回顾: Linux简单编程

- vim编辑器
  - ➡ 配置文件**~/.vimrc**, 参考**amix/vimrc**

- Linux程序编译指令: **g++ -o {可执行文件} {源程序}**
  - ➡ **-o**选项: 指定产生的可执行文件名称

- Makefile用法
  - ➡ 规则: 目标, 依赖, 指令
  - ➡ **make**: 编译makefile的**第一条**规则
  - ➡ **make {target}**: 编译makefile中以target为目标的规则

```
<target>:  <prerequisites>
[tab]      <commands>
```

提示: bitsToInteger

- **bitStr[0..i] = 2*bin_str[0..i-1] + bin_str[i]** (i >= 1)
  - binStr[0..i]表示**binStr**中前i+1个二进制字符对应的十进制数

```
1 int bitsToInteger(unsigned char *bitStr, int length) {
2   value = 0;
3   for i from 0 to length-1
4     if bitStr[i] == 1
5       value++;
6     endif
7     value = value << 1;
8   end for
9   value = value >> 1;
10  return value;
11 }
```

提示: integerToBits

```
unsigned char *integerToBits(int value, int *pLength) {
  binStr = "";
  while value > 0
    if value % 2 == 1
      insertFromFront(binStr, '1');
    else
      insertFromFront(binStr, '0');
    end if
    value = value >> 1;
  end while
  *pLength = strlen(binStr);
  return binStr;
}
```

数论基础实验-模指数运算

模指数运算

- 模指数运算: 已知a, e, m, **计算a^e mod m**
  - ➡ a: 底数, e: 指数, m: 模数

- 应用: 公钥密码体制

- 示例: 计算2^{90} **mod** 13
  - ➡ 指数函数: **pow**(a, e)
    - ✳ 依赖**math.h**
  - ➡ 模运算: **%**

如何有效计算模指数

当e很大时, a^e溢出: 运算结果超出固定分配空间能够存储的最大值

模指数运算: 分治原理

- 分治法: 分, 治, 归并

a^e **mod** m = a^{e1+e2} **mod** m = (a^{e1} **mod** m) * (a^{e2} **mod** m) **mod** m

➡分别计算a^{e1} **mod** m和a^{e2} **mod** m (不会发生溢出)

➡由a^{e1} **mod** m和a^{e2} **mod** m归并出a^e **mod** m

- 示例: 计算2^{90} **mod** 13

➡ 分: $2^{90} = 2^{50} * 2^{40}$

➡ 治: 2^{40} **mod** 13 = 3, 2^{50} **mod** 13 = 4

✳ 2^{40}和2^{50}在**unsigned long**类型下不会溢出

➡ 归并: 2^{90} **mod** 13 = 4 * 3 **mod** 13 = 12

问题

能否在计算2^{50} **mod** 13时重用2^{40} **mod** 13的部分结果

模指数运算: 二进制算法

- 以**二进制**方式分拆指数

  如果e可以表示为二进制数$\underline{d_{n-1}d_{n-2}...d_1d_0}$, 这里$d_i$ = 0或1

  $e = d_{n-1} * 2^{n-1} + \cdots + d_1 * 2 + d_0 = D_{n-1} + \ldots + D_1 + D_0$ **($D_i = d_i * 2^i$)**

  因此, $a^e = a^{Dn-1} * a^{Dn-2} * \ldots * a^{D1} * a^{D0}$

- 分治计算: $A_i = a^{Di}$ **mod** m, i = 0, 1,..., n-1

  - ➡ $a^{2^{i+1}} \bmod m = (a^{2^i}$ **mod** m) * (a^{2^i} **mod** m) **mod** m
  - ➡ 计算**$O(\log_2 e)$**次

- 归并: ($A_{n-1} * \ldots * A_0$) **mod** m

模指数运算示例 I

计算7^{256} **mod** 13

- **分拆指数:** $256 = 2^8$
- **分治计算**
  - 7^1 **mod** 13 = 7
  - 7^2 **mod** 13 = [(7^1 **mod** 13) * (7^1 **mod** 13)] **mod** 13 = 10
  - 7^4 **mod** 13 = 9, 7^8 **mod** 13 = 3,……
  - 7^{256} **mod** 13 = [(7^{128} **mod** 13) * (7^{128} **mod** 13)] **mod** 13 = 9
- **归并:** 7^{256} **mod** 13 = 9

