| | | | | |
|---|---|---|---|---|
| 128 | 191 | 167 | 3,4 | 12,6 |
| 256 | 383 | 325 | 4 | 15,1 |
| 512 | 767 | 635 | 5 | 17,2 |
| 1024 | 1535 | 1246 | 5 | 18,8 |
| 2048 | 3071 | 2439 | 6 | 20,6 |

$m$-арлы әдіспен берілген үнемдеудің асимптотикалық мәні 33% болады. Бұл тұжырымды дәлелдеу үшін қатынас шегін есептейміз:

$$\lim_{k\to\infty}\frac{2^r-2+k-r+\left(\frac{k}{r}-1\right)\cdot\left(1-2^{-r}\right)}{\frac{3}{2}\cdot(k-1)}=\frac{2}{3}\left(1+\frac{1-2^{-r}}{r}\right)\approx\frac{2}{3}.$$

## 3.11. Адаптивті $m$ -арлы әдісі

Адаптивті әдістер – есептеу процесін кірістегі ақпаратқа тәуелді түрде құрады. Дәрежеге шығарған жағдайда адаптивті әдіс $e$ көрсеткішінің мәніне тәуелді түрде құрылымын өзгертеді. Жоғарыда айтылғандай, егер $e$ көрсеткішін бөлгенде бит-секциялардың арасында $\omega$ барлық мүмкін мәндері болмаса алдын ала есептеу қадамында көбейтулер саны азайды. Сонымен бірге адаптивті алгоритмдер бар, оларда $m$-арлы әдістегі 4b-қадамдағы көбейту амалдарының санын азайтуға нөлдік және нөлдік емес сөздерді қатарға бөлу жүргізіледі.

### 3.11.1. Алдын ала есептеу қадамындағы көбейту амалдарының санын азайту

Көрсеткіштің екілік берілуі алынғаннан кейін оны әр $d$ бит бойынша топтарға бөлеміз. Кейін есептеулер жүргізіп, екілік бөлуде пайда болатын $\omega$ үшін $M^{\omega}(\mathrm{mod}\,n)$-ді аламыз. $k=16$ және $d=4$ үшін келесі көрсеткішті қарастырайық:

$$\underline{1011}\,\underline{0011}\,\underline{0111}\,\underline{1000}.$$

Тек қана келесі түрде алынатын $\omega = 3,7,8,11$ үшін ғана $M^{\omega}(\mathrm{mod}\, n)$-ді есептеу қажеттігін атап өту керек:

$$M^2 = M \cdot M$$
$$M^3 = M^2 \cdot M$$
$$M^4 = M^2 \cdot M^2$$
$$M^7 = M^3 \cdot M^4$$
$$M^8 = M^4 \cdot M^4$$
$$M^{11} = M^8 \cdot M^3$$

Ол үшін 6 көбейту амалы қажет. Алдын ала есептеу қадамында барлық $\omega$ мәндері үшін дәрежелерді есептейтін $m$-арлы әдіс 16-2=14 көбейту амалын қажет етеді. $m$-арлы әдісте үнемдеуге болатын көбейту амалдарының саны жоғарыдан $m-2=2^d-2$ шамасымен шектелген, ол барлық бит-секциялар бірліктерден тұрғанда орын алады: $\underline{0001}\ \underline{0001}\ \underline{0001}\ \underline{0001}$.

Бұл біз ешнәрсені алдын ала есептемей, $M$-ді қолданатынымызды білдіреді. Негізінен барлық $\omega = \omega_0, \omega_1, ..., \omega_{p-1}$ үшін $M^{\omega}(\mathrm{mod}\, n)$-ді есептеуіміз қажет. Егер $\{\omega_i | i = 0,1,..., p-1\}$ жиынының түрлі элементтері $2,3,...,2^d-1$-ң барлық мүмкін мәндерін қабылдайтын болса, онда үнемдеу мүмкін емес. $2^d-2$ көбейту амалдарын орындап, осы мәндерді аламыз. Егер мұндай жиынтық- $2,3,...,2^d-1$ мәндерінің ішкі жиындары болса, онда кейбір үнемдеулерге қол жеткізуге болады, егер $\omega_i$-ді $(i = 0,1,..., p-1)$ $2^d-2$-ден аз көбейту амалдарына қолдансақ, онда есептеуге мүмкіндік болады.

