Prímtesztelés és prímfaktorizáció

Diplomamunka

Írta: Gerbicz Róbert

Alkalmazott matematikus szak

Email : robert.gerbicz@gmail.com

Témavezető:

Gyarmati Katalin, tudományos munkatárs

Algebra és Számelmélet Tanszék

Eötvös Loránd Tudományegyetem, Természettudományi Kar

Eötvös Loránd Tudományegyetem

Természettudományi Kar

2011

Tartalomjegyzék

1. fejezet

Bevezetés

Néhány jelölés:

$\omega(n)$: n különböző prímosztóinak a száma

$\varphi(n)$: Euler-féle phi függvény

$d(n)$: n osztóinak a száma

$\sigma(n)$: n osztóinak az összege

$o_n(a)$: az a rendje modulo n

$\pi(x)$: x-nél nem nagyobb prímek száma

M_n: n-edik Mersenne szám, azaz $2^n - 1$

F_n: n-edik Fermat szám, azaz $2^{2^n} + 1$

$Fib(n)$: n-edik Fibonacci szám

$L_n[\alpha, c] = exp((c + o(1))(\log(n))^{\alpha}(\log\log n)^{1-\alpha})$

$\varphi_n(x)$: az n-edik körosztási polinom; $\varphi_n(b)$ jelölés az $x = b$ helyen vett helyettesítési értéke

Használt definíciók és bizonyítás nélkül tételek:

1. Definíció (Prím). *p prím, ha p nem nulla, nem egység és $p|ab$-ből következik, hogy $p|a$ vagy $p|b$.*

2. Definíció (Felbonthatatlan). *f felbonthatatlan, ha f nem nulla, nem egység, és $f = ab$-ből következik, hogy az egyik egység, másik az f asszociáltja.*

1. Tétel. *$\mathbb{Z}$-ben felbonthatatlan és prím ugyanaz.*

2. Tétel (számelmélet alaptétele). *Ha n nem nulla, nem egység, akkor n felírható prímek szorzataként, ahol ez a felírás asszociáltság erejéig és sorrendtől eltekintve egyértelmű.*

3. Tétel (Eukleidész). *Végtelen sok prím van.*

4. Tétel (prímszámtétel). $\lim_{x\to\infty} \frac{\Pi(x)}{\frac{x}{\ln(x)}} = 1$

5. Tétel (Dirichlet). *Legyen $b > 0$ és d egészek relatív prímek, ekkor az $a_n = bn+d$ számtani sorozat végtelen sok prímet tartalmaz.*

6. Tétel (Mertens első tétele). $\log n - \sum_{p\leq n} \frac{\log n}{p} = O(1)$, *ha* $n \to \infty$

7. Tétel (Mertens második tétele). $\lim_{n\to\infty}(-\lg\lg n + \sum_{p<n} \frac{1}{p}) = C$, *ahol C a Mertens konstans.*

8. Tétel (Mertens harmadik tétele). $\lim_{n\to\infty}(\lg n \prod_{p<n}(1 - \frac{1}{p})) = e^{-\gamma}$, *ahol γ az Euler-Mascheroni állandó.*

A prím a matematika egyik legrégibb fogalma, már Eukleidész i.e. 4. században íródott Elemek című művében megjelenik. Ebben szerepel Eukleidész bizonyítása, hogy végtelen sok prím van. Továbbá a számelmélet alaptételéhez hasonló tulajdonságokat lát be a VII. könyvben, amit először korrektül Gauss mond ki és bizonyít 1801-ben. Még ő fogalmazta meg mindössze 15 évesen Lambert (meglehetősen pontatlan) 102000-ig terjedő prímtáblázatainak tanulmányozása során a prímszámok eloszlására vonatkozó $\pi(x) \approx \frac{x}{\log x}$ sejtést, amit Hadamard és de la Vallée Poussin igazolt 1896-ban.

