

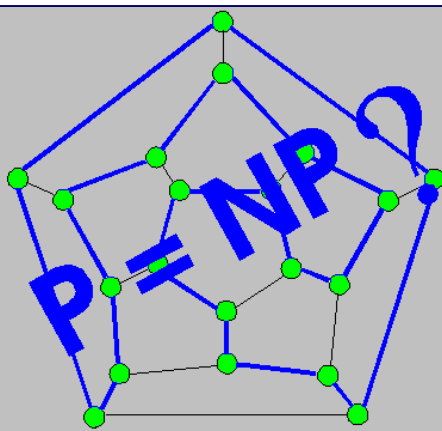


# 3-蛮力法

陆伟

算法设计与分析

Introduction to the Design and Analysis of Algorithms



September 21, 2022

# Lecture Overview

1

- 蛮力法基本思想

2

- 排序问题

3

- 查找问题

4

- 字符串匹配问题

5

- 最近对问题

6

- 搜索问题

7

- 组合问题

8

- 方程问题


# 蛮力法基本思想

- 蛮力法是一种简单直接地解决问题的方法，常常直接基于问题的描述和所涉及的概念定义。
- 也可以用“just do it!”来描述蛮力法的策略。
- 一般来说蛮力策略也常常是最容易实现的方法。

$$a^n = \underbrace{a \times a \times \dots \times a}_n$$

中国古代数学家张丘建在《算经》中提出了著名的“百钱百鸡问题”：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买百鸡，翁、母、雏各几何？

# 蛮力法基本思想

- 虽然巧妙和高效的算法很少来自于蛮力法，但它在算法设计策略中仍然具有重要地位。
  - 蛮力法适应能力强，是唯一一种几乎什么问题都能解决的一般性方法。
  - 蛮力法一般容易实现,在问题规模不大的情况下，蛮力法能够快速给出一种可接受速度下的求解方法。
  - 虽然通常情况下蛮力法效率很低，但可以作为衡量同类问题更高效算法的准绳。
  - 不要小看蛮力法 

# 排序问题

问题：给定一个可排序的 $n$ 个元素序列（数字、字符或字符串），对它们按照非降序方式重新排列。

- 1、理解问题
- 2、选择策略
- 3、算法设计
- 4、正确性证明
- 5、算法分析
- 6、程序设计

# 排序问题

思想：首先扫描整个序列，找到其中一个最小元素，然后和第一个元素交换，将最小元素归位。然后从第二个元素开始扫描序列，找到后 $n-1$ 个元素中的一个最小元素，然后和第二个元素交换，将第二小元素归位。进行 $n-1$ 遍扫描之后，排序完成。

# 排序问题

## ■ 流程图或伪代码

**算法 selectSort(A[n])**

//用选择法对给定数组排序

//输入： 一个可排序数组A[0.. $n-1$ ]

//输出： 升序排序的数组A[0.. $n-1$ ]

for  $i \leftarrow 0$  to  $n-2$  do

$min \leftarrow i$

    for  $j=i+1$  to  $n-1$  do

        if  $A[j] < A[min]$   $min \leftarrow j$

    swap A[i] and A[min]

# 排序问题

## ■ 小规模实例分析

1	2	3	4	5	6	$n = 7$
<u>89</u>	45	68	90	29	34	17
17	<u>45</u>	68	90	29	34	89
17	29	<u>68</u>	90	45	34	89
17	29	34	<u>90</u>	45	68	89
17	29	34	45	<u>90</u>	68	89
17	29	34	45	68	<u>90</u>	89
17	29	34	45	68	89	90



# 排序问题

- 正确性证明
- 算法分析
  - 输入规模：序列元素个数 $n$
  - 基本操作：比较次数 $A[j] < A[\min]$
  - 影响基本操作执行次数的因素： $n$
  - 建立基本操作求和表达式
  - 利用求和公式分析算法时间复杂性

$$\begin{aligned}T(n) &= \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [n - (i + 1)] = \sum_{i=0}^{n-2} [n - i - 1] \\&= (n - 1) + (n - 2) + (n - 3) + \cdots + 3 + 2 + 1 \\&= \frac{n(n-1)}{2} \in \Theta(n^2)\end{aligned}$$

# 排序问题

实践

- 程序设计
  - 算法程序编码实现 ([MIT6005 lec03 Specification](#))
  - 程序测试 ([MIT6005 lec02 Test first programming](#))

编码

测试

# 排序问题

## ■ 冒泡排序

思想：扫描整个序列，比较相邻元素，如果它们是逆序的话就交换位置，重复n-1次扫描，排序完成。  
第i遍需要扫描的元素为0 ~ n-1-i。

$$A_0, \dots, A_j \leftrightarrow A_{j+1}, \dots, A_{n-i-1} \mid A_{n-i} \leq \dots \leq A_{n-1}$$

