

课程内容

- $1 + 2 + 3 + 4 + \cdots + 100 = ?$
- 方法一：
 $1 + 2 = 3$
 $3 + 3 = 6$
 $6 + 4 = 10$
...
 $4950 + 100 = 5050$
- 方法二： $(1 + 100) * 50 = 5050$



数学王子 高斯

课程内容

- $1 + 2 + 3 + 4 + \cdots + 100 = ?$

- 方法一： $1 + 2 = 3$
 $3 + 3 = 6$
 $6 + 4 = 10$
...
 $4950 + 100 = 5050$

- 方法二： $(1 + 100) * 50 = 5050$



数学王子 高斯

显然方法二更高效

课程内容

- 求解同一问题通常存在多种方法，其效率可能不同

课程内容

- 求解同一问题通常存在多种方法，其效率可能不同
- 如何判断哪个算法更高效？
 - 需**分析**比较算法运行效率

课程内容

- 求解同一问题通常存在多种方法，其效率可能不同
- 如何判断哪个算法更高效？
 - 需**分析**比较算法运行效率
- 如何设计正确高效的算法？
 - 需掌握算法**设计**的方法论

课程内容

- 求解同一问题通常存在多种方法，其效率可能不同
- 如何判断哪个算法更高效？
 - 需**分析**比较算法运行效率
- 如何设计正确高效的算法？
 - 需掌握算法**设计**的方法论

设计高效算法，分析算法性能，两者交互验证

提纲

算法的由来

算法的定义

算法的性质

算法的表示

算法的分析

提纲

算法的由来

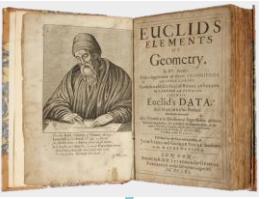
算法的定义

算法的性质

算法的表示

算法的分析

算法的由来

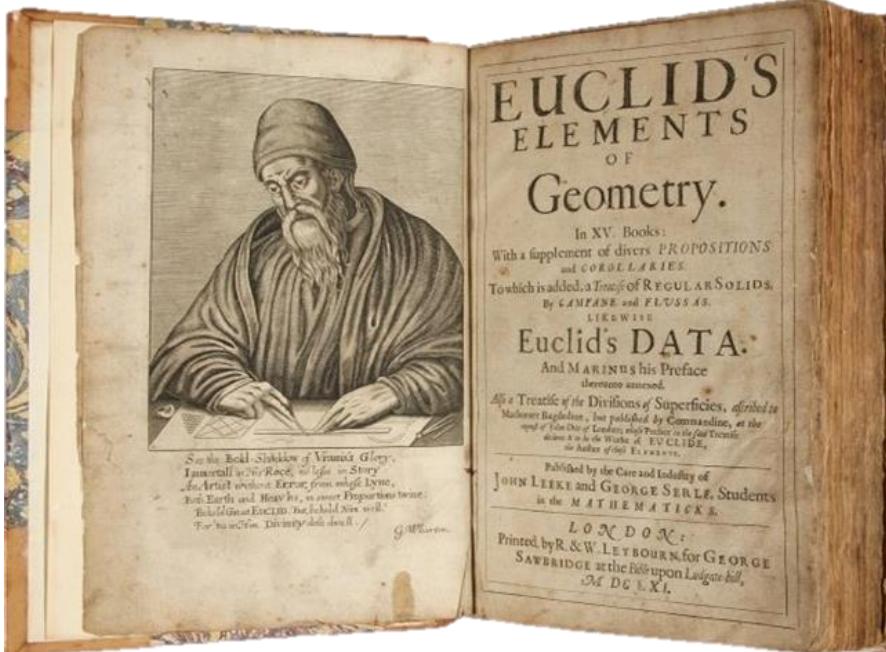


公元前300年
辗转相除法
《几何原本》

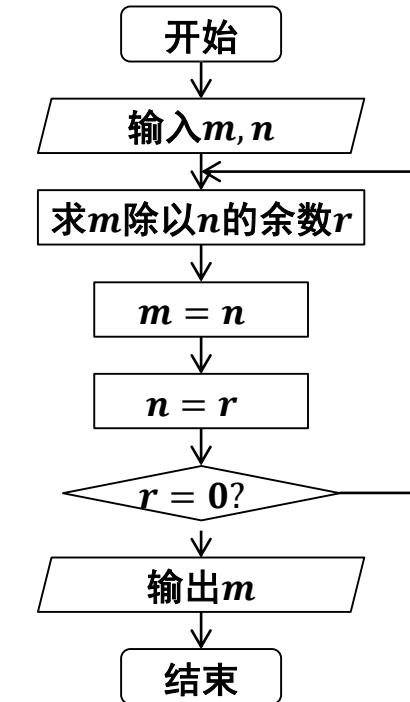
算法的由来

● 辗转相除法

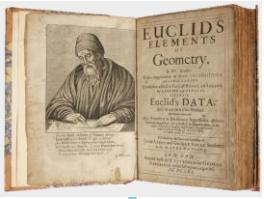
- 用于计算两个整数的最大公约数
- 约公元前300年由欧几里德提出



欧几里得《几何原本》

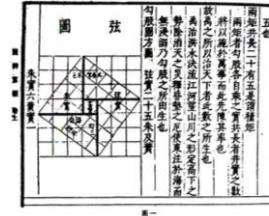


算法的由来



公元前300年
辗转相除法
《几何原本》

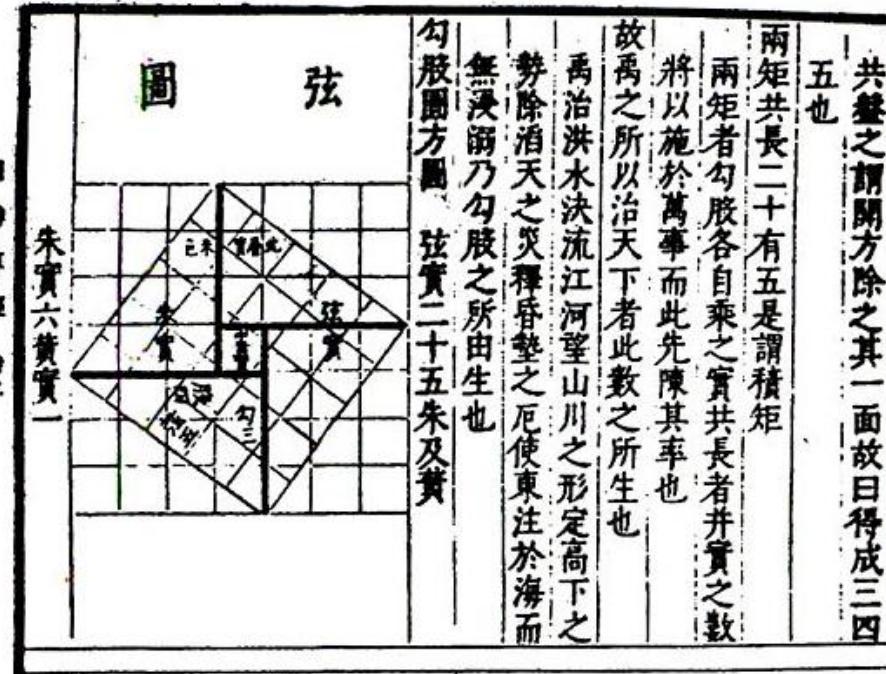
算法
《周髀算经》
公元前1世纪



算法的由来

● 名称由来

- 中文名称“算法”出自约成书于公元前1世纪的《周髀算经》

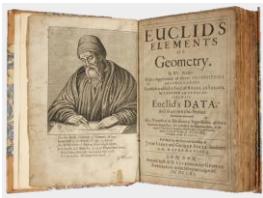


介绍了勾股定理：“以日下为勾，日高为股，勾股各自乘，并而开方除之”

欽定四庫全書	子部六
周髀算經	天文算法類一屬
提要	推步之
臣等謹案周髀算經二卷音義一卷案隋書	
經籍志天文類首列周髀一卷趙嬰注又一 卷甄鸞重述唐書藝文志李淳風釋周髀二 卷與趙嬰甄鸞之注列之天文類而復列李 淳風注周髀算經二卷於曆算類蓋一書重	
欽定四庫全書	
周髀算經	
出也是書內稱周髀長八尺夏至之日晷一 尺六寸蓋髀者股也於周地立八尺之表以 為股其影為勾故曰周髀其首章周公與商 高相問答實勾股之鼻祖故	
御定數理精蘊載在卷首而詳釋之稱為成周六藝 之遺文榮方問于陳子以下徐光啟謂為千 古大愚今詳考其文惟論南北影差以地為 平遠復以平遠測天誠為臆說然與本文已	

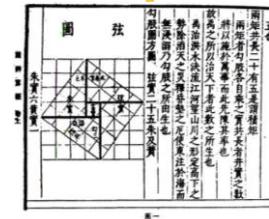
采用简便可行方法确定天文历法
揭示日月星辰运行规律

算法的由来



公元前300年
辗转相除法
《几何原本》

算法
《周髀算经》
公元前1世纪

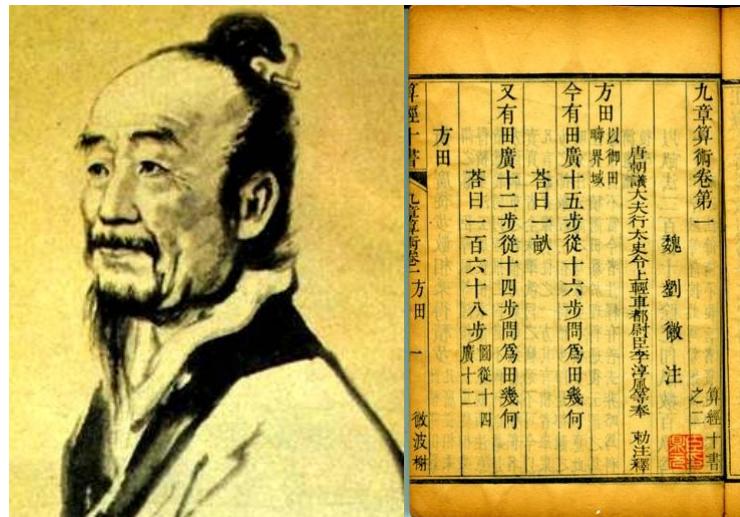


魏晋时期
割圆术
《九章算术注》

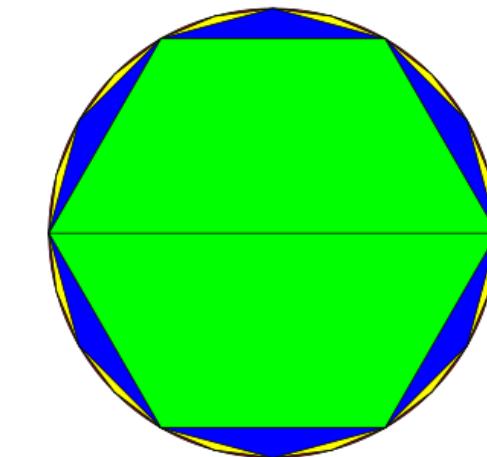
算法的由来

● 割圆术

- 内接正多边形去无限逼近圆，并以此求取圆周率的方法
- 魏晋时期的数学家刘徽在《九章算术注》中首创

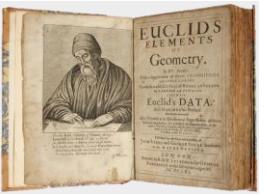


刘徽与《九章算术注》



割圆术示意图

算法的由来



公元前300年
辗转相除法
《几何原本》



7世纪
《算经十书》

算法
《周髀算经》
公元前1世纪



魏晋时期
割圆术
《九章算术注》

算法的由来

● 算经十书

- 唐高宗显庆元年（公元656年），规定将十部汉、唐一千多年间的十部著名数学著作作为国家最高学府的算学教科书，用以进行数学教育和考试，后世通称为《算经十书》



唐高宗李治



周髀算经



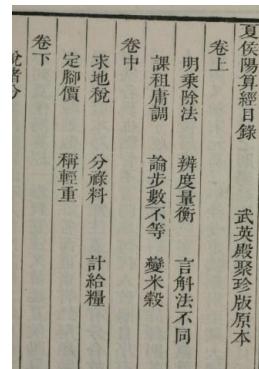
九章算术



海岛算经



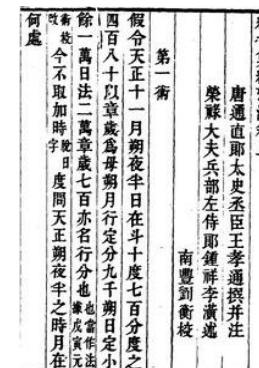
张丘建算经



夏侯阳算经



五经算术



缀术

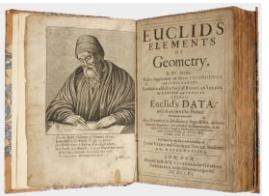


五曹算经



孙子算经

算法的由来



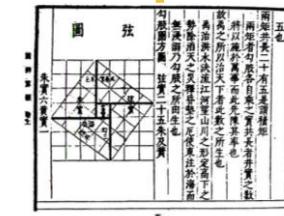
算法
《周髀算经》
公元前1世纪



公元前300年
辗转相除法
《几何原本》



7世纪
《算经十书》



魏晋时期
割圆术
《九章算术注》

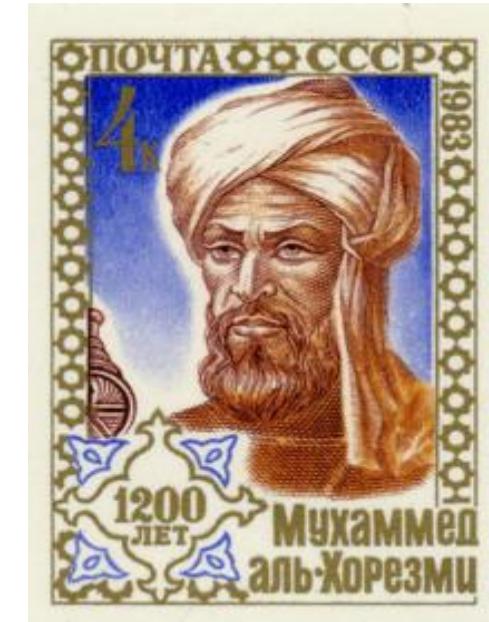
Algorithm
阿尔·花拉子密
9世纪



算法的由来

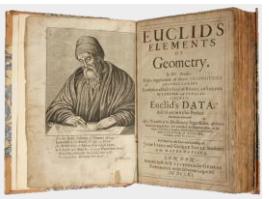
● 名称由来

- 波斯著名的数学家、天文学家、地理学家阿尔·花拉子密在公元825年写成《印度数字算术》一书，对于印度-阿拉伯数字系统在中东及欧洲的传播起到了重要作用
- 该书被翻译成拉丁语“*Algoritmi de numero Indorum*”，花拉子密的拉丁文音译即为“算法”（Algorithm）一词的由来



苏联在1983年发行邮票纪念花拉子密1200岁生辰

算法的由来



算法
《周髀算经》
公元前1世纪



公元前300年
辗转相除法
《几何原本》



7世纪
《算经十书》



Algorithm
阿尔·花拉子密
9世纪



魏晋时期
割圆术
《九章算术注》

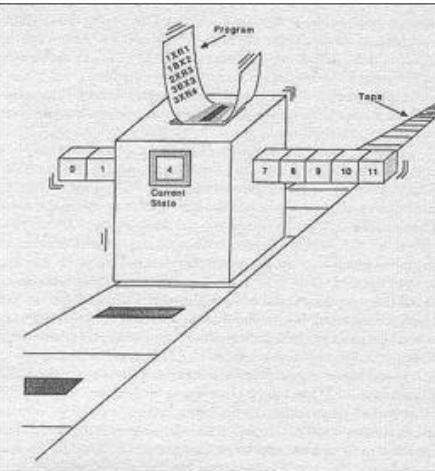


20世纪30年代~40年代
艾伦·图灵与冯·诺依曼

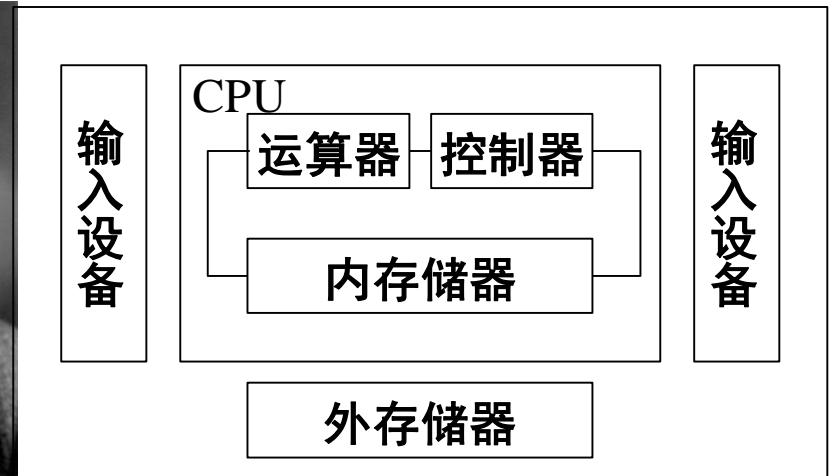
算法的由来

● 算法与计算机的结合

- 1936年，艾伦·图灵提出**图灵机**，通过建立通用计算机模型，刻画计算机的计算行为
- 1946年，冯·诺依曼提出**存储程序原理**



理论计算机科学与人工智能之父
艾伦·图灵
Alan Turing



现代计算机之父
约翰·冯·诺伊曼
John von Neumann

图灵奖

- 1966年由计算机协会(ACM)设立，奖励对计算机事业做出突出贡献的个人



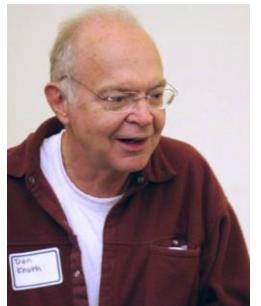
艾伦·图灵
Alan Turing



计算机界的“诺贝尔奖”

在1966~2024年的58年中，共计79位图灵奖获得者

算法与计算复杂性领域图灵奖得主



Donald E. Knuth
1974, USA
算法分析之父



Michael O. Rabin
1976, Israeli
非确定自动机
素数判定随机算法



Dana S. Scott
1976, USA
非确定自动机



Robert W. Floyd
1978, USA
最短路径Floyd算法



Stephen A. Cook
1982, USA
NP完全性



Richard M. Karp
1985, USA
NP完全性与
网络流算法



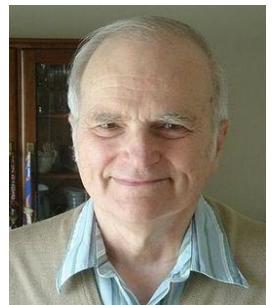
John Hopcroft
1986, USA
最差情况分析
数据结构与算法



Robert Tarjan
1986, USA
数据结构与图算法



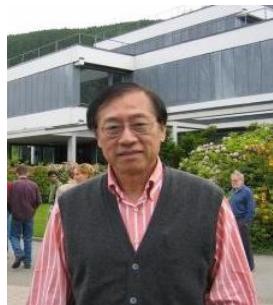
Juris Hartmanis
1993, Latvia
计算复杂性理论



Richard E. Stearns
1993, USA
计算复杂性理论



Manuel Blum
1995, Venezuela
计算复杂性理论



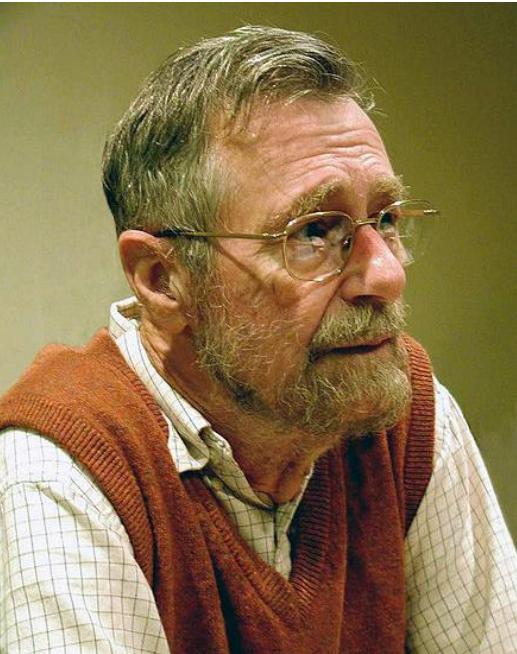
Andrew Yao
2000, China
伪随机数生成
与通信复杂性



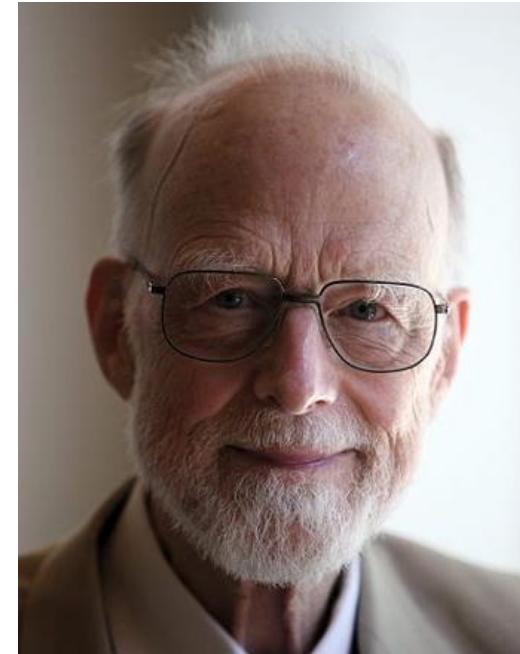
Leslie G. Valiant
2010, Hungarian
#P完全性与
计算学习理论

算法与计算复杂性领域图灵奖得主共计19人，占比约24.1%

其他相关图灵奖得主



Edsger W. Dijkstra
1972, Netherlands
ALGOL之父
提出单源最短路径Dijkstra算法



Tony Hoare
1980, UK
霍尔逻辑
提出快速排序算法

提纲

算法的由来

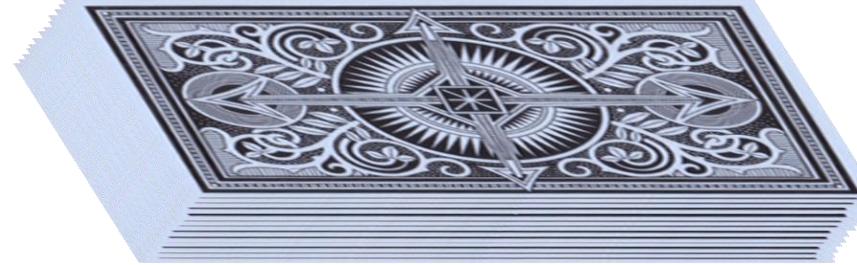
算法的定义

算法的性质

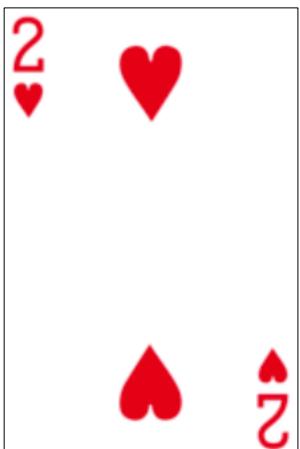
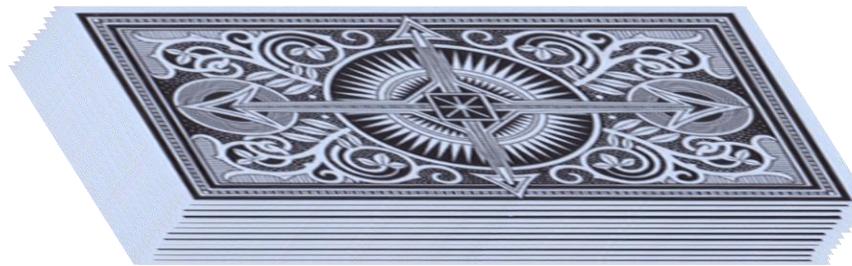
算法的表示

算法的分析

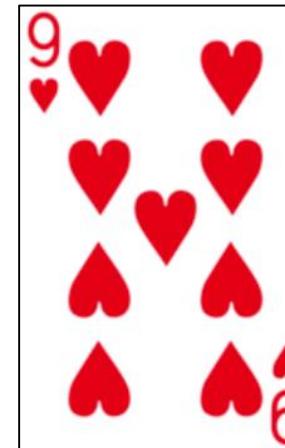
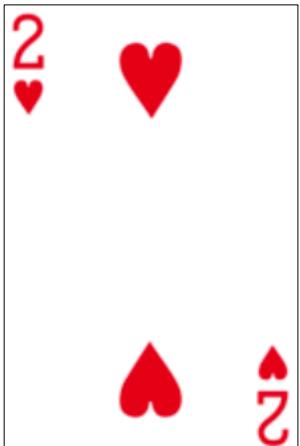
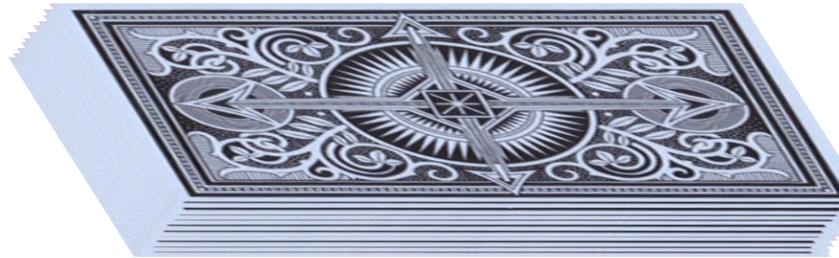
生活中的算法



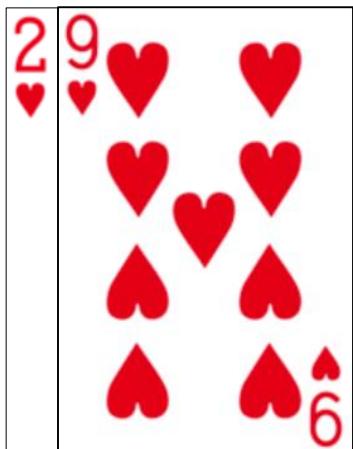
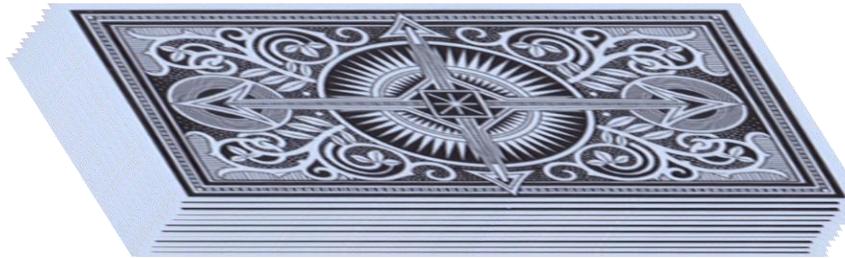
生活中的算法



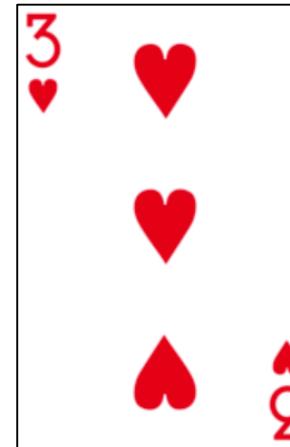
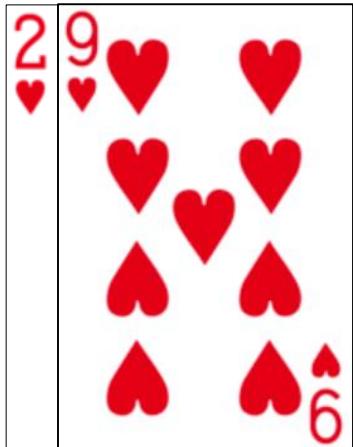
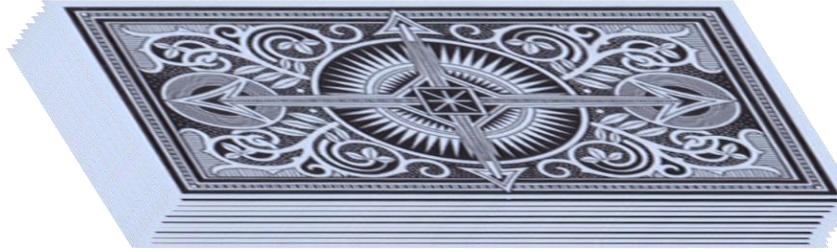
生活中的算法



