第 11 章 查找

11.1 静态查找

林劼
电子科技大学

查找表分类

- 静态查找表

仅作查询和检索操作的查找表。

- 动态查找表

“查询”结果“不在查找表中”→数据元素插入到查找表中；
“查询”结果为“在查找表中”的数据元素→删除。

查找过程中，往往是依据数据元素的某个数据项进行查找，这个数据项通常是数据的关键字。

关键字：是数据元素中某个数据项的值，用以标识一个数据元素。

若关键字能标识唯一的一个数据元素，则称谓**主关键字。**

若关键字能标识若干个数据元素，　则称谓**次关键字。**

张三 2016010002 男 成都 1.75

平均查找长度 ASL

$$ASL = P_1 C_1 + P_2 C_2 + \ldots + P_n C_n$$

P_i——查找第 i 个元素的概率

C_i——查找第 i 个元素需要的比较次数

提纲

11.1.1 顺序查找

11.1.2 折半查找

11.1.3 索引查找

11.1.4 作业

11.1.1 顺序查找

	52	7	13	9	41

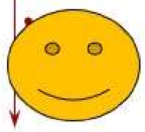
11.1.1 顺序查找

顺序查找基本思想

从表中指定位置（一般为最后一个或第0个位置设为岗哨）的记录开始，沿某个方向将记录的关键字与给定值相比较，若某个记录的关键字和给定值相等，则查找成功；

反之，若找完整个顺序表，都没有与给定关键字值相等的记录，则此顺序表中没有满足查找条件的记录，查找失败。

11.1.1 顺序查找



R

64	21	37	88	19	92	05	64	56	80	75	13	
0	1	2	3	4	5	6	7	8	9	10	11	

key=64

R.Length

R

i	i	i	i	i	i	i	i	i	i	i	i	
60	21	37	88	19	92	05	64	56	80	75	13	
0	1	2	3	4	5	6	7	8	9	10	11	

key=60

R.Length

高等教育出版社

11.1.1 顺序查找

性能分析

空间复杂度： O(1)

时间复杂度：

查找算法的基本运算是给定值与顺序表中记录关键字值的比较。

最好情况： O(1)

最坏情况： O(n)

平均情况： O(n)

11.1.1 顺序查找

顺序表上顺序查找的平均查找长度

平均查找长度（ASL）：给定值与关键字比较次数的期望值。对于具有n个记录的顺序表，查找成功时的平均查找长度为：

$$ASL=\sum_{i=1}^{n}P_iC_i$$

P_i——查找第i个记录的概率

C_i——找到第i个记录数据需要比较的次数，

对于顺序表，$C_i = n-i+1$

11.1.1 顺序查找

等概率情况

$$P_i = \frac{1}{n}$$

$$ASL = \frac{1}{n}\sum_{i=1}^{n}(n-i+1) = \frac{n+1}{2}$$

不等概率

--每个元素的查找概率已知

--每个元素的查找概率未知

顺序查找的应用：查找最大值

问题描述：查找序列（顺序表）$a[1 \dots n]$ $(n > 0)$中的最大元素。

```
1. max_val ← a[1]      //最大元素的初始值
2. for i ← 2 to n do   //依次比较每个元素
3. |   if max_val < a[i] then
4. |   |   max_val ← a[i]
5. |   end
6. end
```

- 比较次数：n-1

顺序查找的应用：查找最大和最小值

问题描述：查找序列（顺序表）$a[1 \ldots n]\ (n > 0)$中的最大元素和最小元素，比较次数不超过$\frac{3}{2}n$。

顺序查找的应用：查找最大和最小值

问题描述：查找顺序表$a[1 \dots n]$ $(n > 0)$中的最大值和最小值，比较次数不超过$\frac{3}{2}n$。

- **算法1**：朴素查找法

```
max_val ← a[1]    //最大元素的初始值
min_val ← a[1]    //最小元素的初始值
for i ← 2 to n do    //对每个元素依次判断
|  if max_val < a[i] then    //比较1
|  |  max_val ← a[i]
|  else if min_val > a[i] then //比较2
|  |  min_v ← a[i]
|  end
end
```

每次循环只与单个元素比较

比较次数：2(n-1)（最坏情况）

高等教育出版社

顺序查找的应用：查找最大和最小值

问题描述：查找顺序表$a[1 \dots n]$ $(n > 0)$中的最大值和最小值，比较次数不超过$\frac{3}{2}n$。

- 每次只与序列中的一个元素比较，总的比较次数达到 2(n-1)
- 可以考虑每次同时比较多个元素，从中找最大值和最小值
- **思考**：为更新当前的最大值和最小值，与序列多少个元素一起比较效率高？如何比较？