模指数运算示例 II

计算5^{117} **mod** 19

- **分拆指数:** 117 = 1 + 4 + 16 + 32 + 64
- **分治计算**
  - 5^1 **mod** 19 = 5
  - 5^2 **mod** 19 = [(5^1 **mod** 19) * (5^1 **mod** 19)] **mod** 19 = 6
  - 5^4 **mod** 19 = 17; ..... 5^{16} **mod** 19 = 16; 5^{32} **mod** 19 = 9
  - 5^{64} **mod** 19 = [(5^{32} **mod** 19) * (5^{32} **mod** 19)] **mod** 19 = 5
- **归并:** 5^{117} **mod** 19 = $5^{1+4+16+32+64}$ **mod** 19 = 1

数论基础实验-素性检测

基于Eratosthenes筛选法的素性测试方法

Miller-Rabin素性测试方法

素数

- 如果n (n > 1) 是**素数**, 当且仅当n只有因子1和它本身
  - ➡ 注: **1不是素数**
- 数学应用: 自然数可由**全体素数基底**表达
  - ➡ **算术基本定理**: 任意整数a > 1都可以唯一地分解为: $a = n_1^{k1} X n_2^{k2} X...$, 这里$n_i$均为素数, ki为正整数
- 密码学应用: 构建基于**RSA**的公钥密码学内核
- **素性测试:** 判定一个数n是否为素数

素性判定

- 第一种方法: 判断n是否被i整除 **(i = 2,..., n-1)**
  - 素数n只有1和n两个因子
  - **n-2**个判断条件

问题
如何减少判断次数

- 第二种方法: 判断n是否被i整除 **(i = 2, 3,..., $n^{1/2}$)**
  - **如果j > $n^{1/2}$是n的因子, 总能找到i < $n^{1/2}$, 满足n = i * j**
  - 判断条件减少为**$n^{1/2}$ -1**次, 判断复杂度由O(n)降低为O($n^{1/2}$)

问题
如何进一步减少判断次数

一种确定性素性测试思想

- 确定性: 判断n**一定是/不是**素数

- 思想: 在**{2, 3,..., $n^{1/2}$}**集合基础上, 找出其中的**素数**, 判断n是否含有这些素因子
  - ➡ {2, 3,..., $n^{1/2}$}可能存在合数, **合数含有素因子导致重复判断**

- 步骤: **(1)**确定待筛选集合{2, 3,..., $n^{1/2}$}; **(2)**基于{2, 3,..., $n^{1/2}$}的**Eratosthenes筛选; (3)**筛选后元素与n整除判定