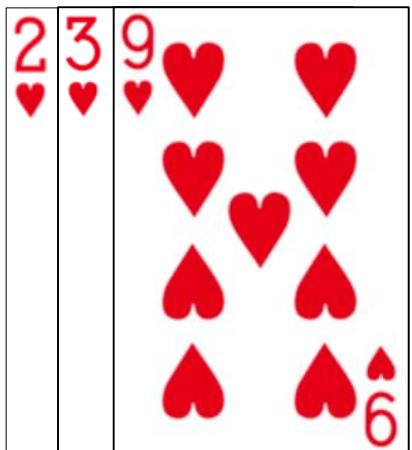
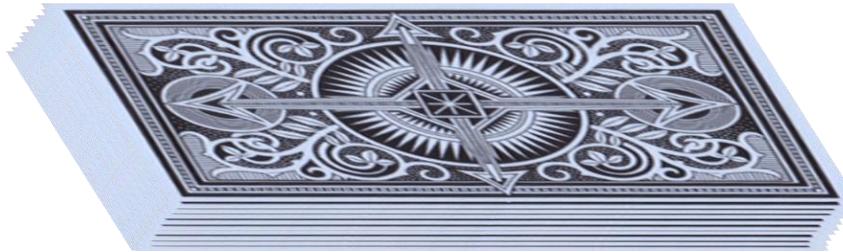
生活中的算法



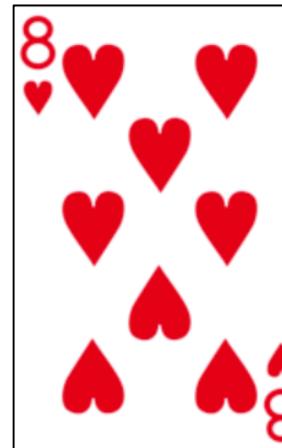
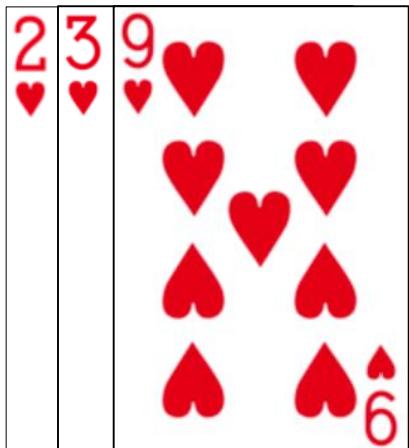
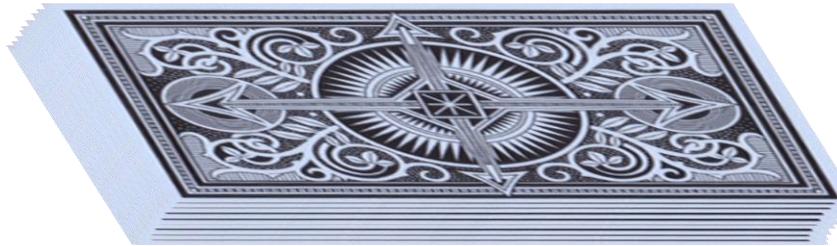
生活中的算法



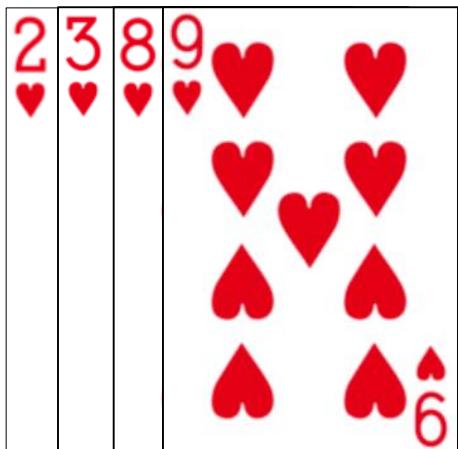
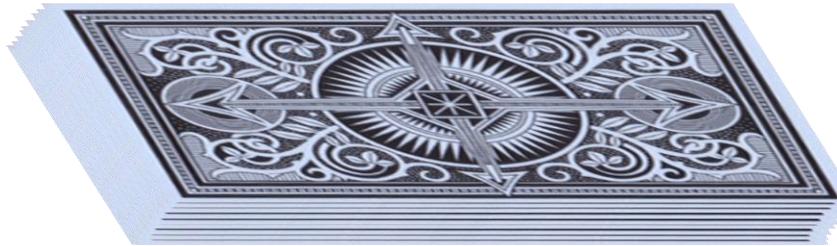
生活中的算法



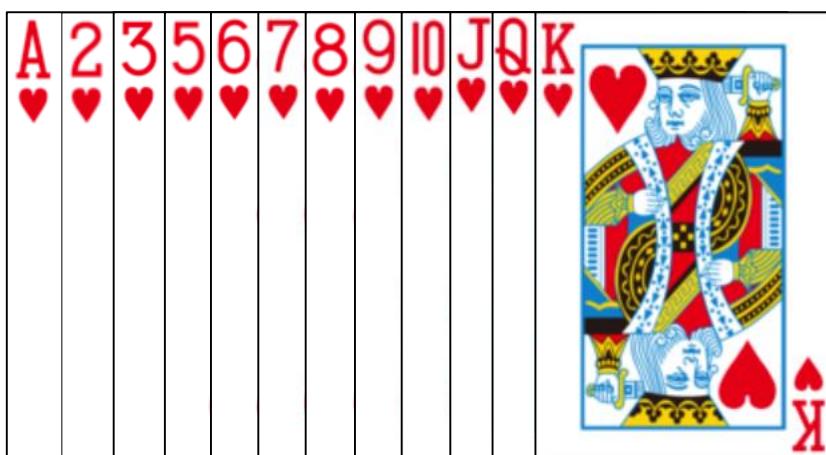
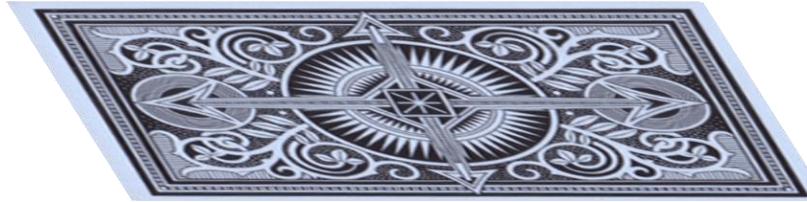
生活中的算法



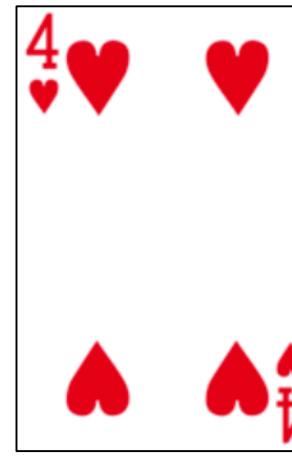
生活中的算法



生活中的算法



生活中的算法



生活中的算法



计算问题

- 定义
 - 给定数据输入，计算满足某种性质输出的问题

计算问题

- 定义
 - 给定数据输入，计算满足某种性质输出的问题
- 示例

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

计算问题

- 定义
 - 给定数据输入，计算满足某种性质输出的问题
- 示例

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

输入：
包含16个数字的数组

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

计算问题

- 定义
 - 给定数据输入，计算满足某种性质输出的问题
- 示例

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

输入：
包含16个数字的数组

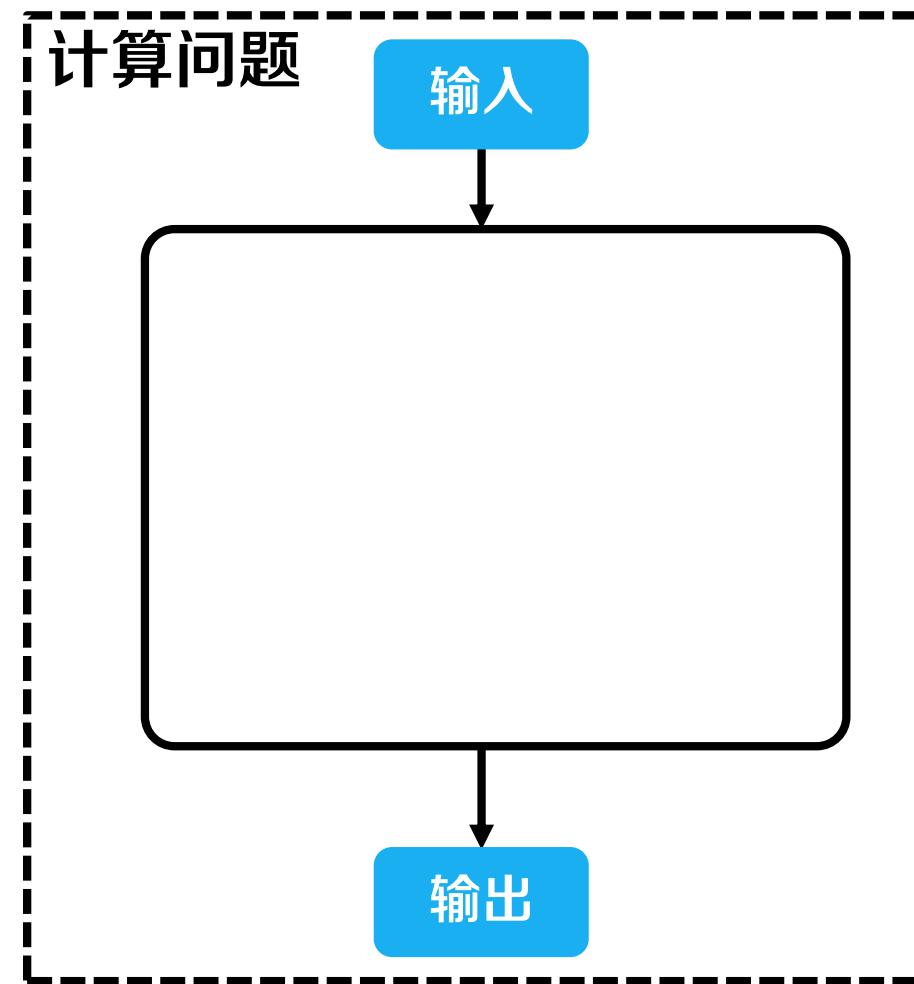
24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

输出：
满足升序性质的输入数组

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

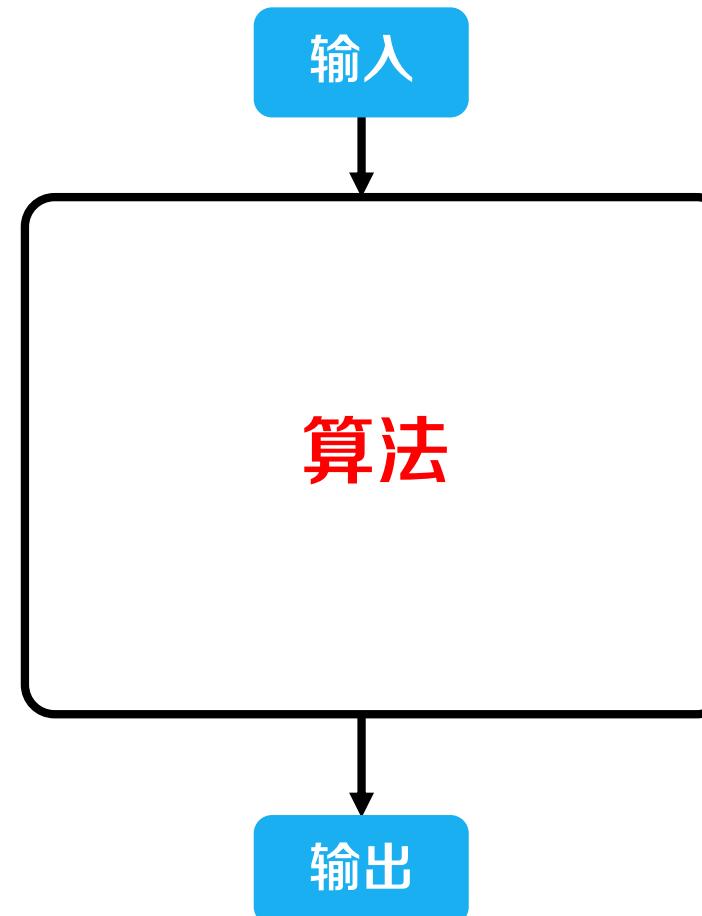
算法的定义

- 给定计算问题



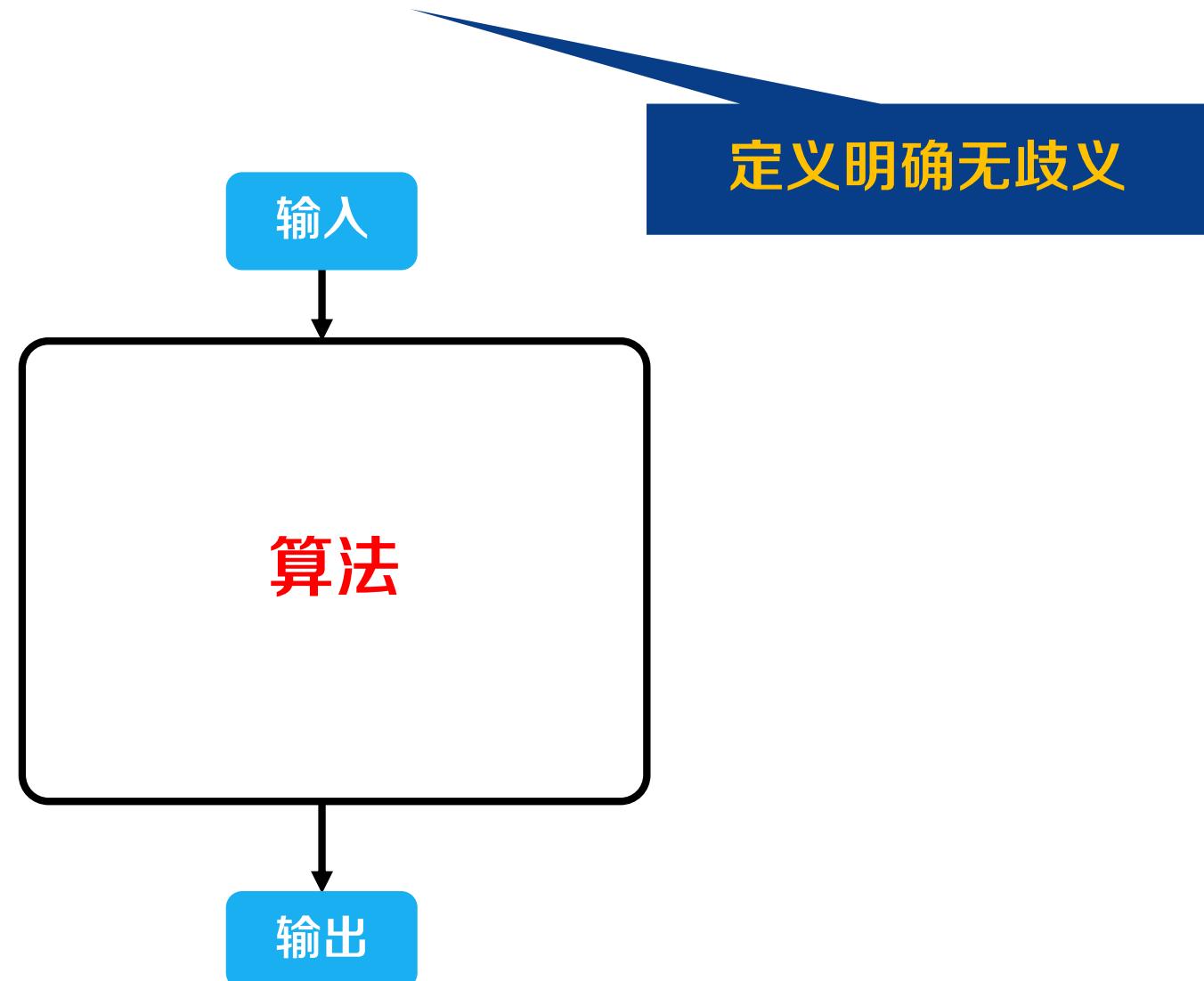
算法的定义

- 给定计算问题，**算法**是一系列良定义的计算步骤



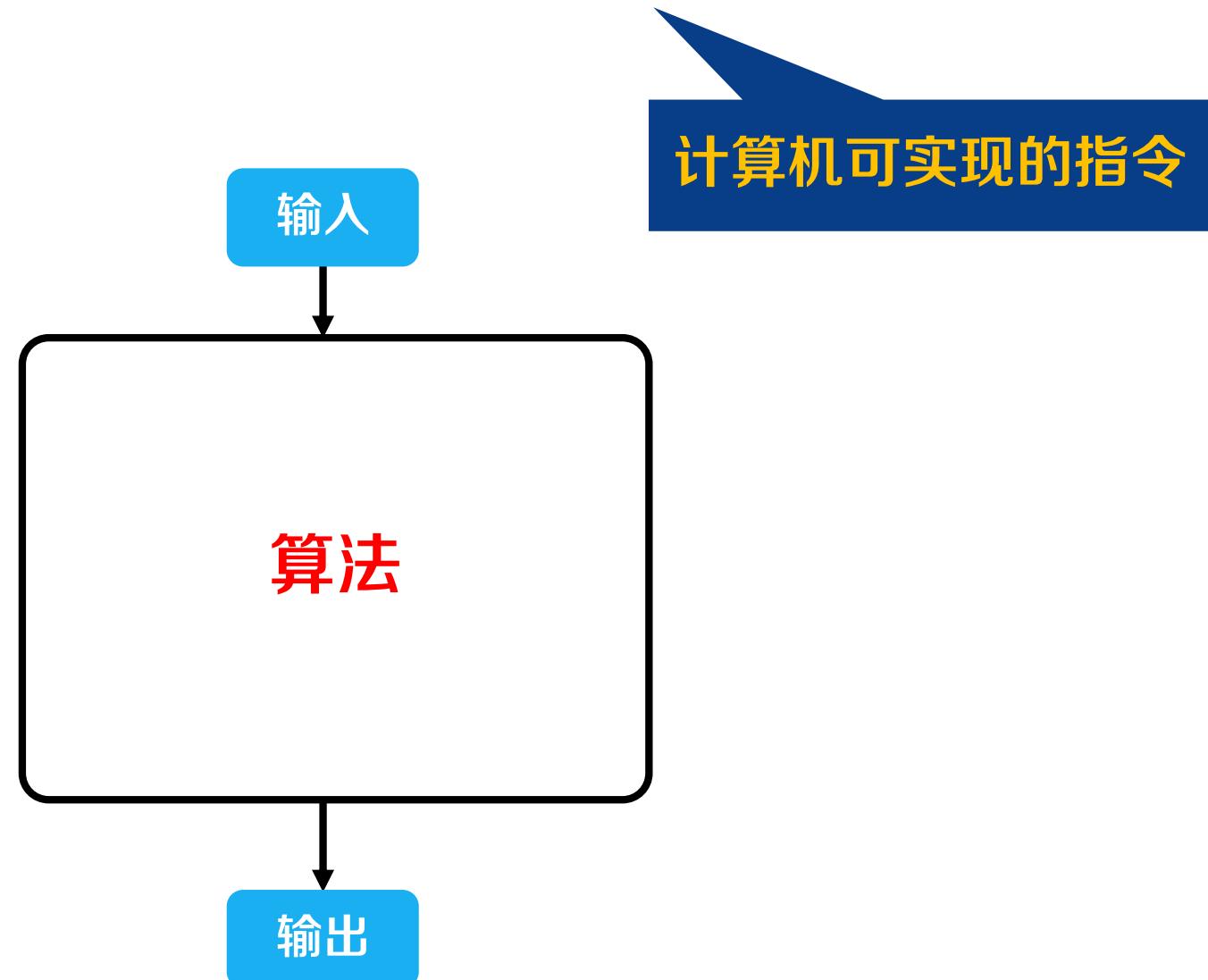
算法的定义

- 给定计算问题，算法是一系列**良定义**的计算步骤



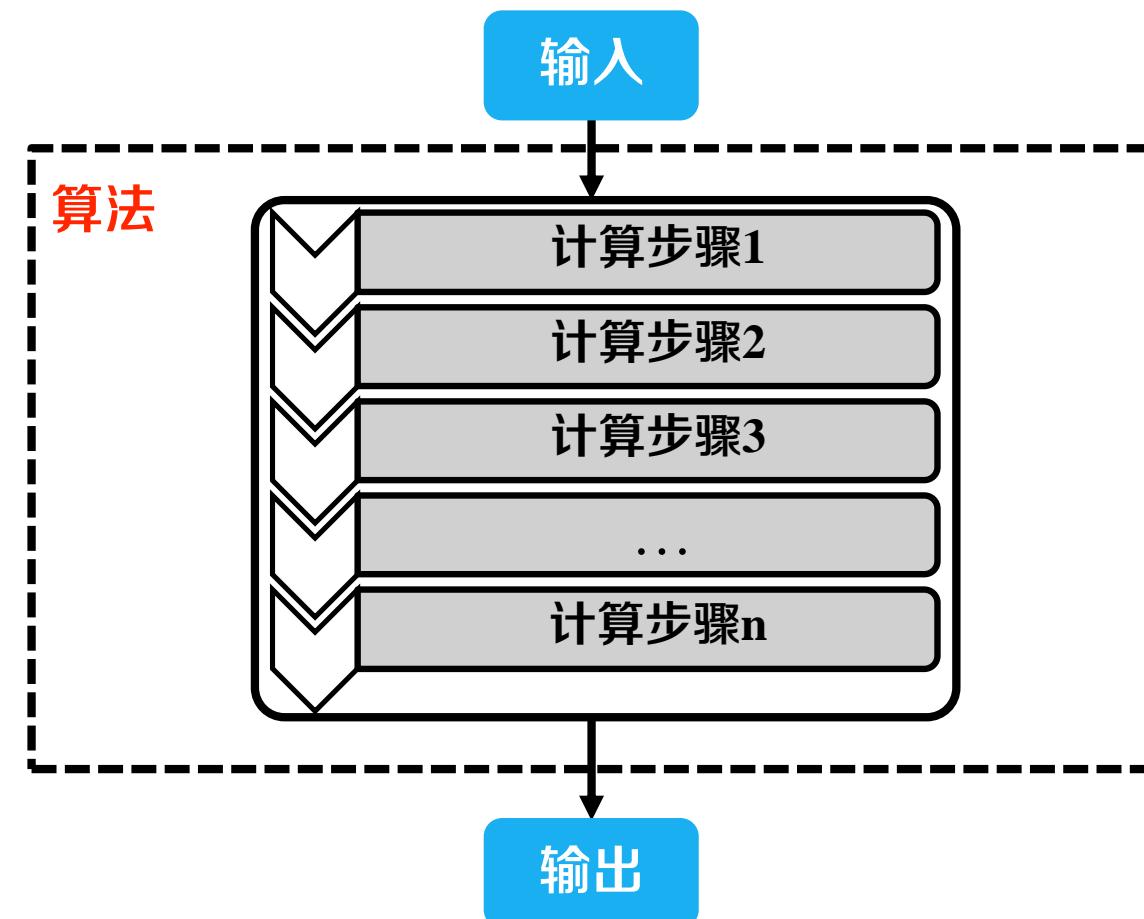
算法的定义

- 给定计算问题，算法是一系列良定义的**计算步骤**



算法的定义

- 给定计算问题，算法是一系列良定义的计算步骤，逐一执行计算步骤即可得预期的输出



插入排序

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

● 插入排序算法

- 将数组待排序元素依次插入到已排序部分，使已排序部分保持升序的性质

插入排序：算法实例

- 输入：

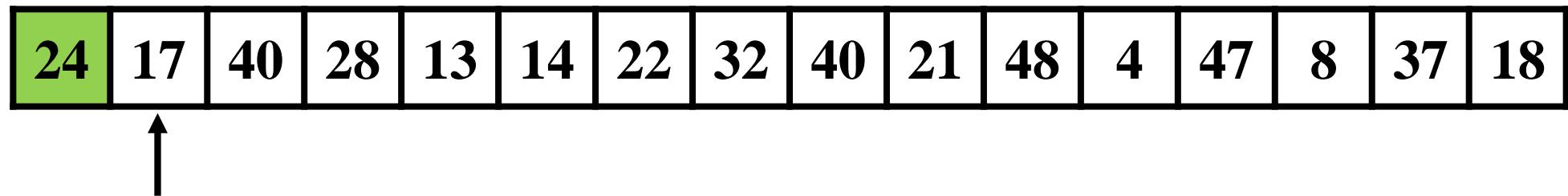
- <24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

