



算法分析与设计

Analysis and Design of Algorithm

任课教师：金嘉晖 副教授

办公室：计算机楼368

Email: jjin@seu.edu.cn

助教：吴碧伟（220191682@seu.edu.cn）

什么是算法 (Algorithm)

输入

2 4 3 1 5

{排序算法}

输出

1 2 3 4 5

起点: 东南大学-桃园食堂

终点: 东南大学-计算机楼

{寻路算法}



{图像识别算法}

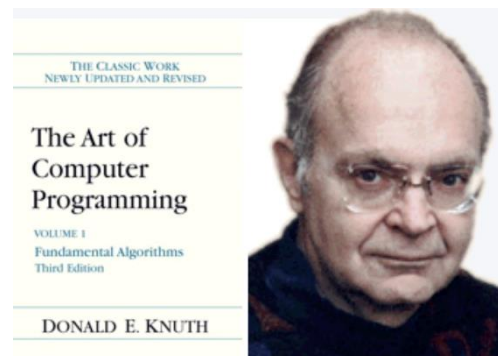
Cat

什么是算法 (Algorithm)

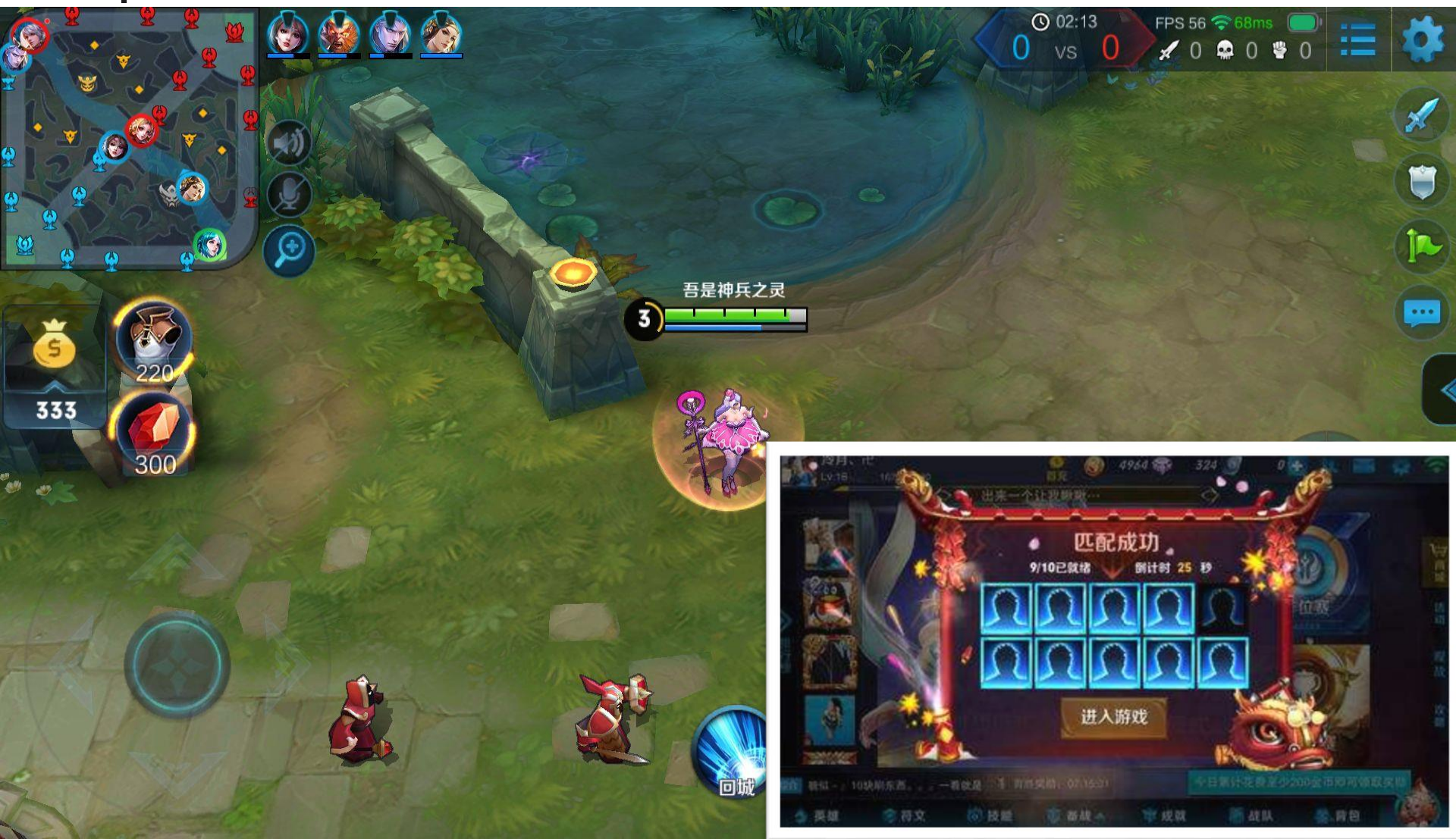
百度百科：算法是指解题方案的准确而完整的描述，是一系列解决问题的清晰指令，算法代表着用系统的方法描述解决问题的策略机制。



