时间复杂度

吴 铤

2003 年 12 月 18 日

时间复杂度

时间复杂度

时间复杂度

时间复杂度

定义1 (时间复杂度)

令M是在所有输入上都停机的确定型图灵机，M的运行时间或时间复杂度是指函数$f:\mathbb{N}\rightarrow\mathbb{N}$，函数值$f(n)$表示$M$在所有长度为$n$的输入上运行所经过的最大步数。

如果$f(n)$是M的运行时间，则称M在时间$f(n)$内运行，M是$f(n)$时间图灵机。

- 最坏情况分析：某特定长度的所有输入上的最长运行时间
- 平均情况分析：某特定长度的所有输入上的运行时间的平均值

定义1 (时间复杂度)

令M是在所有输入上都停机的确定型图灵机，M的**运行时间**或**时间复杂度**是指函数$f:\mathbb{N}\to\mathbb{N}$，函数值$f(n)$表示$M$在所有长度为$n$的输入上运行所经过的最大步数。

如果$f(n)$是M的运行时间，则称M在时间$f(n)$内运行，M是$f(n)$时间图灵机。

- 最坏情况分析：某特定长度的所有输入上的最长运行时间
- 平均情况分析：某特定长度的所有输入上的运行时间的平均值

定义1 (时间复杂度)

令M是在所有输入上都停机的确定型图灵机，M的运行时间或时间复杂度是指函数$f:\mathbb{N}\rightarrow\mathbb{N}$，函数值$f(n)$表示$M$在所有长度为$n$的输入上运行所经过的最大步数。

如果$f(n)$是M的运行时间，则称M在时间$f(n)$内运行，M是$f(n)$时间图灵机。

- 最坏情况分析：某特定长度的所有输入上的最长运行时间
- 平均情况分析：某特定长度的所有输入上的运行时间的平均值

定义1 (时间复杂度)

令M是在所有输入上都停机的确定型图灵机，M的**运行时间**或**时间复杂度**是指函数$f:\mathbb{N}\to\mathbb{N}$，函数值$f(n)$表示$M$在所有长度为$n$的输入上运行所经过的最大步数。

如果$f(n)$是M的运行时间，则称M在时间$f(n)$内运行，M是$f(n)$时间图灵机。

- **最坏情况分析**：某特定长度的所有输入上的最长运行时间
- **平均情况分析**：某特定长度的所有输入上的运行时间的平均值

定义1 (时间复杂度)

令M是在所有输入上都停机的确定型图灵机，M的运行时间或时间复杂度是指函数$f:\mathbb{N}\rightarrow\mathbb{N}$，函数值$f(n)$表示$M$在所有长度为$n$的输入上运行所经过的最大步数。

如果$f(n)$是M的运行时间，则称M在时间$f(n)$内运行，M是$f(n)$时间图灵机。

- 最坏情况分析：某特定长度的所有输入上的最长运行时间
- 平均情况分析：某特定长度的所有输入上的运行时间的平均值

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n\log_2 n + 5n\log_2\log_2 n$：$f(n) = O(n\log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)}), \delta$是大于零的实数。

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n \log_2 n + 5n \log_2 \log_2 n$：$f(n) = O(n \log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)}), \delta$是大于零的实数。

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n\log_2 n + 5n\log_2\log_2 n$：$f(n) = O(n\log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)})$，δ是大于零的实数。

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n\log_2 n + 5n\log_2\log_2 n$：$f(n) = O(n\log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)}), \delta$是大于零的实数。

定义2

设函数$f, g: \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n\log_2 n + 5n\log_2\log_2 n$：$f(n) = O(n\log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)}), \delta$是大于零的实数。

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的**渐进上界**，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n \log_2 n + 5n \log_2 \log_2 n$：$f(n) = O(n \log n)$
- **多项式界**：$O(n^c)$
- 指数界：$O(2^{(n^\delta)})$, δ是大于零的实数。

定义2

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若存在正整数$c, n_0$，使得当$n \geq n_0$时，有

$$f(n) \leq cg(n),$$

则称$g(n)$是$f(n)$的渐进上界，记为$f(n) = O(g(n))$。

- $f(n) = 5n^3 + 2n^2 + n + 1$：$f(n) = O(n^3)$
- $f(n) = 3n\log_2 n + 5n\log_2\log_2 n$：$f(n) = O(n\log n)$
- 多项式界：$O(n^c)$
- 指数界：$O(2^{(n^\delta)}), \delta$是大于零的实数。

定义3

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

则称$f(n) = o(g(n))$。

- $\sqrt{n} = o(n)$
- $n = o(n \log(\log n))$
- $n \log n = o(n^2)$

定义3

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

则称$f(n) = o(g(n))$。

- $\sqrt{n} = o(n)$
- $n = o(n \log(\log n))$
- $n \log n = o(n^2)$

定义3

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

则称$f(n) = o(g(n))$。

- $\sqrt{n} = o(n)$
- $n = o(n \log(\log n))$
- $n \log n = o(n^2)$

定义3

设函数$f, g : \mathbb{N} \to \mathbb{R}^+$，若

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$$

则称$f(n) = o(g(n))$。

- $\sqrt{n} = o(n)$
- $n = o(n \log(\log n))$
- $n \log n = o(n^2)$

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；$(O(n))$
2. 如果带上既有0也有1，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个0和一个1；
3. 如果所有1/0都被删除后还有0/1，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；$(O(n))$
2. 如果带上既有0也有1，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个0和一个1；
3. 如果所有1/0都被删除后还有0/1，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A = \{0^k1^k | k \geq 0\}$对应图灵机$M$的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作 $(n/2 \cdot O(n) = O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1/0*都被删除后还有*0/1*，则拒绝，否则接受。$(O(n))$

即$A \in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A = \{0^k1^k | k \geq 0\}$对应图灵机$M$的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作 $(n/2 \cdot O(n) = O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1*/*0*都被删除后还有*0*/*1*，则拒绝，否则接受。$(O(n))$