高等教育出版社

顺序查找的应用：查找最大和最小值

问题描述：查找顺序表 $a[1 \ldots n]\ (n>0)$中的最大值和最小值，比较次数不超过 $\frac{3}{2}n$ 。

- **算法2**：快速查找法

比较次数：$\frac{3}{2}n$

```
max_val ← a[1]
min_val ← a[1]
k ← (n % 2) + 1   //n是奇数，k 从2开始；否则从1开始
while k < n do
|  if a[k] < a[k+1] then   //比较1：两个元素先比较
|  |  if min_val > a[k] then      //比较2：
|  |  |  min_val ← a[k]           //较小值与min_val比较
|  |  end
|  |  if max_val < a[k+1] then   //比较3：
|  |  |  max_val ← a[k+1]         //较大值与max_val比较
|  |  end
|  else     //a[k] > a[k+1]        //比较1'
|  |  if min_val > a[k+1] then   //比较2'
|  |  |  min_val ← a[k+1]
|  |  end
|  |  if max_val < a[k] then      //比较3'
|  |  |  max_val ← a[k]
|  |  end
|  end
|  k ← k + 2   //每次同时比较两个元素
end
```

- 思考题：如果每次同时比较三个元素，能否进一步减少比较次数?

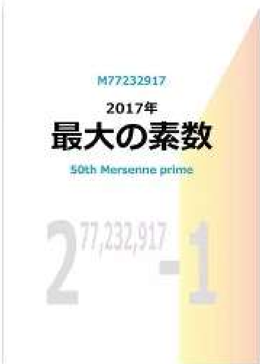
顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>1)$内的所有质数。

质数筛选的意义：

- 质数是数论研究的基础---费马大定理、黎曼猜想等
- 质数在密码学中具有重要的应用---RSA加密算法基于质数的乘法和因数分解
- 算法设计中广泛使用了质数的概念和性质---哈希表、哈希函数等

因特网梅森素数大搜索（GIMPS, Great Internet Mersenne Prime Search）



日本虹色社出版

- ✓ 世界上第一个基于互联网的分布式计算项目，参与者可自行下载prime95和MPrime软件（开放源代码）来搜索梅森素数；成功者可获5-25万美金的奖励！
- ✓ 梅森素数是可以被写成 2^n-1形式的质数，以17 世纪的法国数学家马丁·梅森命名
- ✓ 中国数学家周海中于1992年首次给出了梅森素数分布的准确表达式，被国际上命名为“周氏猜测”

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 1)$内的所有质数。

- **算法1**：试除法　时间复杂度：$O(n\sqrt{n})$
  ---小学生都会，但也只有小学生才用吧？！ (^_^)
- **算法2**：埃氏筛选法　时间复杂度：$O(n \operatorname{loglog}(n))$
  ---不是最优，但最为经典，算法简洁，使用广泛！
- ***算法3**：合数限定法　时间复杂度：$O(n)$
  ---时间最优，但空间复杂度高，纯自研（嗨）算法，仅供参考（拍砖）(@~_~@)
- ***算法4**：欧拉筛选法　时间复杂度：$O(n)$
  ---时间最优，大师创作，但算法较复杂，n越大效率越高！

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法2：埃氏筛选法

- 由希腊数学家Eratosthenes在公元250年提出的一种简单检定质数的算法

- **思路**：把整数2到n排列起来，首先标记2是最小的质数，然后删除2后面所有2的倍数（偶数）。可以发现，2后面第一个没删除的数是3，标记3是下一个质数，再把3后面3的倍数都删除；质数3后面第一个未删除的数是5，说明5是质数，再把5后面所有能被5整除的数都删除。。。这样一直做下去，就会把不超过n的全部合数都筛掉，留下的就是不超过n的全部质数。

- **关键数据结构**：顺序表 is_prime[1...n]

  is_prime[k] = true/false 标注 正整数 k 是否质数

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 1)$内的所有质数。

算法2：埃氏筛选法

```
1.  is_prime[1] ← false    //1不是质数，也不是合数
2.  for k ← 2 to n do
3.  |   is_prime[k] ← true  //初始化，先假设[2...n]都是质数
4.  end
5.  for k ← 2 to n do
6.  |   if is_prime[k] = true then  //k是质数
7.  |   |   m ←  2 * k          //k的最小倍数
8.  |   |   while m ≤ n do
9.  |   |   |   is_prime[m] = false    //k的倍数都不是质数，删除
10. |   |   |   m ←  m + k      //下一个倍数
11. |   |   end
12. |   end
13. end
```