算法和程序设计技术的先驱者Donald Knuth：算法是带有输入输出的、有限的、确定的、有效的过程。



算法有哪些应用



算法有哪些应用



算法有哪些应用

互联网：网页搜索、网络路由、BitTorrent...

生物信息：人类基因组计划、蛋白质结构分析...

计算机图形：电影、游戏、虚拟现实...

计算机安全：手机、电子商务、投票系统...

多媒体：MP3、JPG、HDTV...

人工智能：人脸识别、AlphaGo、聊天机器人...

社会网络：推荐系统、新闻推送、广告...

物理学：离子对撞机、反物质暗物质探测...



本科生学算法有什么用

求职面试： 谷歌、百度、阿里…

读研究生： 考研面试、论文专利…

创业就业： 开发产品、性能优化…





算法课与其他课程的关系

编译
原理

操作
系统

计算机
网络

数据库
原理

算法分析与设计

数据结构与算法

程序设计基础及语言

面向对象程序设计



本课程的内容

NP完全性理论与近似算法

算法高级理论

随机化算法

线性规划与网络流

高级算法

递归
分治

动态
规划

贪心
算法

回溯与
分支限界

基础算法

算法分析与问题的计算复杂性

算法基础理论



参考书目

主要教材

- 王晓东. 算法设计与分析. 电子工业出版社

参考教材

- Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman. The Design and Analysis of Computer Algorithms. 1974年影印本, 铁道出版社
- Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman. 数据结构与算法. 1983年影印本, 清华大学出版社
- Thomas H. Cormen 等4人. 算法导论. MIT第2版影印本, 高教出版社



课程信息

■ 考核方式

- 平时（点名、作业、互动）：**30%**；期末（考试）：**70%**

■ 作业要求

- **习题类型**：算法分析题+算法实现题
- **截至日期**：一周之内完成
- **提交形式**：电子版（关于算法实现题，电子版需要截图、源码、可执行程序）
- **提交方式**：发邮件至jjin@seu.edu.cn，抄送220191682@seu.edu.cn
邮件格式：算法作业X_学号_姓名（X为第几次，如1,2,3……）
- **作业格式**：推荐用Latex编写作业（下载CTex并自学Latex）

■ 答疑方式

- 课程QQ群、计算机楼368室、联系助教



第一章 算法概述

■ 学习要点:

- 理解算法的概念
- 理解什么是程序，程序与算法的区别和联系
- 掌握描述算法的方法
- 掌握算法的计算复杂性概念
- 掌握算法渐近复杂性的数学表述
- 了解NP类问题的基本概念

基础知识和算法设计

什么是算法

输入

2 4 3 1 5

{排序算法}

输出

1 2 3 4 5

起点：东南大学-桃园食堂

终点：东南大学-计算机楼

{寻路算法}



{图像识别算法}

Cat



什么是算法

- 算法(Algorithm)

- 一个（由人或机器进行）关于某种运算规则的集合



确定性 清晰、无歧义

有限性 指令执行次数、时间

- 特点：

- 执行时，不能包含任何主观的决定；
 - 不能有类似直觉/创造力等因素。

例子：

- 日常生活中做菜的过程，可否用算法描述？

✓ 如：“淡了”、“放点盐”、“再煮一会”。

✓ 可否用计算机完成？



- 算法必须规定明确的量与时间；
- 不能含糊字眼。

算法描述：

```
if (咸度 < 5) { 放1克盐; 煮半分钟; }
```

随机算法与近似算法

- 当然不是所有算法都要明确的选择，有些概率算法进行选择。

- “随机” 不等于 “随意”