即$A \in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；($O(n)$)
2. 如果带上既有0也有1，则重复以下操作
   ($n/2\cdot O(n)=O(n^2)$)
   1. 扫描带，删除一个0和一个1；
3. 如果所有1/0都被删除后还有0/1，则拒绝，否则接受。
   ($O(n)$)

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；$(O(n))$
2. 如果带上既有0也有1，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
    1. 扫描带，删除一个0和一个1；
3. 如果所有1/0都被删除后还有0/1，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作
   $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1/0*都被删除后还有*0/1*，则拒绝，否则接受。
   $(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；$(O(n))$
2. 如果带上既有0也有1，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个0和一个1；
3. 如果所有1/0都被删除后还有0/1，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1/0*都被删除后还有*0/1*，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1/0*都被删除后还有*0/1*，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例1 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M的算法分析)

M对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；$(O(n))$
2. 如果带上既有*0*也有*1*，则重复以下操作 $(n/2\cdot O(n)=O(n^2))$
   1. 扫描带，删除一个*0*和一个*1*；
3. 如果所有*1/0*都被删除后还有*0/1*，则拒绝，否则接受。$(O(n))$

即$A\in \mathrm{TIME}(n^2)$,其中$\mathrm{TIME}(t(n))$表示为由$O(t(n))$时间的图灵机判定的所有语言构成的集合，称为时间复杂类。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0, 1的总数是奇数还是偶数。若是奇数，则拒绝;
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*;
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*;
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0, 1的总数是奇数还是偶数。若是奇数，则拒绝;
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*;
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*;
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0,1的总数是奇数还是偶数。若是奇数，则拒绝;
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*;
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*;
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0, 1的总数是奇数还是偶数。若是奇数，则拒绝；
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*；
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*；
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的$0, 1$的总数是奇数还是偶数。若是奇数，则拒绝；
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*；
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*；
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的$0,1$的总数是奇数还是偶数。若是奇数，则拒绝；
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*；
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*；
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A=\{0^k1^k|k\geq 0\}$对应图灵机M_1的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0,1的总数是奇数还是偶数。若是奇数，则拒绝;
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*;
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*;
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

例2 (语言$A = \{0^k1^k | k \geq 0\}$对应图灵机$M_1$的算法分析)

M_1对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 如果带上既有*0*也有*1*，则重复以下操作
   1. 扫描带，检查剩余的0, 1的总数是奇数还是偶数。若是奇数，则拒绝;
   2. 再次扫描带，从第一个*0*开始，隔一个删除一个*0*;
   3. 再次扫描带，从第一个*1*开始，隔一个删除一个*1*;
3. 如果带上没有*0*和*1*，则接受，否则拒绝。

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析:

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析:

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1+\log_2 n$
- M_1运行时间$O(n)+(1+\log_2 n)O(n) = O(n\log n)$，$A \in \mathrm{TIME}(n\log n)$

对于输入$0^{13}1^{13}$，有

- 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 (13个)
- 0 0 0 0 0 0 1 1 1 1 1 1 1 (6个)
- 0 0 0 1 1 1 (3个)
- 0 1 (1个)

个数序列为奇、偶、奇、奇 反转编码后为$(1101)_2 = 13$

运行时间分析：

- 每个步骤所需时间$O(n)$
- 循环次数$1 + \log_2 n$
- M_1运行时间$O(n) + (1 + \log_2 n)O(n) = O(n \log n)$，$A \in \mathrm{TIME}(n \log n)$

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上;
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*;
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
    - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*；
    - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*；
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in\mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在1的右侧发现0，则拒绝；
2. 扫描带1上的0，直至第一个1时停止。将0复制到带2上；
3. 扫描带1上的1直至输入的末尾。
   - 每次从带1上读到一个1，就在带2上输出一个0；
   - 如果在读完1之前所有的0都被删除，则拒绝。
4. 如果所有的0都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上;
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*;
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝;
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上;
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*;
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*；
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*；
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*；
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：*A*的复杂度与选取的计算模型有关。

例3 (语言$A=\{0^k1^k|k\geq 0\}$对应双带图灵机M_2的算法分析)

M_2对于输入串w，执行

1. 扫描带。如果在*1*的右侧发现*0*，则拒绝；
2. 扫描带*1*上的*0*，直至第一个*1*时停止。将*0*复制到带*2*上；
3. 扫描带*1*上的*1*直至输入的末尾。
   - 每次从带*1*上读到一个*1*，就在带*2*上输出一个*0*;
   - 如果在读完*1*之前所有的*0*都被删除，则拒绝。
4. 如果所有的*0*都被删除，则接受，否则拒绝。

在双带图灵机模型下，$A\in \mathrm{TIME}(n)$，即其是线性时间。

注意：A的复杂度与选取的计算模型有关。

定理1

设$t(x)$是一个函数，$t(n) \geq n$，则

- 每个$t(n)$时间的多带图灵机和某个$O(t^2(n))$时间的单带图灵机等价。
- 每个$t(n)$时间的非确定型单带图灵机和某个$2^{O(t(n))}$时间的确定型单带图灵机等价。
- 设N是非确定型图灵机，且是判定机，N的运行时间是函数$f : \mathbb{N} \to \mathbb{N}$，其中$f(n)$是在任何长度为$n$的输入上所有计算分支中的最大步数。

定理1

设$t(x)$是一个函数，$t(n) \geq n$，则

- 每个$t(n)$时间的多带图灵机和某个$O(t^2(n))$时间的单带图灵机等价。
- 每个$t(n)$时间的非确定型单带图灵机和某个$2^{O(t(n))}$时间的确定型单带图灵机等价。
- 设N是非确定型图灵机，且是判定机，N的运行时间是函数$f: \mathbb{N} \to \mathbb{N}$，其中$f(n)$是在任何长度为$n$的输入上所有计算分支中的最大步数。