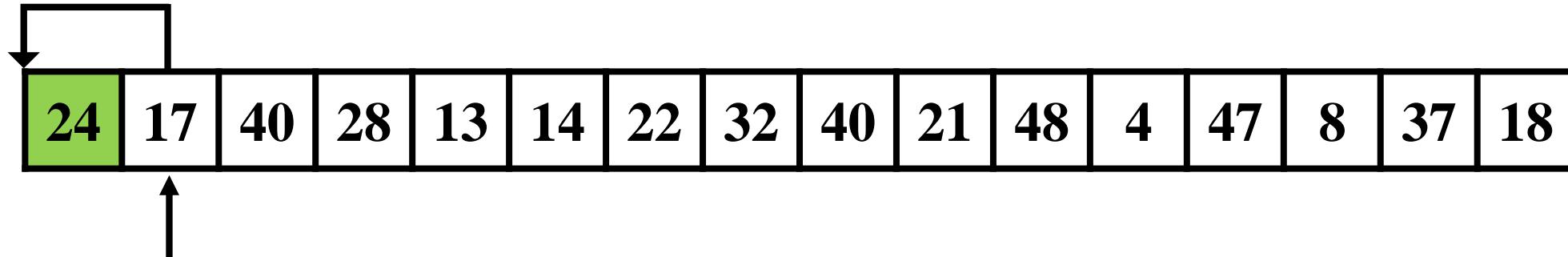
插入排序：算法实例

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

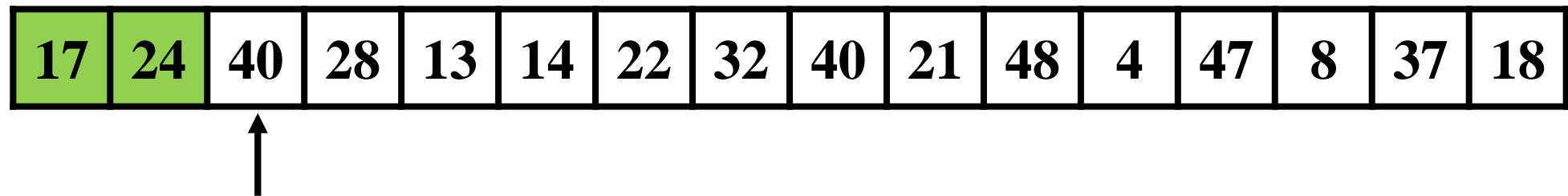
插入排序：算法实例



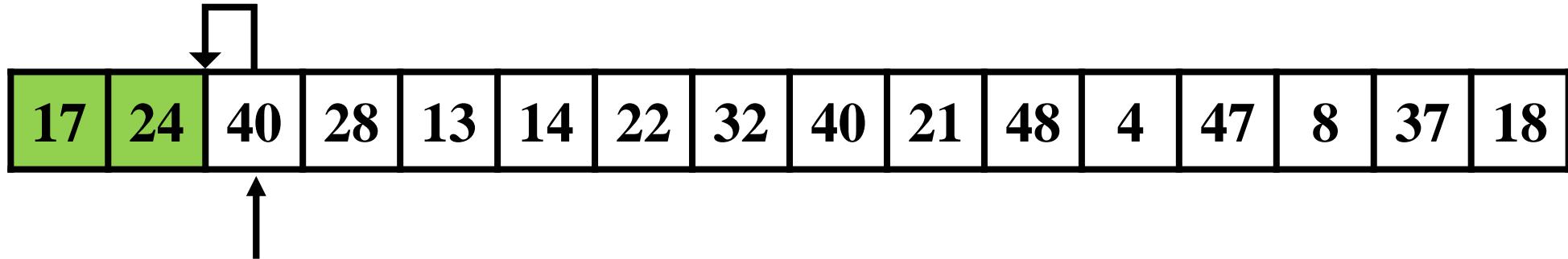
插入排序：算法实例



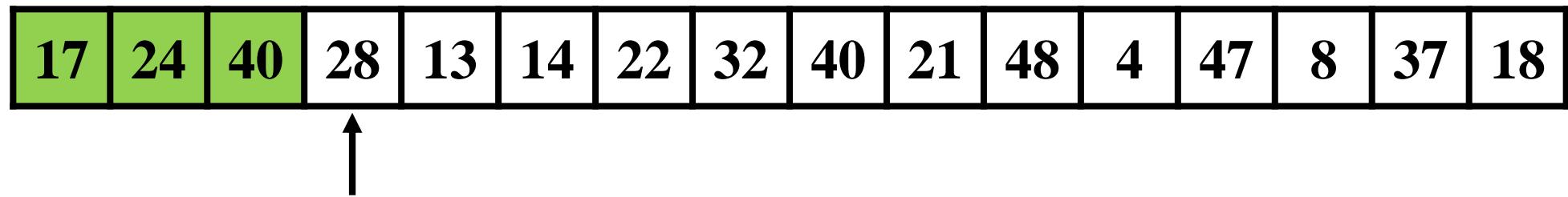
插入排序：算法实例



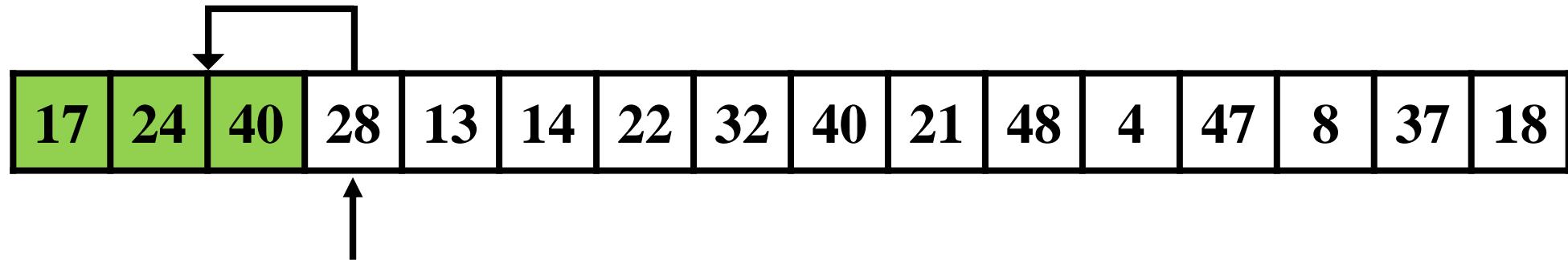
插入排序：算法实例



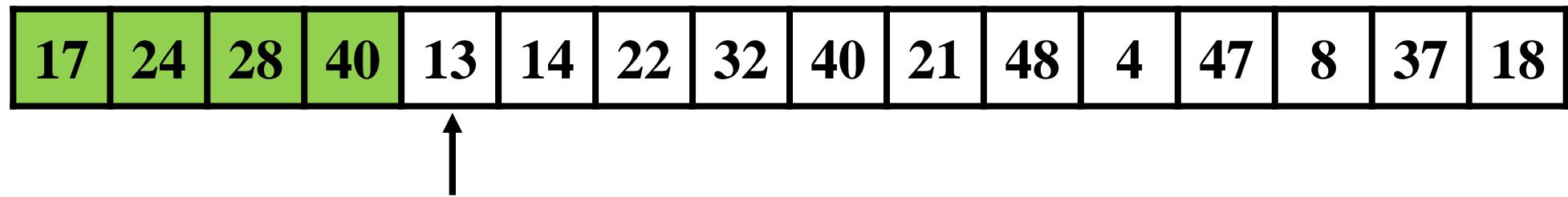
插入排序：算法实例



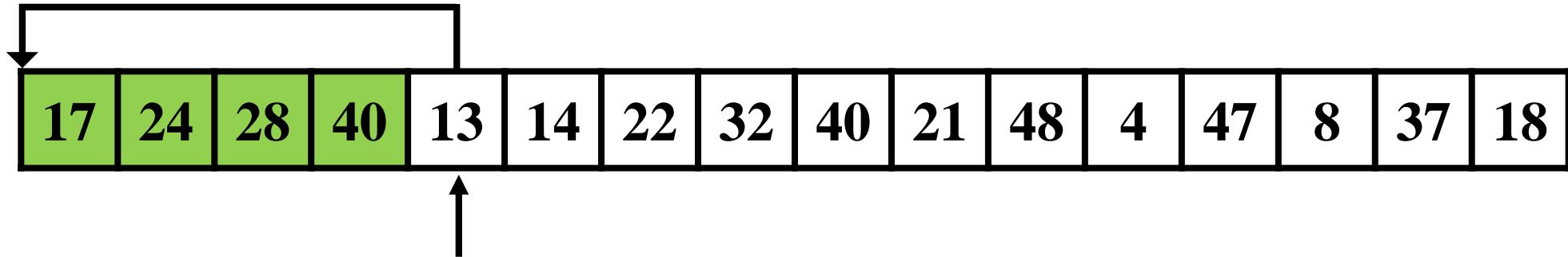
插入排序：算法实例



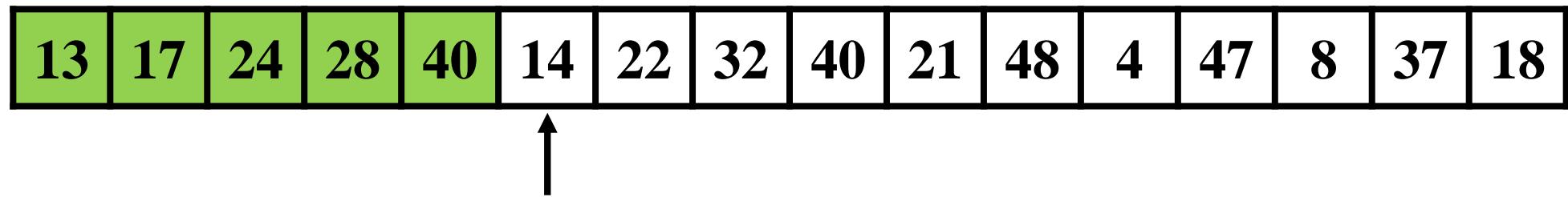
插入排序：算法实例



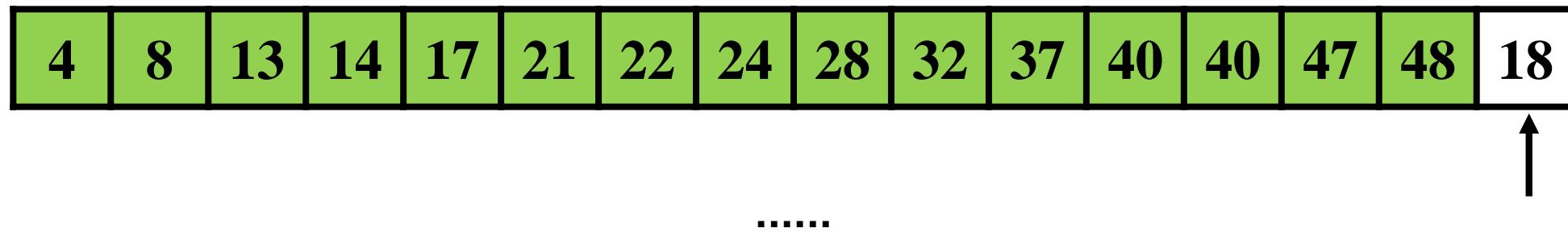
插入排序：算法实例



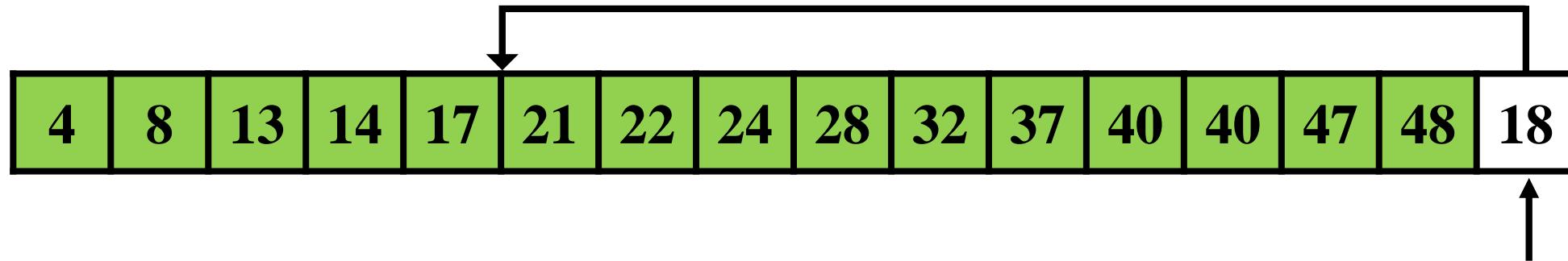
插入排序：算法实例



插入排序：算法实例



插入排序：算法实例



插入排序：算法实例

- **输入：**
 - <24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>
- **输出：**
 - <4, 8, 13, 14, 17, 18, 21, 22, 24, 28, 32, 37, 40, 40, 47, 48>



插入排序：算法实例

- **输入：**
 - <24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>
- **输出：**
 - <4, 8, 13, 14, 17, 18, 21, 22, 24, 28, 32, 37, 40, 40, 47, 48>

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

问题：排序问题是否存在其他算法？

选择排序

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

● 选择排序算法

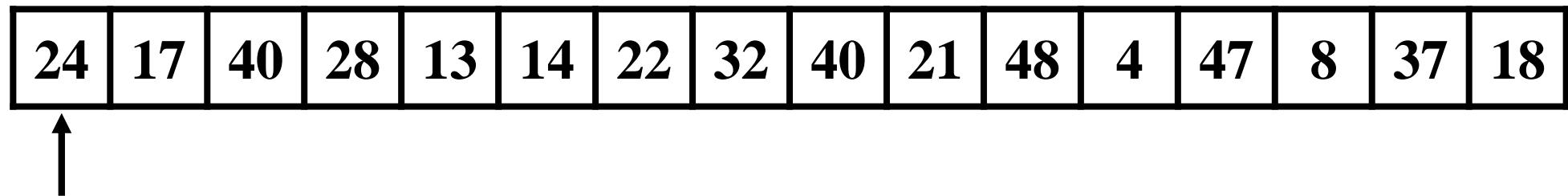
- 第一次遍历找到最小元素
- 第二次在剩余数组中遍历找到次小元素
- ...
- 第 n 次在剩余数组中遍历找到第 n 小元素

选择排序：算法实例

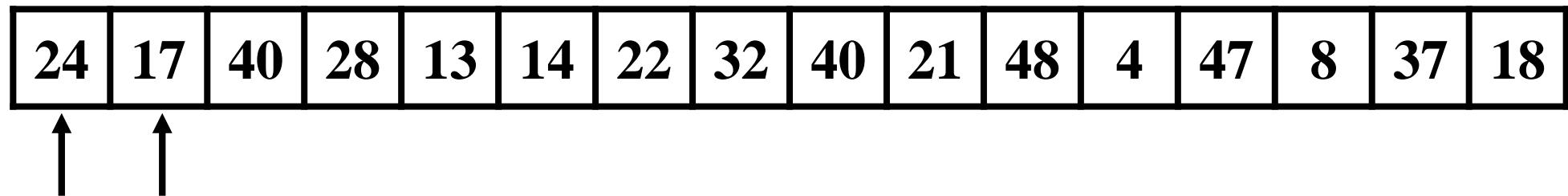
- 输入： $<24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>$

