第四届（2019）全国高校密码数学挑战赛

加法链问题（赛题三）

参赛队员：赵晓鹏　钱佳威　刘竹森　　指导教师：曹珍富　董晓蕾

计算机科学与软件工程学院 · 密码与网络安全系

August 5, 2019

華東師範大學
EAST CHINA NORMAL UNIVERSITY

加法链介绍及相关经典研究

适用于挑战赛题的加法链求解方法

赛题样例分析

总结与思考

什么是加法链?

加法链的历史

确定某个正整数的最短加法链的长度这个问题最早是在 1894 年 H.Dellac 提出来的。同年，E. de Jonquières[1]用因子方法给出了这个问题的部分答案。

定义

正整数 n 的一条长度为 r 的加法链是整数递增序列：

$$1 = a_0 < a_1 < a_2 < \ldots < a_r = n$$

并且，对于每个 $i = 1, 2, \ldots, r$，都存在正整数 $k \leq j < i$ 满足

$$a_i = a_j + a_k$$

如果对于每个 i 都有 $j = i - 1$，则我们称这条加法链是星链。

例

$1 \to 2 \to 3 \to 6 \to 7 \to 14 \to 15$ 是 15 的长度为 6 的加法链，但它的最短加法链长度是 5，满足条件的加法链（不唯一）可以是 $1 \to 2 \to 3 \to 6 \to 12 \to 15$。

華東師範大學 EAST CHINA NORMAL UNIVERSITY
计算机科学与软件工程学院 School of Computer Science and Software Engineering

[1] [*l'Intermèdiaire des Mathèmaticiens* **1** (1894), 20, 162~164]

函数定义

设 n 是一个正整数，为了更清楚地描述与分析加法链的性质，我们定义：

函数	定义
$\lambda(n)$	$\lfloor \log_2(n) \rfloor$
$\mu(n)$	n 的二进制表示中 1 的个数
$\ell(n)$	n 的最短加法链长度
$\ell^*(n)$	n 的最短星链长度
$\ell(a_1, a_2, \cdots)$	包含 $a_1, a_2, \cdots$ 的最短加法链长度

表格: 函数定义

设集合

$$\mathfrak{S} = \{(a_1, \ldots, a_k, \ell) \mid \text{存在长度为 } \ell \text{ 且包含 } a_1, \ldots, a_k \text{ 的加法链}\}$$

Downey,Leong 和 Sethi 证明了 $\mathfrak{S}$ 是 NP-难的[2]。但对 $k = 1$ 情况的子集合并没有证明是 NP-难的。

華東師範大學 EAST CHINA NORMAL UNIVERSITY
计算机科学与软件工程学院 School of Computer Science and Software Engineering

[2] [Downey P, Leong B, Sethi R., *SIAM Journal on Computing*, 1981, **10**, 638–646].

为什么要研究加法链?

对 $C, n \in \mathbb{Z}$ 和 $A, B \in \mathbb{Z}_n$，模指数运算

$$B = A^C \pmod{n}$$

在纠错码理论和密码理论中有很大应用。例如著名的 Rivest-Shamir-Adleman (RSA) 密码系统就包含这类运算，并且计算它们的代价往往是昂贵的。因此，如何使运算的乘法次数最少是解决这类问题的关键。

如果我们预先计算出 C 的最短加法链，那么就可以达到目标。例如 $C = 60$ 的最短加法链是:

$$1 \to 2 \to 3 \to 6 \to 12 \to 15 \to 30 \to 60$$

那么我们可以通过下列方法计算出 A^{60}。

$$A^1; A^2 = A^1 A^1; A^4 = A^2 A^2; A^6 = A^4 A^2; A^{12} = A^6 A^6$$
$$A^{15} = A^{12} A^3; A^{30} = A^{15} A^{15}; A^{60} = A^{30} A^{30}.$$

二进方法

设 n 是正整数，x 属于任何一个含有幺元素的代数系统，其中已定义一个可结合乘法运算记作 $\times$。二进方法是计算 x^n 常用的方便快捷算法。

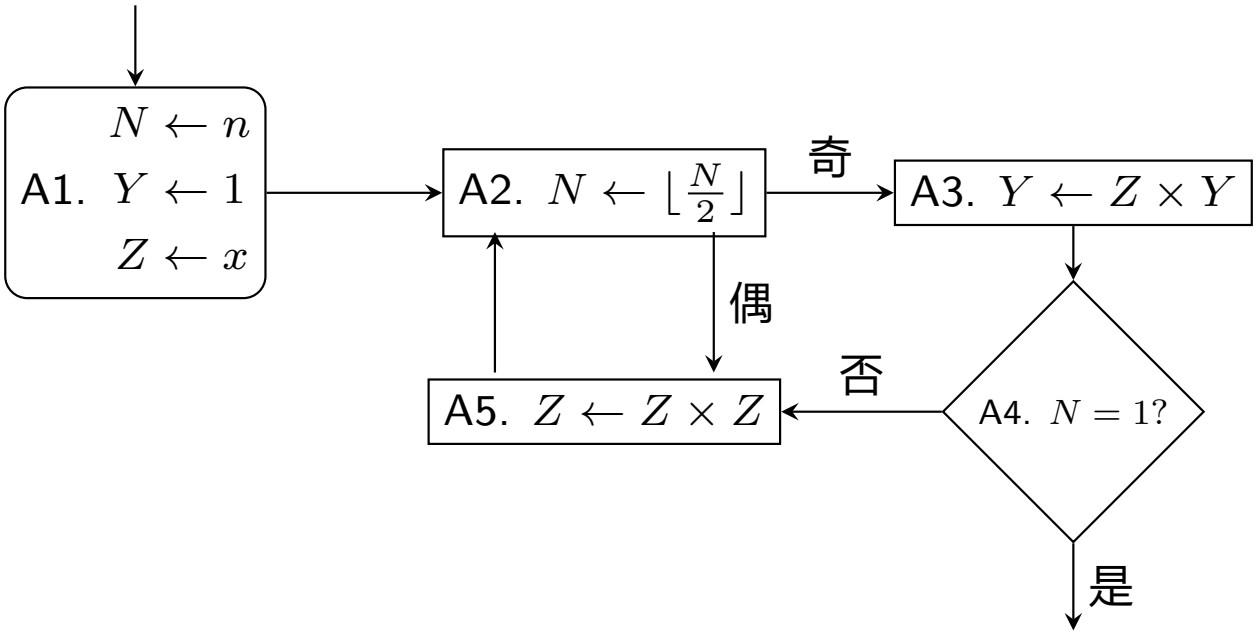


图: 二进方法（自右向左）计算 x^n 需要 $\lambda(n)+\mu(n)$ 次乘法运算

二进方法

设 n 是正整数，x 属于任何一个含有么元素的代数系统，其中已定义一个可结合乘法运算记作 $\times$。二进方法是计算 x^n 常用的方便快捷算法。

例 (二进方法（自右向左）计算 $x^{23=(10111)_2}$ 需要 8 次乘法运算)

	N	Y	Z
步骤 A1 后	23	1	x
步骤 A5 后	11	x	x^2
步骤 A5 后	5	x^3	x^4
步骤 A5 后	2	x^7	x^8
步骤 A5 后	1	x^7	x^{16}
步骤 A4 后	1	x^{23}	x^{16}

因子方法

假设 $1 \to a_1 \to \ldots \to a_{\ell(m)} = m$ 和 $1 \to b_1 \to \ldots \to b_{\ell(n)} = n$ 分别构成正整数 m 和 n 的最短加法链，则可构造 mn 的长度为 $\ell(m) + \ell(n)$ 加法链：

$$1 \to a_1 \to \ldots \to a_{\ell(m)} \to a_{\ell(m)}b_1 \to \ldots \to a_{\ell(m)}b_{\ell(n)}$$

定理

设 m 和 n 都是正整数，有 $\ell(mn) \leq \ell(m) + \ell(n)$

因子方法的过程是通过递归合并上述构造完成的。平均来说，它比二进方法更好一些。

Knuth 方法[3]

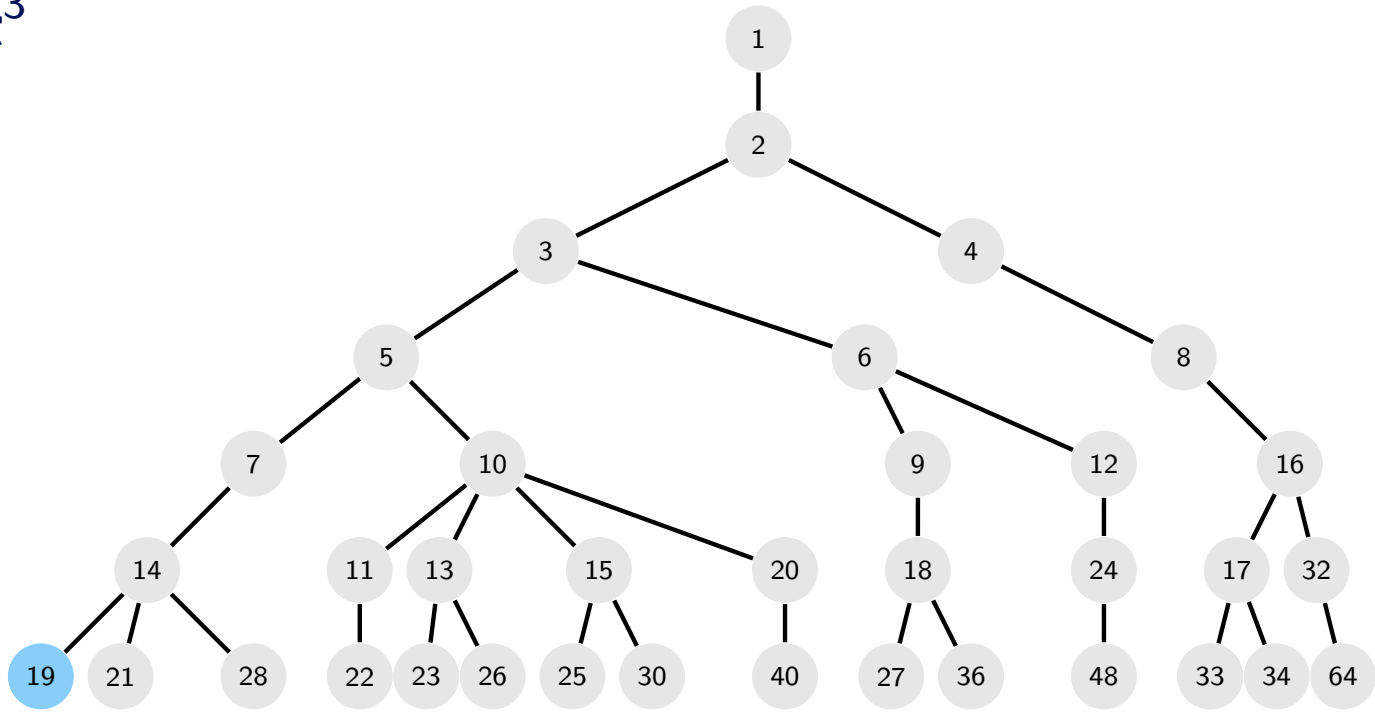


图: 7 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

$$
\begin{aligned}
19 + 1 &= 20 \\
19 + 2 &= 21 \\
19 + 3 &= 22 \\
19 + 5 &= 24 \\
19 + 7 &= 26 \\
19 + 14 &= 33 \\
19 + 19 &= 38
\end{aligned}
$$