Közben megjelentek prímtesztek és prímfaktorizációs módszerek, ez utóbbiak nagy része még mindig Fermat klasszikus faktorizációs trükkjén alapszik: ha n összetett és $n|x^2-y^2 = (x-y)(x+y)$, akkor kb. legalább $\frac{1}{2}$ valószínűséggel n egy nemtriviális osztóját megkaphatjuk: $d = lnko(x+y, n)$. Prímtesztelésről 1971-ben látta be Robinson, hogy NP-ben van (coNP-ben is benne van, de ez triviális), míg 2002-ben bizonyították, hogy P-ben van, ez az AKS teszt. Ahogy $NP \cap coNP$-beli problémákról általában sikerül belátni, hogy P-beliek. Ellenben a faktorizációról máig nem tudjuk, hogy polinom időben elvégezhető lenne Turing gépen, bár vannak erre utaló jelek, mint például Schor tétele ami kvantum Turing gépen faktorizál polinom időben. Fontos lenne tudni a bonyolultságot, mert például a klasszikus rejtjelezési módszerek legnépszerűbbike az RSA titkosítás törhetősége is ezen múlik. Ajánlások szerint 1024 bites RSA-t már nem javasolják. Jelenlegi (publikus) faktorizációs világrekord általános (nem speciális alakú) számok körében 768 bit, anno egyébként 50000 dollárt tűzött ki az rsa.com ennek a számnak a faktorizációjára, de még 2007-ben minden pénzdíjat visszavontak. Speciális alakú számok körében már megdöntötték az $1k = 1024$ bites álomhatárt, a világrekord 1039 bites, ez a $2^{1039} - 1$ faktorizálását jelentette 2007-ben.

A weblapomon levő programok (lásd [9]) mind saját fejlesztésűek. Diplomamunkában közölt futásidők 64 bites 2.1 GHz-es Amd processzor egy magjára vonatkoznak 64 bites Suse Linux operációs rendszer alatt, 2 GB Ram memória mellett. Használtam az (ingyenesen letölthető) 5.0.1-es Gmp könyvtárat és a 4.5.3-as Sage computeralgebrai programot.

2. fejezet

Prímtesztelés

Prímtesztelés során a feladat eldönteni, hogy egy adott n egész szám prím-e. A probléma nem triviális abban a tekintetben, hogy nem lehetséges egy véges hosszú listát ellenőrizni, hogy n rajta van-e, mert végtelen sok prím van Eukleidész tétele szerint:

9. Tétel (Eukleidész). *Végtelen sok prím van.*

Indirekten tegyük fel, hogy véges sok prím van: $p_1 = 2, p_2 = 3, \cdots, p_k$, legyen $n = 1 + \prod_{i=1}^{k} p_i$, ekkor $n > 1$. Számelmélet alaptétele miatt n felírható prímek szorzataként: $n = q_1 q_2 \cdots q_r$, itt a q_i prímek különböznek a p_j prímektől, mert n mindegyik p_j-vel osztva egy maradékot ad. Így van a p_i-ktől különböző prím, ellentmondás.

Eukleidesz tételére nagyon sok bizonyítás ismert. Ezek közül most egy általam talált bizonyítást is szeretnék ismertetni:
Tegyük fel, hogy véges sok prím van, legyen ezek szorzata c. Ismeretes, hogy $n! = \prod_{p \leq n} p^{\alpha(p)}$, ahol Legendre formulája szerint: $\alpha(p) = \sum_{k \geq 1} \lfloor \frac{n}{p^k} \rfloor$, itt $p \geq 2$ miatt $\alpha(p) \leq \sum_{k \geq 1} \lfloor \frac{n}{2^k} \rfloor \leq \sum_{k \geq 1} \frac{n}{2^k} = n$. Továbbá $n! \geq (\frac{n}{2})^{\frac{n}{2}}$ triviális alsó becslés igaz, hiszen legalább $\frac{n}{2}$ tényező $\frac{n}{2}$-nél nem kisebb az $n! = 1 * 2 * \cdots n$ szorzatban. Két becslést összerakva: $(\frac{n}{2})^{\frac{n}{2}} \leq n! = \prod_{p \leq n} p^{\alpha(p)} \leq \prod_{p \leq n} p^n \leq c^n$, n-edik gyököt vonva: $\sqrt{\frac{n}{2}} \leq c$, ami elég nagy n-re ellentmondás. Junho Peter Whang bizonyításához hasonló, de különböző befejezéssel. Lásd [1].