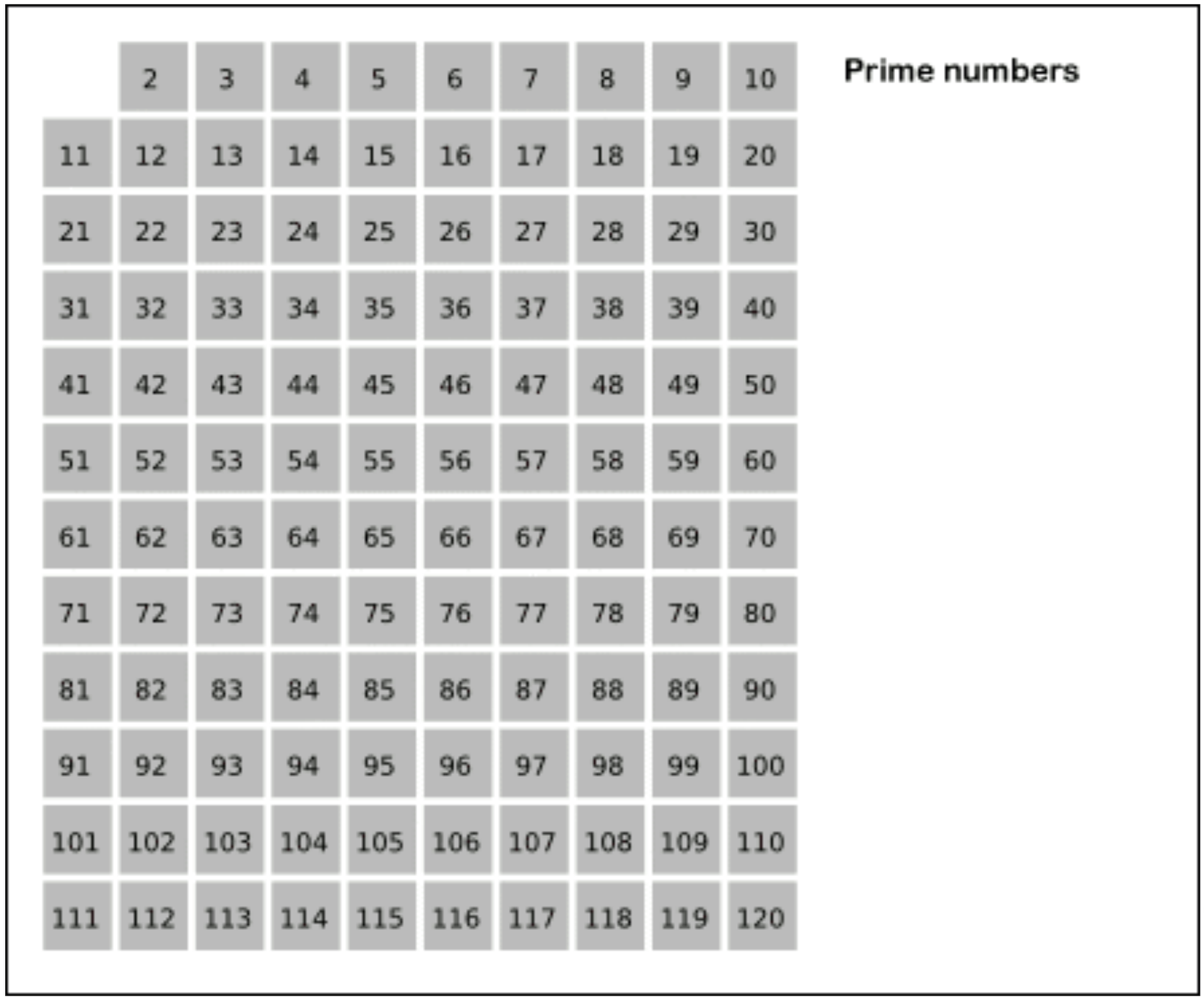
Eratosthenes筛选法

目标: **产生最小的N个素数**

➡ 不适用于计算某个范围内的全部素数

- 取**第一个素数**2, 划去{2,..., N}中除2以外所有2的倍数
- 大于2的第一个正整数(即3)被认定为素数, 在余下的整数中划去除3以外所有3的倍数
- **循环**此过程直到找到{2, 3,..., N}中的所有素数

Eratosthenes筛选法示例



Eratosthenes

更多示例

基于筛选法的素性测试

判断2543是否为素数

- **求平方根:** $2543^{1/2}$ ~= 50, 确定待筛选集合**{2, 3, 4,..., 50}**
- 使用**Eratosthenes筛选法**在集合{2, 3, 4,..., 50}中筛选出所有素数: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
- **整除判定:** 2543除以这15个素数, 结果都不是整数, 因此2543**一定**是素数

课堂实验

- **模指数运算**实验
  - a, e, m均为正整数; 否则返回-1

```
int modExp(int a, int e, int m)
// a: 输入底数
// e: 输入指数
// m: 输入模数
// 返回a^e mod m
```

- 基于**Eratosthenes筛选法**的素性测试实验

```
bool Eratosthenes(int a)
// a: 输入测试数
// 如果a是素数, 返回1; 否则返回0
```