定理1

设$t(x)$是一个函数，$t(n) \geq n$，则

- 每个$t(n)$时间的多带图灵机和某个$O(t^2(n))$时间的单带图灵机等价。
- 每个$t(n)$时间的非确定型单带图灵机和某个$2^{O(t(n))}$时间的确定型单带图灵机等价。
- 设N是非确定型图灵机，且是判定机，N的运行时间是函数$f: \mathbb{N} \to \mathbb{N}$，其中$f(n)$是在任何长度为$n$的输入上所有计算分支中的最大步数。

定理1

设$t(x)$是一个函数，$t(n) \geq n$，则

- 每个$t(n)$时间的多带图灵机和某个$O(t^2(n))$时间的单带图灵机等价。
- 每个$t(n)$时间的非确定型单带图灵机和某个$2^{O(t(n))}$时间的确定型单带图灵机等价。
- 设N是非确定型图灵机，且是判定机，N的**运行时间**是函数$f: \mathbb{N} \to \mathbb{N}$，其中$f(n)$是在任何长度为$n$的输入上所有计算分支中的最大步数。

定义4 (P类)

P是确定型单带图灵机在多项式时间内可判定的语言类。即

$$P = \bigcup_k \mathrm{TIME}(n^k).$$

P问题示例:

- $PATH = \{\langle G, s, t\rangle | G$是具有从$s$到$t$的有向路径的有向图$\}$
- $RELPRIME = \{\langle x, y\rangle | x, y$互素$\}$

定义4 (P类)

P是确定型单带图灵机在多项式时间内可判定的语言类。即

$$P = \bigcup_k \mathrm{TIME}(n^k).$$

P问题示例:

- $PATH = \{\langle G, s, t\rangle | G$是具有从$s$到$t$的有向路径的有向图$\}$
- $RELPRIME = \{\langle x, y\rangle | x, y$互素$\}$

定义4 (P类)

P是确定型单带图灵机在多项式时间内可判定的语言类。即

$$P=\bigcup_{k}\mathrm{TIME}(n^{k}).$$

P问题示例:

- $PATH=\{\langle G,s,t\rangle|G$是具有从$s$到$t$的有向路径的有向图$\}$
- $RELPRIME=\{\langle x,y\rangle|x,y$互素$\}$

定义4 (P类)

P是确定型单带图灵机在多项式时间内可判定的语言类。即

$$P = \bigcup_k \mathrm{TIME}(n^k).$$

P问题示例：

- $PATH = \{\langle G, s, t\rangle | G$是具有从$s$到$t$的有向路径的有向图$\}$
- $RELPRIME = \{\langle x, y\rangle | x, y$互素$\}$

定义4 (P类)

P是确定型单带图灵机在多项式时间内可判定的语言类。即

$$P = \bigcup_k \mathrm{TIME}(n^k).$$

P问题示例:

- $PATH = \{\langle G, s, t\rangle | G$是具有从$s$到$t$的有向路径的有向图$\}$
- $RELPRIME = \{\langle x, y\rangle | x, y$互素$\}$

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m \log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m\log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m \log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m\log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m\log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a,b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m \log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m\log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a,b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m \log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理2 ($PATH \in P$)

- 对于有m个节点的有向图G，PATH问题的暴力算法所需时间为$m! = \Theta(2^{m\log m}) = \Theta(m^m)$
- 多项式时间算法M：
  1. 在结点s上做标记；
  2. 重复以下步骤，直至不再有新结点被标记
     - 扫描G所有的边。如果找到边(a, b)，其中a已被标记而b未被标记，则标记b
  3. 如果t被标记，则接受，否则拒绝。

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y\rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

□

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y \rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y\rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y\rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y \rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

定理3 ($PELPRIME \in P$)

证明.

对于输入的$\langle x, y\rangle$，执行

1. 重复以下操作，直至$y = 0$
   - $x \leftarrow x \mod y$
   - 交换x, y
2. 如果$x = 1$则接受，否则拒绝。

□

定理4 (每个上下文无关语言都是P的成员)

基本思想：利用短子串来确定长子串的生成方法

- 确定长度为1的生成方法
- 对于长子串$w_1 \cdots w_k$，将其分为两个短子串$w_1 \cdots w_t, w_{t+1} \cdots w_k$
    - 若有规则$A \to BC$，而B, C分别可以生成$w_1 \cdots w_t, w_{t+1} \cdots w_k$，则$A$生成$w_1 \cdots w_k$

定理4 (每个上下文无关语言都是P的成员)

基本思想：利用短子串来确定长子串的生成方法

- 确定长度为1的生成方法
- 对于长子串$w_1 \cdots w_k$，将其分为两个短子串$w_1 \cdots w_t, w_{t+1} \cdots w_k$
  - 若有规则$A \to BC$，而B, C分别可以生成$w_1 \cdots w_t, w_{t+1} \cdots w_k$，则$A$生成$w_1 \cdots w_k$

定理4（每个上下文无关语言都是P的成员）

基本思想：利用短子串来确定长子串的生成方法

- 确定长度为1的生成方法
- 对于长子串$w_1 \cdots w_k$，将其分为两个短子串$w_1 \cdots w_t, w_{t+1} \cdots w_k$
  - 若有规则$A \to BC$，而B, C分别可以生成$w_1 \cdots w_t, w_{t+1} \cdots w_k$，则$A$生成$w_1 \cdots w_k$

定理4 (每个上下文无关语言都是P的成员)

基本思想：利用短子串来确定长子串的生成方法

- 确定长度为1的生成方法
- 对于长子串$w_1\cdots w_k$，将其分为两个短子串$w_1\cdots w_t, w_{t+1}\cdots w_k$
  - 若有规则$A\rightarrow BC$，而B,C分别可以生成$w_1\cdots w_t, w_{t+1}\cdots w_k$，则$A$生成$w_1\cdots w_k$

定理4（每个上下文无关语言都是P的成员）

基本思想：利用短子串来确定长子串的生成方法

- 确定长度为1的生成方法
- 对于长子串$w_1 \cdots w_k$，将其分为两个短子串$w_1 \cdots w_t, w_{t+1} \cdots w_k$
  - 若有规则$A \rightarrow BC$，而B, C分别可以生成$w_1 \cdots w_t, w_{t+1} \cdots w_k$，则$A$生成$w_1 \cdots w_k$