- 有些问题没有实用算法（算精确解需要多年）。

✓ 去寻找{规则集}

启发式规则

在可接受的时间内可以算出足够好的近似解



算法和数据结构的关系

- 广义上讲，**算法是某一系列运算步骤**，它表达解决某一类计算问题的一般方法，对这类方法的任何一个输入，它可以按步骤一步一步计算，最终产生一个输出。
- 但是对于所有的计算问题，都离不开要计算的對象或者要处理的信息，而**如何高效的把它们组织起来，就是数据结构关心的问题**，所以算法是离不开数据结构的。

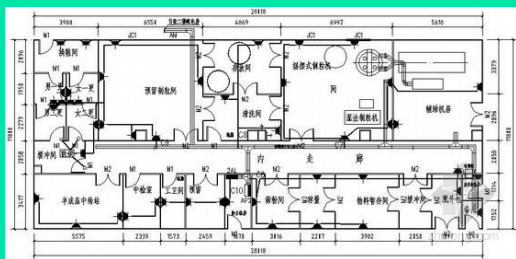


算法和程序的关系

- 程序是算法用某种程序设计语言的具体实现
- 算法和程序的区别主要在于：
 - 在语言描述上，程序必须是用规定的程序设计语言来写，而算法很随意；
 - 在执行时间上，算法所描述的步骤一定是有限的，而程序可以无限地执行下去。
- 例如：操作系统是程序，但不是算法