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

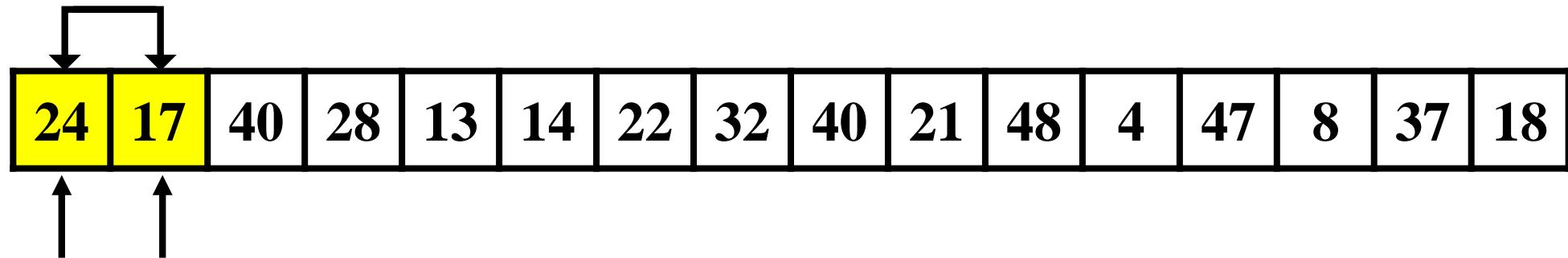
选择排序：算法实例



选择排序：算法实例

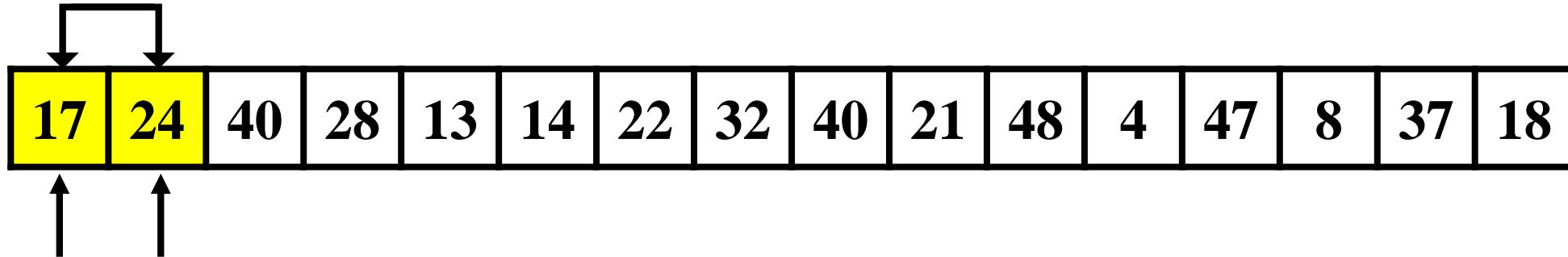


选择排序：算法实例



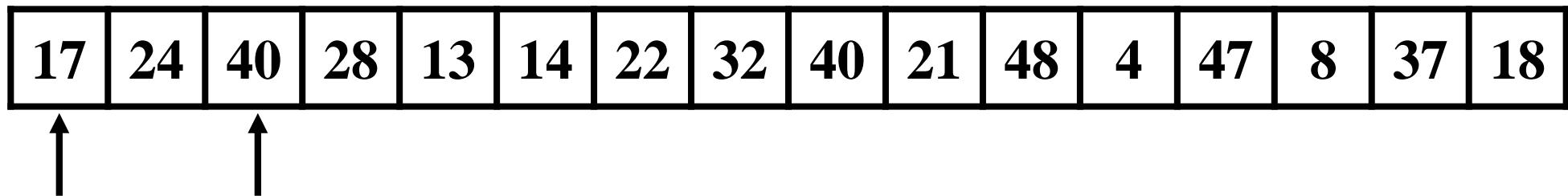
$24 > 17$

选择排序：算法实例



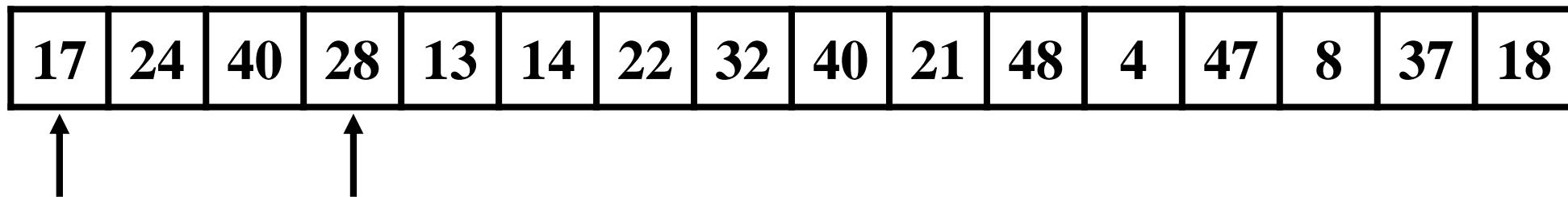
$17 < 24$

选择排序：算法实例



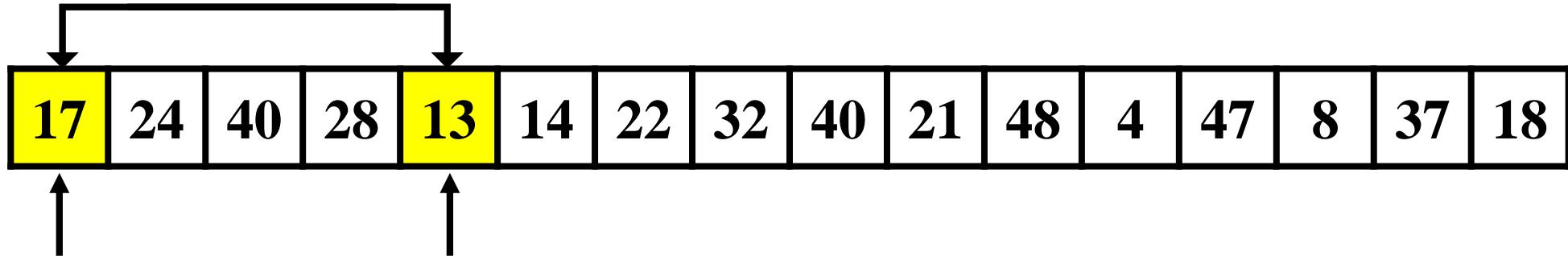
$17 < 40$

选择排序：算法实例



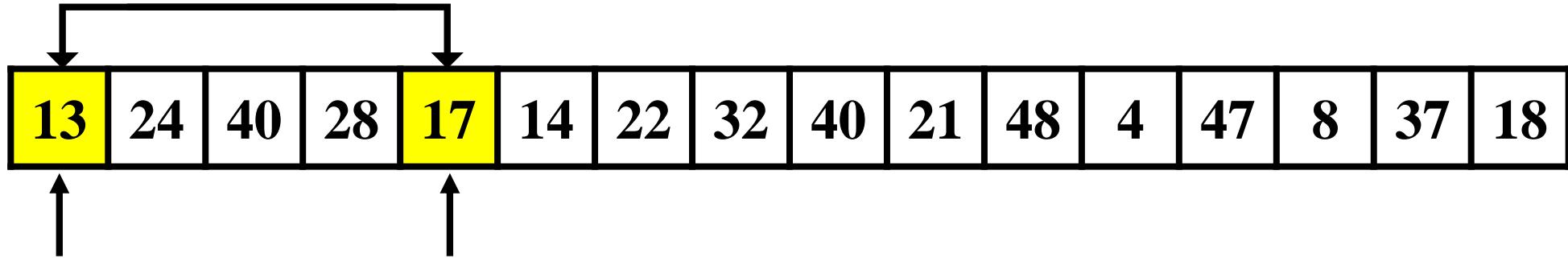
$17 < 28$

选择排序：算法实例



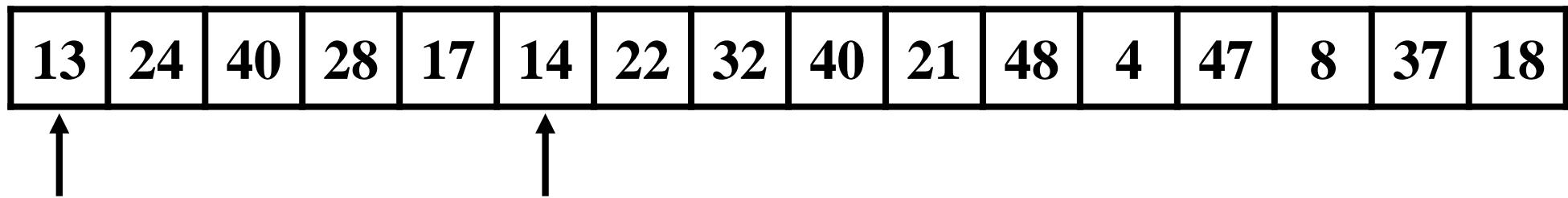
$17 > 13$

选择排序：算法实例



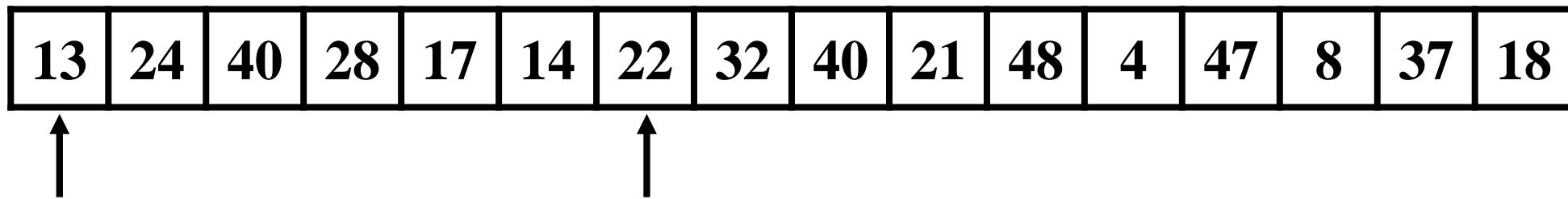
$13 < 17$

选择排序：算法实例



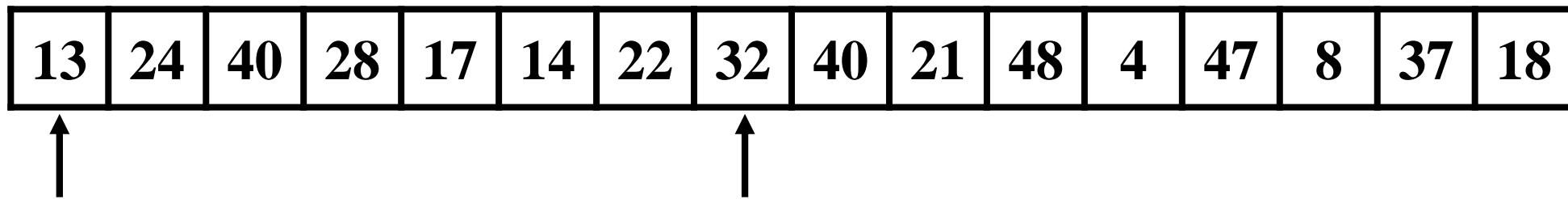
$13 < 14$

选择排序： 算法实例



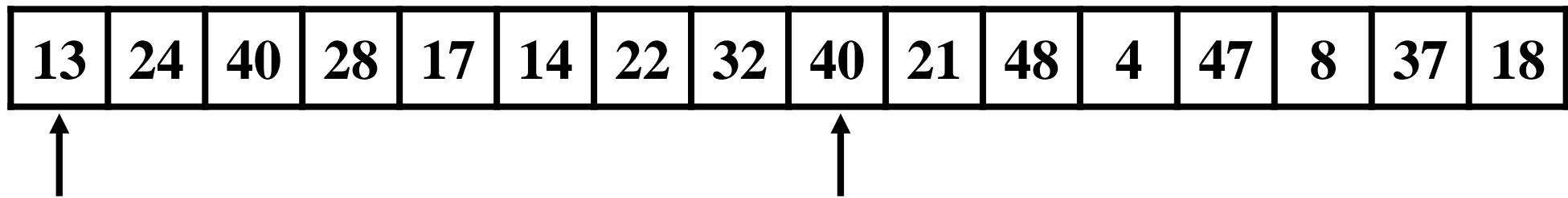
$$13 < 22$$

选择排序：算法实例



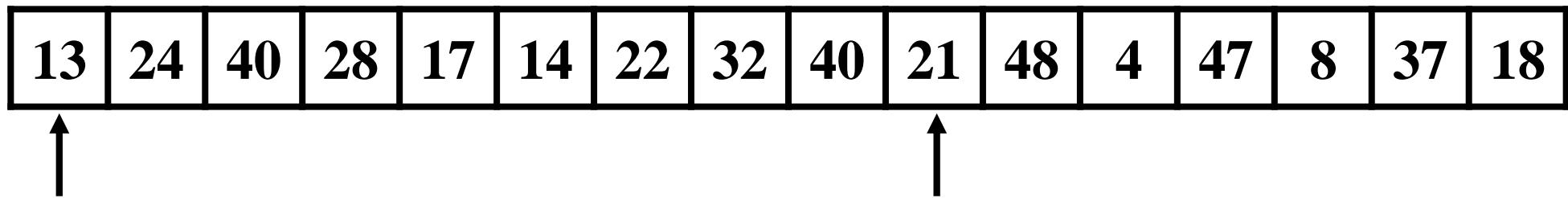
$13 < 32$

选择排序：算法实例



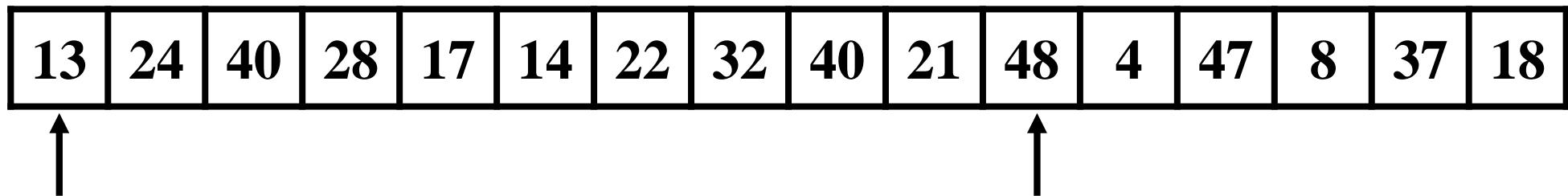
$13 < 40$

选择排序：算法实例



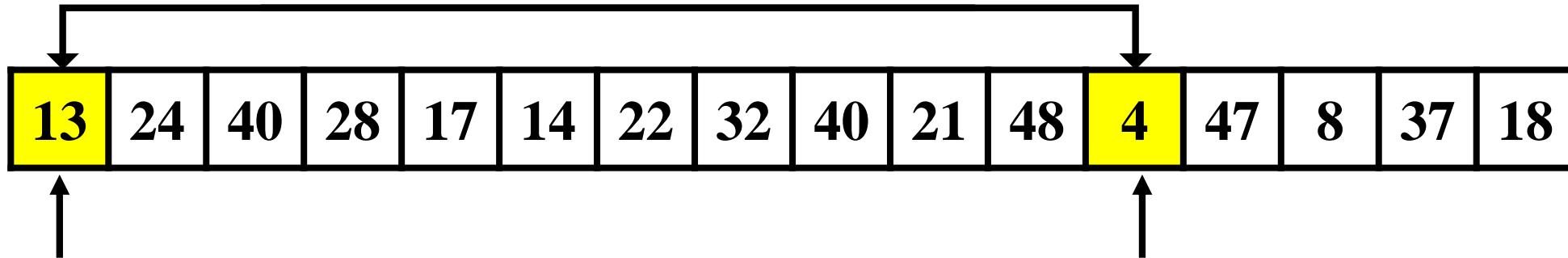
$13 < 21$

选择排序：算法实例



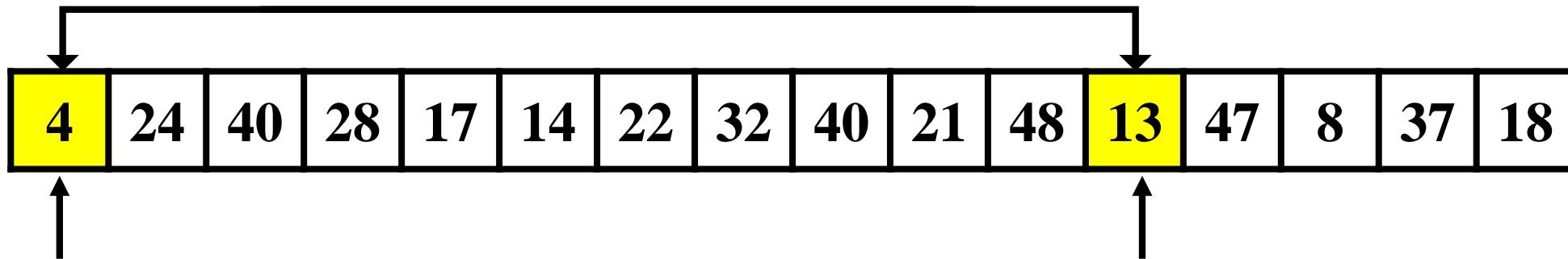
$13 < 48$

选择排序：算法实例



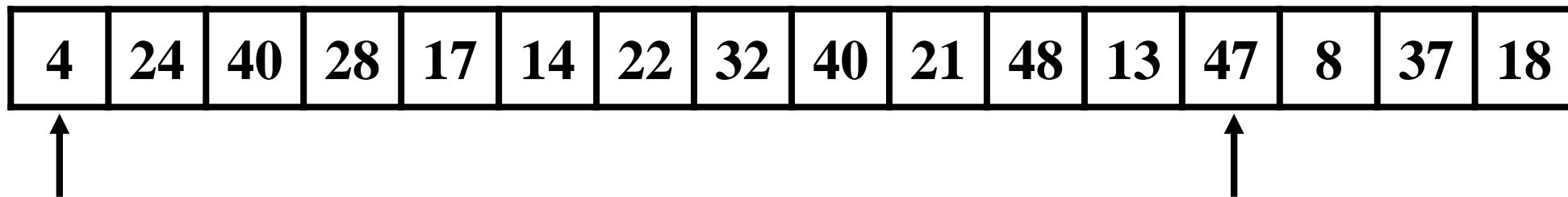
$13 > 4$

选择排序：算法实例



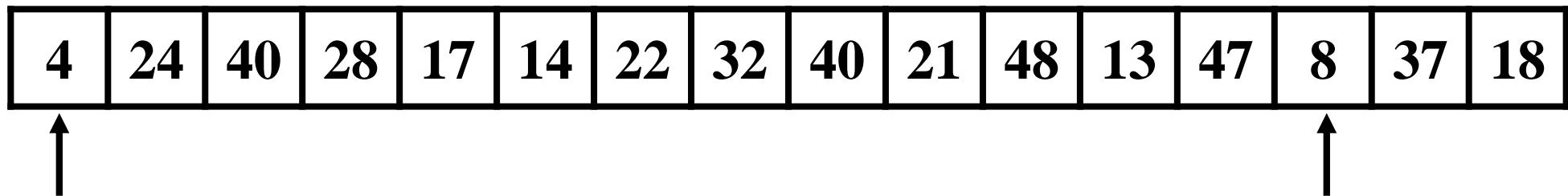
$4 < 13$

选择排序：算法实例



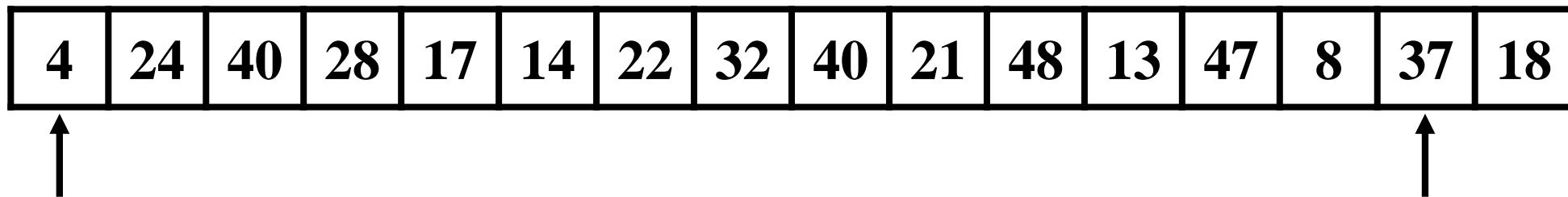
$4 < 47$

选择排序：算法实例



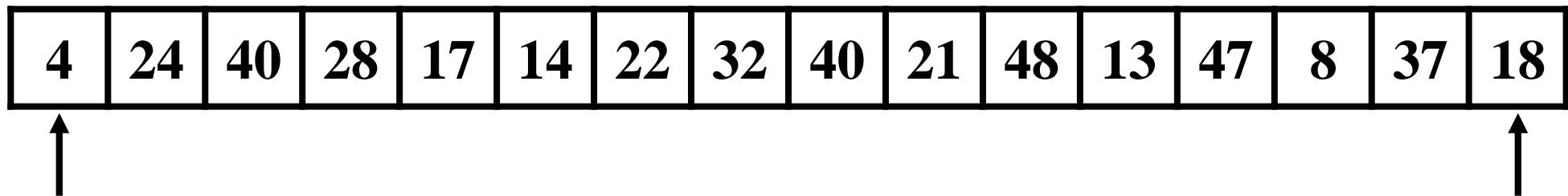
$$4 < 8$$

选择排序：算法实例



$4 < 37$

选择排序：算法实例



$4 < 18$

选择排序：算法实例

4	24	40	28	17	14	22	32	40	21	48	13	47	8	37	18
---	----	----	----	----	----	----	----	----	----	----	----	----	---	----	----

选择排序：算法实例

4	8	40	28	24	17	22	32	40	21	48	14	47	13	37	18
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

选择排序：算法实例

4	8	13	40	28	24	22	32	40	21	48	17	47	14	37	18
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

选择排序：算法实例

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

.....

选择排序：算法实例

- 输入：<24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>
- 输出：<4, 8, 13, 14, 17, 18, 21, 22, 24, 28, 32, 37, 40, 40, 47, 48>

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

选择排序：算法实例

- 输入：<24, 17, 40, 28, 13, 14, 22, 32, 40, 21, 48, 4, 47, 8, 37, 18>
- 输出：<4, 8, 13, 14, 17, 18, 21, 22, 24, 28, 32, 37, 40, 40, 47, 48>
- 满足
 - $4 \leq 8 \leq 13 \leq 14 \leq 17 \leq 18 \leq 21 \leq 22 \leq 24 \leq 28 \leq 32 \leq 37 \leq 40 \leq 40 \leq 47 \leq 48$

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

提纲

算法的由来

算法的定义

算法的性质

算法的表示

算法的分析

算法的性质

有穷性

确定性

可行性

算法的性质

有穷性

确定性

可行性

- 含义

- 算法必须在有限个计算步骤后终止

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须在有限个计算步骤后终止
- 反例：
 - 给定输入数组，不断交换首尾元素的位置

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义

- 算法必须在有限个计算步骤后终止

- 反例：

- 给定输入数组，**不断交换首尾元素的位置**

动作序列没有终结

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须在有限个计算步骤后终止
- 反例：
 - 给定输入数组，不断交换首尾元素的位置
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须在有限个计算步骤后终止
- 反例：
 - 给定输入数组，不断交换首尾元素的位置
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

遍历次数至多与数组元素个数相同

算法的性质

有穷性

确定性

可行性

- 含义

- 算法必须是没有歧义的

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须是没有歧义的
- 反例：
 - 对于给定输入数组，交换两个数的位置

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须是没有歧义的
- 反例：
 - 对于给定输入数组，交换两个数的位置

没有具体指明是哪两个数

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须是没有歧义的
- 反例：
 - 对于给定输入数组，交换两个数的位置
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

算法的性质

有穷性

确定性

可行性

- 含义
 - 算法必须是没有歧义的
- 反例：
 - 对于给定输入数组，交换两个数的位置
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

每一个步骤都是确定的

算法的性质

有穷性

确定性

可行性

- 含义

- 可以机械地一步一步执行基本操作步骤

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 可以机械地一步一步执行基本操作步骤
- 反例：
 - 将大元素放数组后部，小元素放数组前部

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 可以机械地一步一步执行基本操作步骤
- 反例：
 - 将大元素放数组后部，小元素放数组前部

描述含糊，不可拆解为基本操作步骤

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$\langle a'_1, a'_2, \dots, a'_n \rangle$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法的性质

有穷性

确定性

可行性

- 含义
 - 可以机械地一步一步执行基本操作步骤
- 反例：
 - 将大元素放数组后部，小元素放数组前部
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

$$\text{满足 } a'_1 \leq a'_2 \leq \dots \leq a'_n$$

算法的性质

有穷性

确定性

可行性

- 含义
 - 可以机械地一步一步执行基本操作步骤
- 反例：
 - 将大元素放数组后部，小元素放数组前部
- 正例：选择排序
 - 第一次遍历找到最小元素
 - 第二次在剩余数组中遍历找到次小元素
 - ...
 - 第 n 次在剩余数组中遍历找到第 n 小元素

排序问题

Sorting Problem

输入

- 包含 n 个数字的数组 $\langle a_1, \dots, a_n \rangle$

输出

- 升序排列的数组

$$\langle a'_1, a'_2, \dots, a'_n \rangle$$

满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

算法可一步步地执行完成

提纲

算法的由来

算法的定义

算法的性质

算法的表示

算法的分析

算法的表示

- 如何表示一个算法？

算法的表示

- 如何表示一个算法？



算法的表示

- 如何表示一个算法?
 - 自然语言



算法的表示

- 自然语言
 - 方法优势
 - 贴近人类思维，易于理解主旨

选择排序

- 第一次遍历找到最小元素
- 第二次在剩余数组中遍历找到次小元素
- ...
- 第 n 次在剩余数组中遍历找到第 n 小元素

算法的表示

- 自然语言
 - 方法优势
 - 贴近人类思维，易于理解主旨
 - 不便之处
 - 语言描述繁琐，容易产生歧义
 - 使用了“...”等不严谨的描述

选择排序

- 第一次遍历找到最小元素
- 第二次在剩余数组中遍历找到次小元素
- ...
- 第 n 次在剩余数组中遍历找到第 n 小元素

算法的表示

- 如何表示一个算法?

- 自然语言
- 编程语言



算法的表示

- 编程语言
 - 方法优势
 - 精准表达逻辑，规避表述歧义

选择排序

```
void select_sort(int*a,int n)
{
    int i,j,t;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++){
            if(a[i] > a[j]){
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
}
```

C语言

```
def select_sort(a, n):
    for i in range(0, n-1):
        for j in range(i+1,n):
            if a[i] > a[j]:
                tem = a[i]
                a[i] = a[j]
                a[j] = tem
```

Python语言

算法的表示

- 编程语言

- 方法优势

- 精准表达逻辑，规避表述歧义

- 不便之处

- 不同编程语言间语法存在差异

- 过于关注算法实现的细枝末节

选择排序

```
void select_sort(int*a,int n)
{
    int i,j,t;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++){
            if(a[i] > a[j]){
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
}
```

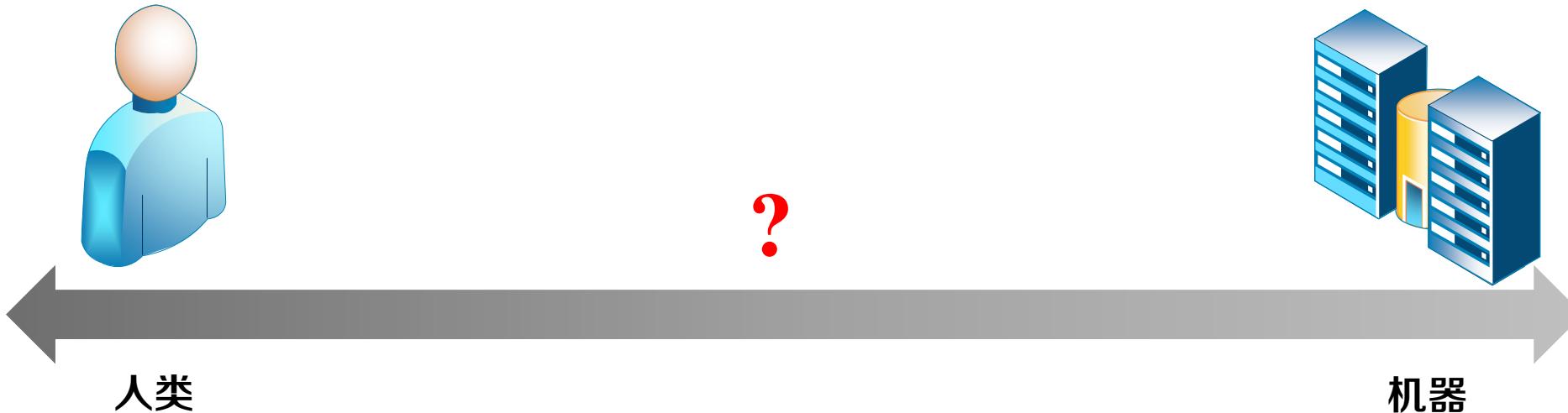
C语言

```
def select_sort(a, n):
    for i in range(0, n-1):
        for j in range(i+1,n):
            if a[i] > a[j]:
                tem = a[i]
                a[i] = a[j]
                a[j] = tem
```

Python语言

算法的表示

- 如何表示一个算法?
 - 自然语言: 贴近人类思维, 易于理解主旨
 - 编程语言: 精准表达逻辑, 规避表述歧义



问题: 可否同时兼顾两类表示方法的优势?

算法的表示

- 如何表示一个算法?
 - 自然语言: 贴近人类思维, 易于理解主旨
 - 编程语言: 精准表达逻辑, 规避表述歧义



问题: 可否同时兼顾两类表示方法的优势?

算法的表示

- 伪代码

- 非正式语言

- 移植**编程语言**书写形式作为基础和框架
 - 按照接近**自然语言**的形式表达算法过程

选择排序

输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for i ← 1 to n – 1 do
    for j ← i + 1 to n do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

算法的表示

● 伪代码

- 非正式语言

- 移植**编程语言**书写形式作为基础和框架
 - 按照接近**自然语言**的形式表达算法过程

- 兼顾自然语言与编程语言优势

- **简洁**表达算法本质，不拘泥于实现细节
 - **准确**反映算法过程，不产生矛盾和歧义

选择排序

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

选择排序

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

定义算法的输入和输出

选择排序

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

循环语句块缩进

选择排序

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

将 $i + 1$ 赋值给j

选择排序

算法的表示

- 伪代码

- 书写约定

输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

注释使用“//”符号

选择排序

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

条件语句块缩进

选择排序

算法的表示

- 伪代码

- 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

不关注交换过程的实现细节

选择排序

算法的表示

- 伪代码
 - 书写约定

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

选择排序

之后出现的算法均使用伪代码描述

算法的表示

- 伪代码

- 示例解读

24	17	40	28	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

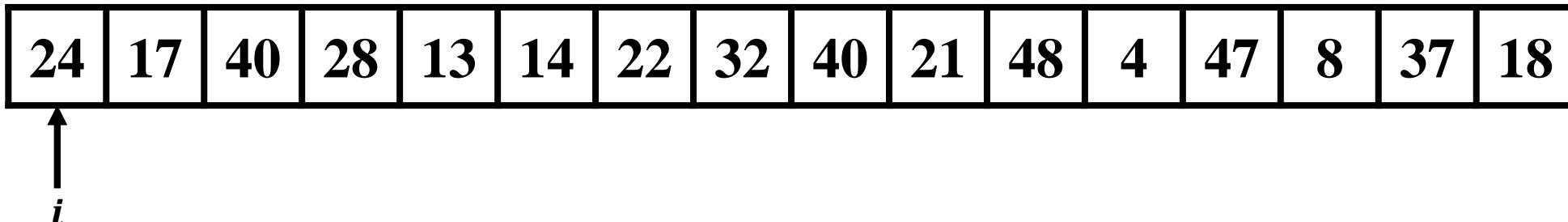
end

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

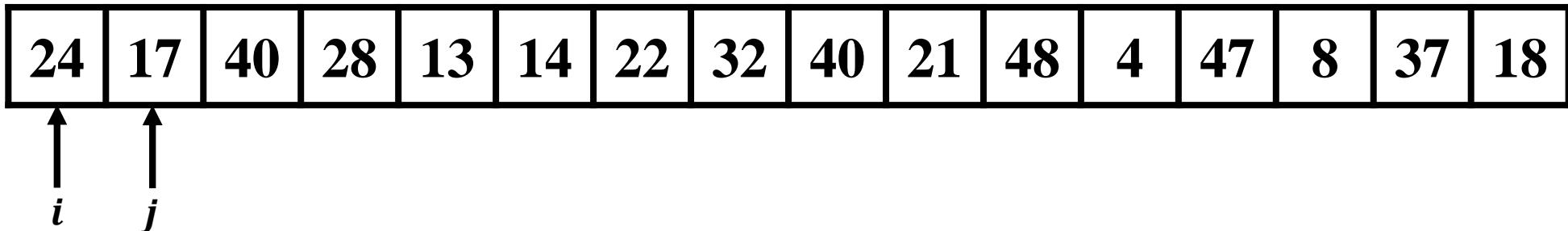
从数组首部开始枚举*i*

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

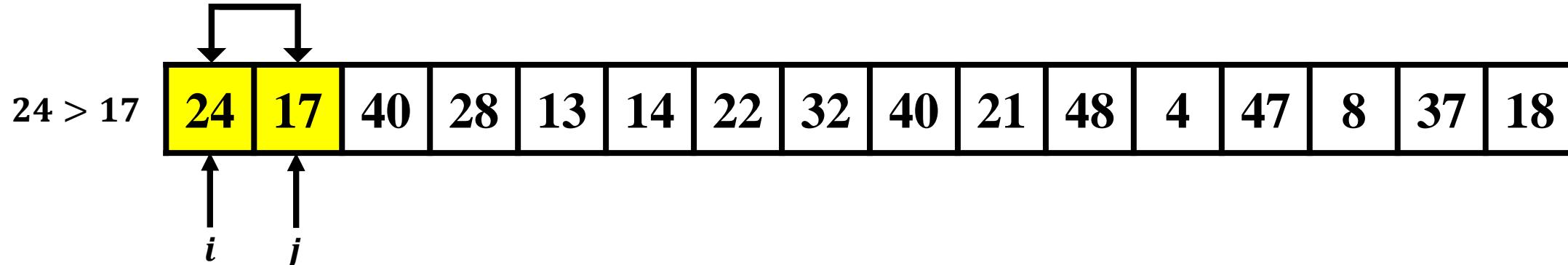
从 $i + 1$ 开始枚举 j

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

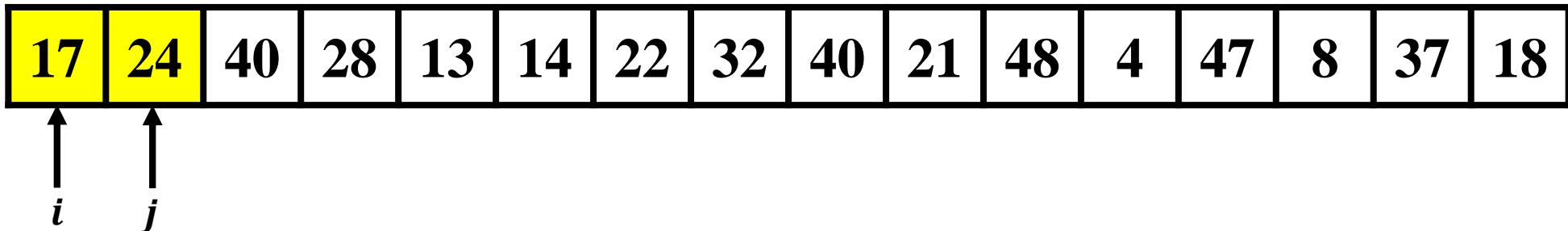
当第 i 个元素大于第 j 个元素时,
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

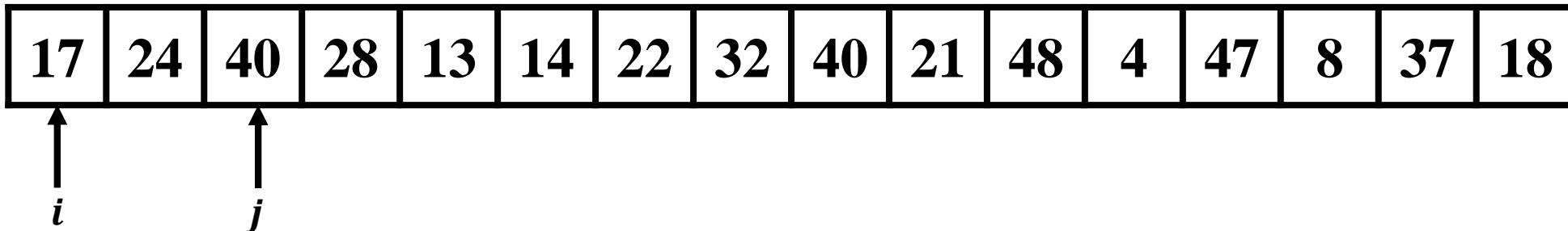
当第 i 个元素大于第 j 个元素时,
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

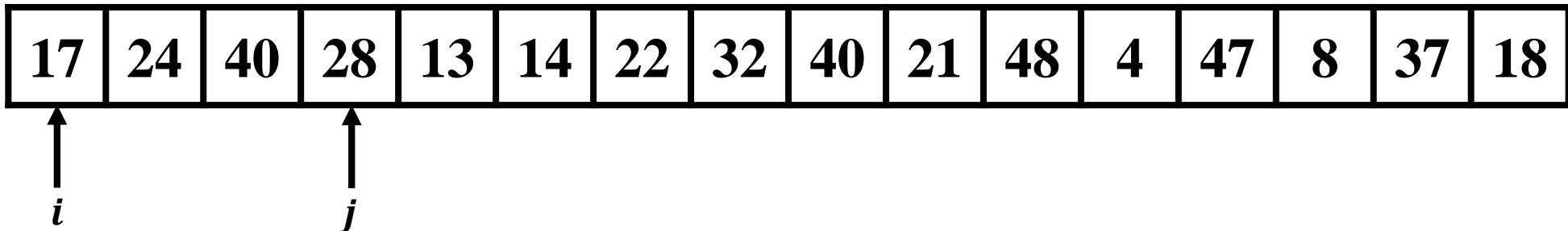
继续枚举 j

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

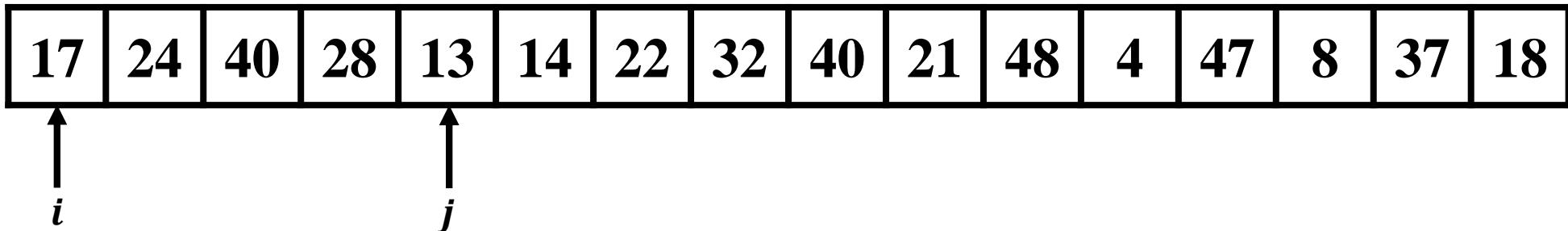
继续枚举 j

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

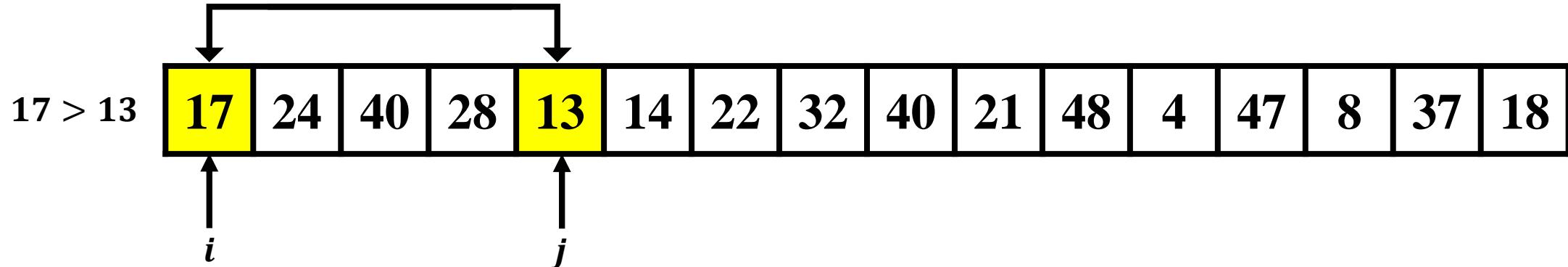
继续枚举 *j*

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

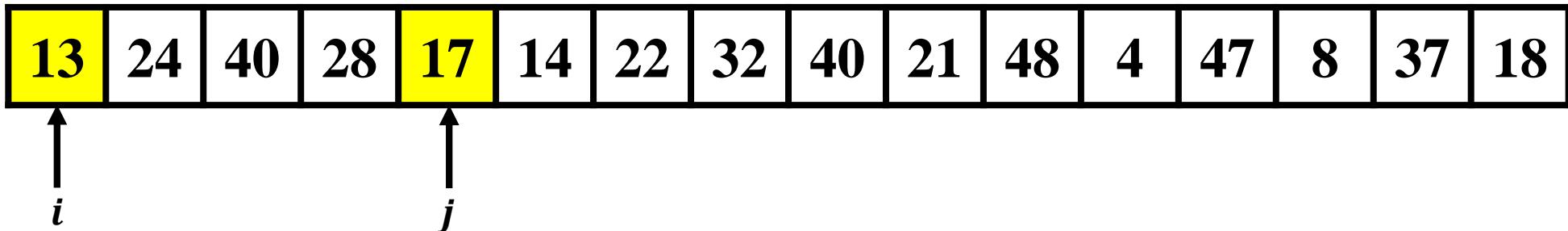
当第*i*个元素大于第*j*个元素时,
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

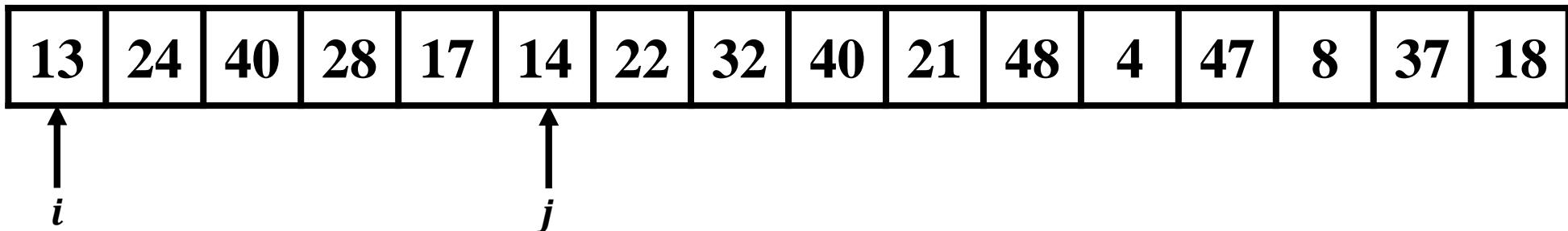
当第*i*个元素大于第*j*个元素时,
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

