放入 Max 中；
将举起重量最大的运动员放入 Pwin 中



复杂度分析
最佳情况：
3 次比较
2 次赋值
最坏情况：
3 次比较
8 次赋值
平均情况：
3 次比较
5 次赋值

2+
4+4+4+
6+6+6+
8 = 40
→ 40/8 = 5

算法的数字排序：冒泡排序

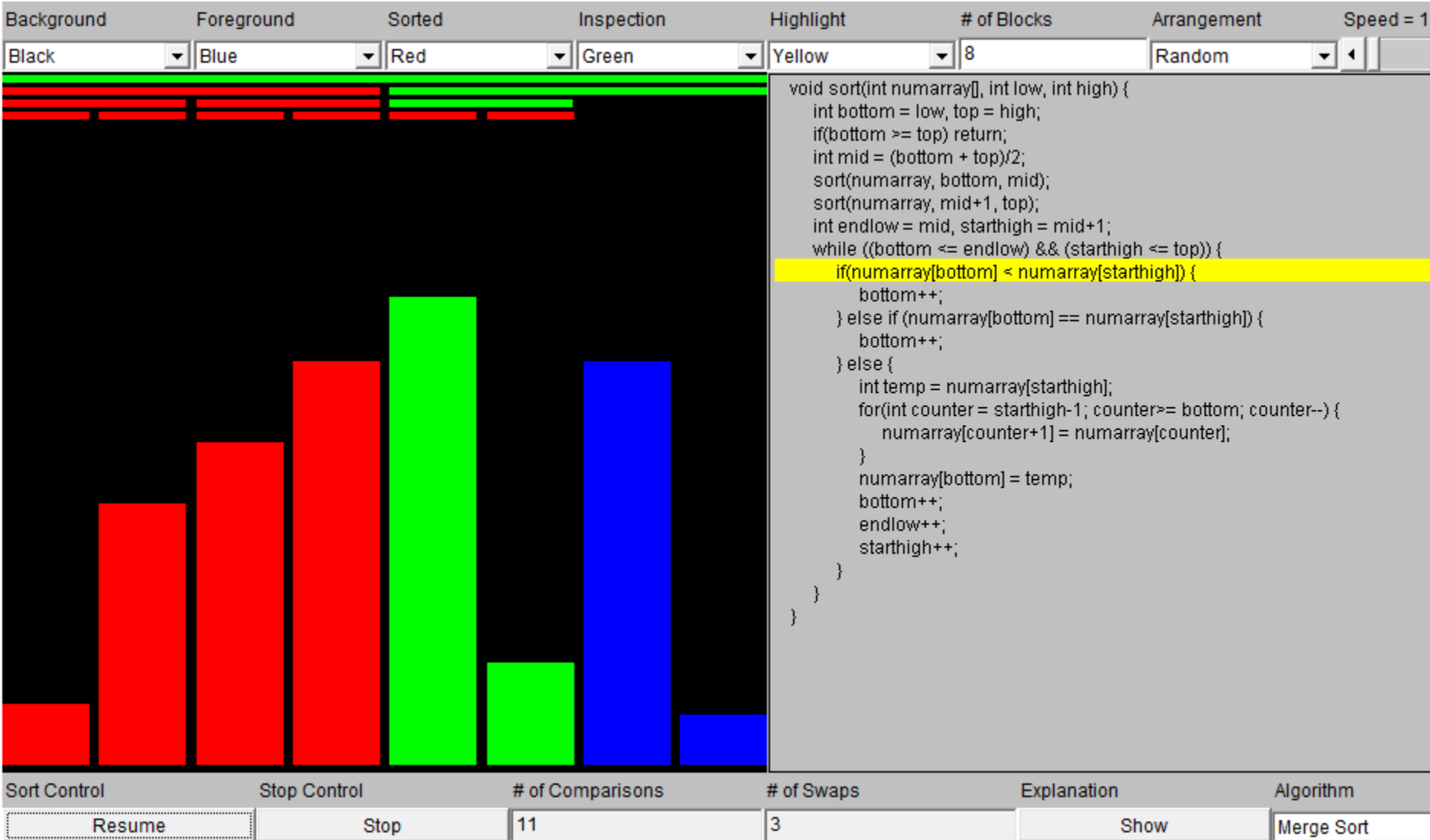


冒泡排序：
给 8 个数字排序，
需进行 28 次比较和
19 次交换。

给 80 个数字排序，
需进行 3160 次比较和
1469 次交换。

相关动画链接：
<https://math.hws.edu/eck/js/sorting/xSortLab.html>

算法的数字排序：归并排序



归并排序：
给 8 个数字排序，
需进行 32 次比较和
9 次交换。

给 80 个数字排序，
需进行 800 次比较
和 195 次交换。

冒泡排序：
给 8 个数字排序，需进行
28 次比较和 19 次交换。
给 80 个数字排序，需进
行 3160 次比较和 1469
次交换。

算法复杂度分析

针对**问题较大**时算法需要使用的**时间**（运算次数）和**空间**（内存）。

不关注输入较小的情况。

大O符号用于估计复杂度的上界。

冒泡排序：

要想给 8 个数字 ($n = 8$) 排序，需进行 28 次比较和 19 次交换。

要想给 80 个数字 ($n = 80$) 排序，需进行 **3160** 次比较和 **1469** 次交换。

复杂度 = $O(n^2)$

大O符号：上界