编写程序如同建房子

设计图纸（算法）



建筑材料（数据结构）



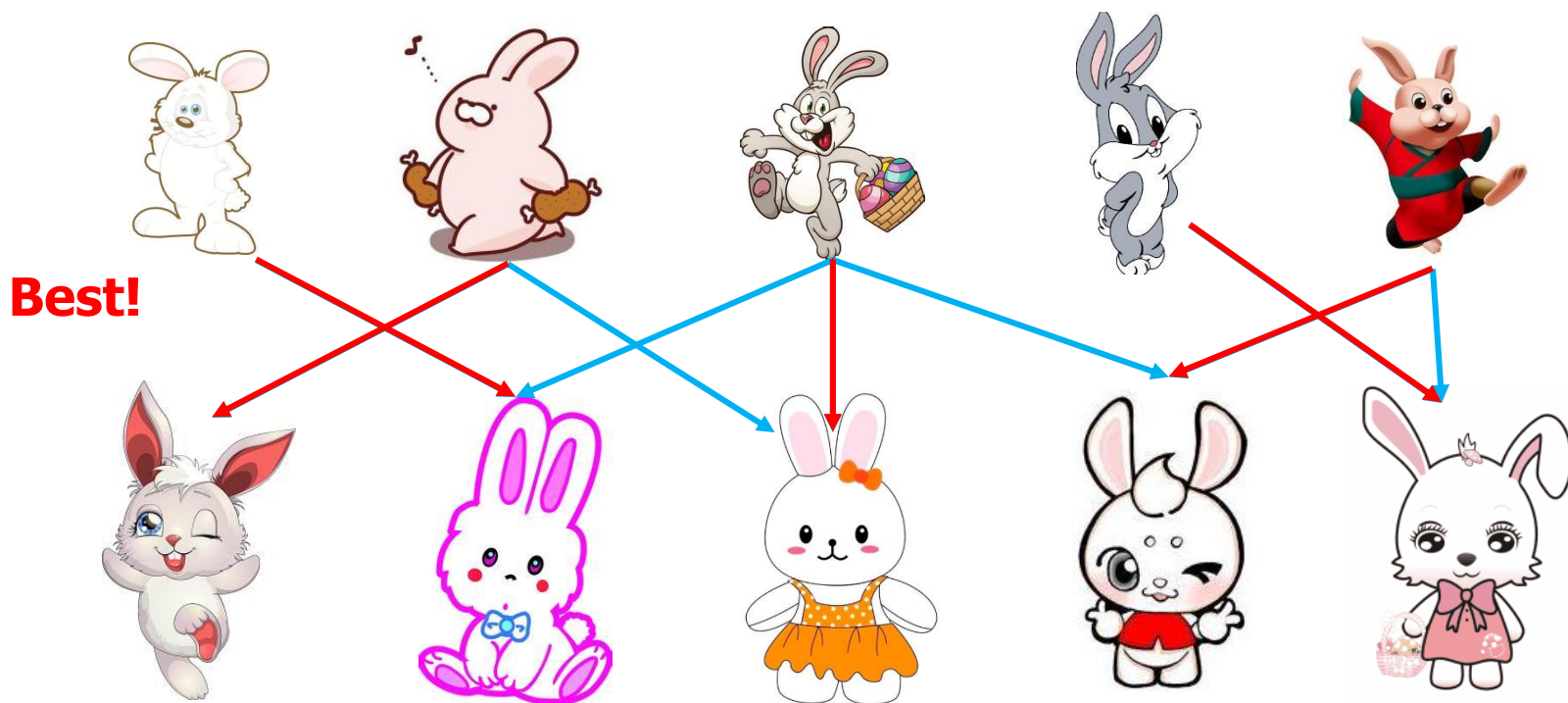
房子（程序）



你们聊！
我搬砖去了

如何设计一个算法（例1）

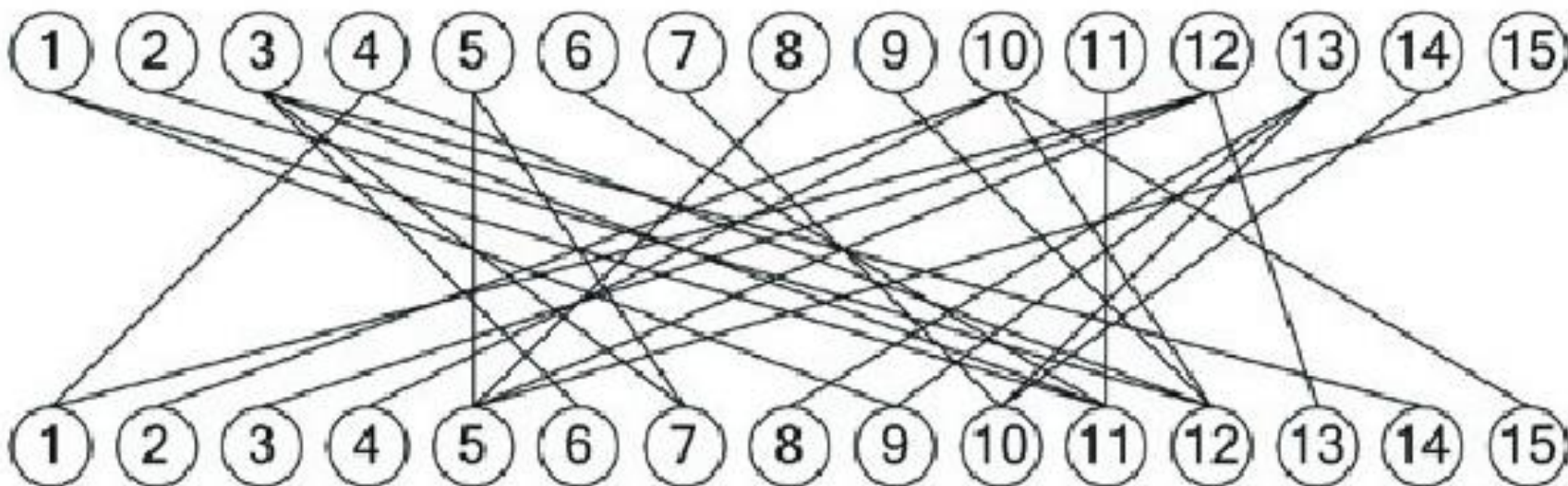
草原上生活着一群可爱的兔子，有5只公兔子和5只母兔子，每只公兔子都对若干母兔子有好感。如何帮助尽可能多的公兔子找到伴侣？