P中的问题举例

对于上下文无关语言L，设G是生成L的乔姆斯基范式的CFG，即其中的规则形式为$A \to BC$，$A \to a$，则对于输入$w = w_1 \cdots w_n$

1. 若$w = \epsilon$且有规则$S \to \epsilon$，则接受，否则拒绝
2. 对$i = 1$ to n　　运行nv次，v为变元数
3. 　对每个变元A，
4. 　　检查是否有规则$A \to w_i$。若有，则把A放入Table(i, i)
5. 对$\ell = 2$ to n　　运行n^3步
6. 　对$i = 1$ to $n - \ell + 1$
7. 　　令$j = i + \ell - 1$
8. 　　对$k = i$ to $j - 1$
9. 　　　对每条规则$A \to BC$
10. 　　　　若Table(i, k)包含B，Table$(k + 1, j)$包含C，则把A放入Tabel(i, j)
11. 若S属于Table$(1, n)$，则接受，否则拒绝

对于上下文无关语言L，设G是生成L的乔姆斯基范式的CFG，即其中的规则形式为$A \to BC$，$A \to a$，则对于输入$w = w_1 \cdots w_n$

1. 若$w = \epsilon$且有规则$S \to \epsilon$，则接受，否则拒绝
2. 对$i = 1$ to n　　运行nv次，v为变元数
3. 　对每个变元A，
4. 　　检查是否有规则$A \to w_i$。若有，则把A放入Table(i, i)
5. 对$\ell = 2$ to n　　运行n^3步
6. 　对$i = 1$ to $n - \ell + 1$
7. 　　令$j = i + \ell - 1$
8. 　　对$k = i$ to $j - 1$
9. 　　　对每条规则$A \to BC$
10. 　　　　若Table(i, k)包含B，Table$(k + 1, j)$包含C，则把A放入Tabel(i, j)
11. 若S属于Table$(1, n)$，则接受，否则拒绝

对于上下文无关语言L，设G是生成L的乔姆斯基范式的CFG，即其中的规则形式为$A \to BC$，$A \to a$，则对于输入$w = w_1 \cdots w_n$

1. 若$w = \epsilon$且有规则$S \to \epsilon$，则接受，否则拒绝
2. 对$i = 1$ to n　　运行nv次，v为变元数
3. 　对每个变元A，
4. 　　检查是否有规则$A \to w_i$。若有，则把A放入$\mathrm{Table}(i, i)$
5. 对$\ell = 2$ to n　　运行n^3步
6. 　对$i = 1$ to $n - \ell + 1$
7. 　　令$j = i + \ell - 1$
8. 　　对$k = i$ to $j - 1$
9. 　　　对每条规则$A \to BC$
10. 　　　　若$\mathrm{Table}(i, k)$包含B，$\mathrm{Table}(k + 1, j)$包含C，则把A放入$\mathrm{Tabel}(i, j)$
11. 若S属于$\mathrm{Table}(1, n)$，则接受，否则拒绝

- $HAMPATH = \{\langle G, s, t\rangle | G$是包含从$s$到$t$的哈密尔顿路径的有向图$\}$
- $COMPOSITES = \{x | x = pq,$ 整数$p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G$是包含$k$团的无向图$\}$
- $SUBSET-SUM = \{\langle s, t\rangle | s = \{x_1, \cdots, x_k\}$且存在$\{y_1, \cdots, y_l\} \subseteq s,$ 满足$\sum y_i = t\}$

多项式可验证性

- $HAMPATH = \{\langle G, s, t\rangle | G\text{是包含从}s\text{到}t\text{的哈密尔顿路径的有向图}\}$
- $COMPOSITES = \{x | x = pq, \text{整数}p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G\text{是包含}k\text{团的无向图}\}$
- $SUBSET-SUM = \{\langle s, t\rangle | s = \{x_1, \cdots, x_k\}\text{且存在}\{y_1, \cdots, y_l\} \subseteq s, \text{满足}\sum y_i = t\}$

多项式可验证性

- $HAMPATH = \{\langle G, s, t\rangle | G$是包含从$s$到$t$的哈密尔顿路径的有向图$\}$
- $COMPOSITES = \{x | x = pq,$ 整数$p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G$是包含$k$团的无向图$\}$
- $SUBSET-SUM = \{\langle s, t\rangle | s = \{x_1, \cdots, x_k\}$且存在$\{y_1, \cdots, y_l\} \subseteq s$, 满足$\sum y_i = t\}$

多项式可验证性

- $HAMPATH = \{\langle G, s, t\rangle | G\text{是包含从}s\text{到}t\text{的哈密尔顿路径的有向图}\}$
- $COMPOSITES = \{x | x = pq, \text{整数}p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G\text{是包含}k\text{团的无向图}\}$
- $SUBSET - SUM = \{\langle s, t\rangle | s = \{x_1, \cdots, x_k\}\text{且存在}\{y_1, \cdots, y_l\} \subseteq s, \text{满足}\sum y_i = t\}$

多项式可验证性

- $HAMPATH = \{\langle G, s, t\rangle | G$是包含从$s$到$t$的哈密尔顿路径的有向图$\}$
- $COMPOSITES = \{x | x = pq,$ 整数$p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G$是包含$k$团的无向图$\}$
- $SUBSET-SUM = \{\langle s, t\rangle | s = \{x_1, \cdots, x_k\}$且存在$\{y_1, \cdots, y_l\} \subseteq s,$ 满足$\sum y_i = t\}$

多项式可验证性

- $HAMPATH =$ $\{\langle G, s, t\rangle | G$是包含从$s$到$t$的哈密尔顿路径的有向图$\}$
- $COMPOSITES = \{x | x = pq,$ 整数$p, q > 1\}$
- $CLIQUE = \{\langle G, k\rangle | G$是包含$k$团的无向图$\}$
- $SUBSET - SUM = \{\langle s, t\rangle | s =$ $\{x_1, \cdots, x_k\}$且存在$\{y_1, \cdots, y_l\} \subseteq s,$ 满足$\sum y_i = t\}$