- 时间复杂度：

$$O\left(\frac{n}{2}+\frac{n}{3}+\frac{n}{5}+\cdots\right) = O(n\ \mathrm{loglog}(n))$$

单选题 1分

将整数$a \in [1 \dots n]$因数分解，得到$a = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$，其中$p_1, \dots, p_k$是不同的质数（k >1）且指数$n_1, \dots, n_k$都大于0。则在埃氏筛选法中，a会被删除几次？即 $is\_prime[a] \leftarrow$ false 被执行几次?

A. k

B. $n_1 + n_2 + \cdots + n_k$

C. $\max\{n_1, n_2, \dots, n_k\}$

D. $p_1 + p_2 + \cdots + p_k$

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法3：合数限定法

- 筛选法的问题在于一个合数会被多次删除，造成时间浪费。如 6,12,18,36是质数2和3的倍数

  问：整数k会被删除几次?

  答：有多少个不同的质因数就被删除几次

  如12 = 2 x 2 x 3，会被删除2次

- **思路**：为使每个合数只删除一次，不能简单地删除质数的所有倍数，而是删除由当前找到的质数合成的数。

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法3：合数限定法

- **思路**：为使每个合数只删除一次，不能简单地删除质数的所有倍数，而是删除由当前找到的质数合成的数。
- **关键数据结构：**
  (1) 顺序表 is_prime[1…n]
  (2) 队列 M
  ---设当前找到k-1个质数: $p_1 < p_2 < \cdots < p_{k-1}$
  --- 用队列M存放由这些质数合成的数值
  即 M = { $p_1^{n_1} p_2^{n_2} \ldots p_{k-1}^{n_{k-1}} \leq n \mid \forall i \in [1, k): n_i \geq 0)$ }

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法3：合数限定法

- 关键步骤：更新合数队列M

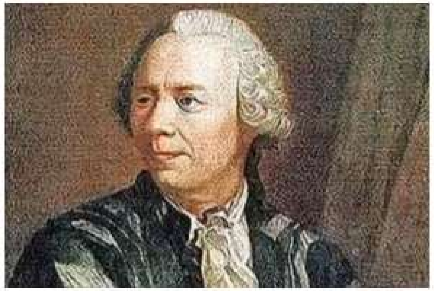
```
            //设找到第k个素数p_k   (is_prime[p_k] = true)
Q ← M   //新建队列Q，然后将M中的所有合数移到Q
while IsEmpty(Q) = false do
|  m ← DeQueue(Q)        //取出合数，与p_k相乘
|  while m * p_k ≤ n do           //相乘结果在区间范围内
|  |  is_prime[m * p_k] ← false  //删除m * p_k ，且该合数是第一次删除 (?)
|  |  EnQueue(M, m)      //将合数m放入队列M
|  |  m ← m * p_k           //更新合数，继续与p_k相乘
|  end
end
```

算法细节省略，因为更牛的在后面。。。

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：欧拉筛选法 （线性筛选法）



Leonhard Euler

- **思路**：在埃氏筛选法的基础上，让每个合数只被它的最小质因数删除一次，以达到不重复的目的
- **关键数据结构**：（1）顺序表 is_prime[1...n]