如何设计一个算法（例1）

草原上生活着一群可爱的兔子，有5只公兔子和5只母兔子，每只公兔子都对若干母兔子有好感。如何帮助尽可能多的公兔子找到伴侣？

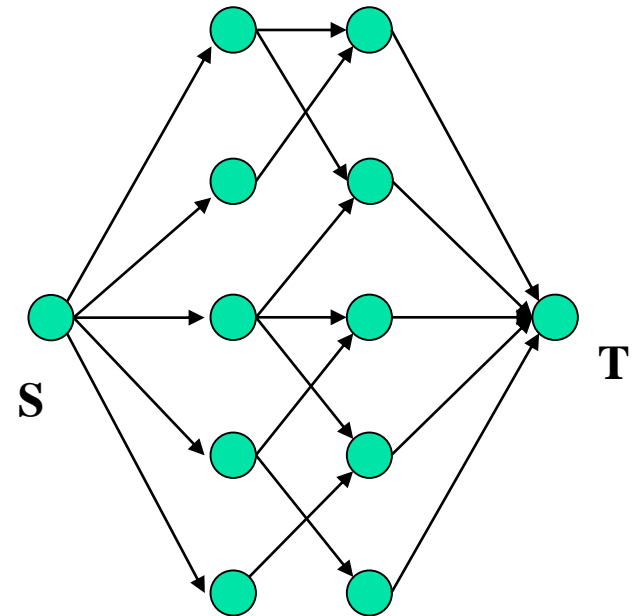
如果兔子数量很多，该如何算？



如何设计一个算法（例1）

- 步骤一：抽象为数学问题
 - 将公兔子和母兔子看作图中的点
 - 若公兔子a喜欢母兔子b，则在a和b之间连一条边
 - 增加一个源点S和终点T

最后，兔子匹配问题抽象成为
最大流问题（第8章）



- 步骤二：将求解过程用计算机语言描述
(较为复杂，此处省略，见第8章)



如何设计一个算法（例2）

每个月一对成熟的兔子会产生一对后代，而这对后代2个月后又会繁殖。即第1个月有1对小兔子；第2个月仍只有1对；第3个月有2对...假设兔子不死亡，**请问10个月后草原有几对兔子？**

	1月	2月	3月	4月	5月
刚出生	1	0	1	1	2
一个月	0	1	0	1	1
成熟	0	0	1	1	2
总数	1	1	2	3	5



如何设计一个算法（例2）

- 兔子数序列：0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...
- 步骤一：抽象为数学问题
 - Fibonacci序列
 - 序列的表达式
$$\begin{cases} f_0 = 0; & f_1 = 1 \\ f_n = f_{n-1} + f_{n-2}, & n \geq 2 \end{cases}$$
- 步骤二：将求解过程用计算机语言描述

```
int Fibonacci(int n){  
    if(n<2) then return n  
    else return Fibonacci(n-1) + Fibonacci(n-2)  
}
```

更多算法例子请见：1.2 算法设计的两个例子



如何描述算法

- 算法的描述方法
 - 自然语言
 - 流程图
 - 程序设计语言：C、C++、Java、Python、...
 - 伪代码(Pseudocode)——算法语言
 - 类C/类Pascal语言
 - 结构化编程语言



插入排序算法的插入操作

输入

5	7	1	3	6	2	4
---	---	---	---	---	---	---

前面已经排好序，插入2

插入2

1	3	5	6	7	2	4
---	---	---	---	---	---	---

插入后

1	2	3	5	6	7	4
---	---	---	---	---	---	---

插入排序算法的运行实例

输入

5	7	1	3	6	2	4
---	---	---	---	---	---	---

初始

5	7	1	3	6	2	4
---	---	---	---	---	---	---

插入7

5	7	1	3	6	2	4
---	---	---	---	---	---	---

插入1

1	5	7	3	6	2	4
---	---	---	---	---	---	---

插入3

1	3	5	7	6	2	4
---	---	---	---	---	---	---

插入6

1	3	5	6	7	2	4
---	---	---	---	---	---	---

插入2

1	2	3	5	6	7	4
---	---	---	---	---	---	---

插入4

1	2	3	4	5	6	7
---	---	---	---	---	---	---



如何描述算法（文字描述）

插入排序算法描述

输入： 长度为 n 的数组 T

输出： 排好序的数组

过程：

- 1、依次遍历 $T[1]$, $T[2]$, ..., $T[n-1]$, 为每个 $T[i]$ 执行步骤2-4;
- 2、令 $x=T[i]$, $j=i-1$, 然后执行步骤3-4, 以确定 x 在排序后数组中的位置
- 3、如果 $x < T[j]$ 且 $j \geq 0$, 则令 $T[j+1] = T[j]$, $j=j-1$, 并重新执行步骤3, 否则执行步骤4
- 4、令 $T[j+1]=x$
- 5、输出数组 T



如何描述算法（类C描述）

Algorithm 1 插入排序算法 Insert

Input: 待排序数组 T .

Output: 排序完成数组 T .

