

向量

位图：典型应用

$\theta_2 - G_2$

Those too big to pass through are our friends.

邓俊辉

deng@tsinghua.edu.cn

小集合 + 大数据：问题

❖ 老问题：int A[n]的元素均取自[0, m)

如何剔除其中的重复者？

❖ 仿照Vector::deduplicate()改进版

先排序，再扫描

—— $O(n \log n + n)$ —— 毫无压力

❖ 新特点：数据量虽大，但重复度极高

- 想想我们电脑里的mp3、mp4
- 还有，朋友圈 ...

❖ 比如：

$$2^{24} = m \ll n = 10^{10}$$

亦即，10,000,000,000个24位无符号整数

❖ 如果采用内部排序算法

至少需要 $4 * n = 40GB$ 内存

——否则，频繁的I/O将导致整体效率的低下

❖ 那么

$m \ll n$ 的条件，又应如何加以利用？

小集合 + 大数据：算法

❖ `Bitmap B(m); //O(m)`

`for (int i = 0; i < n; i++)`

`B.set(A[i]); //O(n)`

`for (int k = 0; k < m; k++)`

`if (B.test(k))`

`/* ... */; //O(m)`

❖ 拓展：搜索引擎的使用规律亦是如此

词表**规模**不大，但**重复度**极高

——如何从中剔除重复的索引词？

❖ 总体运行时间 = $O(n+m) = O(n)$

❖ 空间 = $O(m)$

- 就上例而言，降至：

$$m/8 = 2^{21} = 2\text{MB} \ll 40\text{GB}$$

- 即便 $m = 2^{32}$ ，也不过：

$$2^{29} = 0.5\text{GB}$$

❖ 关键在于

如何将查询词表转换为某一集合

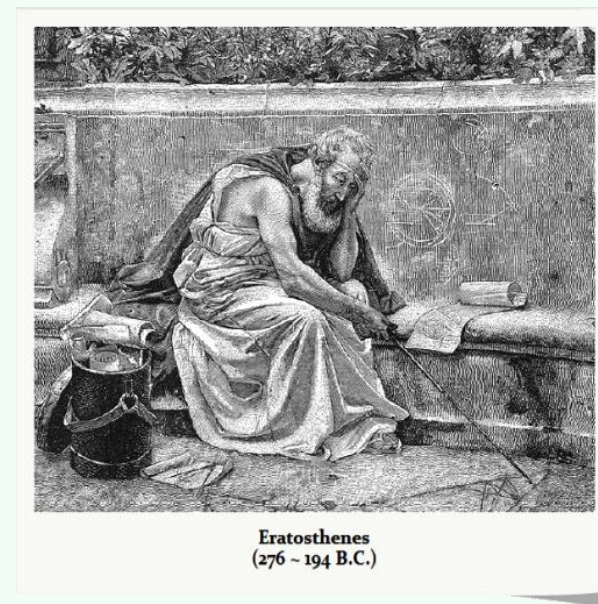
——留作习题

筛法：思路