多项式可验证性

定义5

- 语言A的验证机是一个算法V，其中

$$A = \{w | \text{对某个字符串} c, V \text{接受} \langle w, c \rangle\},$$

其中c被称为A的证书。

- 如果语言A有一个多项式时间验证机，则称其是多项式可验证的。
- 具有多项式时间可验证的语言类被称为NP(非确定型多项式时间)。

定义5

- 语言A的验证机是一个算法V，其中

$$A = \{w | \text{对某个字符串}c, V\text{接受}\langle w, c\rangle\},$$

其中c被称为A的证书。

- 如果语言A有一个多项式时间验证机，则称其是多项式可验证的。
- 具有多项式时间可验证的语言类被称为NP(非确定型多项式时间)。

定义5

- 语言A的验证机是一个算法V，其中

$$A = \{w | \text{对某个字符串} c, V \text{接受} \langle w, c \rangle\},$$

其中c被称为A的证书。

- 如果语言A有一个多项式时间验证机，则称其是多项式可验证的。
- 具有多项式时间可验证的语言类被称为NP(非确定型多项式时间)。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取；
2. 检查序列的重复性。如果有重复，则拒绝；
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝；
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取；
2. 检查序列的重复性。如果有重复，则拒绝；
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝；
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取;
2. 检查序列的重复性。如果有重复，则拒绝;
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝;
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取；
2. 检查序列的重复性。如果有重复，则拒绝；
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝；
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取；
2. 检查序列的重复性。如果有重复，则拒绝；
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝；
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

在非确定型多项式时间内判定HAMPATH问题的非确定型图灵机：

对于输入的$\langle G, s, t\rangle$，其中G是m阶有向图

1. 写出m个数构成的序列$p_1, \cdots, p_m$，其中p_i从$\{1, \cdots, m\}$中非确定地选取；
2. 检查序列的重复性。如果有重复，则拒绝；
3. 检查$p_1 = s, p_m = t$是否都成立，否则拒绝；
4. $\forall i \in \{1, \cdots, m-1\}$，检查$(p_i, p_{i+1})$是否是$G$中的边。如果都是，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：
  对于长为n的输入w：
  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c \rangle$上运行V;
  3. 如果V接受，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：

  对于长为n的输入w：

  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c\rangle$上运行V；
  3. 如果V接受，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：
  对于长为n的输入w：

  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c\rangle$上运行V；
  3. 如果V接受，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：
  对于长为n的输入w：
  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c\rangle$上运行V;
  3. 如果V接受，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：
  对于长为n的输入w：
  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c\rangle$上运行V；
  3. 如果V接受，则接受，否则拒绝。

定理5

一个语言在NP中的充要条件是其能够被某个非确定型多项式时间图灵机(NTM)判定。

- 若$A \in NP$，V是A的多项式时间n^k验证机，则判定A的多项式时间NTM构造如下：
  对于长为n的输入w：
  1. 非确定地选择长度不超过n^k的字符串c;
  2. 在输入$\langle w, c \rangle$上运行V；
  3. 如果V接受，则接受，否则拒绝。

假设A被多项式时间NTM N判定，则构造多项式时间验证机V如下：

- 对于输入$\langle w, c\rangle$
  - 在输入w上模拟N，而把c的每个符号看成是对每次非确定选择的描述；
  - 若N的该计算分支接受，则接收，否则拒绝。

假设A被多项式时间NTM N判定，则构造多项式时间验证机V如下：

- 对于输入$\langle w, c\rangle$
  - 在输入w上模拟N，而把c的每个符号看成是对每次非确定选择的描述；
  - 若N的该计算分支接受，则接收，否则拒绝。

假设A被多项式时间NTM N判定，则构造多项式时间验证机V如下：

- 对于输入$\langle w, c\rangle$
  - 在输入w上模拟N，而把c的每个符号看成是对每次非确定选择的描述；
  - 若N的该计算分支接受，则接收，否则拒绝。

假设A被多项式时间NTM N判定，则构造多项式时间验证机V如下：

- 对于输入$\langle w, c\rangle$
  - 在输入w上模拟N，而把c的每个符号看成是对每次非确定选择的描述；
  - 若N的该计算分支接受，则接收，否则拒绝。

- $\text{NTIME}(t(n)) = \{$能被$O(t(n))$时间NTM判定的语言$\}$
- $NP = \bigcup_k \text{NTIME}(n^k)$
- $P \subseteq NP$
- $P \stackrel{?}{=} NP$, 或 $P \stackrel{?}{\subset} NP$

- $\mathrm{NTIME}(t(n)) = \{$能被$O(t(n))$时间NTM判定的语言$\}$
- $NP = \bigcup_k \mathrm{NTIME}(n^k)$
- $P \subseteq NP$
- $P \stackrel{?}{=} NP$, 或 $P \stackrel{?}{\subset} NP$

- $\mathrm{NTIME}(t(n)) = \{$能被$O(t(n))$时间NTM判定的语言$\}$
- $NP = \bigcup_k \mathrm{NTIME}(n^k)$
- $P \subseteq NP$
- $P \stackrel{?}{=} NP$, 或 $P \stackrel{?}{\subset} NP$

- $\text{NTIME}(t(n)) = \{$能被$O(t(n))$时间NTM判定的语言$\}$
- $NP = \bigcup_k \text{NTIME}(n^k)$
- $P \subseteq NP$
- $P \stackrel{?}{=} NP$, 或 $P \stackrel{?}{\subset} NP$

- $\text{NTIME}(t(n)) = \{$能被$O(t(n))$时间NTM判定的语言$\}$
- $NP = \bigcup_k \text{NTIME}(n^k)$
- $P \subseteq NP$
- $P \stackrel{?}{=} NP$, 或 $P \stackrel{?}{\subset} NP$