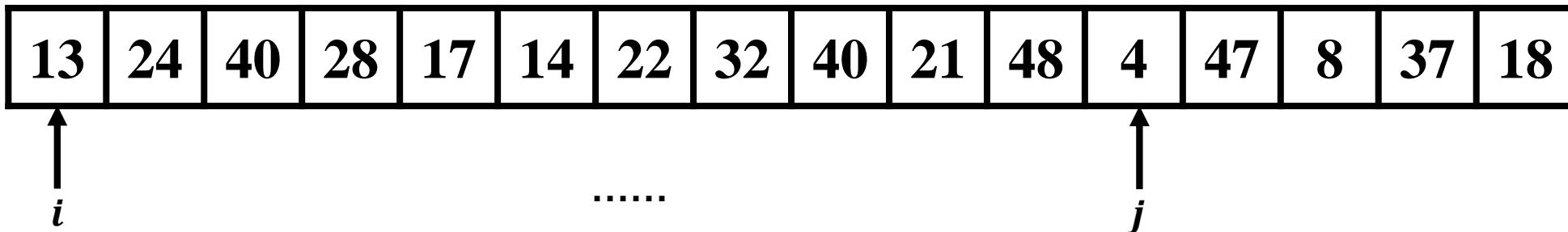
继续枚举 *j*

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 交换 $A[i]$ 和 $A[j]$

 end

end

end

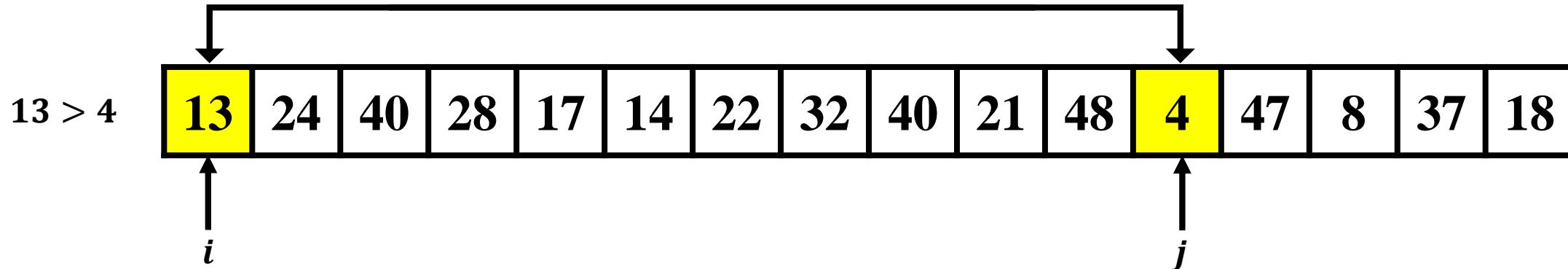
继续枚举 j

选择排序

算法的表示

• 伪代码

● 示例解读



输入：数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A' [a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ **to** $n - 1$ **do**

for $j \leftarrow i + 1$ **to** n **do**

//如果 $A[i]$ 大于 $A[j]$ ，则交换二者位置

if $A[i] > A[j]$ **then**

| 交换 $A[i]$ 和 $A[j]$

end

end

end

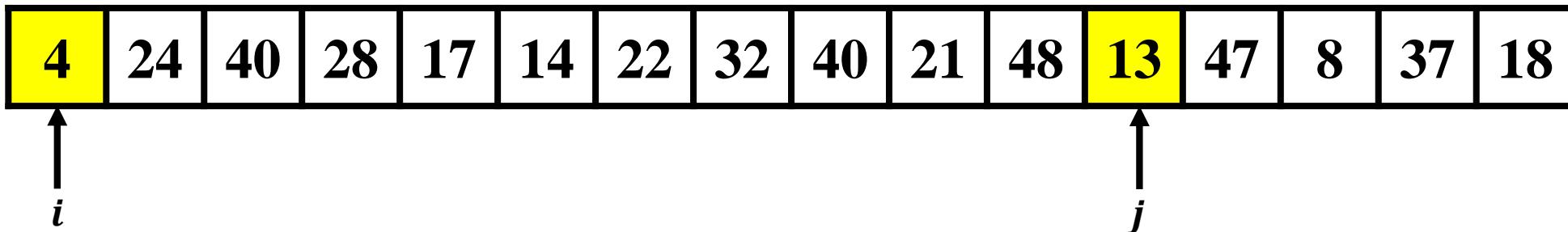
当第*i*个元素大于第*j*个元素时，
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

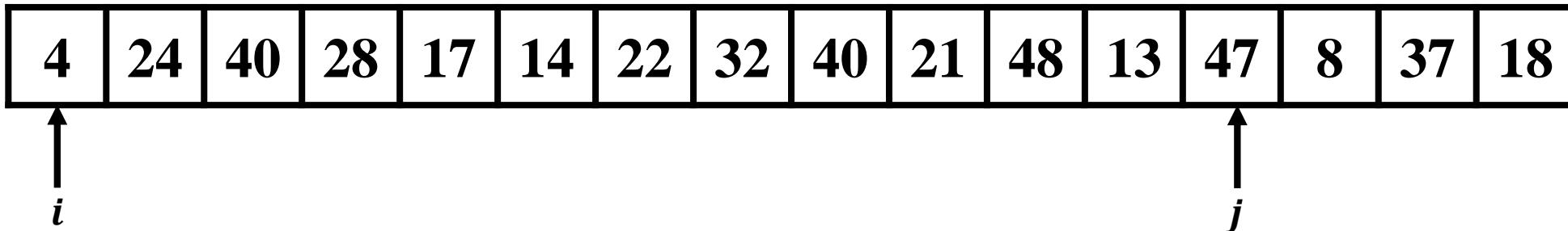
当第*i*个元素大于第*j*个元素时,
交换元素位置

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

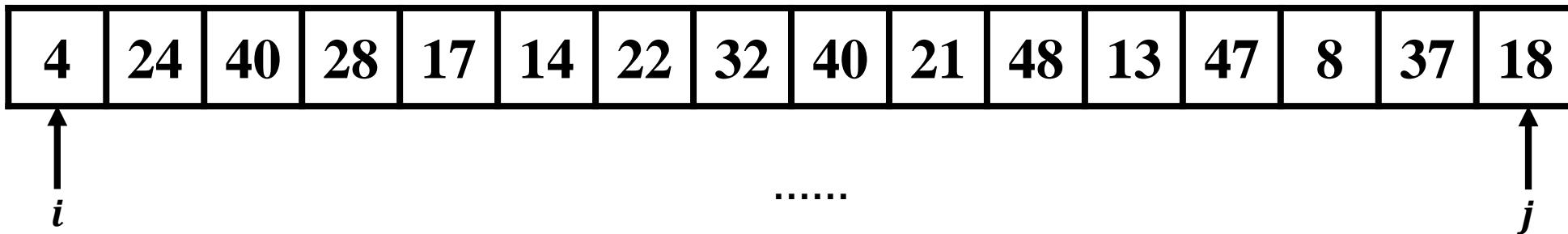
继续枚举 *j*

选择排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$
输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

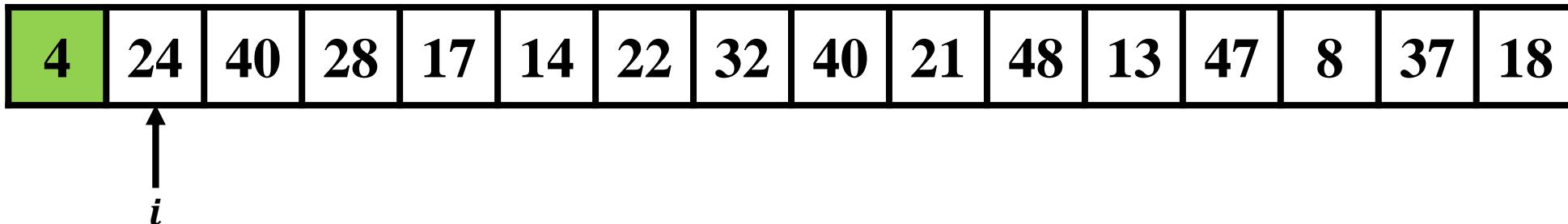
枚举 j 至 n , 结束循环

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

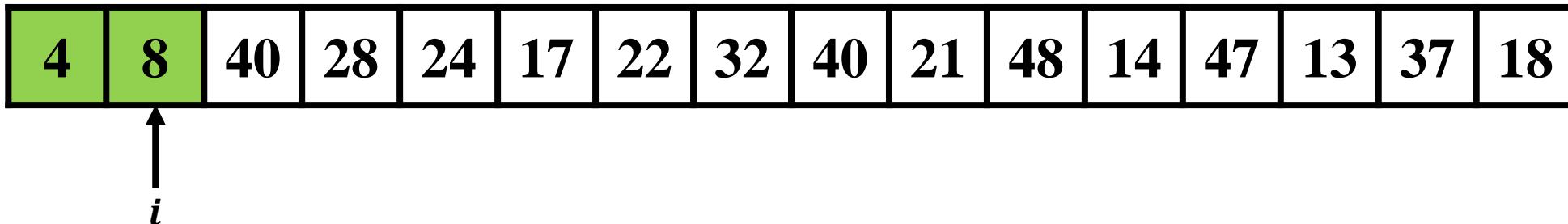
继续枚举*i*

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

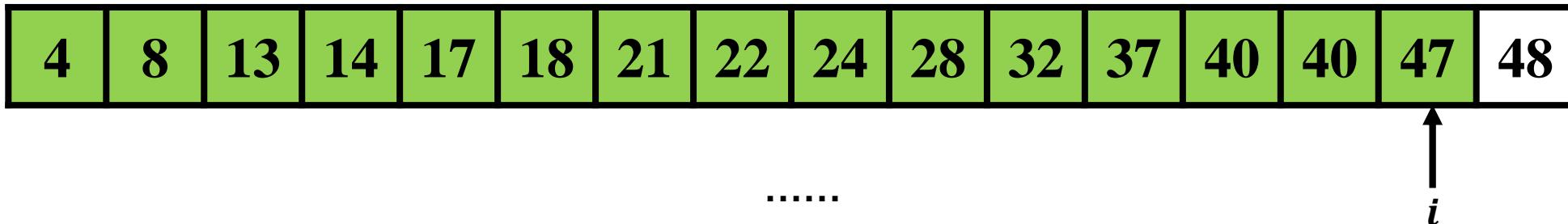
继续枚举*i*

选择排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        //如果  $A[i]$  大于  $A[j]$ , 则交换二者位置
        if  $A[i] > A[j]$  then
            | 交换  $A[i]$  和  $A[j]$ 
        end
    end
end
```

枚举 i 至 $n - 1$, 结束循环

选择排序

算法的表示

- 伪代码

- 示例解读

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $i \leftarrow 1$ to $n - 1$ do

 for $j \leftarrow i + 1$ to n do

 //如果 $A[i]$ 大于 $A[j]$, 则交换二者位置

 if $A[i] > A[j]$ then

 | 交换 $A[i]$ 和 $A[j]$

 end

 end

end

选择排序

算法的表示

- 伪代码

- 示例解读

17	24	28	40	13	14	22	32	40	21	48	4	47	8	37	18
----	----	----	----	----	----	----	----	----	----	----	---	----	---	----	----

输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

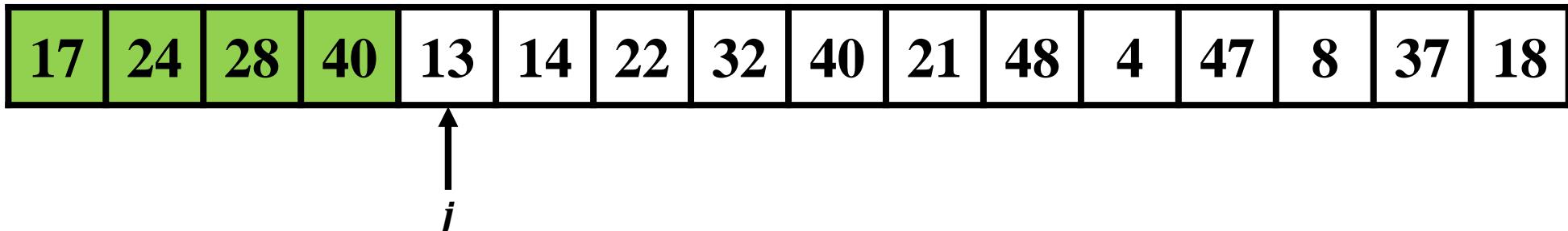
插入排序

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do
 $key \leftarrow A[j]$

将 $A[j]$ 赋值给 key

//将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$
 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$
 $i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

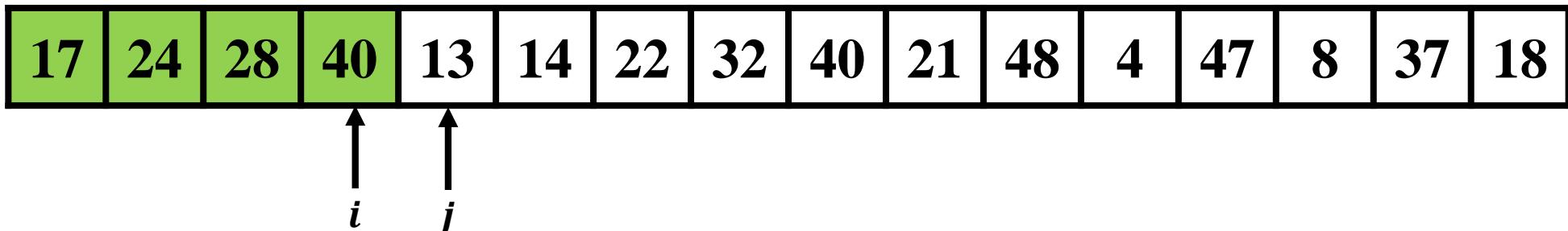
插入排序

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

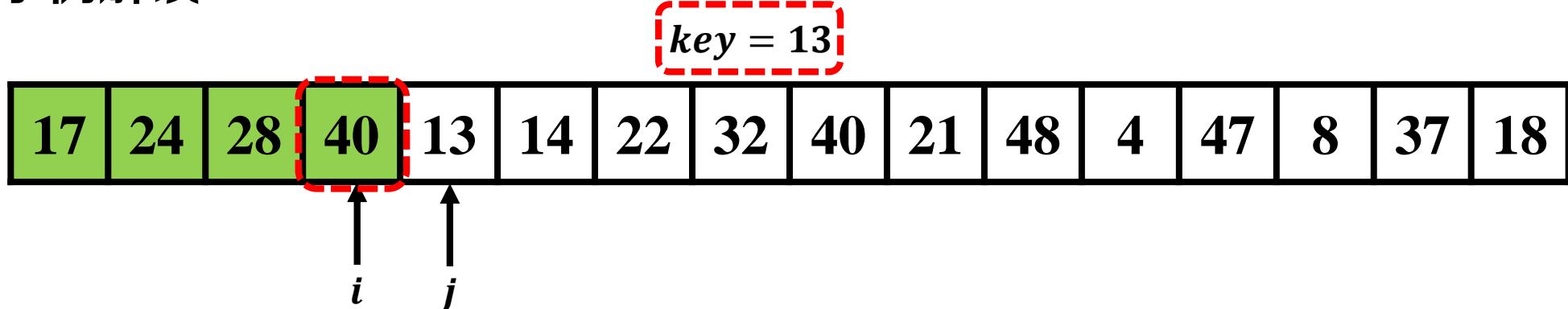
从 $j - 1$ 开始枚举 i

插入排序

算法的表示

- 伪代码

- 示例解读



$i > 0$
 $A[i] > key$

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

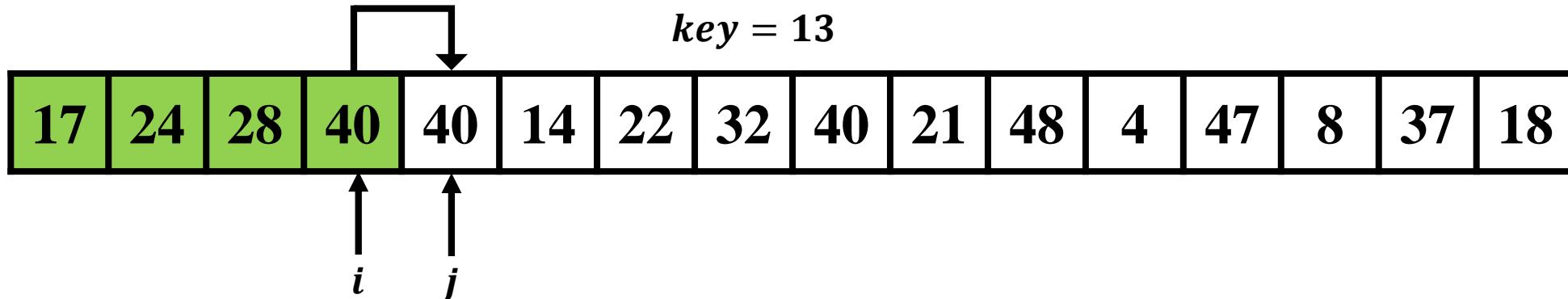
判断循环条件

插入排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

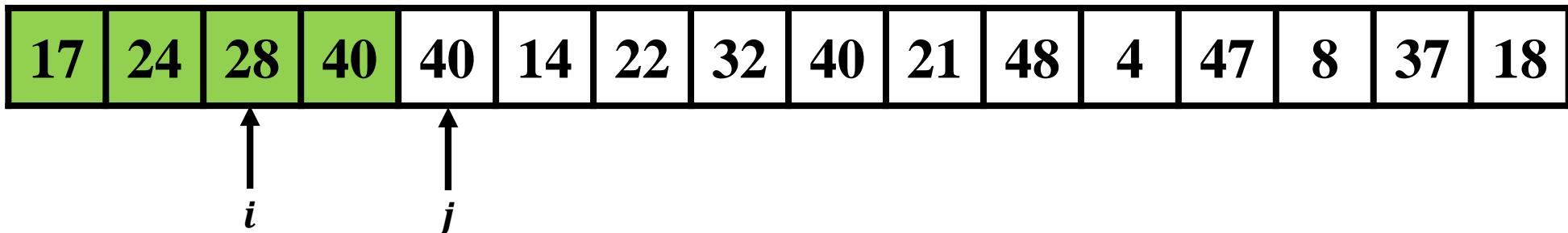
将 $A[i]$ 赋值给 $A[i + 1]$

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

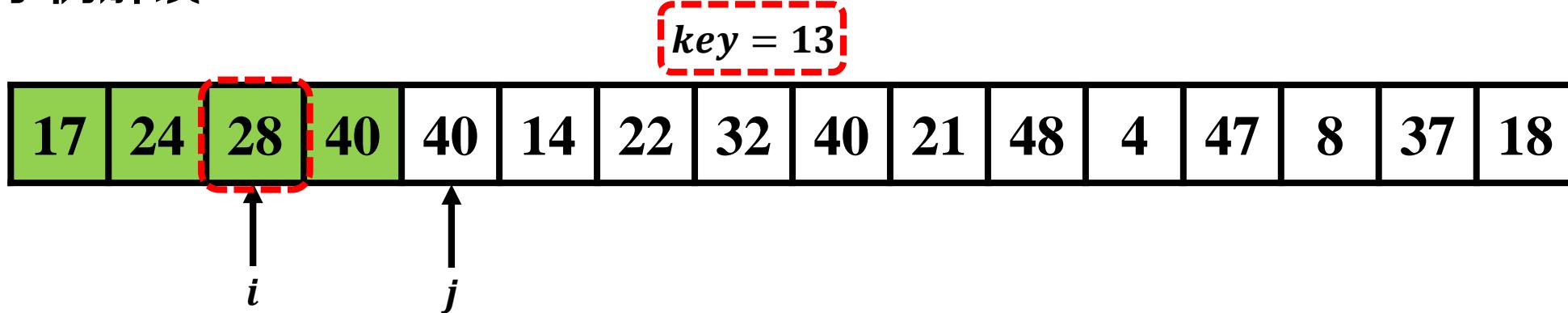
将 i 左移一步

插入排序

算法的表示

- 伪代码

- 示例解读



$i > 0$
 $A[i] > key$

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

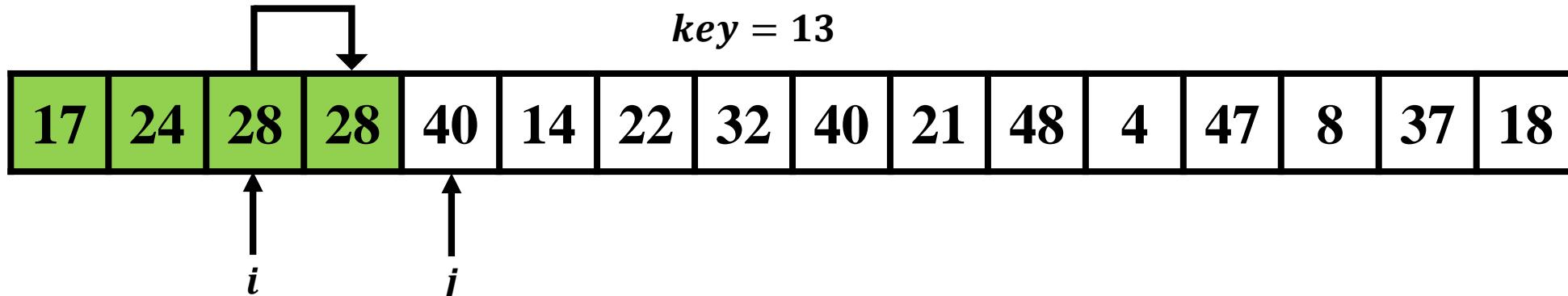
判断循环条件

插入排序

算法的表示

- 伪代码

- 示例解读



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i \geq 0$ and $A[i] > key$ do

$\boxed{A[i + 1] \leftarrow A[i]}$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

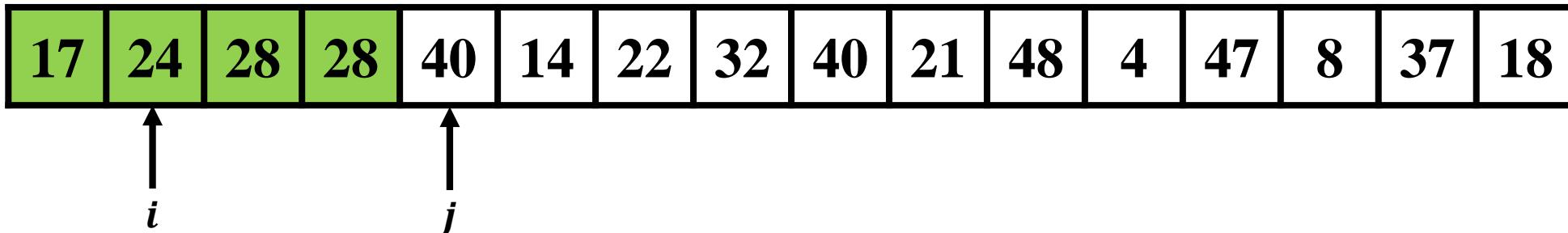
将 $A[i]$ 赋值给 $A[i + 1]$

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

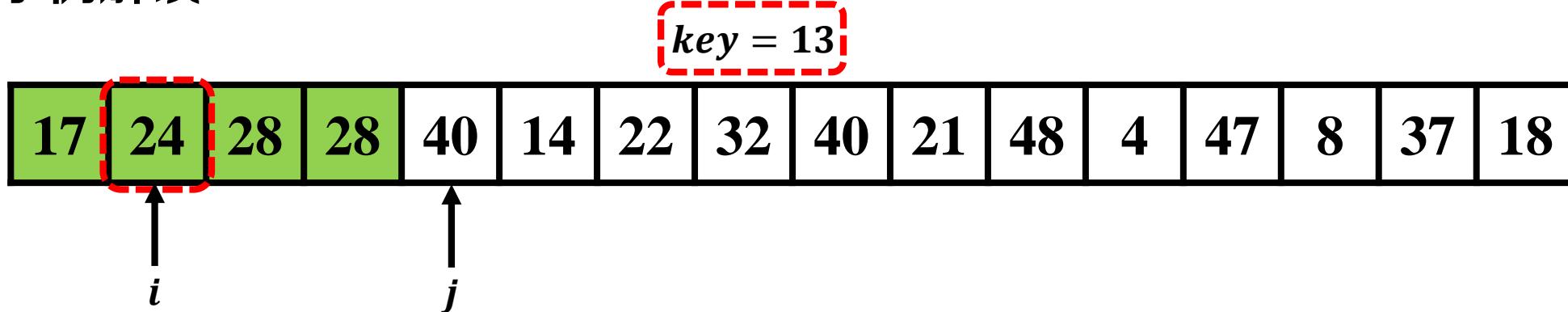
将 i 左移一步

插入排序

算法的表示

- 伪代码

- 示例解读



$i > 0$
 $A[i] > key$

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

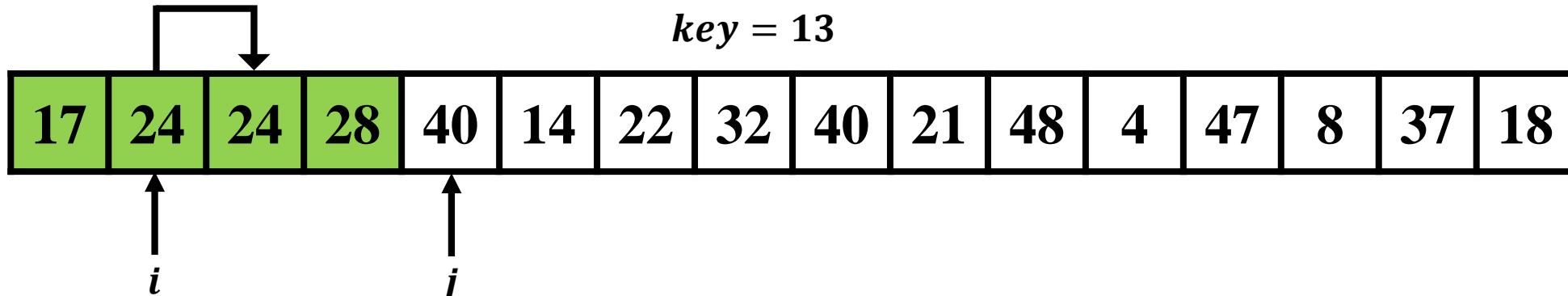
判断循环条件

插入排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

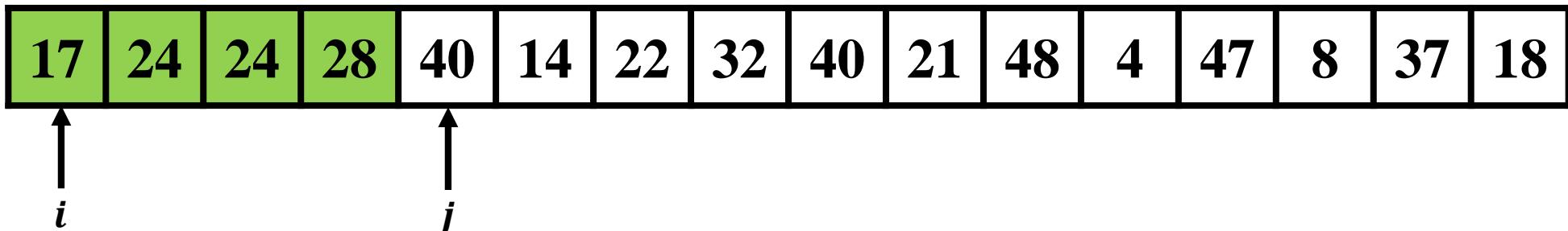
将 $A[i]$ 赋值给 $A[i + 1]$

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

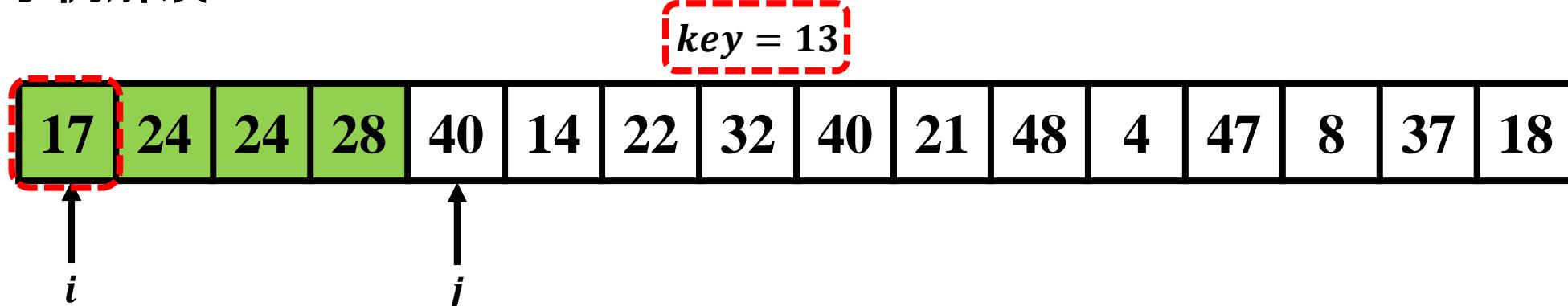
将 i 左移一步

插入排序

算法的表示

- 伪代码

- 示例解读



$i > 0$
 $A[i] > key$

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j-1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

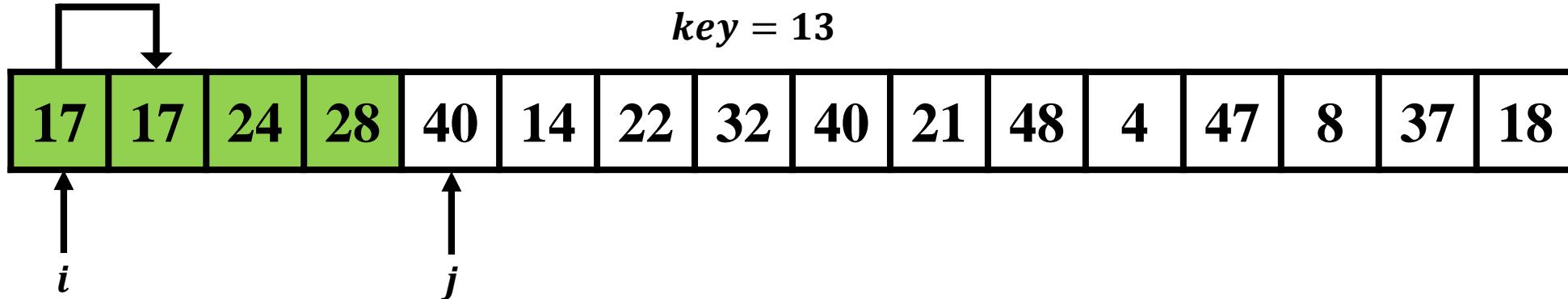
判断循环条件

插入排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
    key  $\leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

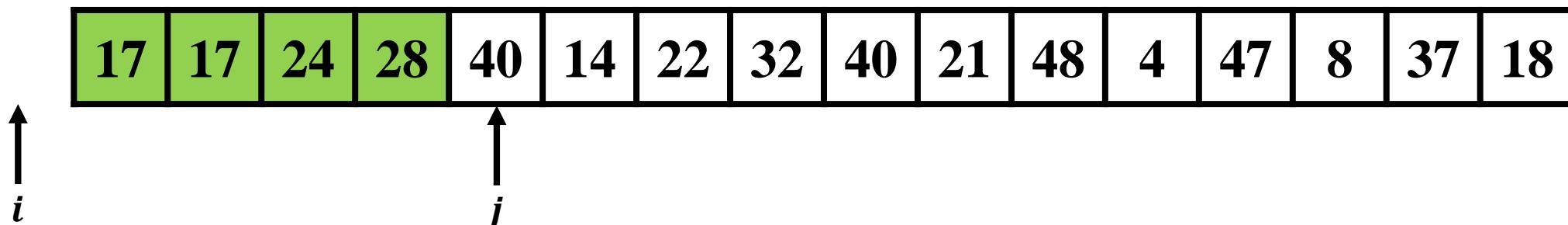
将 $A[i]$ 赋值给 $A[i + 1]$

算法的表示

- 伪代码

- 示例解读

$key = 13$



输入: 数组 $A[a_1, a_2, \dots, a_n]$

输出: 升序数组 $A'[a'_1, a'_2, \dots, a'_n]$, 满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

 //将 $A[j]$ 插入到已排序的数组 $A[1..j - 1]$ 中

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

将 i 左移一步

插入排序

算法的表示

- 伪代码

- 示例解读

$key = 13$



$i = 0$

不满足循环条件