归并排序：

要想给 8 个数字 ($n = 8$) 排序，需进行 32 次比较和 9 次交换。

要想给 80 个数字 ($n = 80$) 排序，需进行 **800** 次比较和 **195** 次交换。

复杂度 = $O(n \log n)$

大O符号：上界

语言和算法的影响

- ❑ 算法、语言和编译器都会影响性能
- ❑ 在 C 语言和 Java 中比较两种排序算法的性能
 - JVM/JIT 为 Sun/Hotspot 1.3.1/1.3.1 版本
- ❑ 观察：谁的影响更大？

算法的影响

语言	翻译方法	编译器优化级别	冒泡排序	快速排序	对比快速排序与冒泡排序
			相对性能		
C	编译器	无	1.00	1.00	2468
C	编译器	优化级1	2.37	1.50	1562
C	编译器	优化级2	2.38	1.50	1555
C	编译器	优化级3	2.41	1.91	1955
Java	解释程序	—	0.12	0.05	1050
Java	JIT 编译器	—	2.13	0.29	338

资料来源：Computer Organization and Design:The Hardware/ Software Interface
by David A. Patterson (UC Berkeley) and John L. Hennessy (Stanford Univ.)

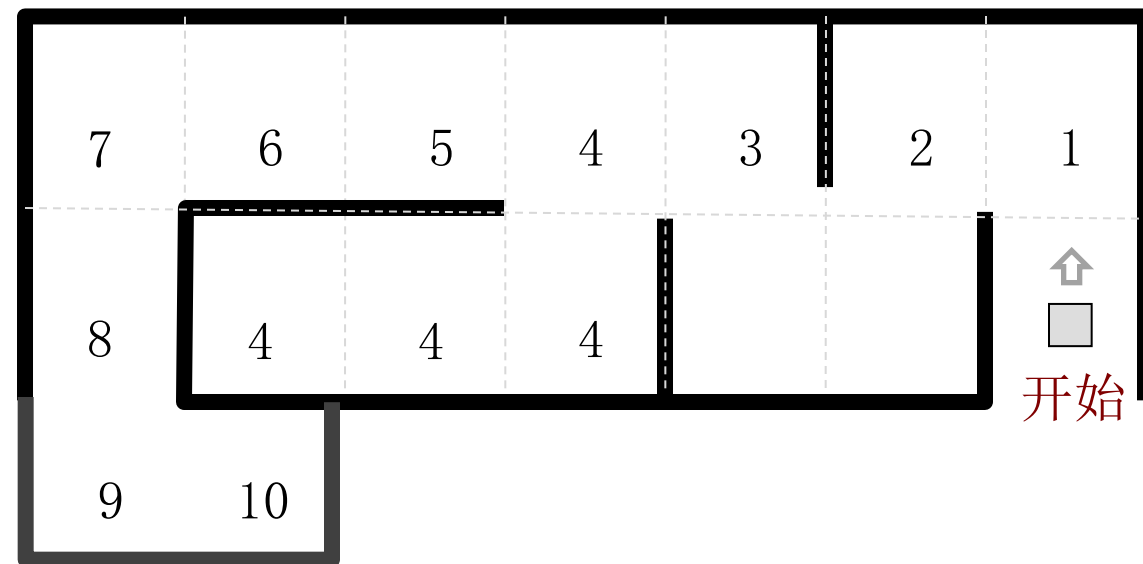
RISC 架构发明者
斯坦福大学校长 Alphabet
执行主席
2018 年图灵奖获得者