# 排序问题

## ■ 冒泡排序

**算法 bubbleSort(A[n])**

//用冒泡法对给定数组排序

//输入：一个可排序数组A[0..*n*-1]

//输出：升序排序的数组A[0..*n*-1]

for *i* ← 0 to *n*-2 do

    for *j*=0 to *n*-2-*i* do

        if A[*j*] > A[*j*+1] *swap* A[*j*] and A[*j*+1]

$$\begin{aligned} T(n) &= \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 = \sum_{i=0}^{n-2} [(n-2-i) - 0 + 1] = \sum_{i=0}^{n-2} [n-i-1] \\ &= (n-1) + (n-2) + (n-3) + \cdots + 3 + 2 + 1 \\ &= \frac{n(n-1)}{2} \in \Theta(n^2) \end{aligned}$$

# 排序问题

- 程序设计

编码

测试

# 查找问题

问题：查找给定序列中固定值——查找键。

思想：查找键与表中元素从头至尾逐个比较。

结果：找到 或 失败

问题：查找给定序列中最大值。

练习

# 字符串匹配问题

问题：给定一个 $n$ 个字符组成的串，称为**文本**，一个 $m$  ( $m \leq n$ ) 个字符组成的串称为**模式**，从文本中寻找匹配模式的子串。

$$\begin{array}{ccccccc} t_0 & \dots & t_i & \dots & t_{i+j} & \dots & t_{i+m-1} & \dots & t_{n-1} \\ & & \updownarrow & & \updownarrow & & \updownarrow & & \\ & & p_0 & \dots & p_i & \dots & p_{m-1} & & \end{array}$$

# 字符串匹配问题

思想：将模式对准文本的前 $m$ 个字符，然后从左到右匹配每一对相应的字符，若遇到一对不匹配字符，模式向右移一位，重新开始匹配；若 $m$ 对字符全部匹配，算法可以停止。注意，在文本中，最后一轮子串匹配的起始位置是 $n-m$ （假设文本的下标从0到 $n-1$ ）

N	O	B	O	D	Y	_	N	O	T	I	C	E	D	_	H	I	M
N	O	T	T	T	T	T	T	T									
N	O	N	N	N	N	N	N	N									



# 字符串匹配问题

- 流程图或伪代码

**算法 bruteForceStringMatch( $T[0..n-1], P[0..m-1]$ )**

**//蛮力字符串匹配算法实现**

**//输入1：一个n个字符的数组 $T[0..n-1]$ 代表一段文本**

**//输入2：一个m个字符的数组 $P[0..m-1]$ 代表一个模式**

**//输出：若查找成功，返回文本第一个匹配子串中的第一个字符的位置，否则返回-1**

**for  $i \leftarrow 0$  to  $n-m$  do**

**$j \leftarrow 0$**

**while  $j < m$  and  $P[j] = T[i+j]$**

**$j \leftarrow j+1$**

**if  $j = m$  return  $i$**

**return -1**

# 字符串匹配问题

---

编码练习

参考MyStringMatch.java

# 最近对问题

问题：找出一个包含 $n$ 个点的集合中距离最近的两个点。

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

# 最近对问题

- 问题理解
- 精确算法
- 算法设计

$$\min_{\substack{0 \leq i \leq n \\ i < j \leq n}} d(P_i, P_j)$$

思想：分别计算每一点对之间的距离，然后从中找出距离最小的那一对。为了避免同一点对计算两次，可以只考虑 $i < j$ 的点 $(P_i, P_j)$

- 流程图或伪代码

# 最近对问题

## 算法 bruteForceClosesPoints(P)

//蛮力法求解平面中距离最近的两点

//输入：一个 $n(n \geq 2)$ 个点的列表 $P$ ,  $P_1=(x_1, y_1), \dots, P_n=(x_n, y_n)$

//输出：两个最近点的下标

$d_{\min} \leftarrow \infty$

for  $i \leftarrow 0$  to  $n-2$  do

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

$d \leftarrow (x_i - x_j)^2 + (y_i - y_j)^2$

    if  $d < d_{\min}$

$d_{\min} \leftarrow d$ ;  $\text{index1} \leftarrow i$ ;  $\text{index2} \leftarrow j$ ;

return  $\text{index1}, \text{index2}$

# 最近对问题

---

- 小规模实例分析
  - point1(0,0), point1(1,1), point1(1,2), point1(3,3)
- 正确性证明
- 复杂性分析
  - $O(n^2)$
- 程序设计

# 最近对问题

---

编码练习

参考ClosesPoints.java

# 搜索问题

---



# 组合问题

---

# 方程问题

---

# Discuss

---