2.1. Kis számok prímtesztelése, Eratoszthenész-i szita

Ha nagyon kicsi számokról szeretnénk eldönteni, hogy prím-e, akkor a legjobb módszer adott n-ig meghatározni az összes prímet úgy, hogy az n méretű tömb k-adik eleme legyen 1, ha k prím, 0 különben. Ekkor, ha $n_0 \leq n$-ről szeretnénk eldönteni,

hogy prím-e, csak konstans időben meg kell nézni a tömb n_0-adik elemét. Adott n-ig a prímek meghatározása Eratoszthenész-i szitával történhet:

Írjuk fel a számokat 1-től n-ig. Az 1-et húzzuk át (az 1 az egység Z-ben). Majd sorban haladjunk a számokon, ha találunk át nem húzott számot, akkor a nála nagyobb többszöröseit húzzuk át. Az eljárás végén az át nem húzott számok pontosan az n-nél nem nagyobb prímek lesznek. Hiszen prímet nem húzhattunk át, mert annak nincs valódi osztója. Az összetett számokat mind áthúztuk, mert azoknak van valódi osztójuk.

Megvalósítva az Eratoszthenész-i szitát ez $O(n)$ memóriát igényel és Mertens 2. tételét használva $O(n \lg \lg n)$ időbe kerül.

2.1.1. Erdős sejtése

Végtelen sok $n^2 + 1$ alakú prím van.

A szita módszer egy alkalmazására nézzük meg hogyan lehet adott N-ig meghatározni az összes $n \leq N$-et, amelyre $n^2 + 1$ prímszám. Ismert, hogy $n^2 + 1$-nek csak a 2, illetve $4k+1$ alakú prímosztói lehetnek. Legyen $p = 4k+1$, szeretnénk az összes $n \leq N$-et meghatározni, amelyre $p|n^2 + 1$ teljesül, azaz $n^2 \equiv -1 \pmod{p}$. Legyen az a kvadratikus nemmaradék modulo p (tudjuk, hogy ilyen van), azaz $(\frac{a}{p}) = -1$, így $a^{\frac{p-1}{2}} \equiv -1 \pmod{p}$, tehát $b \equiv a^{\frac{p-1}{4}} \pmod{p}$ választással $b^2 + 1 \equiv 0 \pmod{p}$. Ha $n^2 + 1 \equiv 0 \pmod{p}$, akkor $p|b^2 - n^2 = (b - n)(b + n)$, de p prím, így $n \equiv \pm b \pmod{p}$. Azaz kettő maradékosztály elemeit kell kihúznunk, kivéve esetleg az első elemet, amikor $b^2 + 1 = p$ vagy $(p - b)^2 + 1 = p$. $p \leq N$-ig elég elmenni a szitában, hiszen $n^2 + 1$-nek, ha összetett, akkor van $p \leq \lfloor \sqrt{n^2 + 1} \rfloor = n \leq N$ prímosztója.

c nyelven programot írva a problémára 10^{21}, illetve 10^{22}-ig sikerült megkapnom az $x^2 + 1$ alakú prímek számát a szitával. Lásd Neil Sloane online adatbázisában az `http://oeis.org/A083844` -es sorozatot. Korábban Marek Wolf 10^{20}-ig számolta ki (lásd [2]), de ő csupán véletlen prímteszteket használt. Saját szitámmal azonban Marek Wolf $x^2 + 1 < 2^{84}$-ig kiszámolta, hogy 107533484876 ilyen prím van, és ezek reciprokösszege 30 tizedesjegyre kerekítve 0.814596571702967656907692644179, innen a többi nagyobb $x^2 + 1$ alakú prímek reciprokösszegét becsülve 19 tizedesjegy pontossággal 0.8145965717029728452 (lásd `http://oeis.org/A172168`) lesz a reciprokösszeg, ez itt csupán sejtés. Hasonlóan ahhoz ahogy az ikerprímeknél becsülik a Brun konstanst.

Ha $e(N)$ jelöli az N-nél nem nagyobb $n^2 + 1$ alakú prímek számát, akkor szintén egy régi sejtés, hogy $e(N) = \theta(\frac{\sqrt{N}}{\log N})$.