  is_prime[k] = true/false 标注 正整数 k 是否质数

  （2）线性表 prime_list：按升序存储筛选出的质数

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：欧拉筛选法

- 欧拉对质数p的倍数 k * p（k>1) 的分析

Q1：在埃氏筛选法中，合数k * p 什么情况下会被多次删除？

A1：将k质因数分解为 $k = q_1 q_2 \dots q_j$ 且 $q_1 \le q_2 \le \cdots \le q_j$

如果 $\exists\, i \in [1..j]: q_i \neq p$，则k*p至少会被 p 和 q_i 删除两次

Q2：合数k * p 第一次被删除是在找到哪个质数后发生的？

A2： $k * p = \min\{q_1, p\} * q_2 \dots q_j * \max\{q_1, p\}$

因此，k*p第一次删除发生在找到质数$\min\{q_1, p\}$ 时！

如果$q_1 < p$，p的倍数k*p在筛选出p之前就已经被删除了！

欧拉定理

对于每个正整数k, 它只需与小于等于其最小质因数的质数相乘，并删除相乘后的合数

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 1)$内的所有质数。

算法4：欧拉筛选法

```
InitList(prime_list)    //初始化存放质数的线性表
is_prime[1] ← false
for k ← 2 to n do
|  is_prime[k] ← true  //初始化，先假设[2...n]都是质数
end
for k ← 2 to n do   //从2开始筛选
|  if is_prime[k] = true then        //k是质数
|  |  Append(prime_list, k)      //将k从后添加至线性表（升序排列）
|  end
     //判断k是否质数后，再作为系数与已找到的质数相乘，并删除合数
|  j ← 0            //从线性表中的最小质数2开始
|  while k * prime_list[j] ≤ n do     //合数在[1...n]区间内
|  |  is_prime[ k * prime_list[j] ] ← false   //删除合数
|  |  if k % prime_list[j] == 0 then   //prime_list[j] 是k的最小质因数
|  |  |  break                          //结束删除
|  |  else
|  |  |  j ← j + 1              //否则，k继续与下一个质数相乘
|  |  end
|  end
end
```

- 如果k是质数，已添加至prime_list末尾
- 如果k是合数，则其最小质因数一定在prime_list中 (?)

因此，用k删除合数的运行时间为 O(|prime_list|)

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○		✖																	

k = 2

最小质因数：2

在埃氏筛选法中，2会删除所有偶数，并且删除的合数数量最多！，

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	×		×			×												

k = 3

最小质因数：3

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖		✖		✖	✖												

k = 4

最小质因数：2

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖		✖	✖	✖					✖						

k = 5

最小质因数：5

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	×	○	×		×	×	×		×			×						

k = 6

最小质因数：2

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖		✖		✖	✖						✖

k = 7

最小质因数：7

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖		✖		✖	✖	✖					✖

k = 8

最小质因数：2

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖		✖		✖	✖	✖		✖			✖

k = 9

最小质因数：3

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖		✖		✖	✖	✖		✖		✖	✖

k = 10

最小质因数：2

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖	○	✖		✖	✖	✖		✖		✖	✖

k = 11

最小质因数：11

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

对于每个正整数k，它只需与小于等于其最小素因子的质数相乘，并删除相乘后的合数

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	○	○	✖	○	✖	○	✖	✖	✖	○	✖	○	✖	✖	✖	○	✖	○	✖	✖

k = 12,13,...,21 （倍数都大于21！）

高等教育出版社

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

***算法证明**：

- 正确性：[1,n]中所有合数都被删除

（证明） 设合数$a \in [1, n]$，且$a = p_1 p_2 \dots p_j$ (质因数$p_1 \le p_2 \le \cdots \le p_j$)

⬇

因为$j > 1$ (?)，设$k = p_2 \dots p_j$，得到$a = k * p_1$

⬇

由于整数k的最小质因数 $p_2 \ge p_1$，根据算法，一定和p_1相乘得a

⬇

合数a被删除

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1,n]$ $(n>0)$内的所有质数。

算法4：线性筛选法 （欧拉筛选法）

***算法证明**：

- 线性时间效率：[1,n]中所有合数只被删除1次！

（反证法）假设合数$a\in[1,n]$被删除两次，即存在整数$k_1<k_2$，使得
$a=k_1*p_1=k_2*p_2$ (p_1,p_2都是质数，且 $p_1>p_2$)

矛盾

根据算法，k_1*p_1说明k_1的最小质因数大于等于p_1(?)，证明 p_1 是合数 a 的最小质因数

同理，p_2 也是合数 a 的最小质因数

$p_1=p_2$

顺序查找的应用：查找区间内所有质数

问题描述：查找正整数区间$[1, n]$ $(n > 0)$内的所有质数。

算法比较：

	埃氏筛选法	合数限定法	欧拉筛选法	质数总数
n = 10^2	0.006ms	0.01ms	0.013ms	25
n = 10^4	0.14ms	0.27ms	0.28ms	1229
n = 10^6	10.6ms	15.8ms	13.6ms	78498
n = 10^8	1.8s	1.7s	1.4s	5761455
n = 10^9	23s	20s	14s	50847534

硬件配置：2.3 GHz 双核 Intel Core i5，8GB内存；编译工具：Xcode；编程语言：C++

高等教育出版社