确定关键步骤：迷宫导航算法

❑ 处理代表机器人动作的计算步骤和机械步骤。

❑ 哪个部分更耗时？

- 转湾的角度数
- 行动距离



❑ 利用从起点到出口（终点）所需的转动次数或角度以及距离单元；


❑ 评估不同的算法

- 随机算法
- 局部最佳决策的启发式算法

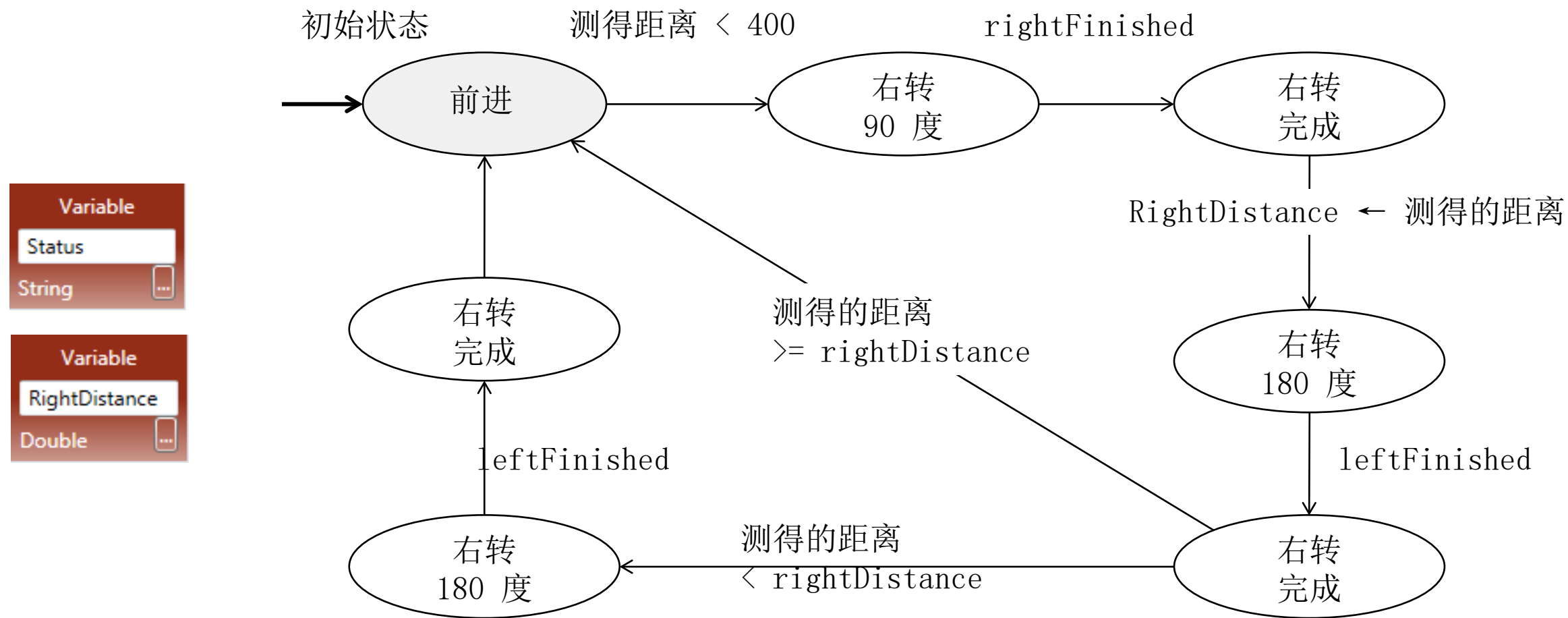
➡ • 沿墙算法

➡ • 基于第一个可用解的贪婪算法

自动迷宫分步遍历算法

- 
1. 机器人被设置为“前进”状态并向前移动;
 2. 如果距离传感器测得的前方距离小于 400 毫米, 则机器人将向右转动 (90 度);
 3. 在事件 “rightFinished” 发生后, 它将把测得的距离保存到变量 RightDistance;
 4. 然后, 机器人会向左旋转 180 度, 测量另一侧的距离;
 5. 在事件 “leftFinished” 发生后, 它将测得的距离与保存在变量 RightDistance 中的数值进行比较;
 6. 如果当前的距离更长, 则跳到步骤 1 并继续向前;
 7. 反之, 则 (旋转 180 度) 回到另一个方向;
 8. 然后, 跳到步骤 1 并继续向前。

有限状态机的自主迷宫遍历算法



分步的左侧沿墙算法

- ➡ 1. Variable DV = 传感器测得的左侧初始距离
- 2. 向前移动时，机器人会循环重复以下步骤，直到按下触控传感器；
 - 1) 机器人会在一定的时间间隔内持续测量左侧距离，并将新测得的距离与存储在 variable DV 中的距离进行比较。
 - 2) 如果新测得的距离大于 $DV+1$ ，机器人则会向左转动一度，再返回到步骤 2；//调整与墙的距离
 - 3) 如果新测得的距离小于 $DV-1$ ，机器人则会向右转动一度，再返回到步骤 2；
 - 4) 如果新测得的距离大于 $2*DV$ ，机器人则会向左转动 90 度，再返回步骤 2；
- 3. 触控传感器被按下；机器人向后移动 0.5 圈，再向右转动 90 度；
- 4. 返回到步骤 2。

有限状态机的左侧沿墙算法