- $SAT = \{\langle\phi\rangle : \phi$是可满足的布尔公式$\}$
- $3SAT = \{\langle\phi\rangle : \phi$是可满足的3CNF公式$\}$
- 对于语言A, B，若存在多项式时间可计算函数$f : \sum^* \to \sum^*$，使得

$$w \in A \Longleftrightarrow f(w) \in B$$

则称A可以多项式时间映射可归约到B，记为$A \leq_P B$。

- 若$A \leq_P B$且$B \in P$，则$A \in P$

- $SAT = \{\langle\phi\rangle : \phi$是可满足的布尔公式$\}$
- $3SAT = \{\langle\phi\rangle : \phi$是可满足的3CNF公式$\}$
- 对于语言A, B，若存在多项式时间可计算函数$f : \sum^* \to \sum^*$，使得

$$w \in A \Longleftrightarrow f(w) \in B$$

则称A可以多项式时间映射可归约到B，记为$A \leq_P B$。
- 若$A \leq_P B$且$B \in P$，则$A \in P$

- $SAT = \{\langle\phi\rangle : \phi$是可满足的布尔公式$\}$
- $3SAT = \{\langle\phi\rangle : \phi$是可满足的3CNF公式$\}$
- 对于语言A, B，若存在多项式时间可计算函数$f : \sum^* \to \sum^*$，使得

$$w \in A \Longleftrightarrow f(w) \in B$$

则称A可以多项式时间映射可归约到B，记为$A \leq_P B$。

- 若$A \leq_P B$且$B \in P$，则$A \in P$

- $SAT = \{\langle\phi\rangle : \phi$是可满足的布尔公式$\}$
- $3SAT = \{\langle\phi\rangle : \phi$是可满足的3CNF公式$\}$
- 对于语言A, B，若存在多项式时间可计算函数$f : \sum^* \rightarrow \sum^*$，使得

$$w \in A \Longleftrightarrow f(w) \in B$$

则称A可以多项式时间映射可归约到B，记为$A \leq_P B$。

- 若$A \leq_P B$且$B \in P$，则$A \in P$

- $SAT = \{\langle\phi\rangle : \phi$是可满足的布尔公式$\}$
- $3SAT = \{\langle\phi\rangle : \phi$是可满足的3CNF公式$\}$
- 对于语言A, B，若存在多项式时间可计算函数$f : \sum^* \to \sum^*$，使得

$$w \in A \Longleftrightarrow f(w) \in B$$

则称A可以多项式时间映射可归约到B，记为$A \leq_P B$。

- 若$A \leq_P B$且$B \in P$，则$A \in P$

定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

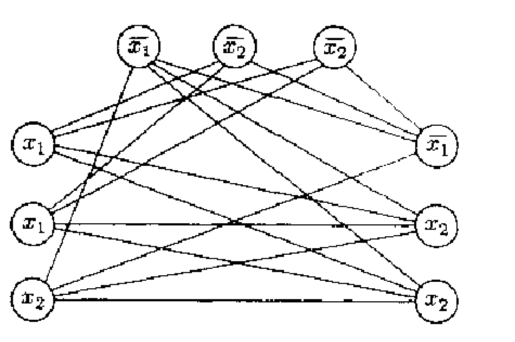
定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

定理6 ($3SAT \leq_q CLIQUE$)

- $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$
- 点：对应于每个变量（或其否定）
- 边：连接所有的点对，除非
  - 同一个三元组内的节点无边相连
  - 相反标记的两个节点无边相连

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



定义6 (NP完全)

若语言B满足

- $B \in NP$
- NP中每个A都可以多项式时间归约到B

则称B是NP完全的。

- 若B是NP完全的且$B \in P$，则$P = NP$
- 若B是NP完全的，且$B \leq_P C$，其中$C \in NP$，则C是NP完全的。

定义6 (NP完全)

若语言B满足

- $B \in NP$
- NP中每个A都可以多项式时间归约到B

则称B是NP完全的。

- 若B是NP完全的且$B \in P$，则$P = NP$
- 若B是NP完全的，且$B \leq_P C$，其中$C \in NP$，则C是NP完全的。

定义6 (NP完全)

若语言B满足

- $B \in NP$
- NP中每个A都可以多项式时间归约到B

则称B是NP完全的。

- 若B是NP完全的且$B \in P$，则$P = NP$
- 若B是NP完全的，且$B \leq_P C$，其中$C \in NP$，则C是NP完全的。

定理7 (库克-列文定理：SAT是NP完全的)

- $SAT \in NP$;
- 设A是任意NP问题，N是在时间n^k内判定A的非确定型图灵机。
- 画面：$n^k \times n^k$表，其中每行表示N在输入w上的某个计算分支的格局

#	q_0	w_1	$\cdots$	w_n	␣	$\cdots$	␣	#	起始格局
#								#	第2个格局
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	第$*$个格局
#								#	第n^k个格局

定理7 (库克-列文定理：SAT是NP完全的)

- $SAT \in NP$;
- 设A是任意NP问题，N是在时间n^k内判定A的非确定型图灵机。
- 画面：$n^k \times n^k$表，其中每行表示N在输入w上的某个计算分支的格局

$\#$	q_0	w_1	$\cdots$	w_n	$\sqcup$	$\cdots$	$\sqcup$	$\#$	起始格局
$\#$								$\#$	第2个格局
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	第$*$个格局
$\#$								$\#$	第n^k个格局

定理7 (库克-列文定理：SAT是NP完全的)

- $SAT \in NP$;
- 设A是任意NP问题，N是在时间n^k内判定A的非确定型图灵机。
- 画面：$n^k \times n^k$表，其中每行表示N在输入w上的某个计算分支的格局

#	q_0	w_1	$\cdots$	w_n	␣	$\cdots$	␣	#	起始格局
#								#	第2个格局
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	第$*$个格局
#								#	第n^k个格局

定理7 (库克-列文定理：SAT是NP完全的)

- $SAT \in NP$;
- 设A是任意NP问题，N是在时间n^k内判定A的非确定型图灵机。
- 画面：$n^k \times n^k$表，其中每行表示N在输入w上的某个计算分支的格局