类C语言

```
template<class Type>
void Insert(Type T[], int n)
//从第 2 列到第 n 个元素,
//把该元素插入到之前的数组相应位置上
    for(int i=1; i<n; i++){
        Type x = T[i]; int j = i-1;
        while(j >= 0 && x < T[j]){
            T[j+1] = T[j]; j = j-1;
        }
        T[j+1] = x;
    }
}
```



小结

- 算法的概念
- 算法和数据结构的关系
- 算法和程序的关系
- 设计算法的方法
- 描述算法的方法

算法分析



如何选择算法

- 当解决一个问题时，存在几种算法可供选择，如何决定哪个最好？
- 以排序算法为例：

算法	最坏情况	平均情况
插入排序	$O(n^2)$	$O(n^2)$
冒泡排序	$O(n^2)$	$O(n^2)$
快速排序	$O(n^2)$	$O(n\log n)$
归并排序	$O(n\log n)$	$O(n\log n)$

冒泡排序算法的运行实例

输入	5	7	1	3	6	2	4
巡回1	5	1	3	6	2	4	7
巡回2	1	3	5	2	4	6	7
巡回3	1	3	2	4	5	6	7
巡回4	1	3	2	4	5	6	7
巡回5	1	2	3	4	5	6	7
巡回6	1	2	3	4	5	6	7

快速排序一次递归运行

输入

5	8	1	3	6	2	4	7
---	---	---	---	---	---	---	---

交换1

5	4	1	3	6	2	8	7
---	---	---	---	---	---	---	---

交换2

5	4	1	3	2	6	8	7
---	---	---	---	---	---	---	---

划分

2	4	1	3	5	6	8	7
---	---	---	---	---	---	---	---

子问题

2	4	1	3	5	6	8	7
---	---	---	---	---	---	---	---



二分归并排序运行实例

输入

5	8	1	3	6	2	4	7
---	---	---	---	---	---	---	---

划分

5	8	1	3	6	2	4	7
---	---	---	---	---	---	---	---

递归
排序

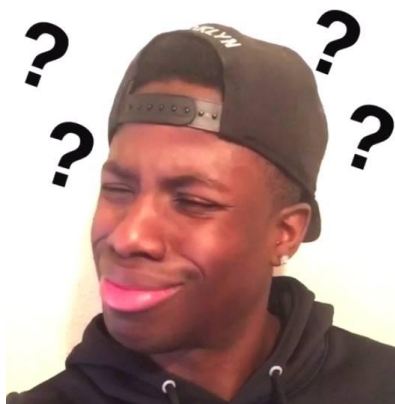
1	3	5	8	2	4	6	7
---	---	---	---	---	---	---	---

1	3	5	8	2	4	6	7
---	---	---	---	---	---	---	---

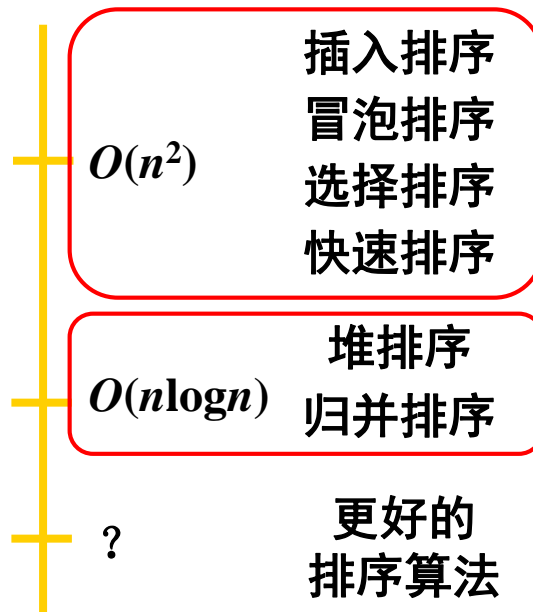
问题的计算复杂度分析

■ 问题

- 哪个排序算法效率最高？
- 是否可以找到更好的排序算法？
- 排序问题计算难度如何？
- 问题计算复杂度的估计方法



哪个排序算法效率最高？
如何分析排序问题计算难度？



评估算法的执行效率

■ 两种评估方法：

- **经验(Empirical)**：对各种算法编程，用不同实例进行实验；
- **理论(Theoretical)**：以数学化的方式确定算法所需要资源数与实例大小之间函数关系。