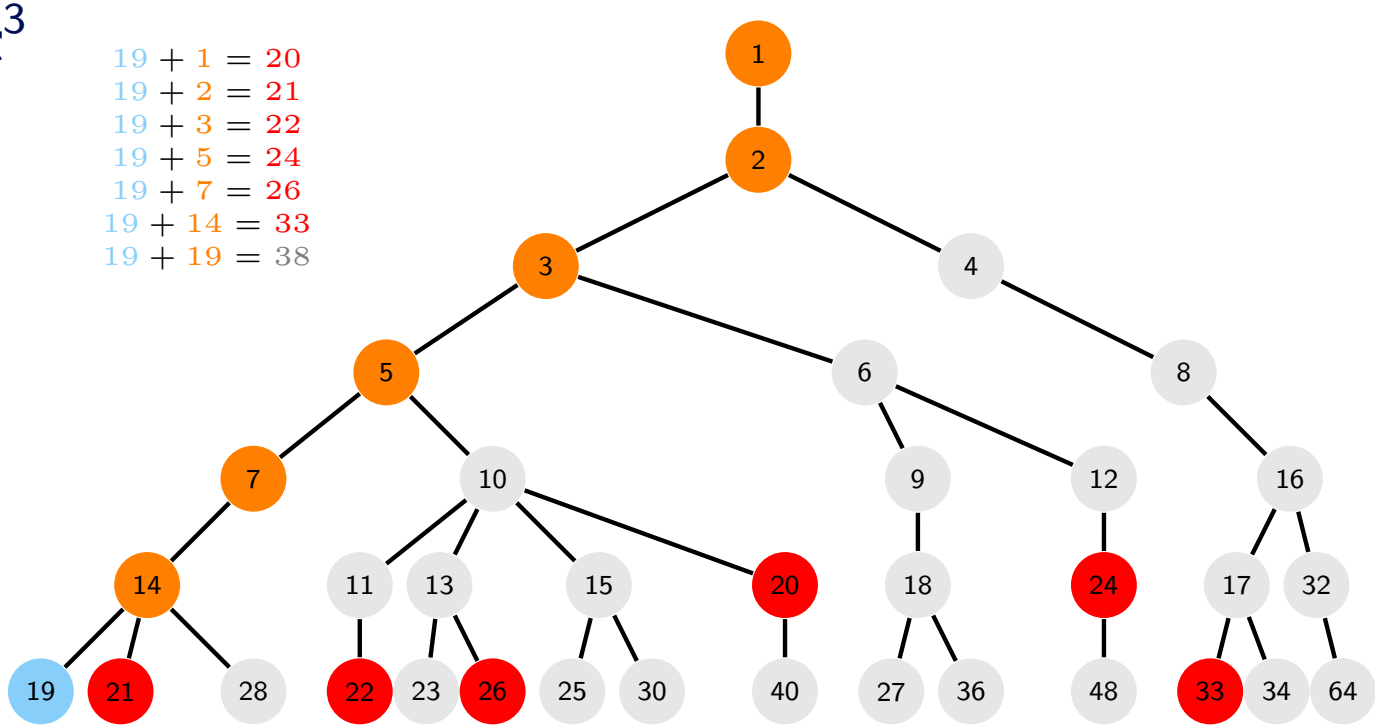


图: 7 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

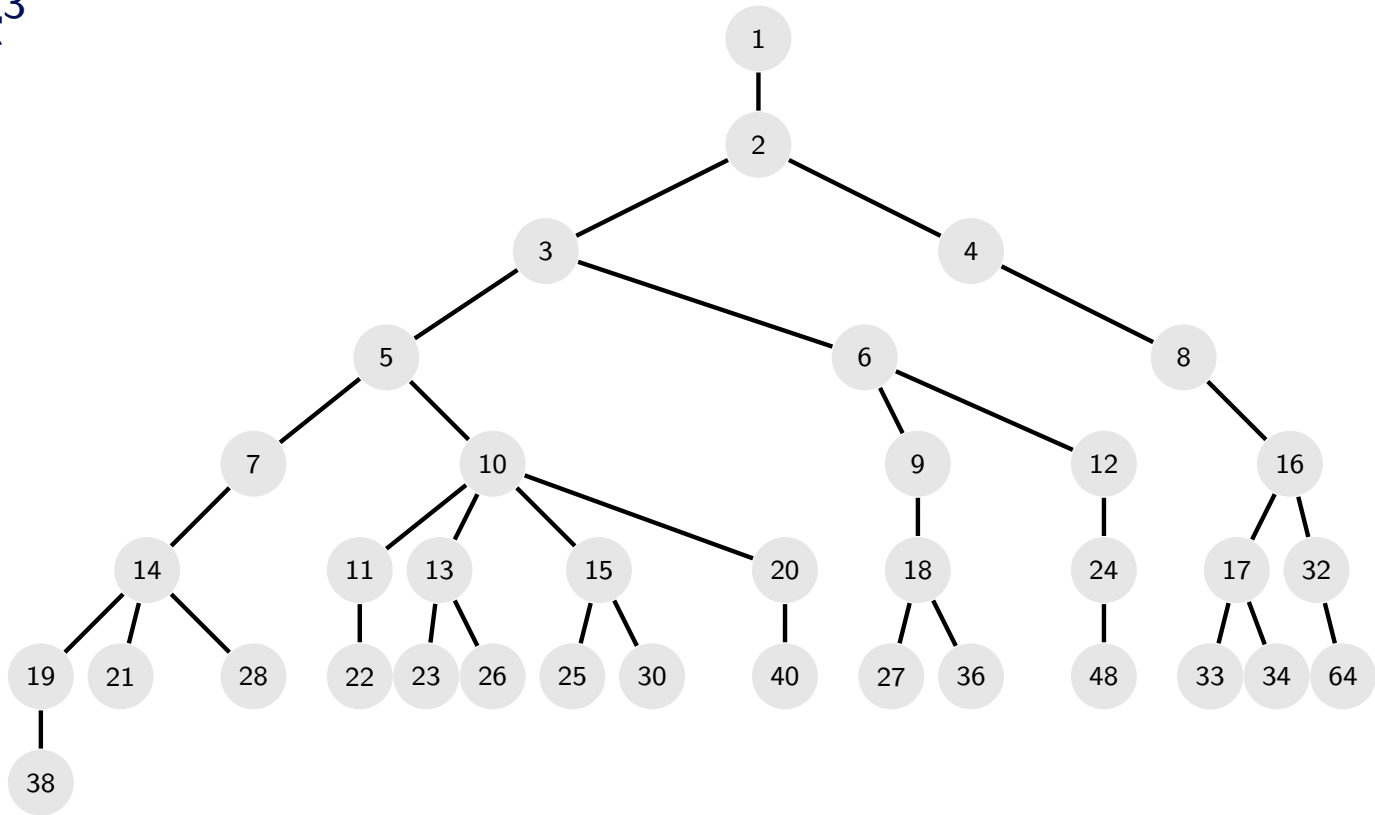


图:

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

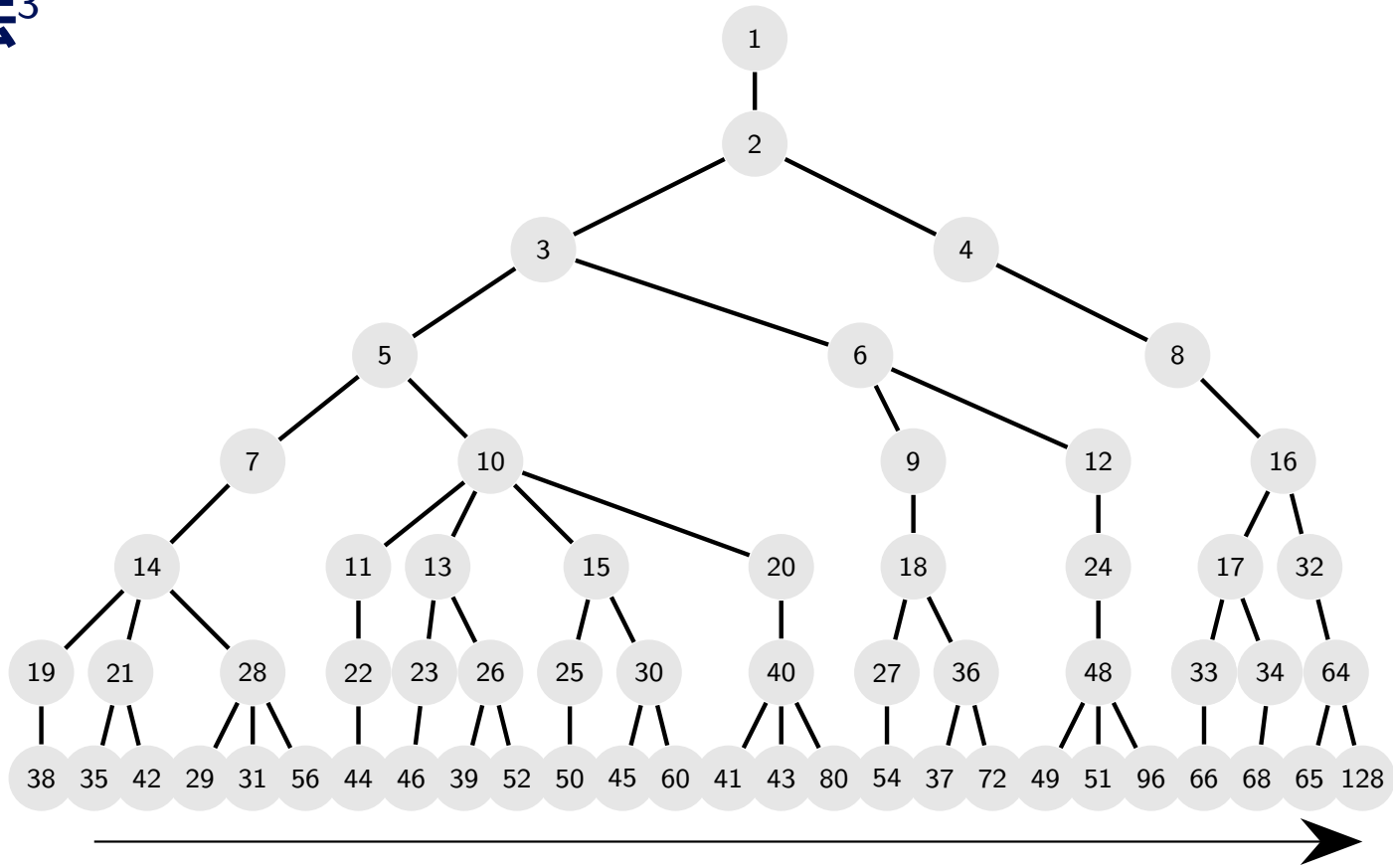


图: 8 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

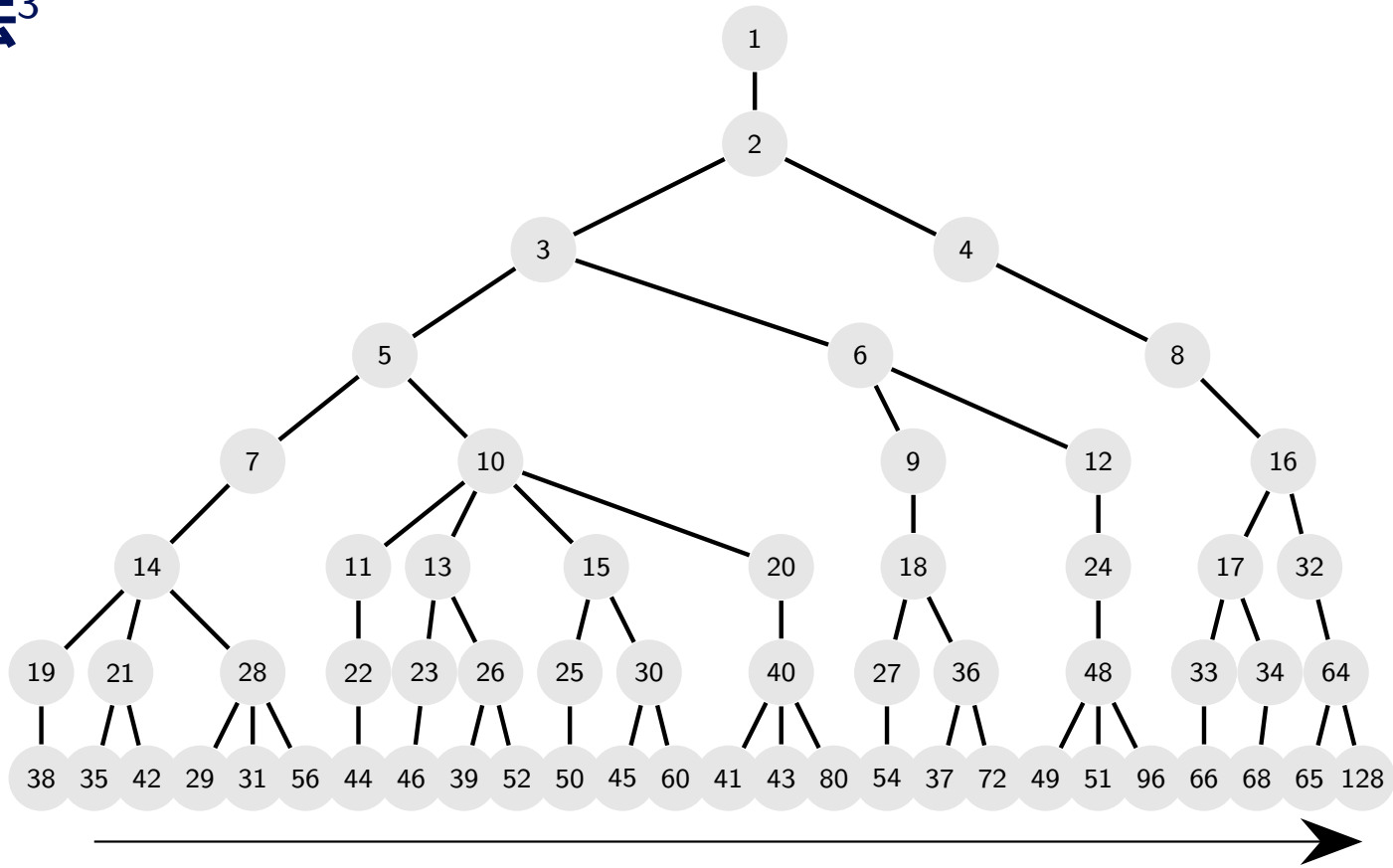


图: 8 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

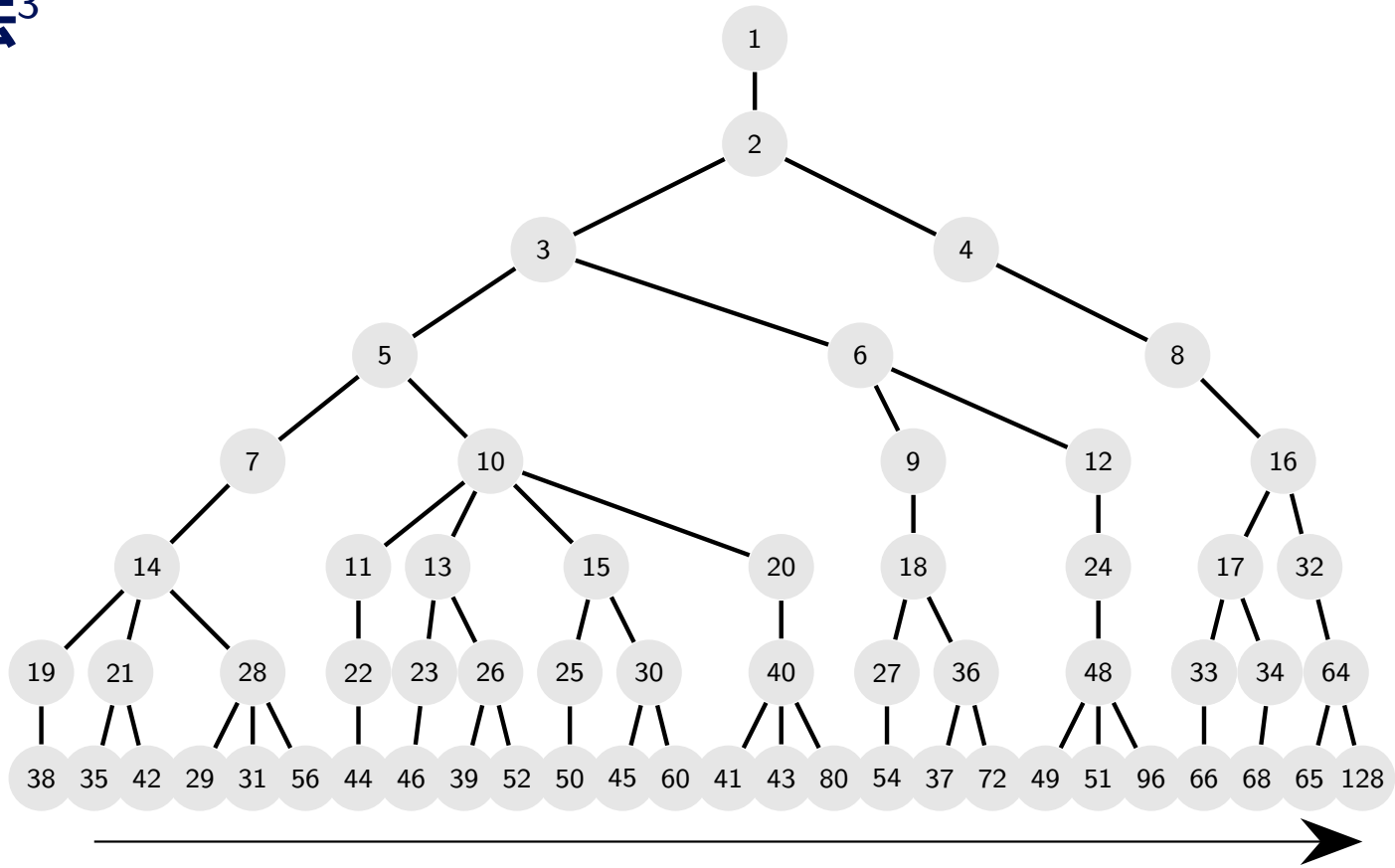


图: 8 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

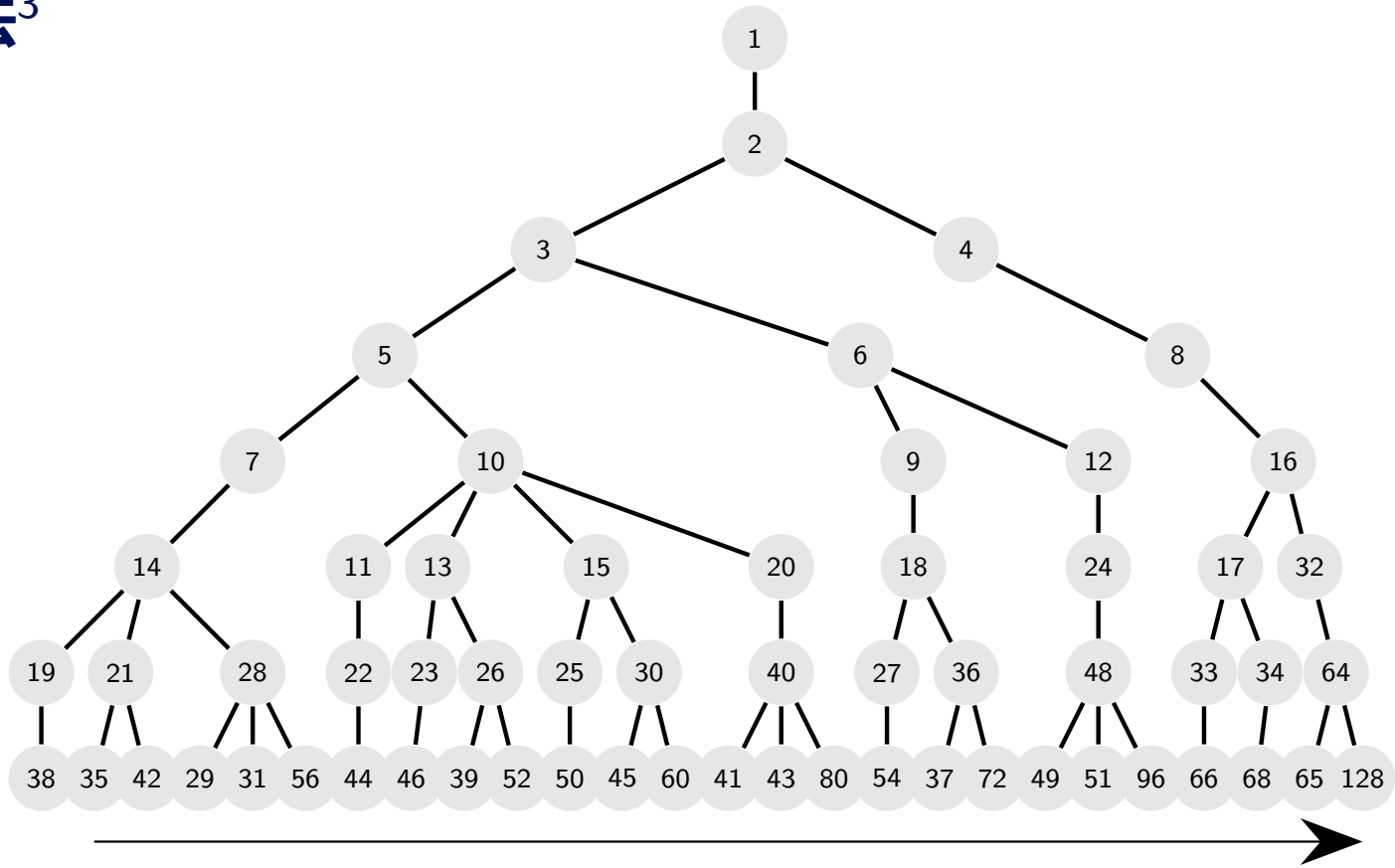


图: 8 层 Knuth 幂树

- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

Knuth 方法[3]

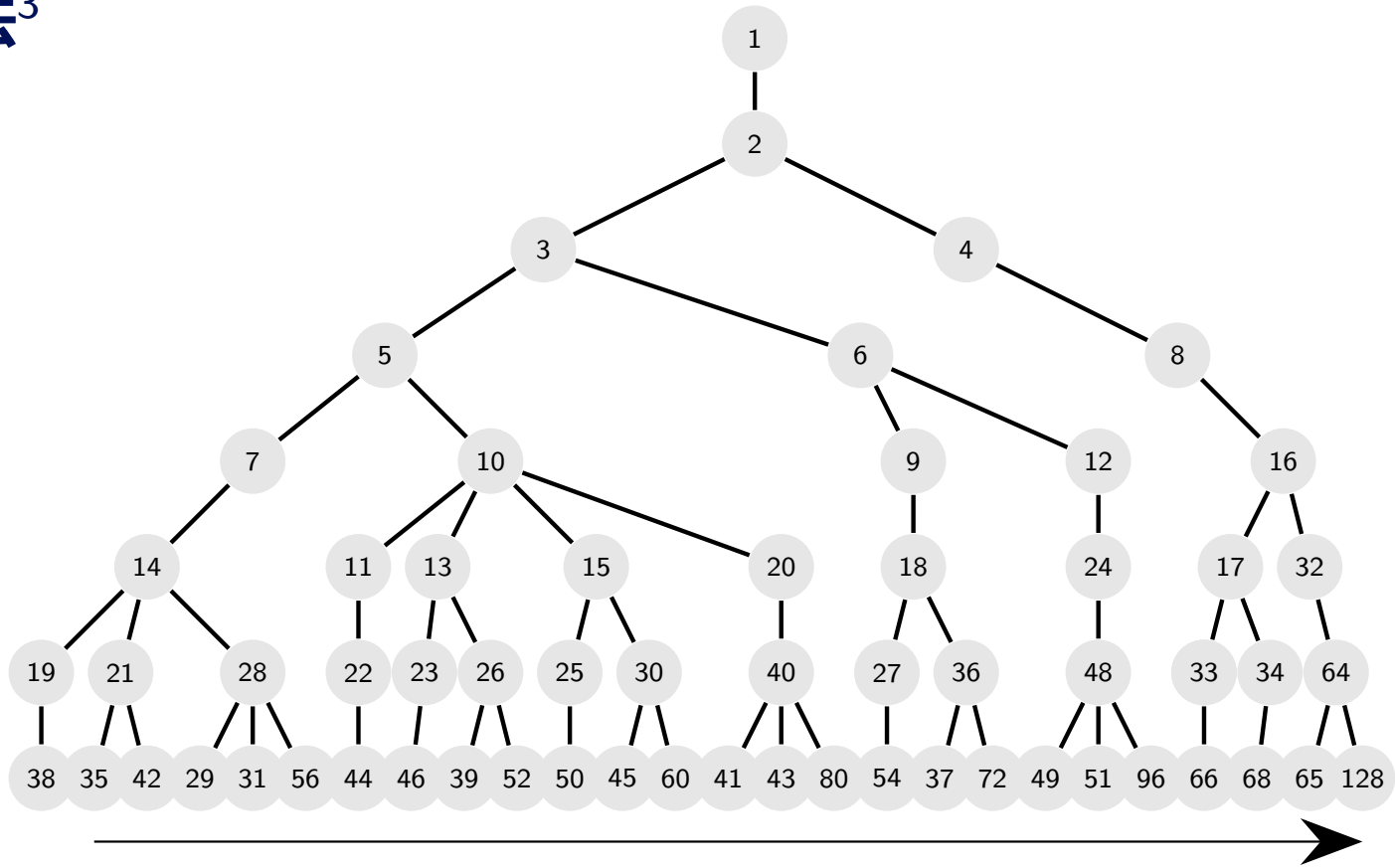
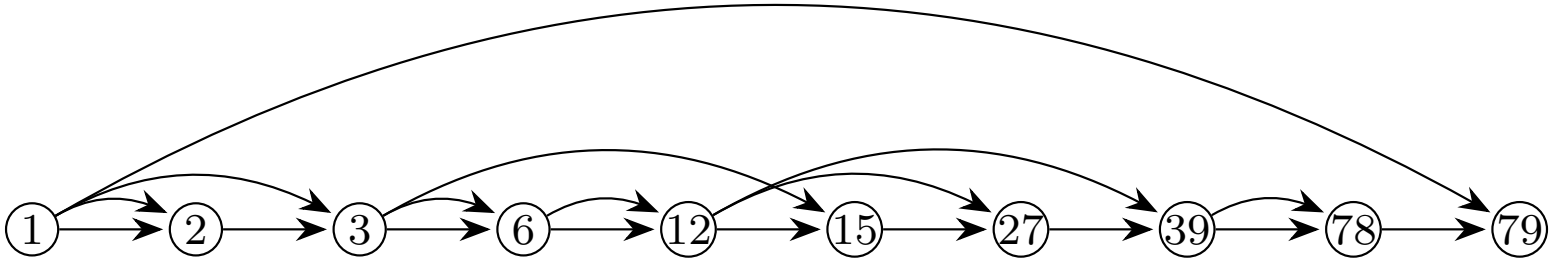


图: 8 层 Knuth 幂树

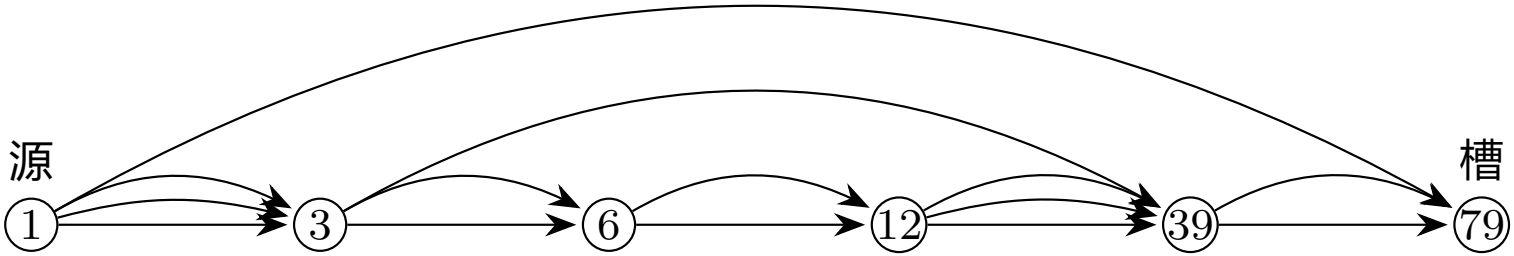
- 按层序扩展，所以一个数只在树中出现一次，树根到一个节点的节点链就是叶子节点的加法链
- 简单易实现，但精度不高，常常用于计算上界
- 验证表明：对所有小于 24924 的数，Knuth 方法生成幂树的结果可以获得 85% 的准确率，且不正确的结果的次优比例高达 99.9%
- 一种改进方法：随机幂树法

[3] [E.Knuth Donald. The art of computer programming, vol. 2: Seminumerical algorithms 1981]