Кез келген $p$ мәліметтерінің экспонентін есептеу алгоритмі векторлық аддитивті тізбек деп аталады, ал $p=1$ жағдайында аддитивті тізбек деп аталады. Минималды ұзындықты аддитивті тізбекті алу тапсырмасы NP-толық болып табылады.

## 3.11.2. Жылжитын терезелер тәсілі

$m$-арлы әдіс көрсеткіш биттерін $d$-биттік сөздерге бөледі. $d$ сөзінің ұзындығы нөлдік екендігінің ықтималдығы 0 және 1 бірдей ықтималдықты деп алғанда $2^{-d}$-ға тең. $m$-арлы әдістің 4b қадамында көбейту амалы жүргізілмейді. $d$ мәнінің өсуімен бірге көбейту

амалынан босатылу ықтималдығы жоғарылайды. Бірақ көбейтулер $d$ азайған сайын өседі. Жылжитын терезелер алгоритмі белгілі бір жағдайларды қамтамасыз етеді, бұл стратегия нөлдік сөздерді көбейтіп, бір уақытта $d$ үшін жоғары мәндерді қолдануды көздейді. Жылжитын терезелер экспонентін есептеу алгоритмі бірінші $e$ көрсеткішінің декомпозициясын $L(F_i)$ ұзындығы бар $F_i$ нөлдік және нөлдік емес сөзіне жүзеге асырады. $p$ терезелер саны $k/d$-ға тең болуы мүмкін. Жалпы жағдайда, терезелер бірдей ұзындықта болуы шарт емес. $d$ есебінде ең ұзын терезе ұзындығын аламыз, яғни $i = 0,1,...,k-1$ үшін $d = \max(L(F_i))$. Бірақ $F_i$-кез келген нөлдік емес терезе болса, $F_i$ терезесінің ең кіші биті 1-ге тең болуы керек, себебі көрсеткішті ең кіші биттен бастап бөлеміз. Сәйкесінше, алдын ала есептеулерде көбейту амалдарының саны шамамен жартысына дейін қысқарады, себебі $x^{\omega}$ дәрежелері $\omega$-ның тақ көрсеткіштері үшін есептеледі.

*Жылжитын терезелер әдісі*

Кірісінде: $M, e, n$

Шығысында: $C = M^e \pmod n$

1. Барлық $\omega = 3,5,7,...,2^d - 1$ үшін $M^{\omega} \pmod n$ есептеп, сақтау керек
2. $i = 0,1,2,..., p-1$ үшін $L(F_i)$ ұзындықты $F_i$ нөлдік және нөлдік емес терезесіне $e$-ны тарату.
3. $C := M^{F_{k-1}} \pmod n$
4. 0-ге дейін $i = p-2$ үшін

4a. $C := C^{2^{L(F_i)}} \pmod n$

4b. Егер $F_i \neq 0$ болса, онда $C := C \cdot M^{F_i} \pmod n$

5. $C$ мәнін қайтару.

## 3.11.3 Ұзындықтары бірдей нөлдік емес терезелер тәсілі (CLNW)

CLNW тәсілінің бөлу алгоритмін Кнут ұсынды [15]. Ол көрсеткіш биттерін кішісінен үлкеніне көшіреді. Кез келген уақытта алгоритм не нөлдік терезені (ZW), немесе нөлдік емес терезені (NW) қалыптастырады. Алгоритм төменде келтірілген:

ZW: кірістегі бірлік битті тексереміз: егер ол 0 болса, онда ZW қадамында қаламыз, әйтпесе NW қадамына барамыз.

NW: NW қадамында $d$ биттерді жинағанша қаламыз. Одан кейін кірістегі бір битті тексереміз: егер ол 0-ге тең болса, онда ZW қадамына, әйтпесе NW қадамына барамыз.

NW қадамында NW қадамына ену және шығуды айтып өткен жөн. Алғашқысында біз сол нөлдік терезені қалыптастыруды жалғастырамыз, ал екіншісінде бұл жаңа нөлдік емес терезені білдіреді. Ұзындықтары бірдей нөлдік емес терезелер алгоритмі еркін ұзындықтағы нөлдік терезелерді, ал $d$ ұзындықтағы нөлдік емес терезені жүргізеді. Екі аралас нөлдік терезе болмайды, олар міндетті түрде ілігеді, ал нөлдік емес терезелер көршілес бола алады.