$\#$	q_0	w_1	$\cdots$	w_n	$\sqcup$	$\cdots$	$\sqcup$	$\#$	起始格局
$\#$								$\#$	第2个格局
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	第$*$个格局
$\#$								$\#$	第n^k个格局

定理7 (库克-列文定理：SAT是NP完全的)

- $SAT \in NP$;
- 设A是任意NP问题，N是在时间n^k内判定A的非确定型图灵机。
- 画面：$n^k \times n^k$表，其中每行表示N在输入w上的某个计算分支的格局

#	q_0	w_1	$\cdots$	w_n	␣	$\cdots$	␣	#	起始格局
#								#	第2个格局
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	第$*$个格局
#								#	第n^k个格局

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \iff$单元$cell[i,j]$包含s
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量:
    - 对每个$i, j, 1 \le i, j \le n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \iff$单元$cell[i,j]$包含s
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化:

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f: 令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量:
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

A到SAT的约化：

- N在w上的每个对应画面对应N在w上的一个计算分支
- N是否接受w等价于N在w上是否有接受画面(即某行是接受格局)
- 归约函数f：令$C = Q \cup \Gamma \cup \{\#\}$,其中$Q, \Gamma$分别表示状态集和带字母表
  - 变量：
    - 对每个$i, j, 1 \leq i, j \leq n^k$以及每个$s \in C$，定义变量$x_{i,j,s}$
    - $x_{i,j,s} = 1 \Longleftrightarrow$单元$cell[i, j]$包含$s$
  - 公式$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{move} \wedge \phi_{accept}$

- ϕ_{cell}： $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\ne t}\left(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}}\right)\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}$, $s\in C$中有且仅有一个是1
- ϕ_{start}： $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}： $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}: $\bigwedge\limits_{1\le i,j\le n^k}\left[\left(\bigvee\limits_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge\limits_{s\ne t}\left(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}}\right)\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}$, $s\in C$中有且仅有一个是1
- ϕ_{start}: $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}: $\bigvee\limits_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}: $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}$, $s\in C$中有且仅有一个是1
- ϕ_{start}: $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}: $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}: $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\ne t}\left(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}}\right)\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}, s\in C$中有且仅有一个是1
- ϕ_{start}: $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}: $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}： $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\ne t}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}$, $s\in C$中有且仅有一个是1
- ϕ_{start}： $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}： $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}：$\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\ne t}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}, s\in C$中有且仅有一个是1
- ϕ_{start}：$x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}：$\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}：$\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}, s\in C$中有且仅有一个是1
- ϕ_{start}：$x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}：$\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}: $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t}\left(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}}\right)\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}$, $s\in C$中有且仅有一个是1
- ϕ_{start}: $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}: $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{cell}: $\bigwedge_{1\le i,j\le n^k}\left[\left(\bigvee_{s\in C} x_{i,j,s}\right)\wedge\left(\bigwedge_{s\neq t}(\overline{x_{i,j,s}}\vee\overline{x_{i,j,t}})\right)\right]$
  - 保证对于每个单元$cell[i,j]$，在所有$s_{i,j,s}, s\in C$中有且仅有一个是1
- ϕ_{start}: $x_{1,1,\#}\wedge x_{1,2,q_0}\wedge x_{1,3,w_1}\wedge\cdots\wedge x_{1,n+2,w_n}\wedge x_{1,n+3,\sqcup}\wedge\cdots\wedge x_{1,n^k-1,\sqcup}\wedge x_{1,n^k,\#}$
  - 保证第一行是N在w上的起始格局
- ϕ_{accept}: $\bigvee_{1\le i,j\le n^k} x_{i,j,q_{accept}}$
  - 保证接受格局出现在画面中

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a,b,c，$\delta(q_1,a)=\{(q_1,b,R)\},\delta(q_1,b)=\{(q_2,c,L),(q_2,a,R)\}$
-

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

-

a	b	a
a	b	q_2

b	b	b
c	b	b

a	b	a
a	a	a

a	q_1	b
q_1	a	a

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a,b,c，$\delta(q_1,a)=\{(q_1,b,R)\},\delta(q_1,b)=\{(q_2,c,L),(q_2,a,R)\}$
-

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

$\#$	b	a
$\#$	b	a

-

a	b	a
a	b	q_2

b	b	b
c	b	b

a	b	a
a	a	a

a	q_1	b
q_1	a	a

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a, b, c，$\delta(q_1, a) = \{(q_1, b, R)\}, \delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$
-

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

-

a	b	a
a	b	q_2

b	b	b
c	b	b

a	b	a
a	a	a

a	q_1	b
q_1	a	a

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a, b, c，$\delta(q_1, a) = \{(q_1, b, R)\}, \delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	b	b
c	b	b

a	b	a
a	a	a

a	q_1	b
q_1	a	a

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a,b,c，$\delta(q_1,a)=\{(q_1,b,R)\},\delta(q_1,b)=\{(q_2,c,L),(q_2,a,R)\}$
-

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

$\#$	b	a
$\#$	b	a

-

a	b	a
a	b	q_2

b	b	b
c	b	b

a	b	a
a	a	a

a	q_1	b
q_1	a	a

- ϕ_{move}：保证每一行都是由上一行按照规则合法转移得到的格局
- 手段：合法2×3窗口
- 例：设字母表为a,b,c，$\delta(q_1,a)=\{(q_1,b,R)\},\delta(q_1,b)=\{(q_2,c,L),(q_2,a,R)\}$
- $\begin{array}{ccc} a & q_1 & b \\ q_2 & a & c \end{array}\quad \begin{array}{ccc} a & q_1 & b \\ a & a & q_2 \end{array}\quad \begin{array}{ccc} a & a & q_1 \\ a & a & b \end{array}\quad \begin{array}{ccc} \# & b & a \\ \# & b & a \end{array}$
- $\begin{array}{ccc} a & b & a \\ a & b & q_2 \end{array}\quad \begin{array}{ccc} b & b & b \\ c & b & b \end{array}\quad \begin{array}{ccc} a & b & a \\ a & a & a \end{array}\quad \begin{array}{ccc} a & q_1 & b \\ q_1 & a & a \end{array}$