图表示法



删去其出口次数为 1 的顶点，并把来自它的前驱的有向边附加到它的后继（约简图）：



则寻找 n 的最短加法链问题等价于所有约简图中寻找从源到槽有 n 个有向通路并且满足

$$（有向边数）-（顶点数）+1$$

最小的问题。

2016 年，微软研究员 Clift 用图表示法计算出所有小于 2^{36} 的正整数的最短加法链[4]，并对任意的 $\ell(n) \leq 8$，证明了 $\ell(2^n - 1) = n + \ell(n) - 1$。

[4] http://wwwhomes.uni-bielefeld.de/achim/addition_chain.html

ℓ 和 ℓ^* 函数的性质和界

著名定理和猜想

1. **二进方法.** $\lambda(n) \le \ell(n) \le \lambda(n) + \mu(n) - 1$
2. **Brauer（2^k 进）方法.**[5] $\lim_{n\to\infty} \frac{\ell^*(n)}{\lambda(n)} = \lim_{n\to\infty} \frac{\ell(n)}{\lambda(n)} = 1$
3. **Scholz-Brauer 猜想.** $\ell(2^n - 1) \le n - 1 + \ell(n)$.
   125 的最短星链（$\ell^*(125) = 9$）：
   $1 \to 2 \to 4 \to 5 \to 10 \to 15 \to 30 \to 60 \to 120 \to 125$

   $(1)_2 - (10)_2 - (11)_2 - (110)_2 - (1100)_2 - (1111)_2 - (11110)_2 - (11111)_2 - \overset{\text{加倍 5 次}}{\cdots}$

   $-(1111100000)_2 - (1111111111)_2 - \overset{\text{以此类推}}{\cdots} - (111\overset{\text{125 个 1}}{\cdots})_2$

   表格: $2^{125} - 1$ 的一条长度为 $125 - 1 + \ell^*(125) = 133$ 的加法链

4. **Schönhage 定理和猜想.**[6]
   定理：$\ell(n) \ge \lceil \log_2(n) + \log_2(\mu(n)) - 2.13 \rceil$.
   猜想：$\ell(n) \ge \lambda(n) + \lceil \log_2(\mu(n)) \rceil$.
5. **Yao[7]给出界：**

$$\ell(a_1, a_2, \cdots, a_k) \le \log_2(a_k) + \left(2 + \frac{4}{\sqrt{\log_2(a_k)}}\right) \frac{k \log_2(a_k)}{\log_2(\log_2(a_k))}$$

[5] [A.Brauer, *Bull.Amer.Math.Soc.* **45** (1939), 736~739].
[6] [A. Schönhage, *Theoretical Computer Science.* **1** (1975), 1~12].
[7] [Andrew Yao, *Siam J. Comput.* **5** (1976), 100~103].

ℓ 和 ℓ^* 函数的性质和界

著名定理和猜想

1. **二进方法.** $\lambda(n) \le \ell(n) \le \lambda(n) + \mu(n) - 1$
2. **Brauer（2^k 进）方法.**[5] $\lim_{n\to\infty} \frac{\ell^*(n)}{\lambda(n)} = \lim_{n\to\infty} \frac{\ell(n)}{\lambda(n)} = 1$
3. **Scholz-Brauer 猜想.** $\ell(2^n - 1) \le n - 1 + \ell(n)$.
   125 的最短星链 $(\ell^*(125) = 9)$:
   $1 \to 2 \to 4 \to 5 \to 10 \to 15 \to 30 \to 60 \to 120 \to 125$

   $(1)_2 - (10)_2 - (11)_2 - (110)_2 - (1100)_2 - (1111)_2 - (11110)_2 - (11111)_2 - \overset{\text{加倍 5 次}}{\cdots}$
   $-(111100000)_2 - (111111111)_2 - \overset{\text{以此类推}}{\cdots} - (111\overset{125\text{ 个 }1}{\cdots})_2$

   表格: $2^{125} - 1$ 的一条长度为 $125 - 1 + \ell^*(125) = 133$ 的加法链
4. **Schönhage 定理和猜想.**[6]
   定理: $\ell(n) \ge \lceil \log_2(n) + \log_2(\mu(n)) - 2.13 \rceil$.
   猜想: $\ell(n) \ge \lambda(n) + \lceil \log_2(\mu(n)) \rceil$.
5. **Yao[7]给出界:**

$$\ell(a_1, a_2, \cdots, a_k) \le \log_2(a_k) + \left(2 + \frac{4}{\sqrt{\log_2(a_k)}}\right) \frac{k \log_2(a_k)}{\log_2(\log_2(a_k))}$$

[5] [A.Brauer, *Bull.Amer.Math.Soc.* **45** (1939), 736~739].
[6] [A. Schönhage, *Theoretical Computer Science.* **1** (1975), 1~12].
[7] [Andrew Yao, *Siam J. Comput.* **5** (1976), 100~103].

ℓ 和 ℓ^* 函数的性质和界

著名定理和猜想

1. **二进方法.** $\lambda(n) \le \ell(n) \le \lambda(n) + \mu(n) - 1$
2. **Brauer（2^k 进）方法.**[5] $\lim_{n\to\infty} \frac{\ell^*(n)}{\lambda(n)} = \lim_{n\to\infty} \frac{\ell(n)}{\lambda(n)} = 1$
3. **Scholz-Brauer 猜想.** $\ell(2^n - 1) \le n - 1 + \ell(n)$.
   125 的最短星链（$\ell^*(125) = 9$）：
   $1 \to 2 \to 4 \to 5 \to 10 \to 15 \to 30 \to 60 \to 120 \to 125$

   $(1)_2 - (10)_2 - (11)_2 - (110)_2 - (1100)_2 - (1111)_2 - (11110)_2 - (11111)_2 - \overset{\text{加倍 5 次}}{\cdots}$

   $-(1111100000)_2 - (1111111111)_2 - \overset{\text{以此类推}}{\cdots} - (111\overset{\text{125 个 1}}{\cdots})_2$

   表格: $2^{125} - 1$ 的一条长度为 $125 - 1 + \ell^*(125) = 133$ 的加法链

4. **Schönhage 定理和猜想.**[6]
   定理：$\ell(n) \ge \lceil \log_2(n) + \log_2(\mu(n)) - 2.13 \rceil$.
   猜想：$\ell(n) \ge \lambda(n) + \lceil \log_2(\mu(n)) \rceil$.
5. **Yao[7]给出界：**

$$\ell(a_1, a_2, \cdots, a_k) \le \log_2(a_k) + \left(2 + \frac{4}{\sqrt{\log_2(a_k)}}\right) \frac{k \log_2(a_k)}{\log_2(\log_2(a_k))}$$

[5] [A.Brauer, *Bull.Amer.Math.Soc.* **45** (1939), 736~739].
[6] [A. Schönhage, *Theoretical Computer Science.* **1** (1975), 1~12].
[7] [Andrew Yao, *Siam J. Comput.* **5** (1976), 100~103].

ℓ 和 ℓ^* 函数的性质和界

著名定理和猜想

1. **二进方法.** $\lambda(n) \le \ell(n) \le \lambda(n) + \mu(n) - 1$
2. **Brauer（2^k 进）方法.**[5] $\lim_{n\to\infty} \frac{\ell^*(n)}{\lambda(n)} = \lim_{n\to\infty} \frac{\ell(n)}{\lambda(n)} = 1$
3. **Scholz-Brauer 猜想.** $\ell(2^n - 1) \le n - 1 + \ell(n)$.
   125 的最短星链（$\ell^*(125) = 9$）：
   $1 \to 2 \to 4 \to 5 \to 10 \to 15 \to 30 \to 60 \to 120 \to 125$

   $(1)_2 - (10)_2 - (11)_2 - (110)_2 - (1100)_2 - (1111)_2 - (11110)_2 - (11111)_2 -$ 加倍 5 次 $\cdots$

   $-(1111100000)_2 - (1111111111)_2 -$ 以此类推 $\cdots -$ $(111\overset{125\text{ 个 }1}{\cdots})_2$

   表格: $2^{125} - 1$ 的一条长度为 $125 - 1 + \ell^*(125) = 133$ 的加法链

4. **Schönhage 定理和猜想.**[6]
   定理：$\ell(n) \ge \lceil \log_2(n) + \log_2(\mu(n)) - 2.13 \rceil$.
   猜想：$\ell(n) \ge \lambda(n) + \lceil \log_2(\mu(n)) \rceil$.
5. **Yao[7]给出界：**

$$\ell(a_1, a_2, \cdots, a_k) \le \log_2(a_k) + \left(2 + \frac{4}{\sqrt{\log_2(a_k)}}\right) \frac{k \log_2(a_k)}{\log_2(\log_2(a_k))}$$

[5] [A.Brauer, *Bull.Amer.Math.Soc.* **45** (1939), 736~739].
[6] [A. Schönhage, *Theoretical Computer Science.* **1** (1975), 1~12].
[7] [Andrew Yao, *Siam J. Comput.* **5** (1976), 100~103].

ℓ 和 ℓ^* 函数的性质和界

著名定理和猜想

1. **二进方法.** $\lambda(n) \le \ell(n) \le \lambda(n) + \mu(n) - 1$
2. **Brauer（2^k 进）方法.**[5] $\lim_{n\to\infty} \frac{\ell^*(n)}{\lambda(n)} = \lim_{n\to\infty} \frac{\ell(n)}{\lambda(n)} = 1$
3. **Scholz-Brauer 猜想.** $\ell(2^n - 1) \le n - 1 + \ell(n)$.
   125 的最短星链（$\ell^*(125) = 9$）：
   $1 \to 2 \to 4 \to 5 \to 10 \to 15 \to 30 \to 60 \to 120 \to 125$

   $(1)_2 - (10)_2 - (11)_2 - (110)_2 - (1100)_2 - (1111)_2 - (11110)_2 - (11111)_2 -$ 加倍 5 次 $\cdots$

   $-(1111100000)_2 - (1111111111)_2 -$ 以此类推 $\cdots -$ $(111\overset{125\text{ 个 }1}{\cdots})_2$

   表格: $2^{125} - 1$ 的一条长度为 $125 - 1 + \ell^*(125) = 133$ 的加法链

4. **Schönhage 定理和猜想.**[6]
   定理：$\ell(n) \ge \lceil \log_2(n) + \log_2(\mu(n)) - 2.13 \rceil$.
   猜想：$\ell(n) \ge \lambda(n) + \lceil \log_2(\mu(n)) \rceil$.
5. **Yao[7]给出界：**

$$\ell(a_1, a_2, \cdots, a_k) \le \log_2(a_k) + \left(2 + \frac{4}{\sqrt{\log_2(a_k)}}\right) \frac{k \log_2(a_k)}{\log_2(\log_2(a_k))}$$

[5] [A.Brauer, *Bull.Amer.Math.Soc.* **45** (1939), 736~739].
[6] [A. Schönhage, *Theoretical Computer Science.* **1** (1975), 1~12].
[7] [Andrew Yao, *Siam J. Comput.* **5** (1976), 100~103].