```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
     $key \leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > key$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

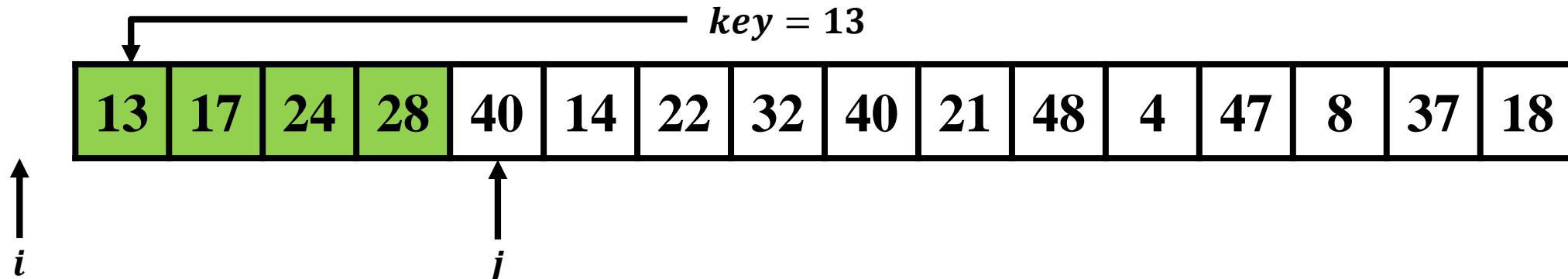
判断循环条件

插入排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
    key  $\leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > \text{key}$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
    A[i + 1]  $\leftarrow \text{key}$ 
end
```

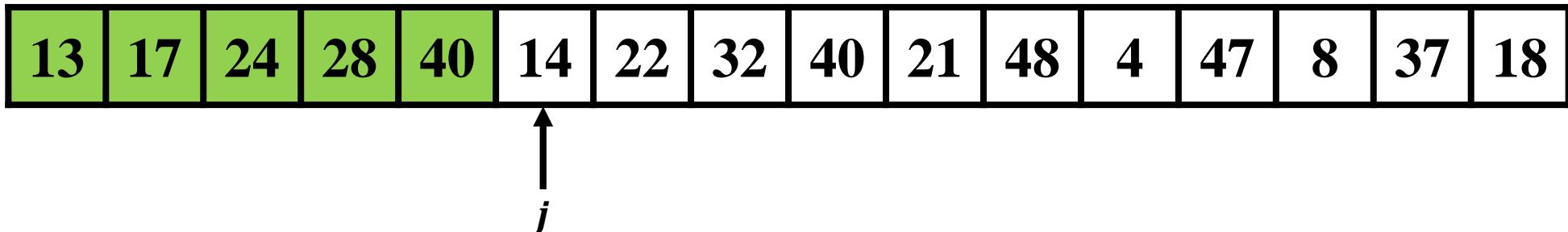
将 *key* 赋值给 $A[i + 1]$

插入排序

算法的表示

- 伪代码

- 示例解读



```
输入: 数组  $A[a_1, a_2, \dots, a_n]$ 
输出: 升序数组  $A'[a'_1, a'_2, \dots, a'_n]$ , 满足  $a'_1 \leq a'_2 \leq \dots \leq a'_n$ 
for  $j \leftarrow 2$  to  $n$  do
    key  $\leftarrow A[j]$ 
    //将  $A[j]$  插入到已排序的数组  $A[1..j - 1]$  中
     $i \leftarrow j - 1$ 
    while  $i > 0$  and  $A[i] > \text{key}$  do
         $A[i + 1] \leftarrow A[i]$ 
         $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow \text{key}$ 
end
```

插入排序

算法的表示方式比较

表示方式	语言特点
自然语言	贴近人类思维，易于理解主旨 表述不够精准，存在模糊歧义
编程语言	精准表达逻辑，规避表述歧义 受限语法细节，增大理解难度
伪代码	关注算法本质，便于书写阅读

提纲

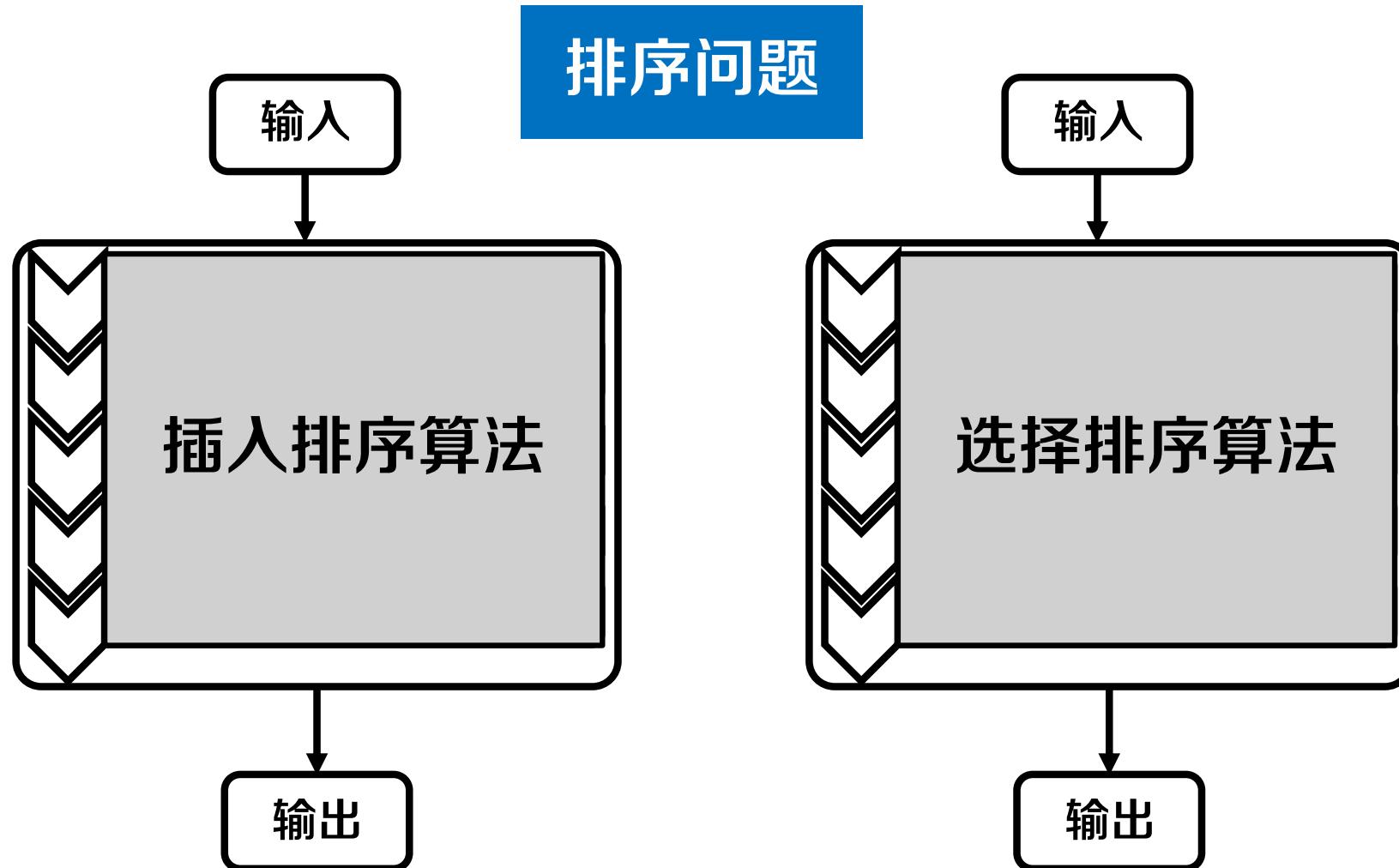
算法的由来

算法的定义

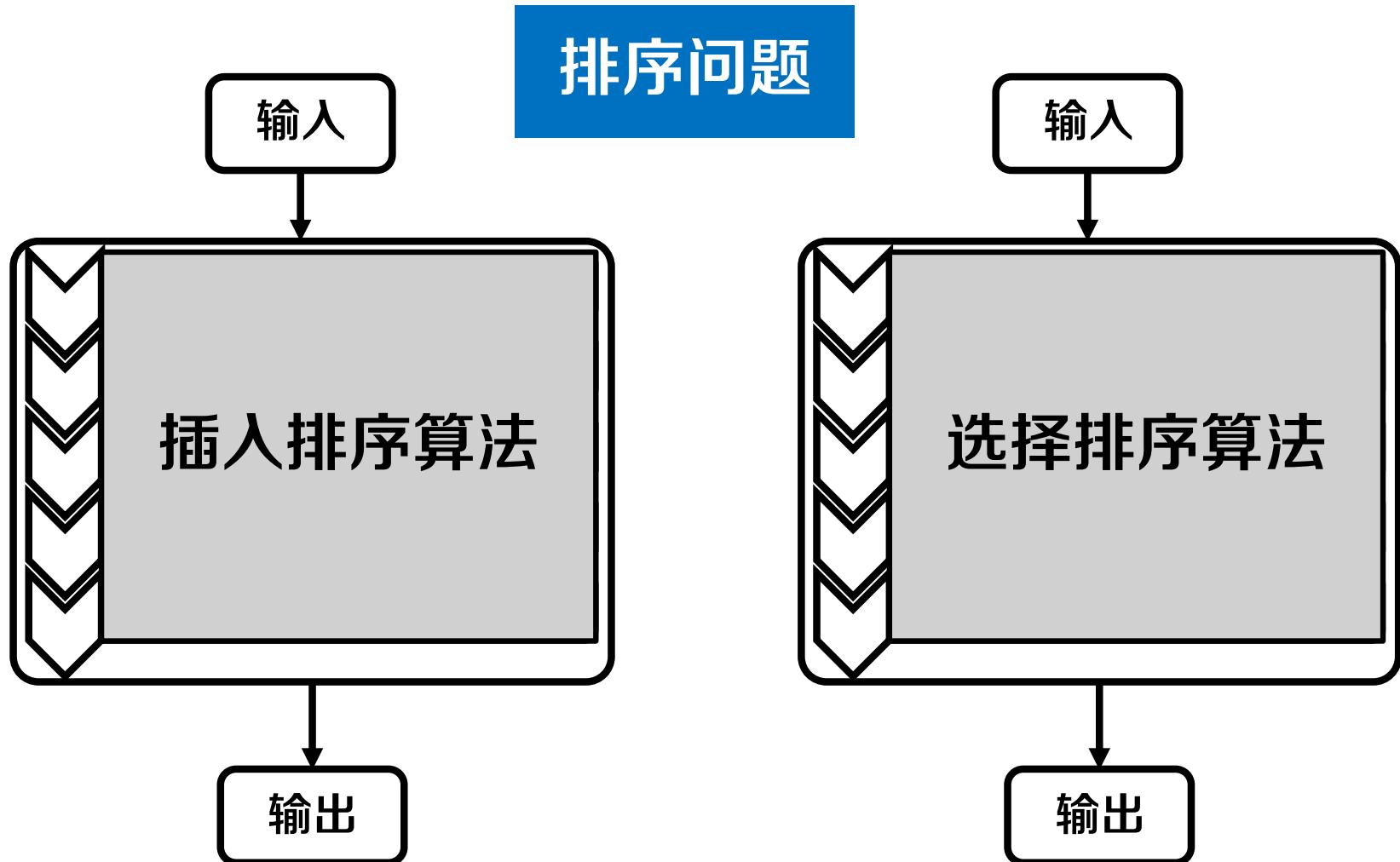
算法的性质

算法的表示

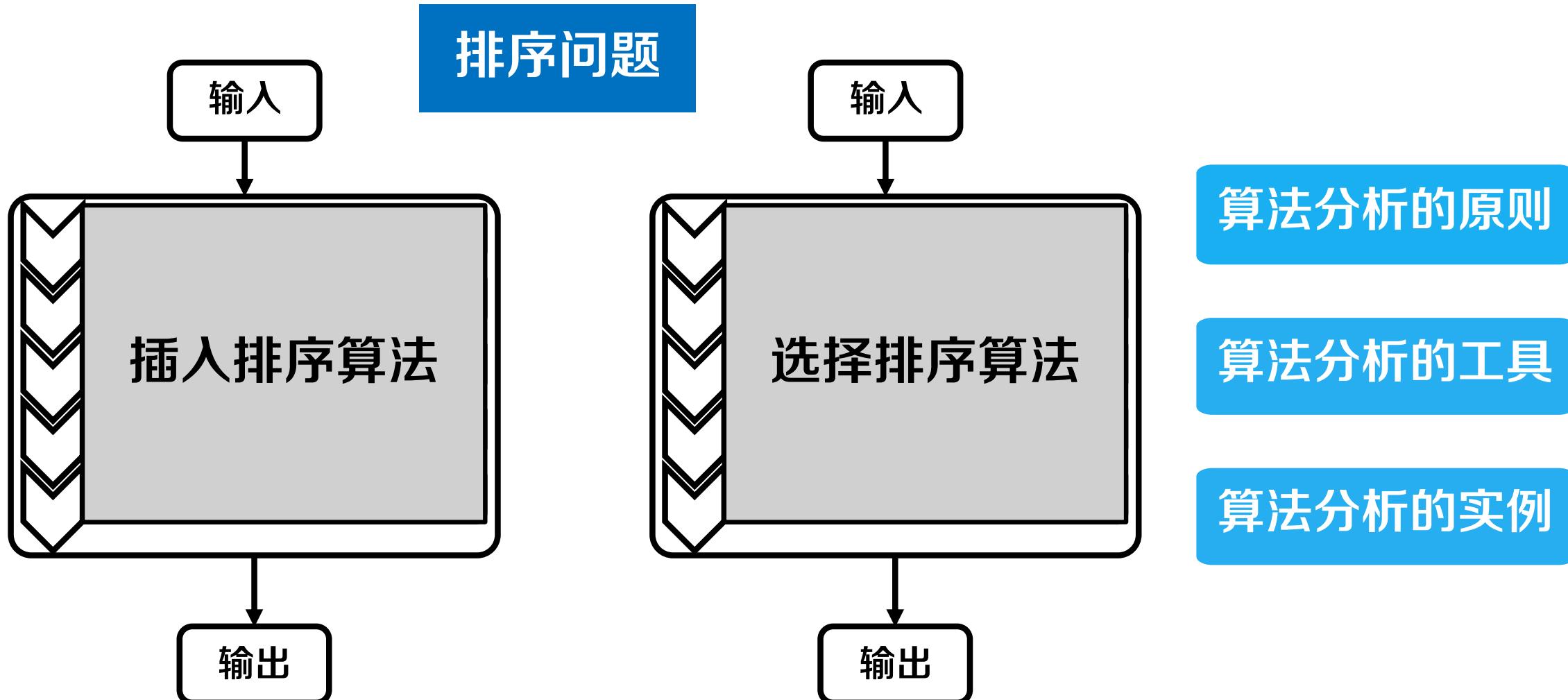
算法的分析



问题：如何比较不同算法性能？



分析算法的运行时间



分析算法的运行时间

算法分析的原则

- 机器的运算速度影响算法的运行时间

机器	运算速度	运行算法	运行时间
天河三号	百亿亿次/秒	插入排序	无法公平比较
个人电脑	十亿次/秒	选择排序	



算法分析的原则

- 机器的运算速度影响算法的运行时间

机器	运算速度	运行算法	运行时间
天河三号	百亿亿次/秒	插入排序	无法公平比较
个人电脑	十亿次/秒	选择排序	



分析算法的运行时间应独立于机器

算法分析的原则

- 归纳基本操作
 - 如：运算、赋值、比较

+	-	×	÷
:=	>	<	=

算法分析的原则

- 归纳基本操作
 - 如：运算、赋值、比较

+	-	×	÷
:=	>	<	=

- 统一机器性能
 - 假设基本操作代价均为1



算法分析的原则

- 归纳基本操作
 - 如：运算、赋值、比较
- 统一机器性能
 - 假设基本操作代价均为1

+	-	×	÷
\coloneqq	$>$	$<$	$=$



统一机器性能后，算法运行时间依赖于问题输入规模与实例

算法分析的原则

- 相同输入规模，实例影响运行

输入：数组 $A[a_1, a_2, \dots, a_n]$

输出：升序数组 $A'[a'_1, a'_2, \dots, a'_n]$ ，满足 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

for $j \leftarrow 2$ to n do

$key \leftarrow A[j]$

$i \leftarrow j - 1$

 while $i > 0$ and $A[i] > key$ do

$A[i + 1] \leftarrow A[i]$

$i \leftarrow i - 1$

 end

$A[i + 1] \leftarrow key$

end

循环次数未知

插入排序算法伪代码

算法分析的原则

- 相同输入规模，实例影响运行

- 插入排序最好情况：数组升序

- 比较次数： $1 + 1 + 1 + \dots + 1 = n - 1$



$n - 1$

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

算法分析的原则

- 相同输入规模，实例影响运行

- 插入排序最好情况：数组升序

- 比较次数： $1 + 1 + 1 + \dots + 1 = n - 1$

$$\underbrace{}_{n-1}$$

4	8	13	14	17	18	21	22	24	28	32	37	40	40	47	48
---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----

- 插入排序最坏情况：数组降序

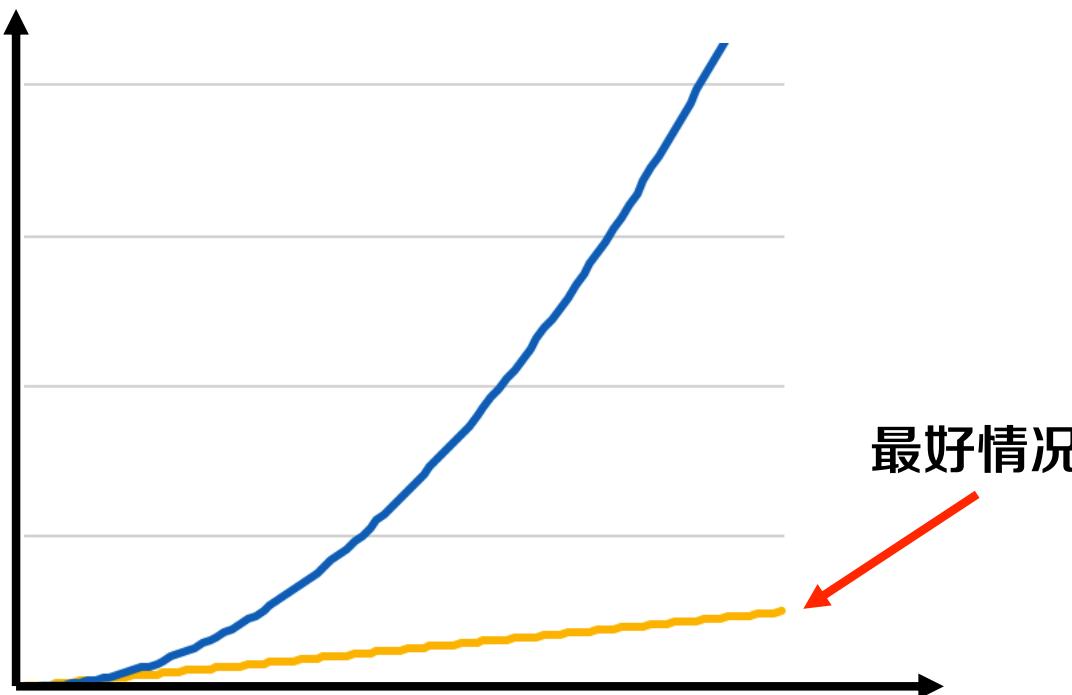
- 比较次数： $1 + 2 + 3 + \dots + (n - 1) = \frac{n(n-1)}{2}$

48	47	40	40	37	32	28	24	22	21	18	17	14	13	8	4
----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---

算法分析的原则

输入情况	情况说明
最好情况	不常出现，不具普遍性

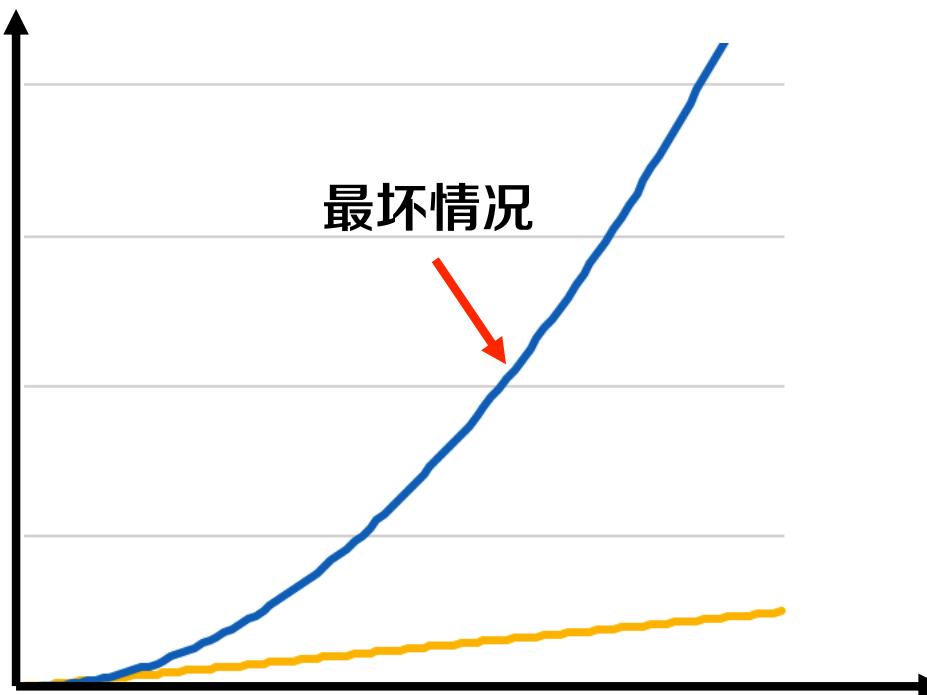
运行时间



算法分析的原则

输入情况	情况说明
最好情况	不常出现，不具普遍性
最坏情况	确定上界，更具一般性

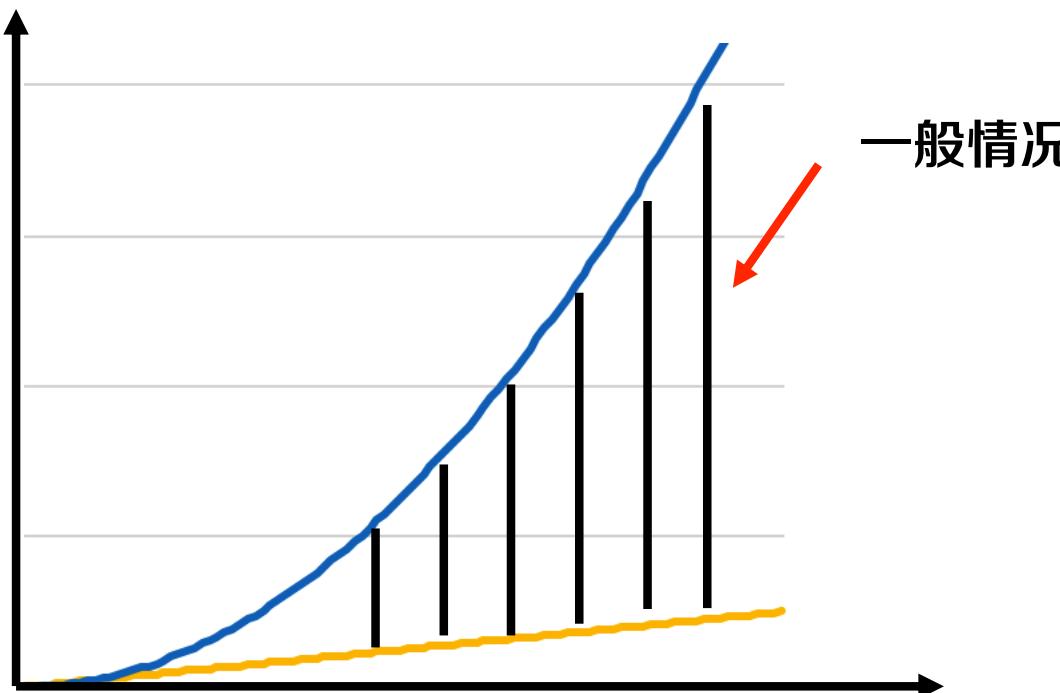
运行时间



算法分析的原则

输入情况	情况说明
最好情况	不常出现，不具普遍性
最坏情况	确定上界，更具一般性
一般情况	情况复杂，分析难度大

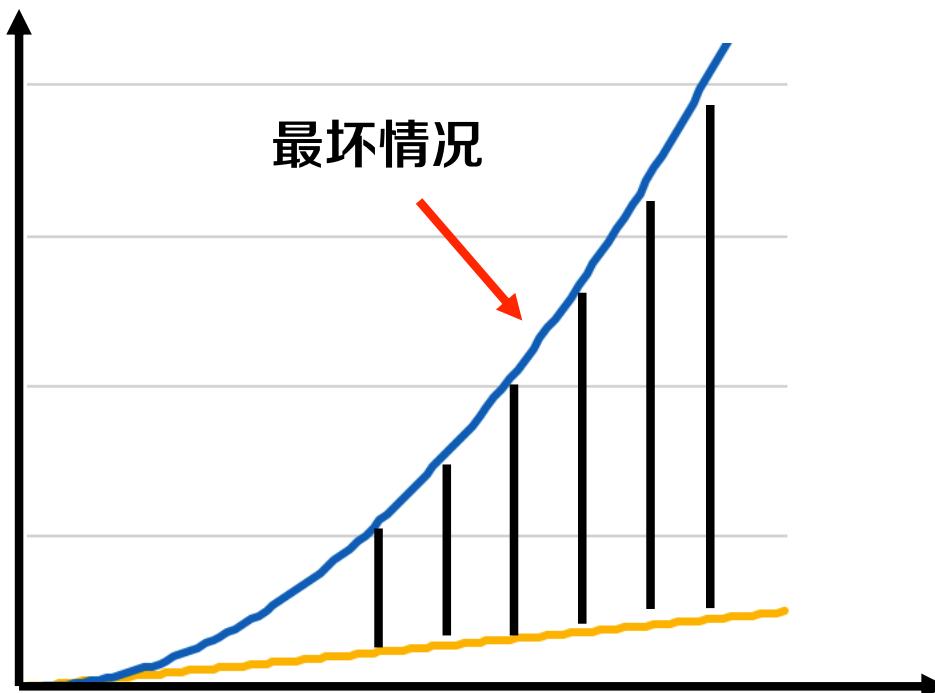
运行时间



算法分析的原则

输入情况	情况说明
最好情况	不常出现，不具普遍性
最坏情况	确定上界，更具一般性
一般情况	情况复杂，分析难度大

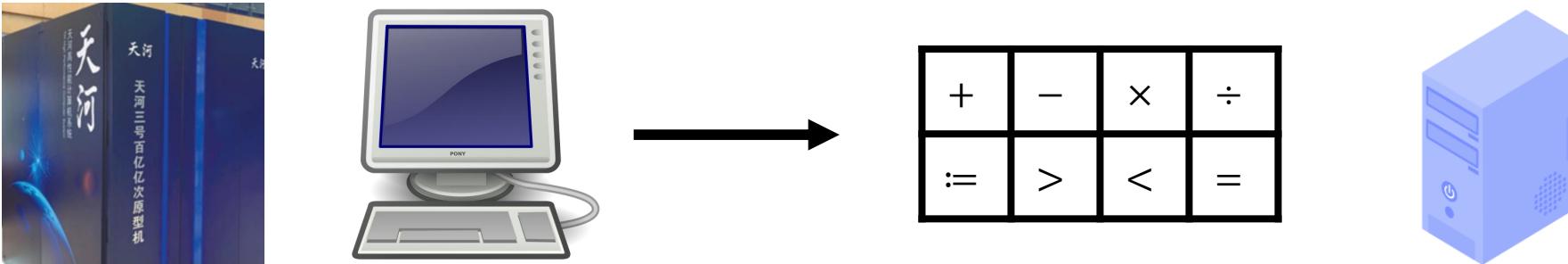
运行时间



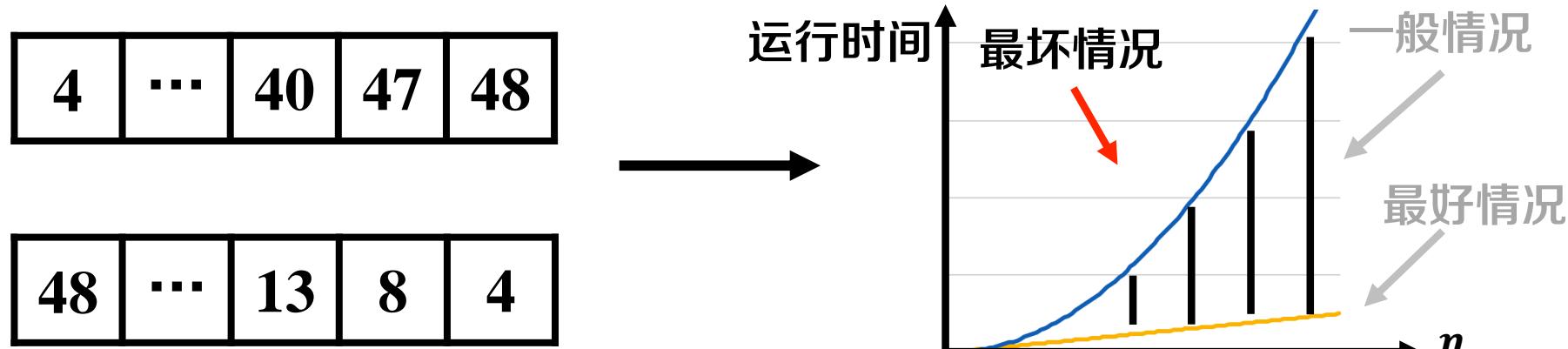
常用最坏情况分析算法运行时间

算法分析的原则

- 统一机器性能



- 分析最坏情况



算法运行时间仅依赖于问题输入规模 n , 表示为 $T(n)$

算法分析的工具

- 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do  
    key  $\leftarrow A[j]$   
     $i \leftarrow j - 1$   
    while  $i > 0$  and  $A[i] > key$  do  
        |  $A[i + 1] \leftarrow A[i]$   
        |  $i \leftarrow i - 1$   
    end  
     $A[i + 1] \leftarrow key$   
end
```