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \leq i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \le i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \le i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \le i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \leq i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1\le i<n^k, 1<j<n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- 如果顶行是起始格局，表中每个窗口都是合法的，那么每一行都是由上一行合法转移得到。
- $\phi_{move} = \bigwedge_{1 \leq i < n^k, 1 < j < n^k} ((i,j)\text{窗口都是合法的})$

$$\text{窗口合法} = X_{i,j-1,a_1} \wedge X_{i,j,a_2} \wedge \cdots \wedge X_{i+1,j+2,a_6}$$

- 变量数目：$O(n^{2k})$
  - 单元数n^{2k}
  - 单元关联的变量数为$|C|$，与输入字符串长度无关

- ϕ_{cell}: $O(n^{2k})$
- ϕ_{start}: $O(n^{k})$
- ϕ_{move}: $O(n^{2k})$
- ϕ_{accept}: $O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^{k})$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^{k})$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^k)$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^k)$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^k)$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

- ϕ_{cell}:$O(n^{2k})$
- ϕ_{start}:$O(n^{k})$
- ϕ_{move}:$O(n^{2k})$
- ϕ_{accept}:$O(n^{2k})$

推论1

- *3SAT是NP完全的;*
- *CLIQUE是NP完全的;*

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$:
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i:

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$:
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i:

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$：
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i：
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1；当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P SUBSET-SUM$：
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i：
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1;
      当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$：
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i：
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1；当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$:
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i:
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1; 当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$：
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i：
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1;
      当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

定理8 (SUBSET-SUM是NP完全的)

证明.

- SUBSET-SUM∈NP;
- 3SAT$\leq_P$ $SUBSET-SUM$:
  - 设$\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_k$，$\phi$中的变量为$x_1, \cdots, x_l$，其中$c_i$为初等和
  - 变量x_i对应一对数y_i, z_i:
    - y_i, z_i十进制的左边部分由1和随后的$l-i$个0组成
    - 右边部分：当初等和c_j包含x_i时，y_i的第j位是1;
      当初等和c_j包含$\overline{x_i}$时，z_i的第j位是1
    - 其余位记为0

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$，其对应的$y_i, z_i$为

- $y_1 = 100010$
- $z_1 = 100000$
- $y_2 = 10001$
- $z_2 = 10000$
- $y_3 = 1000$
- $z_3 = 1011$
- $y_4 = 100$
- $z_4 = 110$

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$： $g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$： $t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$： $g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$： $t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$： $g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$： $t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow$ S的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$： $g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$： $t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow$ S的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow$ S的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow S$的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

- 对每个初等和c_j，取一对数g_i, h_i：
  - $g_i = h_i = 10^{k-j}$：　$g_1 = h_1 = 10, g_2 = h_2 = 1$
- $S = \{y_1, z_1, \cdots, y_l, z_l, g_1, h_1, \cdots, g_k, h_k\}$
- $t = 1^l 3^k$：　$t = 100033$

ϕ可满足$\Longrightarrow$ S的某个子集和为t

- 在ϕ的成真赋值中，
  - 若$x_i = 1$，则取y_i
  - 若$x_i = 0$，则取z_i
- 所选数之和的前l位的每一位是1
- 所选数之和的后k位为1，2，3；
- 取适当的g, h使得后k位的每一位都是3

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$的成真解释$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$有：

- 取$z_1 = 100000, y_2 = 10001, z_3 = 1011, y_4 = 100$
- $z_1 + y_2 + z_3 + y_4 = 111112$
- 取g_1, h_1, g_2
- $z_1 + y_2 + z_3 + y_4 + g_1 + h_1 + g_2 = 111133 = t$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$的成真解释$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$有：

- 取$z_1 = 100000, y_2 = 10001, z_3 = 1011, y_4 = 100$
- $z_1 + y_2 + z_3 + y_4 = 111112$
- 取g_1, h_1, g_2
- $z_1 + y_2 + z_3 + y_4 + g_1 + h_1 + g_2 = 111133 = t$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$的成真解释$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$有：

- 取$z_1 = 100000, y_2 = 10001, z_3 = 1011, y_4 = 100$
- $z_1 + y_2 + z_3 + y_4 = 111112$
- 取g_1, h_1, g_2
- $z_1 + y_2 + z_3 + y_4 + g_1 + h_1 + g_2 = 111133 = t$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$的成真解释$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$有：

- 取$z_1 = 100000, y_2 = 10001, z_3 = 1011, y_4 = 100$
- $z_1 + y_2 + z_3 + y_4 = 111112$
- 取g_1, h_1, g_2
- $z_1 + y_2 + z_3 + y_4 + g_1 + h_1 + g_2 = 111133 = t$

对于$\phi = (x_1 \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2 \vee x_2 \vee \overline{x_3})$的成真解释$x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$有：

- 取$z_1 = 100000, y_2 = 10001, z_3 = 1011, y_4 = 100$
- $z_1 + y_2 + z_3 + y_4 = 111112$
- 取g_1, h_1, g_2
- $z_1 + y_2 + z_3 + y_4 + g_1 + h_1 + g_2 = 111133 = t$

如果S的某个子集(T)和等于t:

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$

如果S的某个子集(T)和等于t：

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$

如果S的某个子集(T)和等于t:

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$

如果S的某个子集(T)和等于t:

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$

如果S的某个子集(T)和等于t：

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$

如果S的某个子集(T)和等于t：

- 如果$y_i \in T$，则令$x_i = 1$，否则令$x_i = 0$
- 由于后k列的每一列和为3，从而子集中的y_i或z_i必定至少提供1
- 如果是y_i，则x_i出现在c_j中且赋值是1，从而c_j被满足
- 如果是z_i，则$\overline{x_i}$出现在c_j中且x_i赋值为0，从而c_j满足
- 计算量$O(n^2)$