赛题描述与分析

加法链问题（ACP） 给定正整数 n，设 $n-5 \leq m \leq n+5$，指出哪一个 m 值具有最短的加法链，并给出其加法链表示。

数据范围

挑战赛题 n	$\lambda(n)$	$\mu(n)$
第一类	43	7
第二类	47	16
第三类	63	5
第四类	127	14
第五类	254	253
第六类	510	510
第七类	1535	784

求解思路

1. 因子方法
2. 遗传算法
3. 滑动窗口法
4. 混合使用上述方法

赛题描述与分析

加法链问题（ACP）

给定正整数 n，设 $n-5 \leq m \leq n+5$，指出哪一个 m 值具有最短的加法链，并给出其加法链表示。

数据范围

挑战赛题 n	$\lambda(n)$	$\mu(n)$
第一类	43	7
第二类	47	16
第三类	63	5
第四类	127	14
第五类	254	253
第六类	510	510
第七类	1535	784

求解思路

1. 因子方法
2. 遗传算法
3. 滑动窗口法
4. 混合使用上述方法

赛题描述与分析

加法链问题（ACP）

给定正整数 n，设 $n-5 \leq m \leq n+5$，指出哪一个 m 值具有最短的加法链，并给出其加法链表示。

数据范围

挑战赛题 n	$\lambda(n)$	$\mu(n)$
第一类	43	7
第二类	47	16
第三类	63	5
第四类	127	14
第五类	254	253
第六类	510	510
第七类	1535	784

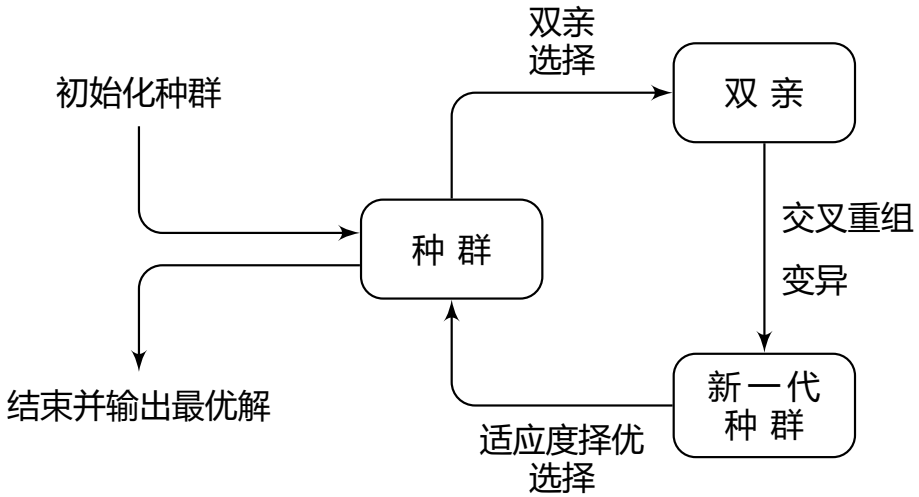
求解思路

1. 因子方法
2. 遗传算法
3. 滑动窗口法
4. 混合使用上述方法

遗传算法

框架与特点

遗传算法 (Genetic Algorithm) 是从达尔文生物进化论的自然选择思想获得灵感而仿生出的算法，通过模拟物种进化过程来搜索最优解。



- 擅长解决全局最优化问题
- 如果适应度函数和交叉算子选择不当，则有可能收敛于局部最优
- 初始种群的数量是重要的参数
- 遗传算法能很快地找到良好的解，调节参数（个体数目，交叉率，变异率）有助于更快地收敛

遗传算法——初始化种群

初始化种群算法

输入: 正整数 n
输出: n 的一条加法链

1: 将第 0 个元素设置为 1，第 1 个元素设置为 2，第 2 个元素随机地置成 3 或 4。
2: 以 10% 的概率将链尾元素加倍，直到它超过 $\frac{n}{2}$ 为止。
3: 检查当前生成的加法链是否合法。
4: 随机选取下述方法以获得链尾的下一个元素，前提是它不超过 n:
    (1). 链中随机选取两个元素求和
    (2). 链尾元素和链中随机选取一个元素求和
    (3). 从链尾元素往前循环，直到找到可以与链尾元素求和的最大元素

遗传算法——交叉算子

单点交叉

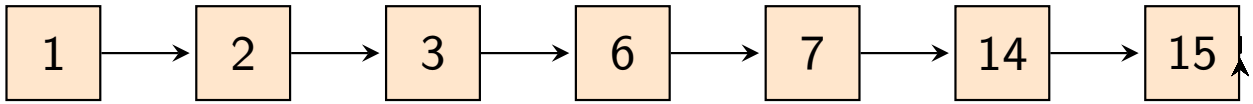
对加法链 P，函数 $FindLowestPair(P, i)$ 返回满足 $P_i = P_{pair_1} + P_{pair_2}$ 且字典序最小的二元组 $(pair_1, pair_2)$。

交叉算子

输入: 正整数 n，n 的双亲加法链 P_1 和 P_2
输出: 后代加法链 $e = \{e_0, e_1, ..., e_k = n\}$

1: $rand \leftarrow random(3, k-1)$
2: 对所有的 $0 \leq i \leq rand$ 执行
$e_i = P1_i$
3: 对所有的 $rand + 1 \leq i \leq k$ 执行
$(pair_1, pair_2) \leftarrow FindLowestPair(P_2, i)$
$e_i = e_{pair_1} + e_{pair_2}$
4: $e \leftarrow RepairChain(e, n)$
5: 输出 e

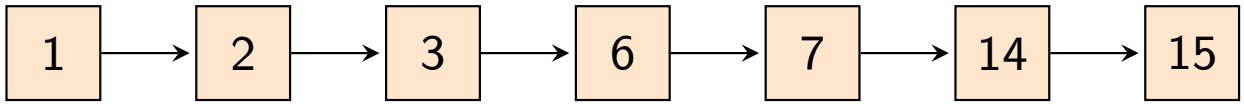
其中 $RepairChain$ 算法以加法链 e 和 n 为输入，并输出一条 n 的包含 e 的加法链。其算法流程与初始化种群算法大同小异。