算法效率 → 算法的快慢

★ 时间/空间

某些方法实例中某些构件数的数量

- 排序：以参与排序的项数表示实例大小；
- 讨论图时，常用图的节点/边来表示

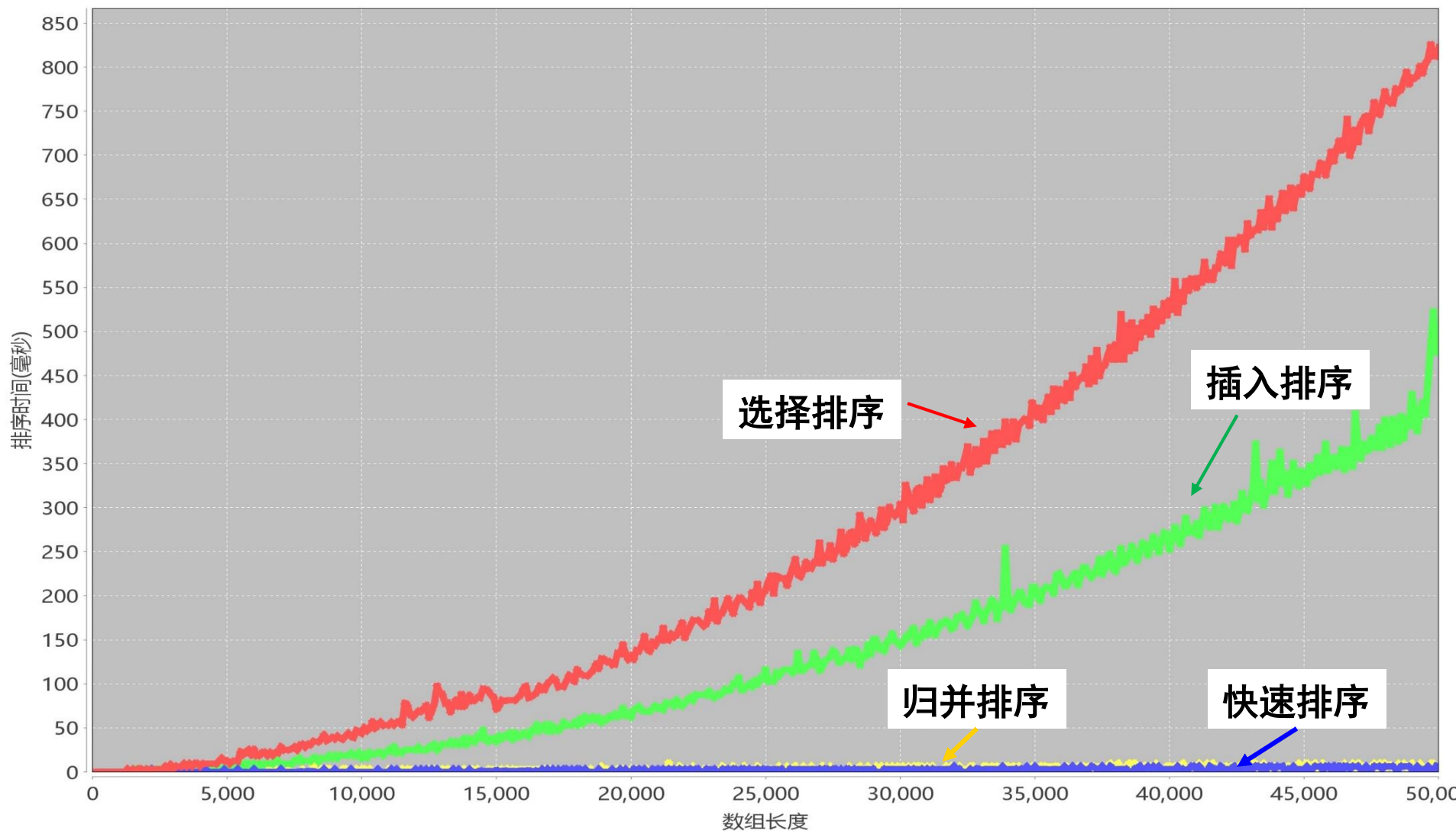


基于经验的评估方法

- 实验方案
 - 生成 n 个随机数并进行排序
($n=100, 200, \dots, 25000$)
 - 记录各个算法的排序时间
 - 用图的形式画出来

基于经验的评估方法

排序算法执行时间比较图



经验法存在的问题

- 经验法的问题
 - 依赖于计算机
 - 依赖于语言/编程技能
 - 需要一定的编程/调试时间
 - 只能评估部分实例的效率



理论法优点：既不依赖于计算机，也不依赖于语言/编程技能。节省了无谓编程时间；可研究任何在实例上算法效率

基于理论的评估方法

■ 执行时间的估计

- 定义评估函数 $T(n)$, n 为输入数据规模
- 如果存在一个正的常数 c , 而该算法对每个大小为 n 的实例的执行时间都不超过 $cT(n)$ 秒
- → 该算法的开销在 $T(n)$ 级内。

为什么定义常数 c ?