- 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

算法分析的工具

- 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do  
    key  $\leftarrow A[j]$   
     $i \leftarrow j - 1$   
    while  $i > 0$  and  $A[i] > key$  do  
        |  $A[i + 1] \leftarrow A[i]$   
        |  $i \leftarrow i - 1$   
    end  
     $A[i + 1] \leftarrow key$   
end
```

- 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

计算每行伪代码操作次数

算法分析的工具

● 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do .....  $n$ 次
    key  $\leftarrow A[j]$  .....  $n - 1$ 次
     $i \leftarrow j - 1$  .....  $n - 1$ 次
    while  $i > 0$  and  $A[i] > key$  do
        |  $A[i + 1] \leftarrow A[i]$ 
        |  $i \leftarrow i - 1$ 
    end
     $A[i + 1] \leftarrow key$ 
end
```

● 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do
    for  $j \leftarrow i + 1$  to  $n$  do
        | if  $A[i] > A[j]$  then
        |   | 交换  $A[i]$  和  $A[j]$ 
        |   end
    end
end
```

计算每行伪代码操作次数

算法分析的工具

● 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do .....  $n$ 次  
    key  $\leftarrow A[j]$  .....  $n - 1$ 次  
     $i \leftarrow j - 1$  .....  $n - 1$ 次  
    while  $i > 0$  and  $A[i] > key$  do .....  $\sum_{k=2}^n k$ 次  
        |  $A[i + 1] \leftarrow A[i]$  .....  $\sum_{k=2}^n k - 1$ 次  
        |  $i \leftarrow i - 1$  .....  $\sum_{k=2}^n k - 1$ 次  
    end  
     $A[i + 1] \leftarrow key$   
end
```

● 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

计算每行伪代码操作次数

算法分析的工具

● 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do .....  $n$ 次  
    key  $\leftarrow A[j]$  .....  $n - 1$ 次  
     $i \leftarrow j - 1$  .....  $n - 1$ 次  
    while  $i > 0$  and  $A[i] > key$  do .....  $\sum_{k=2}^n k$ 次  
        |  $A[i + 1] \leftarrow A[i]$  .....  $\sum_{k=2}^n k - 1$ 次  
        |  $i \leftarrow i - 1$  .....  $\sum_{k=2}^n k - 1$ 次  
    end  
     $A[i + 1] \leftarrow key$  .....  $n - 1$ 次  
end
```

● 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

计算每行伪代码操作次数

算法分析的工具

● 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do .....  $n$ 次  
    key  $\leftarrow A[j]$  .....  $n - 1$ 次  
     $i \leftarrow j - 1$  .....  $n - 1$ 次  
    while  $i > 0$  and  $A[i] > key$  do .....  $\sum_{k=2}^n k$ 次  
        |  $A[i + 1] \leftarrow A[i]$  .....  $\sum_{k=2}^n k - 1$ 次  
        |  $i \leftarrow i - 1$  .....  $\sum_{k=2}^n k - 1$ 次  
    end  
     $A[i + 1] \leftarrow key$  .....  $n - 1$ 次  
end
```

● 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do .....  $n$ 次  
    for  $j \leftarrow i + 1$  to  $n$  do .....  $\sum_{k=0}^{n-2} (n - k)$ 次  
        | if  $A[i] > A[j]$  then .....  $\sum_{k=1}^{n-1} (n - k)$ 次  
        | | 交换  $A[i]$  和  $A[j]$  .....  $\sum_{k=1}^{n-1} k$ 次  
    end  
end
```

计算每行伪代码操作次数

算法分析的工具

● 插入排序最坏情况

```
for j ← 2 to n do ..... n次
    key ← A[j] ..... n - 1次
    i ← j - 1 ..... n - 1次
    while i > 0 and A[i] > key do .....  $\sum_{k=2}^n k$  次
        | A[i + 1] ← A[i] .....  $\sum_{k=2}^n k - 1$  次
        | i ← i - 1 .....  $\sum_{k=2}^n k - 1$  次
    end
    A[i + 1] ← key ..... n - 1次
end
```

● 选择排序最坏情况

```
for i ← 1 to n - 1 do ..... n次
    for j ← i + 1 to n do .....  $\sum_{k=0}^{n-2} (n - k)$  次
        | if A[i] > A[j] then .....  $\sum_{k=1}^{n-1} (n - k)$  次
        | | 交换 A[i] 和 A[j] .....  $\sum_{k=1}^{n-1} k$  次
    end
end
```

$$T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4$$

求和

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

求和

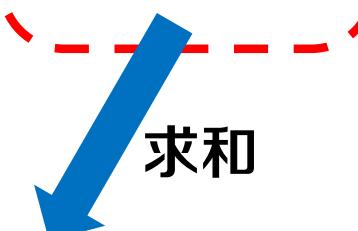
算法分析的工具

● 插入排序最坏情况

```
for j ← 2 to n do ..... n次
    key ← A[j] ..... n - 1次
    i ← j - 1 ..... n - 1次
    while i > 0 and A[i] > key do .....  $\sum_{k=2}^n k$  次
        | A[i + 1] ← A[i] .....  $\sum_{k=2}^n k - 1$  次
        | i ← i - 1 .....  $\sum_{k=2}^n k - 1$  次
    end
    A[i + 1] ← key ..... n - 1次
end
```

● 选择排序最坏情况

```
for i ← 1 to n - 1 do ..... n次
    for j ← i + 1 to n do .....  $\sum_{k=0}^{n-2} (n - k)$  次
        | if A[i] > A[j] then .....  $\sum_{k=1}^{n-1} (n - k)$  次
        | | 交换 A[i] 和 A[j] .....  $\sum_{k=1}^{n-1} k$  次
    end
end
```

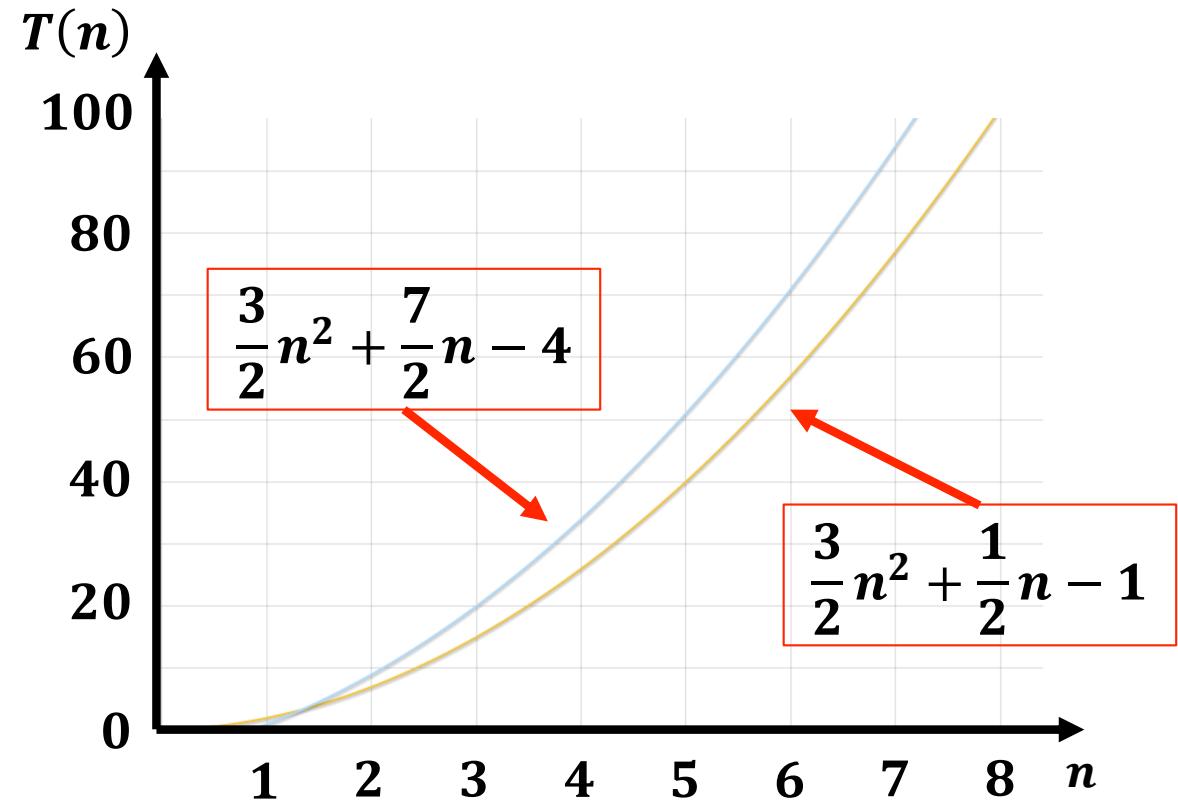


$$T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4$$

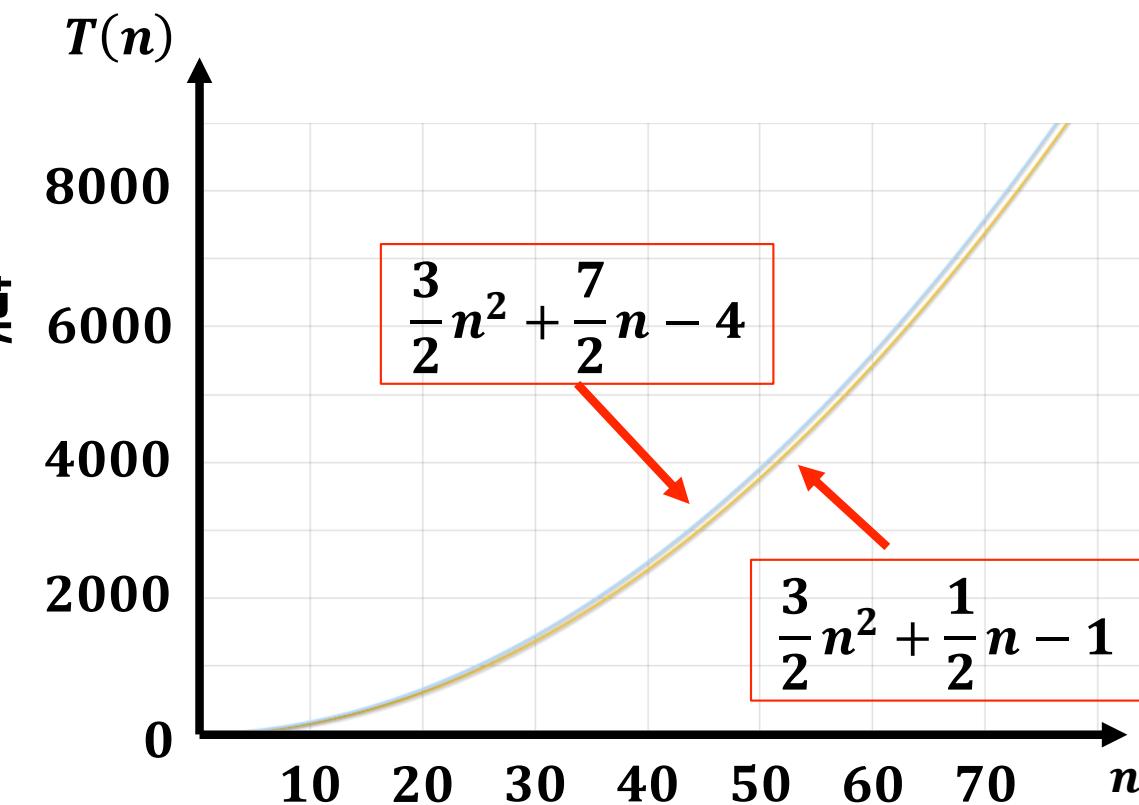
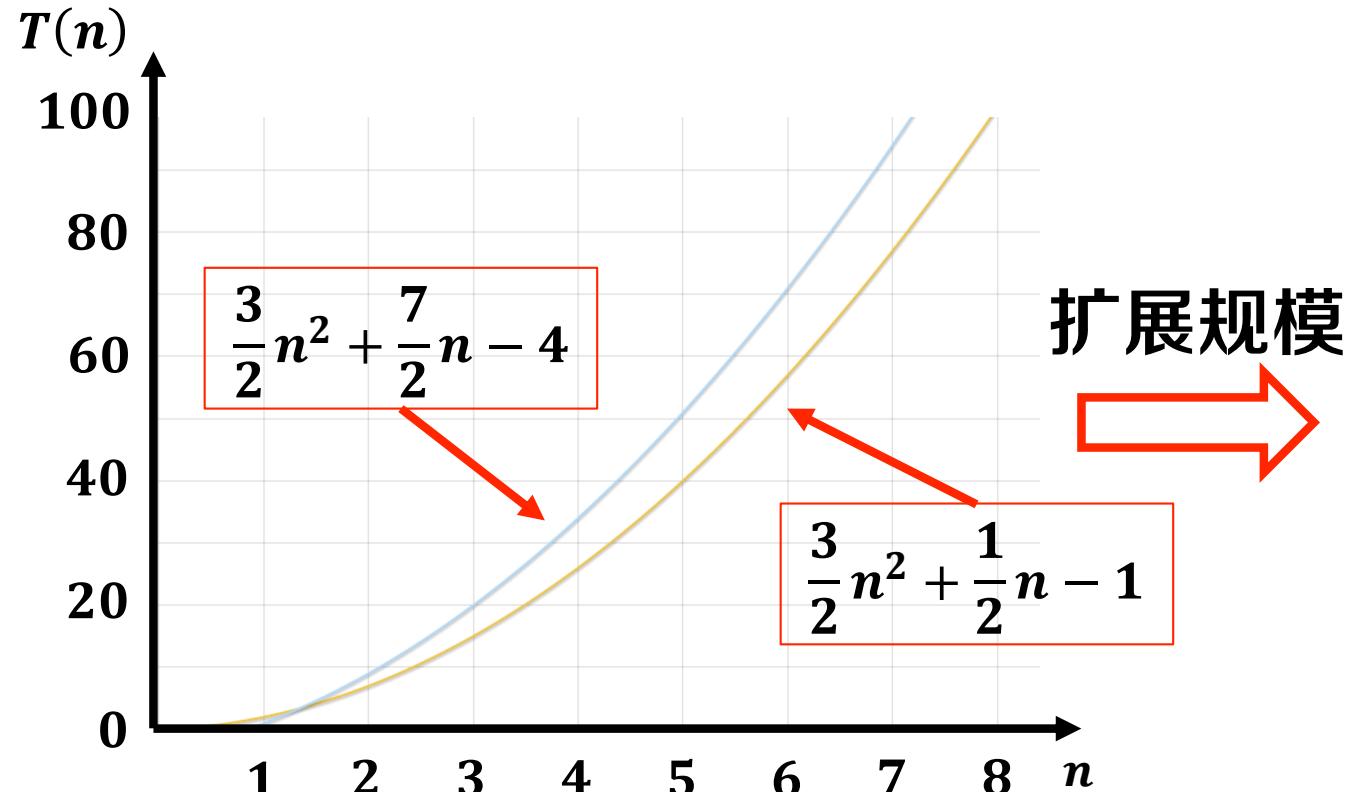
$$T(n) = \frac{3}{2}n^2 + \frac{1}{2}n - 1$$

问题：能否简洁地衡量算法运行时间？

算法分析的工具

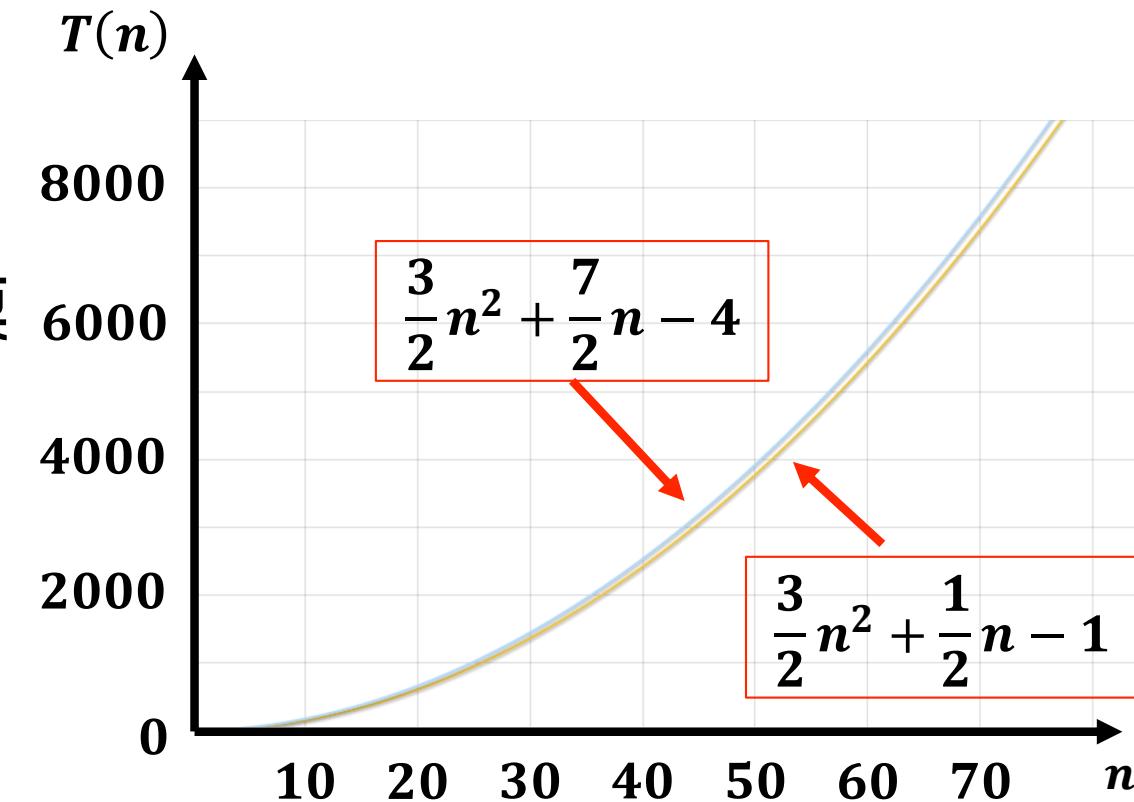
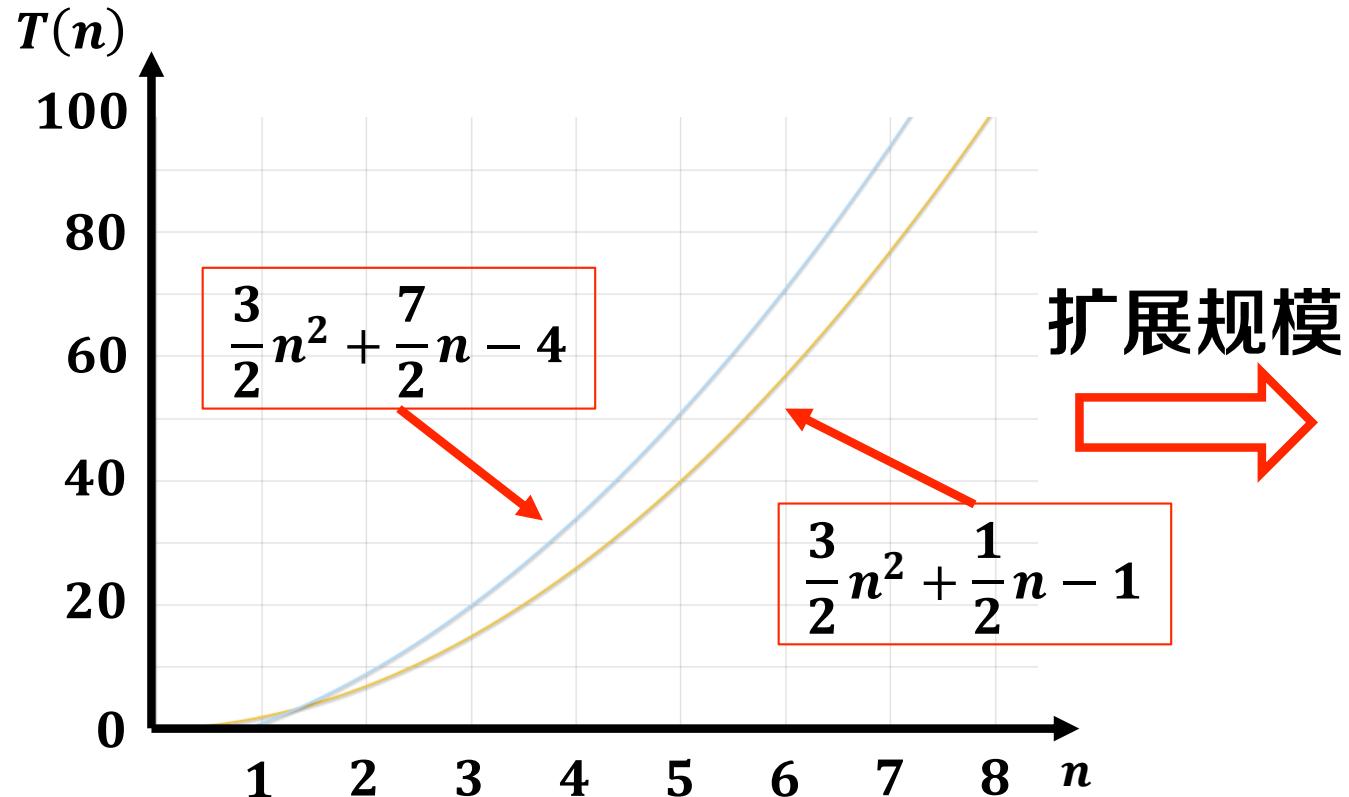


算法分析的工具



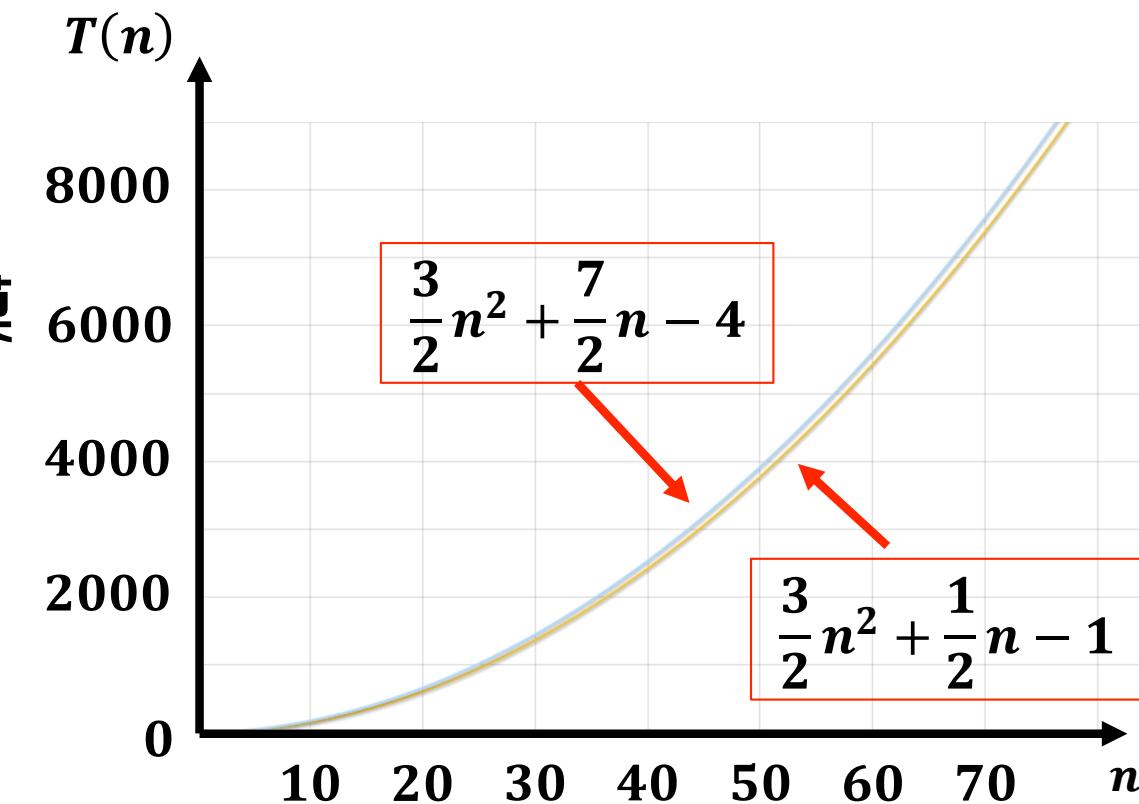
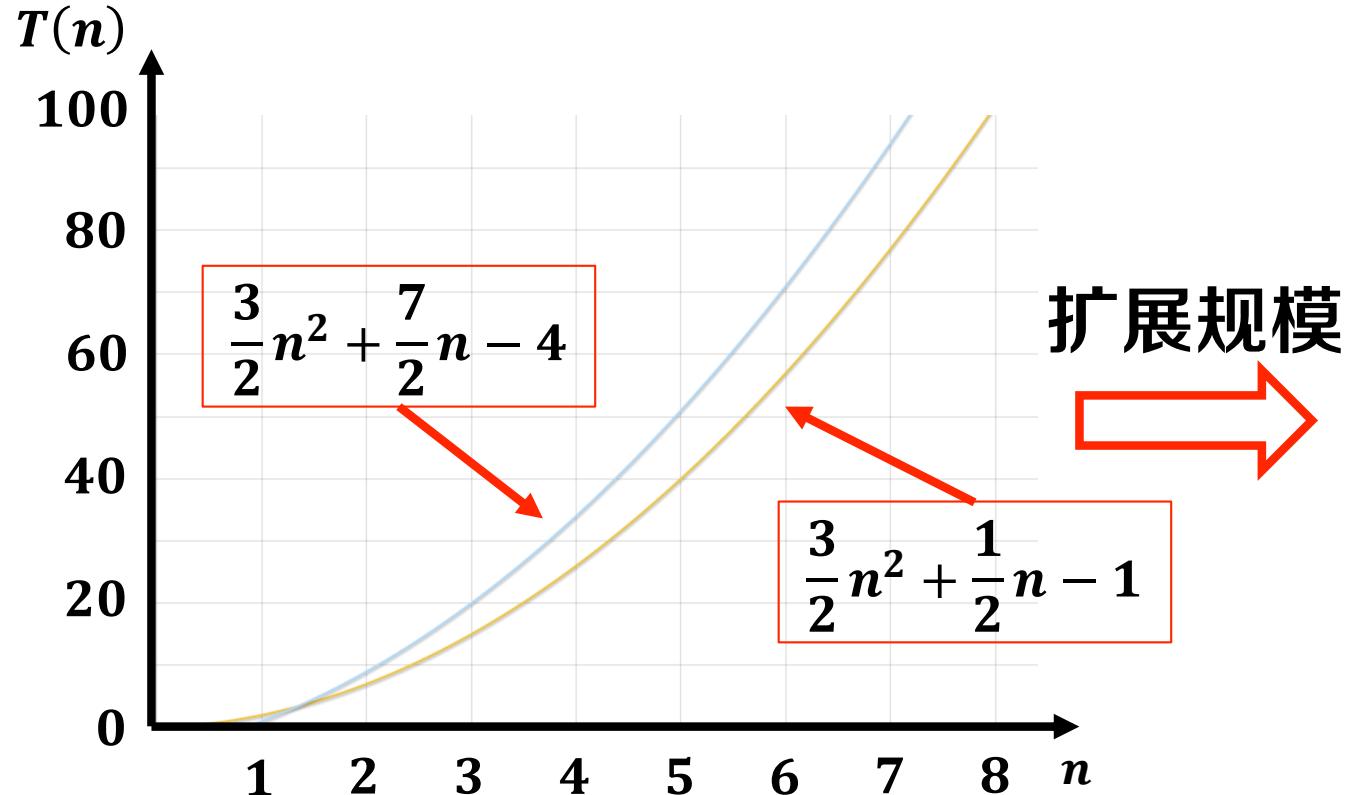
- 在 n 充分大时，两者相差不大

算法分析的工具



- 在 n 充分大时，两者相差不大
- 原因？

算法分析的工具



- 在 n 充分大时，两者相差不大
- 原因：两函数的最高阶项相同

算法分析的工具

- 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do  
    key  $\leftarrow A[j]$   
     $i \leftarrow j - 1$   
    while  $i > 0$  and  $A[i] > key$  do  
        |  $A[i + 1] \leftarrow A[i]$   
        |  $i \leftarrow i - 1$   
    end  
     $A[i + 1] \leftarrow key$   
end
```