双亲 P_1 

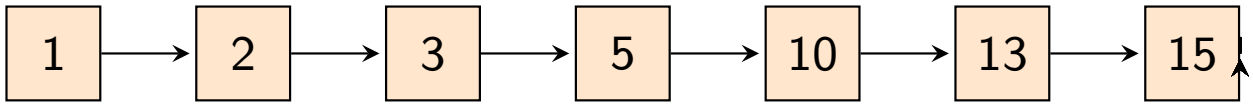
双亲 P_2

后代 e

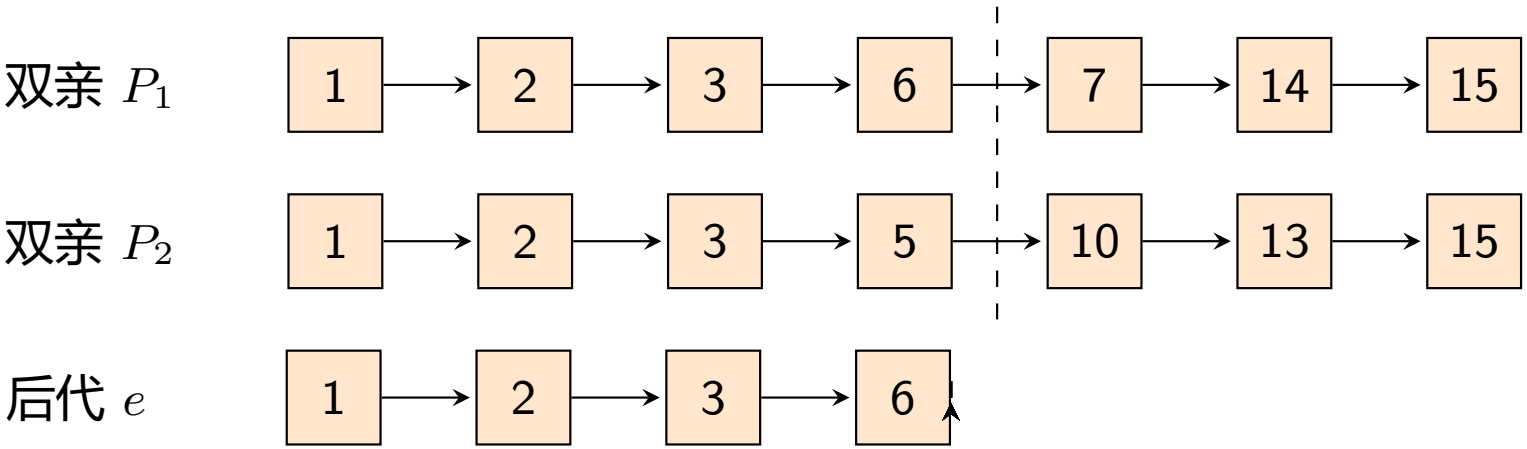
双亲 P_1

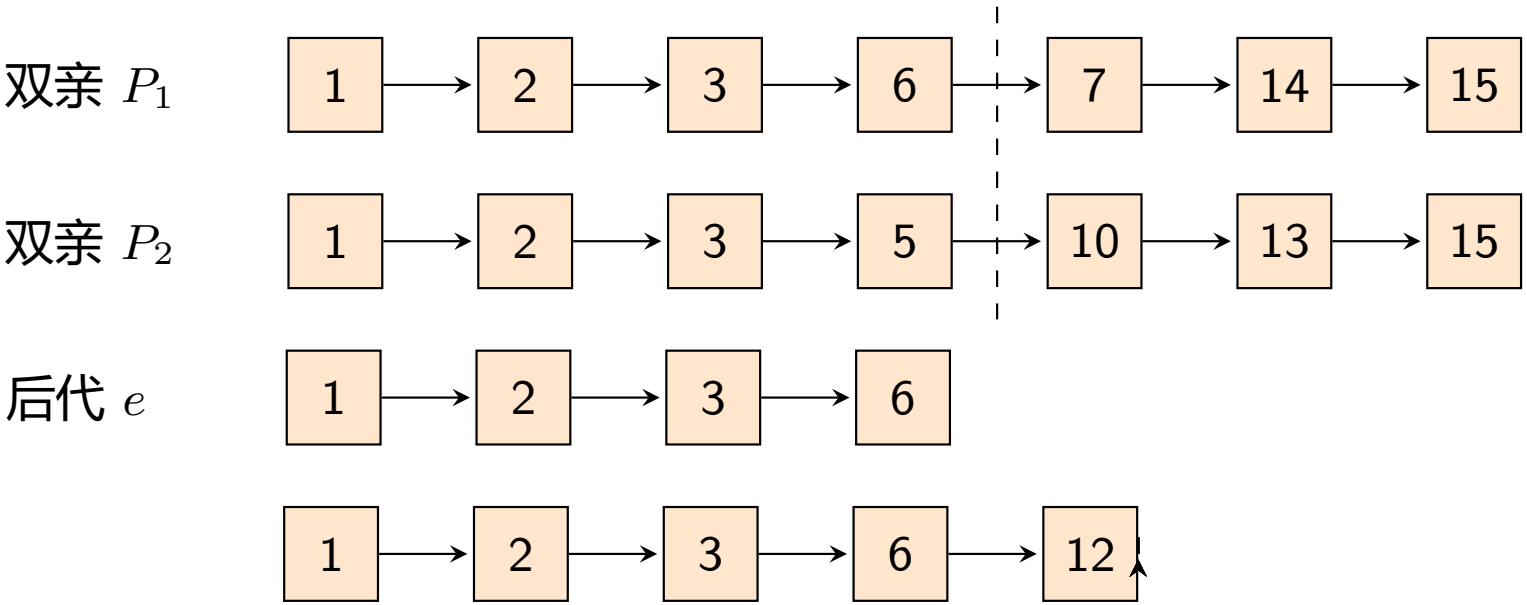


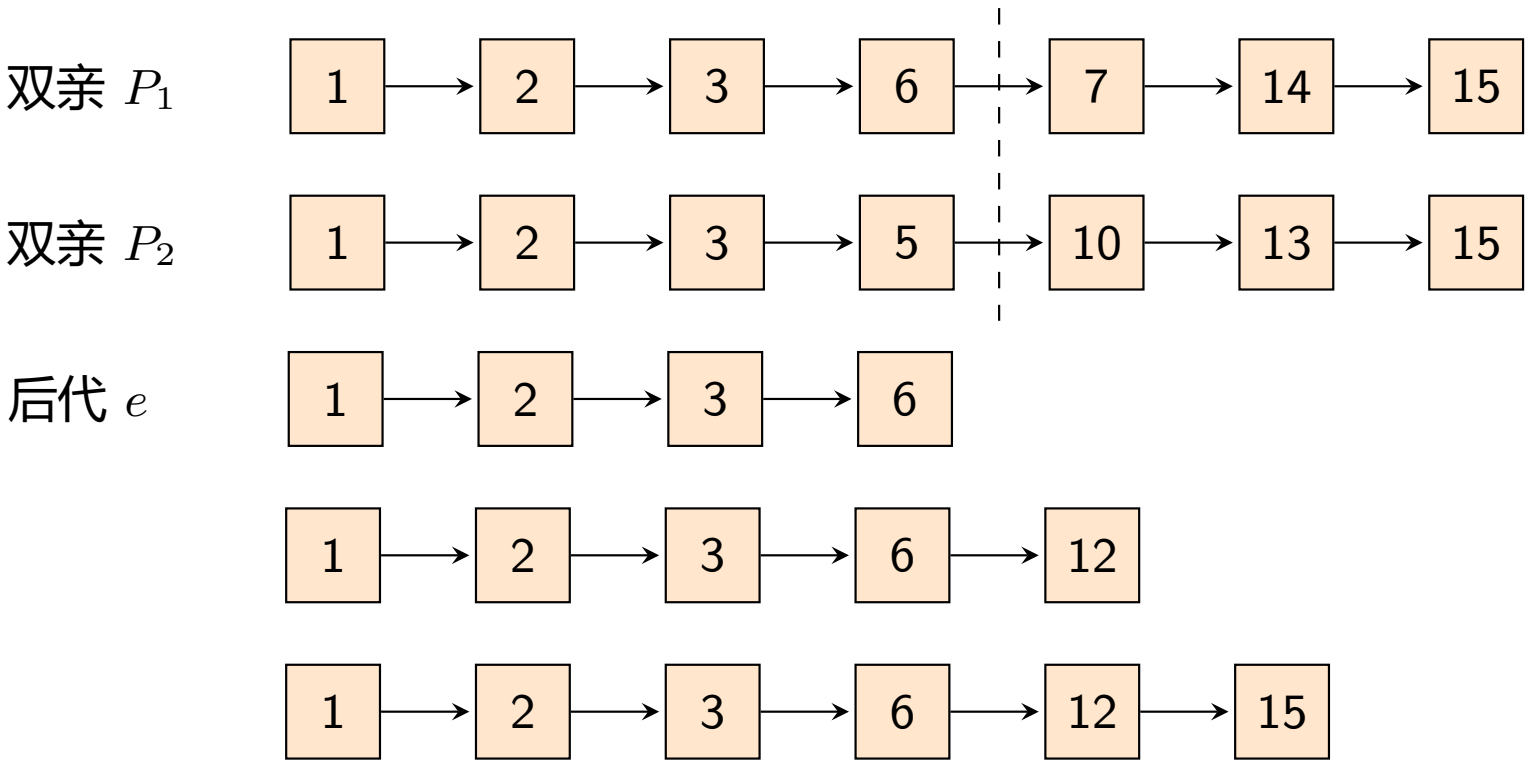
双亲 P_2



后代 e


双亲 P_1
1
2
3
6
7
14
15
双亲 P_2
1
2
3
5
10
13
15
后代 e
1
2
3
6


双亲 P_1
1
2
3
6
7
14
15
双亲 P_2
1
2
3
5
10
13
15
后代 e
1
2
3
6
1
2
3
6
12


双亲 P_1
1 2 3 6 7 14 15
双亲 P_2
1 2 3 5 10 13 15
后代 e
1 2 3 6
1 2 3 6 12
1 2 3 6 12 15

遗传算法——变异算子

变异算子

输入: 正整数 n，加法链 $e = \{e_0, e_1, ..., e_k = n\}$
输出: 变异加法链 $e' = \{e'_0, e'_1, ..., e'_k = n\}$

1: $rand \leftarrow random(2, k-1)$
2: $rand_2 \leftarrow random(0, 1)$
3: 如果 $rand_2 == 1$ 则

$$e_{rand} = e_{rand-1} + e_{rand-2}$$

4: 反之

$$rand_3 \leftarrow random(2, rand-1)$$

$$e_{rand} = e_{rand-1} + e_{rand_3}$$

5: $e' \leftarrow RepairChain(e, n)$
6: 输出 e

華東師範大學 EAST CHINA NORMAL UNIVERSITY
计算机科学与软件工程学院 School of Computer Science and Software Engineering

滑动窗口法

J. Bos 和 M. Coster [8]在 1989 年提出滑动窗口法。该算法的基本思想是将所求大整数用二进制表示，并将其划分为不同的片段（窗口），当窗口大小等于 1 时，就退化成了二进方法。

例

我们以大整数 $262359474289536631831 91$ 为例，窗口规模设为 5：

$$\underbrace{1011}_{11}000\underbrace{111}_{7}00\underbrace{1}_{1}000000\underbrace{11101}_{29}00\underbrace{101}_{5}00\underbrace{11101}_{29}0\underbrace{1}_{1}000000\underbrace{10111}_{23}\underbrace{1}_{1}$$

$$00000\underbrace{11111}_{31}00\underbrace{11001}_{25}0\underbrace{10101}_{21}\underbrace{11}_{3}$$

如果已知含有上述非零窗口的加法链：

$$1 \to 2 \to \underline{3} \to 4 \to \underline{5} \to \underline{7} \to \underline{11} \to 16 \to \underline{21} \to \underline{23} \to \underline{25} \to \underline{29} \to \underline{31}.$$

那么，我们就可以交替使用倍增和加法两种操作构造出它的加法链。可以发现，对于每个非零窗口，执行了一次加法。综上，加法链长度为 12，加倍次数为 71，加法次数是 12，生成的加法链总长为 95。

[8] [Bos J, Coster M, *Conference on the Theory and Application of Cryptology*. (1989) 400~407].