Мысалы, $d = 3$ үшін $e = 3665 = (111001010001)$ былай бөлінеді:

$$e = 111\ 00\ 101\ 0\ 001.$$

Ұзындықтары бірдей нөлдік емес терезелер алгоритмі алдын ала есептеулерде көбейту амалдарын орындайды және $\omega = 3,5,7$ үшін $M^{\omega} (\mathrm{mod}\, n)$ аламыз.

| Разрядтар | $\omega$ | $M^{\omega}$ |
|---|---|---|
| 001 | 1 | $M$ |
| 010 | 2 | $M \cdot M = M^2$ |
| 011 | 3 | $M \cdot M^2 = M^3$ |
| 101 | 5 | $M^3 \cdot M^2 = M^5$ |
| 111 | 7 | $M^5 \cdot M^2 = M^7$ |

Алгоритм $C = M^{F_4} = M^7 (\mathrm{mod}\, n)$ меншіктейді және келесі түрде $M^{3665} (\mathrm{mod}\, n)$ алынғанға дейін орындалады:

| $i$ | $F_i$ | $L(F_i)$ | 4a-қадам | 4b-қадам |
|---|---|---|---|---|
| 3 | 00 | 2 | $(M^7)^4 = M^{28}$ | $M^{28}$ |
| 2 | 101 | 3 | $(M^{28})^8 = M^{224}$ | $M^{224} \cdot M^5$ |
| 1 | 0 | 1 | $(M^{229})^2 = M^{458}$ | $M^{458}$ |
| 0 | 001 | 3 | $(M^{458})^8 = M^{3664}$ | $M^{3664} \cdot M = M^{3665}$ |

Сәйкесінше, модуль бойынша 4+9+2=15 көбейту амалдары орындалады. Көбейту амалдарының орташа санын Марков тізбегі

көмегімен терезелерді бөлу процесін модельдей отырып табуға болады. Мұндай талдаулар [15]-те келтірілген. Төмендегі кестеде $m$-арлы және ұзындықтары бірдей нөлдік емес терезелер алгоритмі үшін көбейту амалдарының орташа саны келтірілген. $m$-арлы әдіс үшін бағанда әр $k$ үшін $d^*$ тиімді мәндері келтірілген. Ұзындықтары бірдей нөлдік емес терезелер алгоритмінде әр $k$ үшін $d$ тиімді мәндері бар. Олар да кестеге қосылған. Кесте бағанының соңғысында көбейту амалдарының орта санының айырмашылық пайызы көрсетілген. Көрсеткішті бөлудің ұзындықтары бірдей нөлдік емес терезелер алгоритмі көбейту амалдарының орташа санын $128 \le k \le 2048$ сандары үшін 3,7%-ға азайтады. Көрсеткішті бөлуге кеткен уақыт есепке алынбайды, бөлуге қажетті бит-амалдар саны $k$ санына пропорционал.

| | $m$-арлы | | CLNW | | $\frac{T-T_1}{T}$ |
|---|---|---|---|---|---|
| $k$ | $d^*$ | $T$ | $d^*$ | $T_1$ | % |
| 128 | 4 | 168 | 4 | 156 | 7.14 |
| 256 | 4 | 326 | 5 | 308 | 5.52 |
| 512 | 5 | 636 | 5 | 607 | 4.56 |
| 768 | 5 | 941 | 6 | 903 | 4.04 |
| 1024 | 5 | 1247 | 6 | 1195 | 4.17 |
| 1280 | 6 | 1546 | 6 | 1488 | 3.75 |
| 1536 | 6 | 1844 | 6 | 1780 | 3.47 |
| 1792 | 6 | 2142 | 7 | 2072 | 3.27 |
| 2048 | 6 | 2440 | 7 | 2360 | 3.28 |