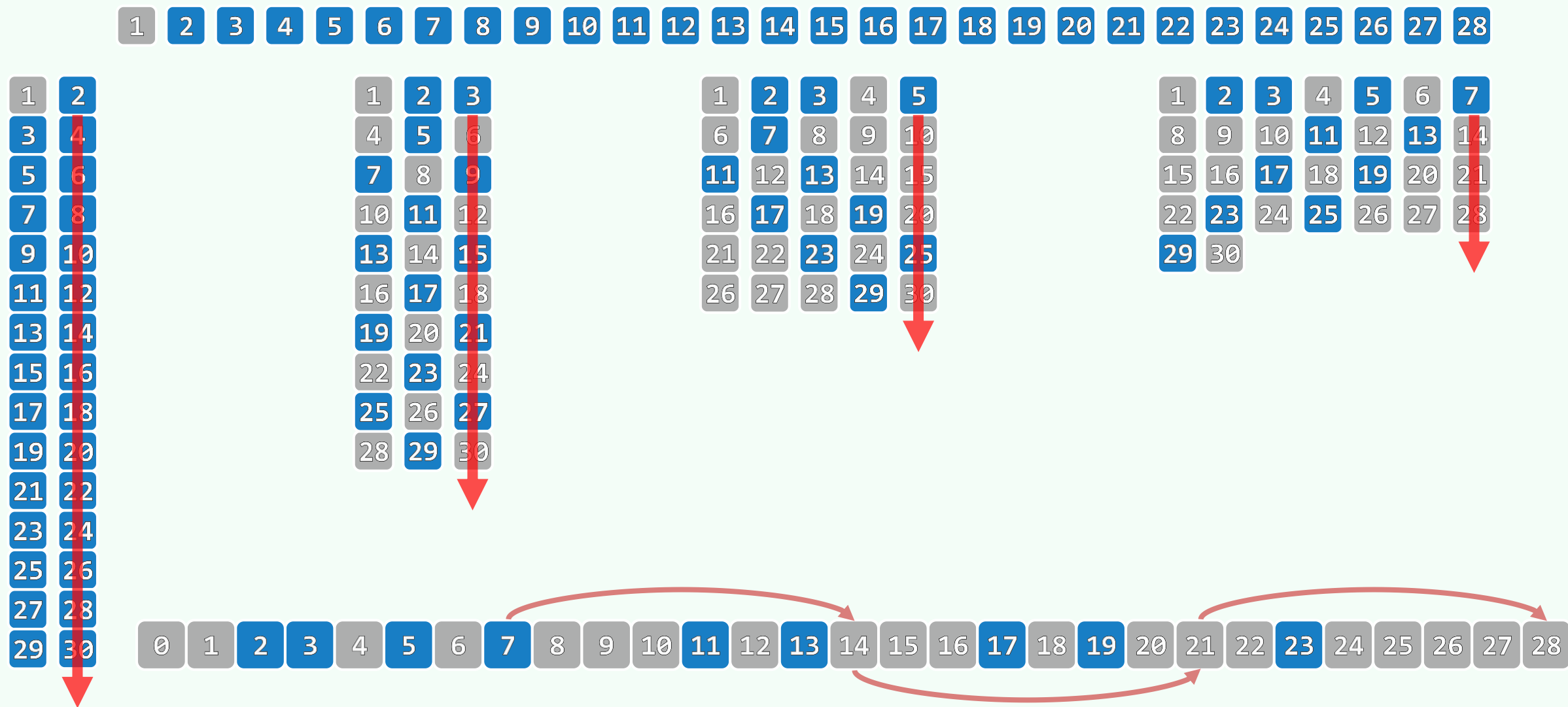


筛法：实现

```
❖ void Eratosthenes( int n, char * file ) {  
    Bitmap B( n );  
    B.set( 0 ); B.set( 1 );  
    for ( int i = 2; i < n; i++ )  
        if ( ! B.test( i ) )  
            for ( int j = 2*i; j < n; j += i )  
                B.set( j );  
    B.dump( file );  
}
```



筛法：过程 + 效果



效率 + 改进

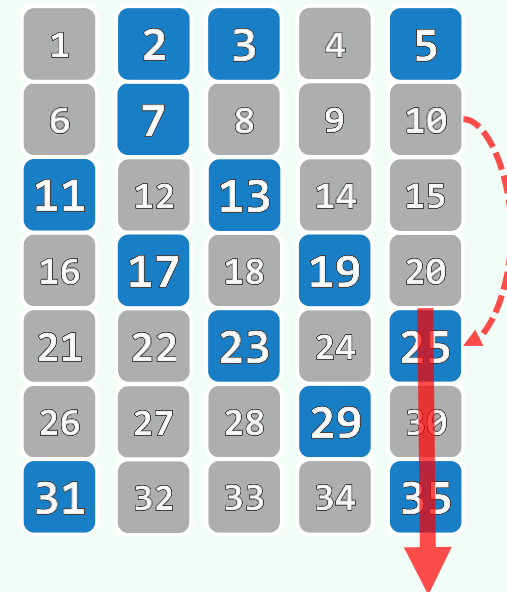
❖ 不计内循环，外循环自身每次仅一次加法、两次判断，累计 $\mathcal{O}(n)$

❖ 内循环每趟迭代 $\mathcal{O}(n/i)$ 步，由素数定理外循环至多 $n/\ln n$ 趟，累计耗时

$$n/2 + n/3 + n/5 + n/7 + n/11 + \dots$$

$$< n/2 + n/3 + n/4 + n/5 + n/6 + \dots + n/(n/\ln n)$$

$$= \mathcal{O}(n \cdot (\ln(n/\ln n) - 1)) = \mathcal{O}(n \cdot \ln n - n \cdot \ln(\ln(n))) = \mathcal{O}(n \cdot \log n)$$



❖ 内循环的起点“ $2*i$ ”可改作“ $i*i$ ”；外循环的终止条件“ $i < n$ ”可改作“ $i*i < n$ ” //为什么？

内循环每趟迭代 $\mathcal{O}(\max(1, n/i - i))$ 步，外循环至多 $\sqrt{n}/\ln \sqrt{n}$ 趟，耗时减少 //从渐近角度看呢？

❖ 基于以上，如何实现`primeNLT(int low)`？