滑动窗口算法

输入: 大整数 n 的二进制表示 $B = (b_t b_{t-1} \dots b_1 b_0)_2$ 与最大窗口规模 k_{max}
输出: 加法链 $A = \{a_0, a_1, a_2, \dots, a_n\}$

1: **初始化参数**

设定最大窗口规模 k_{max}，当前窗口参数 $k = 1$。

2: **求解窗口 k 下的加法链**

2.1 将 B 划分窗口 c_i，并存入集合 $C = (c_0 c_1 \dots c_s)$: 窗口 c_i 当为非零窗口时，满足最高位与最低位都为 1，且元素个数不超过 k。

2.2 对 C 中非零窗口进行 Makesquence 算法，产生最短的包含非零窗口的加法链 $a_0, a_1, \dots, a_h$，并置入集合 A 中。

2.3 对于窗口数组 C，从 c_1 至 c_s 依次进行如下操作：a_h 加倍窗口大小 $n(c_i)$ 次得到相应的加法链元素；如果窗口 c_i 为非零窗口，则加法链最后一个元素加上 c_i 生成下一个链元素，生成的链元素添入 A 中，即得该窗口规模下的加法链构造。

3: **重新选定窗口规模 k**

返回 2，获得窗口规模是 k 的加法链，循环操作，直到达到最大窗口规模 k_{max}。

4: **输出最短的加法链**

比较不同窗口规模求得的加法链链长，将链长最短的加法链输出。

MakeSequence 算法

输入: 正整数 $a_1 < a_2 < \ldots < a_k$
输出: 包含 $a_1, \ldots, a_k$ 的加法链

1: 调用遗传算法生成包含 a_1 的多组加法链，取前 r_1 个最优的保存到桶 B 中。
2: 对所有的 $1 \leq i \leq k$ 执行
　　在初始种群为 B 的基础上，调用遗传算法生成包含 a_i 的多组加法链，取前 r_i 个最优的覆盖桶 B。
3: 输出桶 B 中长度最短的个体。

我们设计的 MakeSequence 算法效果非常好：在对第七类赛题的求解过程中，需要构造包含 95 个数并且最大值是 32151 的加法链，上述算法构造的加法链长度是 173，比 Yao 给出的界：

$$\ell(a_1, a_2, \cdots, a_{95}) \leq \log_2(32151) + \left(2 + \frac{4}{\sqrt{\log_2(32151)}}\right) \frac{95 \log_2(32151)}{\log_2(\log_2(32151))} \approx 1120$$

要小得多。

方法比较

n	$\ell(n)$	二进方法	滑动窗口法（最优）	遗传算法	
				最优	平均
488705	23	26	27	23	23.53
1273909	25	29	30	25	25.87
3399779	25	31	31	25	26.87
5425679	27	32	32	28	28.23
9264263	28	34	34	28	29.63
20279147	29	39	36	30	31.07
51950083	30	34	36	30	31.20
115216741	31	39	38	32	33.60
159963579	23	41	39	34	35.20
310469637	33	36	39	34	34.23
1073740801	35	49	41	35	35.26
17182318319	N/A	49	46	42	46.77
$2^{127}-3$	N/A	251	163	136	161.49

表格: 随机测试

实验概述

1. 所有实验均是在处理器 Core i5、内存 8G 的个人电脑上进行。编程语言为 C++，开发环境是 Visual Studio 2017，并用版本 6.0.0 的 GMP 库进行大数运算。
2. 根据 Schönhage 猜想，对于每类赛题 n，我们用

$$\min_{n-5\leq m\leq n+5}(\lambda(m)+\lceil\log_2(\mu(m))\rceil)$$

   定义理论下界。如果求得结果比对应的理论下界还要小，那么这个猜想就被证否。
3. 3.1 所有实验统一遗传参数设置：初始种群数目是 2000，遗传代数是 600 代，每代交叉取最优秀的 450 个个体，每代变异数目是 10。
   3.2 MakeSequence 算法中所有 r_i 均取 150。
4. 七类赛题中，第一、三类我们的求解结果达到了对应的理论下界，第二、四类相差 2，第五、六类相差 3，第七类相差 217。

我们分别设两类赛题是 n_1 和 n_3，其中第一类赛题 $n_1 = 38 \times 2^{38} + 39$ 是一个质数。分别考虑下述二进制表示：

binary($n_1 - 1$)=
[1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0]

binary($n_3 - 1$)=
[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

对于 $n_1 - 1$ 所以我们构造：

$$(1)_2 - (10)_2 - (100)_2 - (1000)_2 - (1001)_2 - (10010)_2 - (10011)_2 - \overset{\text{加倍 38 次}}{\cdots}$$
$$-(10011\overset{38\text{ 个 }0}{\cdots})_2 \overset{\text{加 19 后再加倍}}{-} binary(n_1 - 1)$$

$n_3 - 1$ 的加法链构造方法与之类似。并且所得的结果均和对应的理论下界值相同，分别为 46 和 65。实现该方法只需要很小的内存和很短的时间。如果使用遗传算法解决两类赛题，也可以得到同样长度的加法链。

华东师范大学 EAST CHINA NORMAL UNIVERSITY 计算机科学与软件工程学院 School of Computer Science and Software Engineering

第六类赛题

第六类赛题 $n_6 = 2^{511} - 17$ 是实现椭圆曲线密码算法常用到的参数。考虑 $n_6 + 1$ 的二进制表示：

$$\text{binary}(2^{511} - 16) = [\overbrace{1, \cdots, 1}^{507\text{个}}, 0, 0, 0, 0]$$

如果我们采用之前分析 Scholz-Brauer 猜想的方法，则只需要构造出 507 的长度为 12 的最短星链：

$$1 \to 2 \to 3 \to 6 \to 12 \to 15 \to 30 \to 60 \to 63 \to 126 \to 252 \to 504 \to 507.$$

这种方法所求的加法链长度是 522，理论下界是 519。由于构造方法简单，运行时间大约是 100ms，内存是 128M。

第二、四、五类赛题

- 对于二、四类赛题 n_2 和 n_4，我们找不到明显的构造方法，所以考虑使用遗传算法。
- 对于第五类赛题 $n_5 = 2^{255} - 19$，虽然可以用求解第六类赛题的方法构造，但是我们也尝试用遗传算法来攻克。值得一提的是，两种方法都得到了相同长度的加法链（构造不同），并且我们实现的效果比 Picek 等人[9]的稍优：他们求解 $2^{255} - 21$ 得到最好的结果是 269 步（已知最优是 265 步）

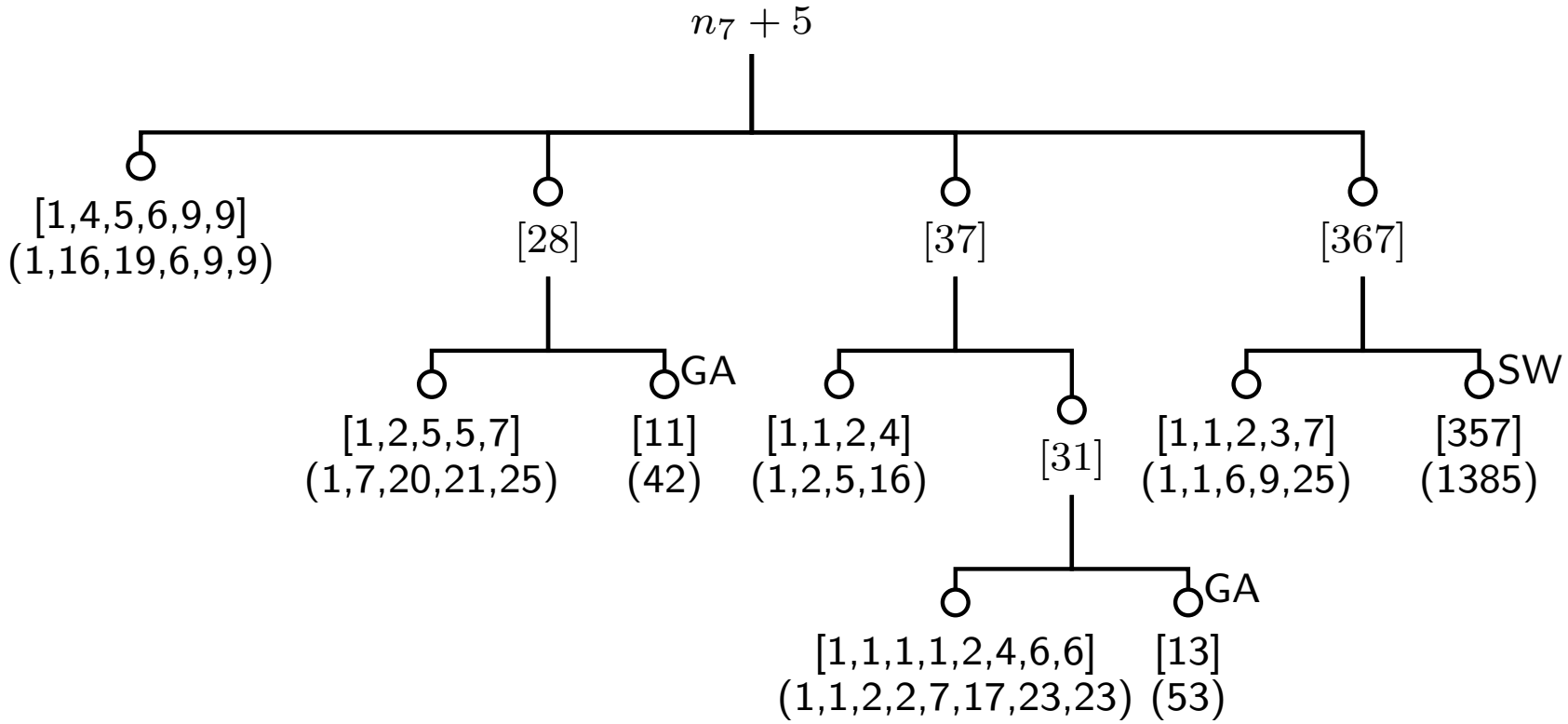
赛题	实验次数	加法链长度	理论下界	运行时间（总）	内存
第二类	≈ 10	53	51	≈ 0.5h	≈ 0.5G
第四类	≈ 20	133	131	≈ 5h	≈ 1G
第五类	≈ 50	265	262	≈ 30h	≈ 2G

表格: 实验结果

[9] [Picek.S et al, *Journal of Heuristics*. (2018) **24**, 457~481].

第七类赛题

第七类赛题 n_7 是 RSA-1536 模数，分解未知，十进制表示 463 位。初赛阶段，我们用窗口规模是 15 的滑动窗口算法求解，得到的加法链长度是 1788，n_7 的理论下界长度是 1545。
后来，我们考虑因子方法并与之前方法混合，新方法的流程图如下：



上述方法得到的加法链长度是 1762，比初赛提交的结果要短一些。因数分解时间大约需要 4 小时，加法链求解时间大约为 2 小时。

总结与思考

- 上个世纪关于加法链的研究产生了数十种求解的算法。通过这次挑战赛，我们学习到了很多算法，并尝试证否一些猜想，改进一些已知结果。
- 目前，Clift 实现的图表示法可以得到 2^{36} 以内所有正整数的最短加法链。是否有更高效的最短加法链求解算法？对于单个数的加法链问题的求解是不是 NP-难的？
- 我们对加法链（Lucas 链）在椭圆曲线密码学的应用的了解还不够深入，希望可以探索更多的应用。
- 加法链在方便计算的同时，会不会受到侧信道攻击？