Apple I
CPU MOS 6502
@ 1 MHz



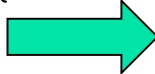
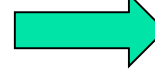
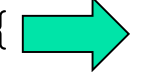
2017

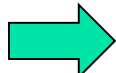
iMac Pro
CPU Intel Xeon W
@ 3.2GHz 八核

回顾：冒泡排序算法

输入	5	7	1	3	6	2	4
巡回1	5	1	3	6	2	4	7
巡回2	1	3	5	2	4	6	7
巡回3	1	3	2	4	5	6	7
巡回4	1	3	2	4	5	6	7
巡回5	1	2	3	4	5	6	7
巡回6	1	2	3	4	5	6	7

冒泡排序复杂度 $T(n)$

```
void bubbleSort(T[] a) {  
    int len = a.length;  只做1次  
    for (int i=0; i<n; i++) {  n次循环  
        for (int j=0; j<n-i-1; j++) {  最多n-1次内循环  
            if (a[j]>a[j+1]) {  
                int temp = a[j];  
                a[j] = a[j+1];  
                a[j+1] = temp;  
            }  
        }  
    }  
}
```

排序时间 $\leq 1+c*n(n-1)$  $T(n) = n^2$

回顾：插入排序算法

输入

5	7	1	3	6	2	4
---	---	---	---	---	---	---

初始

5	7	1	3	6	2	4
---	---	---	---	---	---	---

插入7

比较1次

5	7	1	3	6	2	4
---	---	---	---	---	---	---

插入1

比较2次

1	5	7	3	6	2	4
---	---	---	---	---	---	---

插入3

比较3次

1	3	5	7	6	2	4
---	---	---	---	---	---	---

插入6

比较2次

1	3	5	6	7	2	4
---	---	---	---	---	---	---

插入2

比较5次

1	2	3	5	6	7	4
---	---	---	---	---	---	---

插入4

比较4次


1	2	3	4	5	6	7
---	---	---	---	---	---	---

插入排序复杂度 $T(n)$

```
void insertSort(T[] a) {  
    int n = a.length;           ➡ 只做1次  
    T temp;  
    for (int i = 1; i < n; i++) { ➡ n-1次循环  
        temp = a[i];  
        int j = i - 1;  
        while (j > 0 && a[j] > temp) { ➡ 最多n-1次内循环  
            a[j + 1] = a[j];  
            j--;  
        }  
        a[j + 1] = temp;  
    }  
}
```

排序时间 $\leq 1 + c * (n-1)(n-1) \Rightarrow T(n) = n^2$

思考



冒泡排序和插入排序的复杂度 $T(n)$ 都是 n^2 ,
为什么插入排序实际执行时间比冒泡排序短?



平均和最坏情况分析

- 设 u 和 v 是两个长度为 n 的数组， u 中元素已按升序排序； v 按降序排序。

数组 u

1	2	3	4	5	6	7
---	---	---	---	---	---	---

数组 v

7	6	5	4	3	2	1
---	---	---	---	---	---	---



平均和最坏情况分析

数组 u

1	2	3	4	5	6	7
---	---	---	---	---	---	---

插入排序: n 次比较

冒泡排序: $n(n-1)/2$ 次比较

数组 v

7	6	5	4	3	2	1
---	---	---	---	---	---	---

插入排序: $n(n-1)/2$ 次比较

冒泡排序: $n(n-1)/2$ 次比较

平均和最坏情况分析

- 两个插入排序实例的**时间差**随着元素个数**增加**而**增加**。
 - 算法解决一个实例的时间取决于最坏情况(worst case)。
- 最精确的是分析算法的平均响应时间：
 - 例如：对于插入排序的时间开销在 n 到 n^2 变动。

已排序

最坏情况

因为存在 $n!$ 种初始排列，所以最好求出 $n!$ 种初始排列时间的最坏情况
最坏情况的时间 \geq 排序一个随机次序数组的时间



小结

- 算法的重要性
- 算法的课程框架
- 算法的概念和特点
- 算法、数据结构、程序之间的关系
- 几种排序算法的比较
- 评估算法的性能的两种方法
- 平均和最坏情况分析