- 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

$$T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4$$

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

渐近分析：忽略 $T(n)$ 的系数与低阶项，仅关注高阶项，用记号 Θ 表示

算法分析的工具

- 插入排序最坏情况

```
for  $j \leftarrow 2$  to  $n$  do  
    key  $\leftarrow A[j]$   
     $i \leftarrow j - 1$   
    while  $i > 0$  and  $A[i] > key$  do  
        |  $A[i + 1] \leftarrow A[i]$   
        |  $i \leftarrow i - 1$   
    end  
     $A[i + 1] \leftarrow key$   
end
```

- 选择排序最坏情况

```
for  $i \leftarrow 1$  to  $n - 1$  do  
    for  $j \leftarrow i + 1$  to  $n$  do  
        | if  $A[i] > A[j]$  then  
        | | 交换  $A[i]$  和  $A[j]$   
        | end  
    end  
end
```

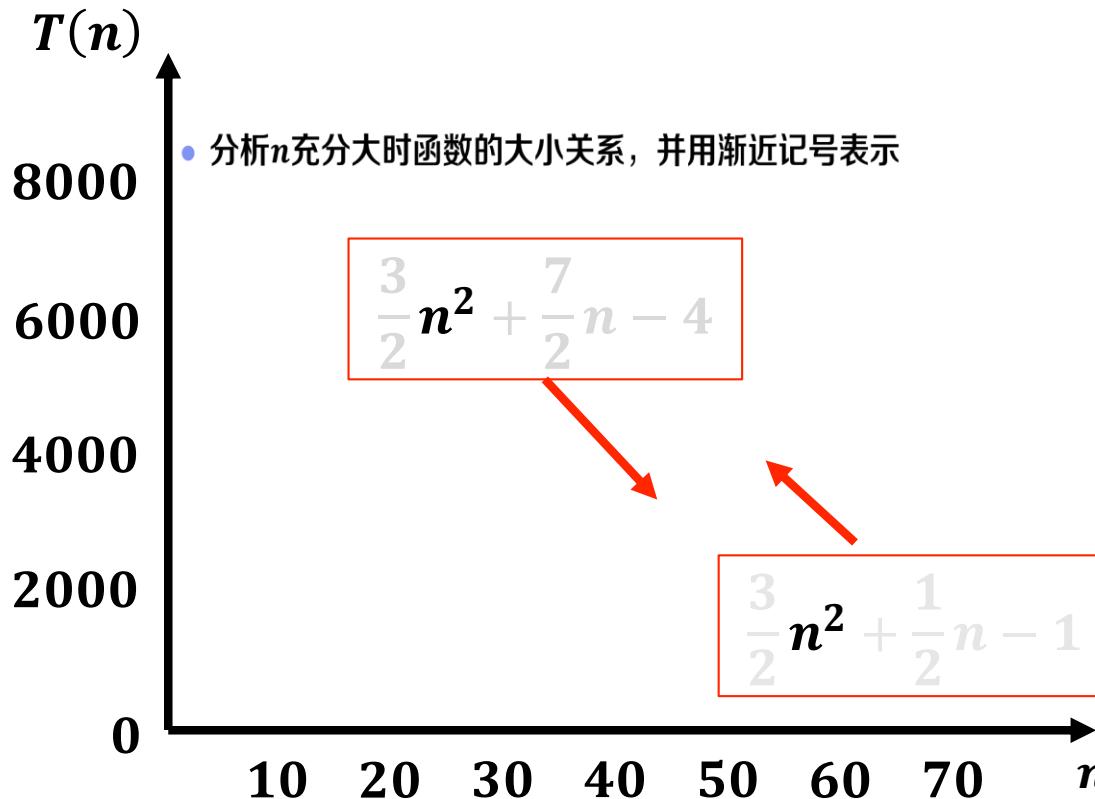
$$T(n) = \Theta(n^2)$$

$$T(n) = \Theta(n^2)$$

渐近分析：忽略 $T(n)$ 的系数与低阶项，仅关注高阶项，用记号 Θ 表示

渐近分析

- 分析 n 充分大时函数的大小关系，并用渐近记号表示



渐近记号	名称
$T(n) = \Theta(g(n))$	渐近紧确界
$T(n) = O(g(n))$	渐近上界
$T(n) = \Omega(g(n))$	渐近下界

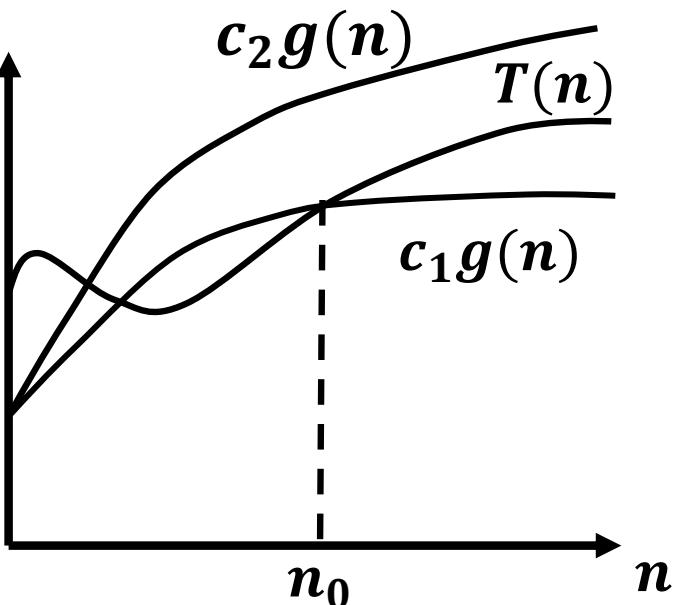
渐近分析：渐近紧确界

Θ记号

定义：

- 对于给定的函数 $g(n)$, $\Theta(g(n))$ 表示以下函数的集合:

$$\Theta(g(n)) = \{T(n) : \exists c_1, c_2, n_0 > 0, \text{使得} \forall n \geq n_0, c_1g(n) \leq T(n) \leq c_2g(n)\}$$



$$T(n) = \Theta(g(n))$$

渐近分析：渐近紧确界

Θ记号

定义：

- 对于给定的函数 $g(n)$, $\Theta(g(n))$ 表示以下函数的集合:

$$\Theta(g(n)) = \{T(n) : \exists c_1, c_2, n_0 > 0, \text{使得} \forall n \geq n_0, c_1 g(n) \leq T(n) \leq c_2 g(n)\}$$

Θ记号示例

- $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4 = ?$

Θ记号

定义：

- 对于给定的函数 $g(n)$, $\Theta(g(n))$ 表示以下函数的集合:

$$\Theta(g(n)) = \{T(n) : \exists c_1, c_2, n_0 > 0, \text{使得} \forall n \geq n_0, c_1 g(n) \leq T(n) \leq c_2 g(n)\}$$

Θ记号示例

- $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4 = ?$
- 令 $n_0 = 2$, 当 $n \geq n_0$ 时, 有
- $\frac{3}{2}n^2 + \frac{7}{2}n - 4 \geq \frac{3}{2}n^2 \geq n^2$

Θ记号

定义：

- 对于给定的函数 $g(n)$, $\Theta(g(n))$ 表示以下函数的集合:

$$\Theta(g(n)) = \{T(n) : \exists c_1, c_2, n_0 > 0, \text{使得} \forall n \geq n_0, c_1 g(n) \leq T(n) \leq c_2 g(n)\}$$

Θ记号示例

- $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4 = ?$
- 令 $n_0 = 2$, 当 $n \geq n_0$ 时, 有
- $\frac{3}{2}n^2 + \frac{7}{2}n - 4 \geq \frac{3}{2}n^2 \geq n^2$
- $\frac{3}{2}n^2 + \frac{7}{2}n - 4 \leq \frac{3}{2}n^2 + \frac{7}{2}n^2 + n^2 = 6n^2$

Θ记号

定义：

- 对于给定的函数 $g(n)$, $\Theta(g(n))$ 表示以下函数的集合:

$$\Theta(g(n)) = \{T(n) : \exists c_1, c_2, n_0 > 0, \text{使得} \forall n \geq n_0, c_1 g(n) \leq T(n) \leq c_2 g(n)\}$$

Θ记号示例

- $T(n) = \frac{3}{2}n^2 + \frac{7}{2}n - 4 = \Theta(n^2)$
- 令 $n_0 = 2$, 当 $n \geq n_0$ 时, 有
- $\frac{3}{2}n^2 + \frac{7}{2}n - 4 \geq \frac{3}{2}n^2 \geq n^2$
- $\frac{3}{2}n^2 + \frac{7}{2}n - 4 \leq \frac{3}{2}n^2 + \frac{7}{2}n^2 + n^2 = 6n^2$
- 故存在 $c_1 = 1, c_2 = 6, n_0 = 2$, 使得 $\forall n \geq n_0, c_1 n^2 \leq T(n) \leq c_2 n^2$

渐近分析：渐近紧确界

- Θ 记号示例

- $\frac{3}{2}n^5 + \frac{7}{2}n - 10 =$

渐近分析：渐近紧确界

- Θ 记号示例

- $\frac{3}{2}n^5 + \frac{7}{2}n - 10 = \Theta(n^5)$

漸近分析：漸近緊確界

- Θ 記号示例

- $\frac{3}{2}n^5 + \frac{7}{2}n - 10 = \Theta(n^5)$
- $n^3 - n^2 + n =$

渐近分析：渐近紧确界

- Θ 记号示例

- $\frac{3}{2}n^5 + \frac{7}{2}n - 10 = \Theta(n^5)$
- $n^3 - n^2 + n = \Theta(n^3)$

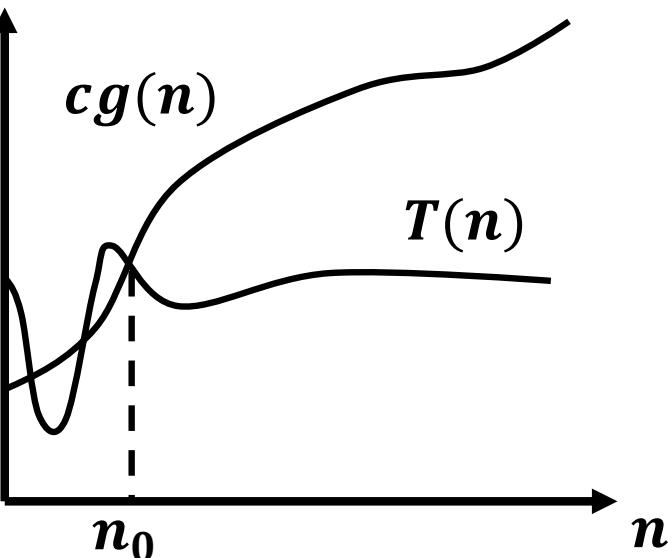
漸近分析：漸近上界

O 记号

定义：

- 对于给定的函数 $g(n)$, $O(g(n))$ 表示以下函数的集合:

$$O(g(n)) = \{T(n) : \exists c, n_0 > 0, \text{使得} \forall n \geq n_0, 0 \leq T(n) \leq cg(n)\}$$



$$T(n) = O(g(n))$$

渐近分析：渐近上界

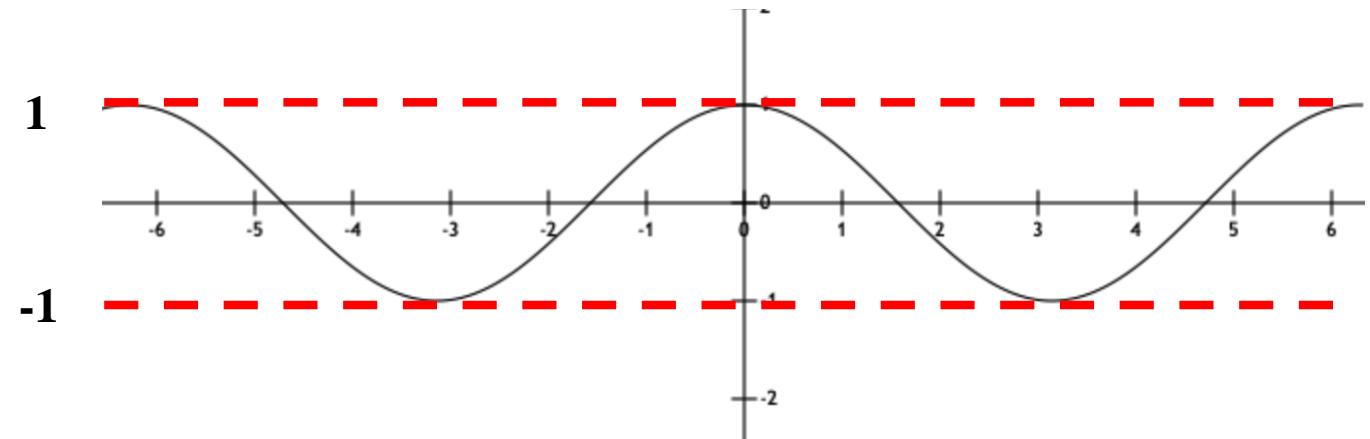
- O 记号示例

- $\cos n$

渐近分析：渐近上界

- O 记号示例

- $\cos n \leq 1$



漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$

漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$

- $\frac{n^2}{2} - 12n =$

漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$

漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n =$

漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} =$

对数换底公式
 $\log_x N = \frac{\log_y N}{\log_y x}$

漸近分析：漸近上界

- O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$

对数换底公式

$$\log_x N = \frac{\log_y N}{\log_y x}$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \cdots + \frac{1}{\frac{n}{2}} + \frac{1}{n}$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$\begin{aligned}&= \frac{1}{1} + \left(\frac{1}{2} + \frac{1}{3} \right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \right) + \left(\frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \dots + \frac{1}{n} \right) \\&< \frac{1}{1} + \left(\frac{1}{2} + \frac{1}{2} \right) + \left(\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} \right) + \left(\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} \right) + \dots + \left(\frac{1}{\frac{n}{2}} + \frac{1}{\frac{n}{2}} \right) + \frac{1}{n}\end{aligned}$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \cdots + \frac{1}{\frac{n}{2}} + \frac{1}{n}$$

$$= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \cdots + \frac{n}{2} \left(\frac{1}{\frac{n}{2}}\right) + \frac{1}{n}$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \cdots + \frac{1}{\frac{n}{2}} + \frac{1}{n}$$

$$= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \cdots + \frac{n}{2} \left(\frac{1}{\frac{n}{2}}\right) + \frac{1}{n}$$



log n 项

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \cdots + \frac{1}{\frac{n}{2}} + \frac{1}{n}$$

$$= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \cdots + \frac{n}{2} \left(\frac{1}{\frac{n}{2}}\right) + \frac{1}{n} = \frac{1}{n} + \sum_{j=0}^{\log n - 1} 1$$

渐近分析：渐近上界

• O 记号示例

- $\cos n = O(1)$
- $\frac{n^2}{2} - 12n = O(n^2)$
- $\log_7 n = \frac{\log_2 n}{\log_2 7} = O(\log_2 n) = O(\log n)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设 n 是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$< \frac{1}{1} + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \cdots + \frac{1}{\frac{n}{2}} + \frac{1}{n}$$

$$= \frac{1}{1} + 2 \cdot \left(\frac{1}{2}\right) + 4 \cdot \left(\frac{1}{4}\right) + 8 \cdot \left(\frac{1}{8}\right) + \cdots + \frac{n}{2} \left(\frac{1}{\frac{n}{2}}\right) + \frac{1}{n} = \frac{1}{n} + \sum_{j=0}^{\log n - 1} 1$$

$$= \log n + \frac{1}{n} = O(\log n)$$

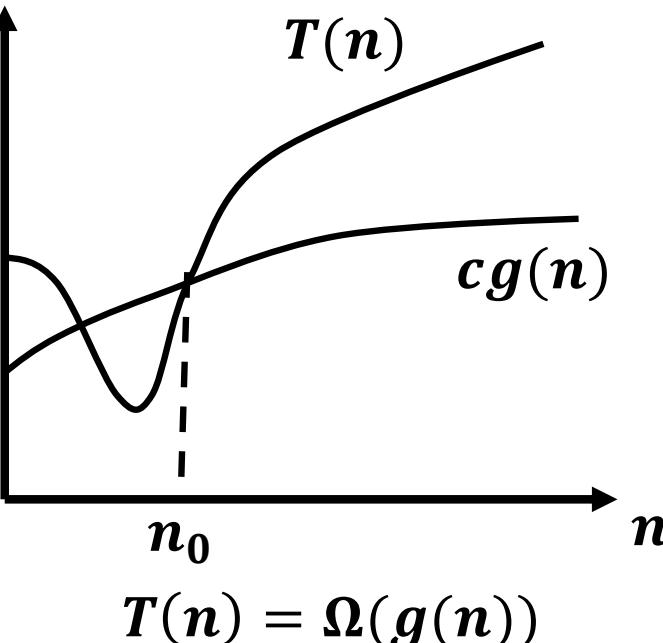
漸近分析：漸近下界

Ω 记号

定义：

- 对于给定的函数 $g(n)$, $\Omega(g(n))$ 表示以下函数的集合:

$$\Omega(g(n)) = \{T(n) : \exists c, n_0 > 0, \text{使得} \forall n \geq n_0, 0 \leq cg(n) \leq T(n)\}$$



渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n =$

漸近分析：漸近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$

漸近分析：漸近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$
- $n^2 + n = \Omega(n^2)$

渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$
- $n^2 + n = \Omega(n^2)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$
- $n^2 + n = \Omega(n^2)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)
 $= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$

渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$

- $n^2 + n = \Omega(n^2)$

- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

$$\begin{aligned}&= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n} \\&> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \cdots + \frac{1}{n}\end{aligned}$$

渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$

- $n^2 + n = \Omega(n^2)$

- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \cdots \frac{1}{n}$$

$$= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \cdots + \frac{n}{2} \left(\frac{1}{n}\right)$$



log n项

渐近分析：渐近下界

• Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$
- $n^2 + n = \Omega(n^2)$
- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$

$$> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \cdots \frac{1}{n}$$

$$= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \cdots + \frac{n}{2} \left(\frac{1}{n}\right)$$

$$= 1 + \sum_{j=1}^{\log n} \frac{1}{2}$$

渐近分析：渐近下界

- Ω 记号示例

- $n^3 - 2n = \Omega(n^3)$

- $n^2 + n = \Omega(n^2)$

- $\sum_{i=1}^n \frac{1}{i}$ (假设n是2的整数幂)

$$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \cdots + \frac{1}{n}$$
$$> \frac{1}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} + \cdots \frac{1}{n}$$

$$= \frac{1}{1} + \frac{1}{2} + 2 \cdot \left(\frac{1}{4}\right) + 4 \cdot \left(\frac{1}{8}\right) + 8 \cdot \left(\frac{1}{16}\right) + \cdots + \frac{n}{2} \left(\frac{1}{n}\right)$$

$$= 1 + \sum_{j=1}^{\log n} \frac{1}{2}$$

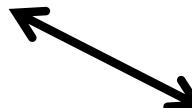
$$= 1 + \frac{1}{2} \log n$$

$$= \Omega(\log n)$$

渐近分析

- $T(n) = \Theta(g(n))$ 等价于: $T(n) = \Omega(g(n))$ 且 $T(n) = O(g(n))$

渐近记号	名称	输入情况	情况说明
Θ	渐近紧确界	最好情况	不常出现, 不具普遍性
O	渐近上界	最坏情况	确定上界, 更具一般性
Ω	渐近下界		



算法运行时间称为算法的时间复杂度, 通常使用渐近记号 O 表示

算法分析的实例

- 插入排序最坏情况

```
for j ← 2 to n do
    key ← A[j]
    i ← j - 1
    while i > 0 and A[i] > key do
        | A[i + 1] ← A[i]
        | i ← i - 1
    end
    A[i + 1] ← key
end
```

$O(n)$ $O(n^2)$

- 选择排序最坏情况

```
for i ← 1 to n - 1 do
    for j ← i + 1 to n do
        | if A[i] > A[j] then
        |   | 交换 A[i] 和 A[j]
        | end
    end
end
```

$O(n)$ $O(n^2)$

算法的分析小结

- 算法分析的原则

+	-	×	÷
:=	>	<	=



统一机器性能

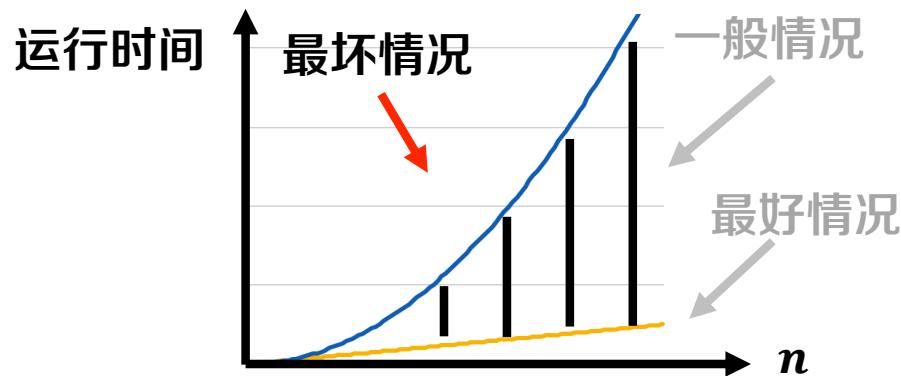
算法的分析小结

● 算法分析的原则

+	-	×	÷
:=	>	<	=



统一机器性能



分析最坏情况

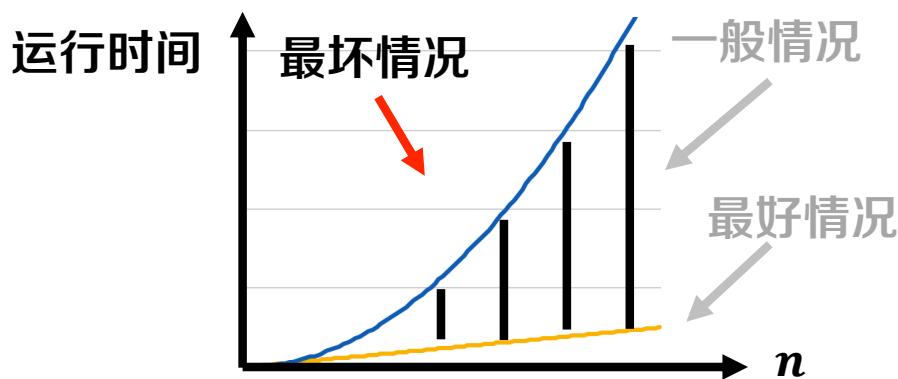
算法的分析小结

- 算法分析的原则

+	-	×	÷
\coloneqq	$>$	$<$	$=$

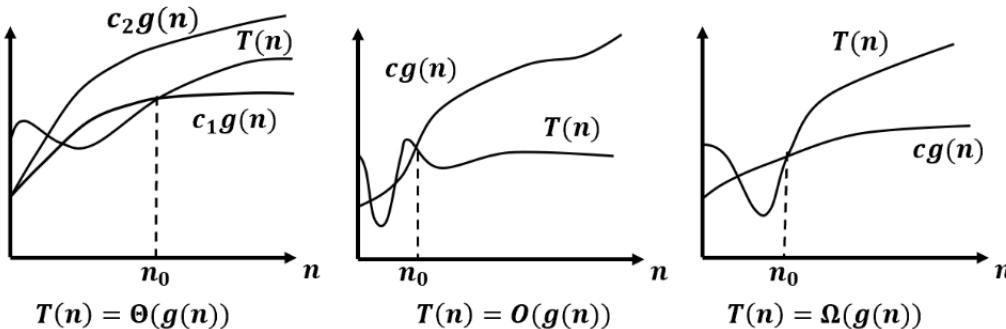


统一机器性能



分析最坏情况

- 算法分析的工具



采用渐近分析

算法设计与分析

分而治之篇

归并排序

递归式求解

最大子数组问题 I

逆序对计数问题

快速排序

次序选择问题

动态规划篇

0-1背包问题

最大子数组问题 II

最长公共子序列问题

最长公共子串问题

编辑距离问题

钢条切割问题

矩阵链乘法问题

部分背包问题

霍夫曼编码

活动选择问题

贪心策略篇