## 3.11.4. Түрлі ұзындықтағы нөлдік емес терезелер тәсілі (VLNW)

CLNW бөлу стратегиясы 1 саны кездескенде нөлдік емес терезені анықтай бастайды. Кірістегі барлық $d-1$ биттер нөлге тең болуы мүмкін, алгоритм оларды ағымдағы нөлдік емес терезеге қосуды жалғастырады. Мысалы, $d=3$, $e=(111001010001)$ көрсеткіші

$$e = \underline{111}\,00\ \underline{101}\,0\ \underline{001}$$

ретінде орналасады. Бірақ түрлі ұзындықтағы нөлдік емес терезелер болса, онда бұл санды былай бөлуімізге болады:

$$e = \underline{111}\,00\ \underline{101}\,000\ \underline{1}.$$

Бұл нөлдік емес терезелердің орта санын азайтатынын көрсетеміз. Түрлі ұзындықтағы нөлдік емес терезелер стратегиясын Bos және Coster ұсынды. Ол нөлдік емес терезені (NW) қалыптастыру кезінде нөлдік терезесіне көшуімізді талап етеді. Түрлі ұзындықтағы нөлдік емес терезелер стратегиясының екі бүтін параметрі бар:

- $d$ - нөлдік емес терезенің ең үлкен ұзындығы;
- $q$ - нөлдік терезені қосу үшін қажетті нольдердің ең кіші саны.

Алгоритм келесі түрде орындалады:

ZW: бір битті кірісінде тексереміз: егер ол нөлдік болса, онда ZW қадамында қаламыз, қарсы жағдайда NW қадамына барамыз.

NW: $q$ биттерді кірісінде тексереміз: егер олардың бәрі нөль болса, онда нөлдік терезе қадамына көшеміз, қарсы жағдайда нөлдік емес терезе қадамында қаламыз. $d = lq + r + 1$ болсын, мұндағы $1 < r \le q$. Нөлдік емес терезе қадамында $lq + 1$ битке жеткенше дейін қаламыз. Соңғы қадамда кірістегі биттер саны $r$ болады. Егер олардың барлығы ноль болса, онда нөлдік терезе қадамына көшеміз, әйтпесе нөлдік емес терезе қадамында қаламыз. Барлық $d$ биттер жиналған соң, кірістегі битті тексереміз: егер ол нольге тең болса, нөлдік терезе қадамына, әйтпесе нөлдік емес терезе қадамына көшеміз.

VLNW бөлу процедурасы 1 санынан басталып, 1 санымен аяқталатын нөлдік емес терезелерді береді. Екі нөлдік емес терезе көршілес бола алады; бірақ нөлдік емес терезелердің міндетті түрде $d$ биттері болады. Екі нөлдік терезелер көршілес бола алмайды. Мысалы, $d = 5$ және $q = 2$ болсын. Онда $5 = 1 + 1 \cdot 2 + 2$ болады, сәйкесінше $l = 1$, ал $r = 2$. Осы берілгендерге сәйкес бөлу былай болады:

$$\underline{101}\,0\ \underline{11101}\,00\ \underline{101}\,\underline{10111}\,000000\ \underline{1}\,00\ \underline{111}\,000\ \underline{1011}$$

Енді $d = 10$, $q = 4$ болсын, онда $l = 2$, $r = 1$. Бұдан:

$$\underline{1011011}0000\ \underline{11}\,0000\ \underline{1111011}00\ \underline{11111101}0000\ \underline{11011}.$$

Көбейту амалдарының орта санын есептеу үшін түрлі ұзындықтағы нөлдік емес терезелер процесі ұзындықтары бірдей нөлдік емес терезелер алгоритмі процесі сияқты Марков тізбегі көмегімен модельдене алады. Бұл талдау [8-18] келтіріліп, $128 \le k \le 2048$ үшін көбейту амалдарының орта саны есептелді. Келесі кестеде бұл мәндер $d$ және $q$ тиімді мәндерімен бірге келтірілген, және оларды $m$-арлы әдістегі көбейту амалдары, сонымен бірге $d$ тиімді мәнімен салыстырамыз. Тәжірибелердің көрсетулері бойынша $128 \le k \le 2048$ және $4 \le d \le 8$ үшін 1 және 3 аралығында $q$-дің ең жақсы мәндері орналасады**.** Түрлі ұзындықтағы нөлдік емес терезелер алгоритмі $m$-арлы әдіске қарағанда 5,8%-ға аз көбейту амалын талап етеді.

Жылжымалы терезелері бар алгоритмдерді программалау оңай. Көбейтулердің санын азайту мұнда басымырақ, мысалы, $n = 612$ үшін $m$-арлы алгоритм 636 көбейтуді қажет етеді, бұл кезде CLNW және VLNW алгоритмдері сәйкесінше 607 және 595 көбейтуді қажет етеді.

| | $m$-арлы | VLNW $T_2 / k$ | $q^*$ үшін $\frac{T_2 - T}{T_2}$ |
|---|---|---|---|
| $k$ | $d^*$ $T / k$ | $d^*$ $q=1$ $q=2$ $q=3$ | % |
| 128 | 4 1.305 | 4 1.204 1.203 1.228 | 7.82 |
| 256 | 4 1.270 | 4 1.184 1.185 1.212 | 6.77 |
| 512 | 5 1.241 | 5 1.163 1.175 1.162 | 6.37 |
| 768 | 5 1.225 | 5 1.155 1.167 1.154 | 5.80 |
| 1024 | 5 1.217 | 6 1.148 1.146 1.157 | 5.83 |
| 1280 | 6 1.207 | 6 1.142 1.140 1.152 | 5.55 |
| 1536 | 6 1.200 | 6 1.138 1.136 1.148 | 5.33 |
| 1792 | 6 1.195 | 6 1.136 1.134 1.146 | 5.10 |
| 2048 | 6 1.191 | 6 1.134 1.132 1.144 | 4.95 |

## 3.12. Фактор-әдіс

Фактор-әдісті Кнут [6] ұсынды. Ол $e$ және $s > 1$-тің ең кіші қарапайым көбейткіші $r$ болып табылатын $e = r \cdot s$ көрсеткішін факторизациялауға негізделген. $M^e$-ны есептейміз, алдымен $M^r$ есептеп, кейін алынған мәнді $s$-ші дәрежеге шығарып аламыз:

$$C_1 = M^r,$$

$$C_2 = C_1^{\,s} = M^{rs} = M^e.$$

Егер $e$ жай сан болса, онда алдымен $M^{e-1}$-ді есептейміз, кейін осы шаманы $M$-ге көбейтеміз. Бұл алгоритм рекурсивті болып табылады. $M^r$-ді есептеу үшін $r_1$ $r$-дің ең кіші жай көбейткіші, ал $r_2 > 1$ болатындай етіп $r = r_1 \cdot r_2$ факторизациялаймыз. Бұл процесс көрсеткіш мәні 2 санына тең болғанға дейін созылады. Мысал ретінде $e = 55 = 5 \cdot 11$ үшін $M^e$-ні келесі түрде көрсетейік:

Есептеу: $M \to M^2 \to M^4 \to M^5$
Меншіктеу: $a := M^5$
Есептеу: $a \to a^2$
Меншіктеу: $b := a^2$
Есептеу: $b \to b^2 \to b^4 \to b^5$
Есептеу: $b^5 \to b^5 \cdot a = M^{55}$.

Фактор-әдіс $M^{55}$ есептеу үшін 8 көбейтуді қажет етеді. Бұл кезде бинарлық әдіске 9 көбейту керек, себебі $e = 55 = (110111)$ үшін 5 квадратқа шығару мен 4 көбейту орындалады.

Өкінішке орай, фактор әдіс көптеген сандар үшін қиындық туғызатын көрсеткіштерді факторизациялауды талап етеді. Сәйкесінше бұл әдісті RSA криптожүйелерінде көрсеткіш мәні аз болғанда, тіпті көрсеткіш оңай факторизацияланатын болса да қолдануға болады.

## 3.13. Қайта кодтау әдістері

Бұл алгоритмнің ерекшелігі - $M^e \pmod n$ тиімді есептеу үшін $n$ модуль бойынша $M$ кері мәнін талап етеді. $k-1$-бұл $M^e$-ні есептеуге қажетті квадратқа шығару санының төменгі шегі, мұндағы $k$-$e$-нің берілуіне қажетті биттер саны. Бірақ көрсеткіштерді қайта кодтау жолымен көбейту амалдарының санын азайтуға болады. Қайта кодтау тәсілдері

$$2^{i+j-1} + 2^{i+j-2} + \ldots + 2^i = 2^{i+j} - 2^i$$

теңдігін бірліктердің блогын бұзып, көрсеткіштің сиретілген мәнін алу үшін қолданады. Бірақ көрсеткішті жазу үшін $\{0,1,-1\}$ цифрларының көп мөлшері қажет болады. Мысалы, $(011110)$ былай жазуға болады:

$$(011110) = 2^4 + 2^3 + 2^2 + 2^1$$

$$(100010) = 2^5 - 2^1.$$

Қайта кодталған көрсеткішті алғаннан кейін $M^e (\mathrm{mod} n)$-ні есептеу үшін бинарлық әдісті кірісінде $M$ және $M^{-1} (\mathrm{mod} n)$-ді қамтамасыз ете отыра қолданамыз. Мысалы, қайта кодтау әдісі бар бинарлық әдіс былай болады:

*Қайта кодтауы бар бинарлық әдіс*
Кірісінде: $M, M^{-1}, e, n$
Шығысында: $C := M^e \bmod n$
0. $e$-нің қайта кодталып берілуі-$f$-ті алу
1. егер $f_k = 1$ болса, онда $C := M$, әйтпесе $C := 1$
2. $i = k-1$-ден бастап, 0-ге дейін
2a. $C := C \cdot C (\mathrm{mod} n)$
2b. егер $f_i = 1$ болса, $C := C \cdot M^{-1} (\mathrm{mod} n)$
3. $C$-ны қайтару.

$e$-ң биттер саны $k$ болса, $f$-ң биттер саны $k+1$ болуы мүмкін. Мысалы, $(111)$ коды $(100\bar{1})$-ге қайта кодталады. Осылайша, қайта кодтауы бар бинарлық әдіс құрамындағы $f$-$k+1$-$e$ көрсеткішінің биттік қайта кодталуы болып табылатын $M^f (\mathrm{mod} n)$ есебі көмегімен $M^e (\mathrm{mod} n)$ есептеу үшін $k$-бинарлық позициясынан басталады. Мысал қарастырайық. $e = 119 = (1110111)$ болсын. Қарапайым бинарлық әдіс $M^{119} (\mathrm{mod} n)$-ді есептеу үшін $6+5=11$ көбейтуді қажет етеді. Қайта кодтауы бар бинарлық әдісте алдымен 119 санын қайта кодтаймыз:

Көрсеткіш: $119 = 01110111$,
Қайта кодталған көрсеткіш: $119 = 10001001$.
Қайта кодтауы бар бинарлық әдіс $M^{119} (\mathrm{mod} n)$ санын келесі түрде есептейді.

Квадратқа шығару мен көбейтулердің жалпы саны $7+2=9$, бұл қарапайым бинарлық әдістен $2$ көбейтуге аз болып табылады.

| $f_j$ | 2a қадамы | 2b қадамы |
|---|---|---|
| 1 | $M$ | $M$ |
| 0 | $(M)^2 = M^2$ | $M^2$ |
| 0 | $(M^2)^2 = M^4$ | $M^4$ |
| 0 | $(M^4)^2 = M^8$ | $M^8$ |
| $\bar{1}$ | $(M^8)^2 = M^{16}$ | $M^{16} \cdot M^{-1} = M^{15}$ |
| 0 | $(M^{15})^2 = M^{30}$ | $M^{30}$ |
| 0 | $(M^{30})^2 = M^{60}$ | $M^{60}$ |
| $\bar{1}$ | $(M^{60})^2 = M^{120}$ | $M^{120} \cdot M^{-1} = M^{119}$ |

## 3.13.1. Booth-алгоритмі және түрлендірілген сызбалар

Booth-алгоритмі $e=(e_{k-1}e_{k-2}...e_1e_0)$ екілік санының разрядтарын оңнан солға көшірледі және қайта кодталған $f$ санынын цифрларын келесі кестені қолдана отырып алады:

| $e_i\ e_{i-1}$ | $f_i$ |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 0 |

$f_0$-і алу үшін $e_{-1}=0$ деп аламыз. Мысалы, $e=(111001111)$ санының қайта кодталуы келесі түрде болады:

$$100\bar{1}01000\bar{1}.$$

Бірақ Booth-алгоритмінің белгілі бір кемшілігі бар. Қайталанатын $(01)$ тізбектілігі қайталанатын $(1\bar{1})$ тізбектілігіне қайта кодталады. Сонда қайта кодтау нәтижесі түпнұсқаға қарағанда сиретілген болып табылады. Қате болған жағдайда $e=(101010101)$ келесі нәтиже береді:

$$1\bar{1}1\bar{1}1\bar{1}1\bar{1}1.$$