2.2. Prímtesztelés és faktorizáció egészértékű NLP feladata

Legyen $n > 1$ egész, és a feladat:

$\min x$

$x \geq 2$

$xy = n$

x, y egész

Ennek egy megengedett megoldása $(x, y) = (n, 1)$ Az optimum célfüggvény érték egyenlő n-nel pontosan akkor, ha n prím, így ez a nemlineáris program egy prímtesztet ad. De a faktorizációt is megoldja, hiszen az optimum az n legkisebb (p) prímosztója, így $\frac{n}{p}$-re ismételten meghívva n-et faktorizálja. Az (x, y) megengedett megoldások egy hiperbola egész pontjai. Speciális alakú számokra az eljárás gyorsítható (lásd Mersenne számokra).

2.3. Prímtesztelés automatákkal

10. Tétel. *Nem létezik a prímeket felismerő véges automata.*

Bizonyítás: A nyelv számrendszere legyen a 10-es (más számrendszerre ugyanúgy elmondható a bizonyítás). d állapota legyen az automatának. Először egy állítást látok be:

Állítás: Minden állapotból eljuthatunk elfogadó állapotba, méghozzá legfeljebb d lépésben

Bizonyítás: Prímszámtételből következik, hogy tetszőleges sztringgel kezdődő prím létezik, így minden állapotból eljuthatunk elfogadó állapotba (ellenkező esetben nem minden prímet ismernénk fel). De akkor az automata egy ilyen működése során a körsétát sétává rövidítve legfeljebb d lépésben is eljuthatunk az adott elfogadó állapotba. Amit bizonyítani kellett.

A bizonyításból következik, hogy a prímek alsó sűrűsége legalább 10^{-d-1}, ami ellentmond a prímszámtételnek. Ugyanis rögzített N-re, minden $0 \leq n < 10^{N-d}$ esetén az automata bármely állapotban is legyen van olyan d hosszú jegysorozat, amire egy elfogadó állapotba fog kerülni. Így 10^N-ig legalább 10^{N-d} prím van, ebből pedig már következik, hogy a prímek alsó sűrűsége legalább 10^{-d-1}.

Másik befejezés: az állítás szerint tetszőleges k számból eljuthatunk egy prímhez, ami azt jelenti, hogy $\exists 0 < f \leq d, \quad 0 \leq g < 10^f$, melyre $k10^f + g$ prím (ekkor f lépésben jutunk prímhez). Ez viszont $k = (2 * 10^d)! - 2$-re nem igaz, hiszen:

$N = ((2*10^d)! - 2)10^f + g$, ekkor $N \geq 2*10^d > 2*10^f - g$ és N osztható $2*10^f - g$-vel, mert $10 \leq 10^f < 2*10^f - g \leq 2*10^d$, így $2*10^f - g$ nemtriviális osztója N-nek, ezért N összetett.

11. Tétel. *Létezik a prímeket felismerő végtelen automata.*

Bizonyítás: Építsünk egy 10-áris (végtelen) fát a nyelvre. Prímeket jelöljük meg elfogadó állapotoknak. Ez jó lesz.

12. Tétel. *Létezik véges automata, ami eldönti az adott prímek valamelyikével osztható számok nyelvét.*

Bizonyítás: $p_1, p_2, \cdots, p_t$ legyenek a prímek, és legyen $M = \prod_{i=1}^{t} p_i$. Az automatának legyen M állapota: $A_0, A_1, \cdots, A_{M-1}$. Kiinduló állapot: A_0. Ha az A_r állapotban van az automata és d jelet kapja, akkor az új állapota legyen A_u, ahol u az a $10r + d$ legkisebb nemnegatív maradéka M-mel osztva. Így, ha az automata végső állapota A_r, akkor könnyen látható, hogy $n \equiv r \pmod{M}$ Így azon A_r állapotokat tegyük elfogadóvá, ahol $lnko(r, M) > 1$. Ez jó lesz, mert $lnko(n, M) > 1$ akkor és csak akkor, ha $lnko(n, M) = lnko(r, M) > 1$.

Későbbiekben prímtesztekhez fogunk használni Jacobi szimbólumot.

13. Tétel. *(Gerbicz) Legyen* $L_a = \{n \in \mathbb{N} : \left(\frac{a}{n}\right) = 1\}$ *Ekkor létezik az* L_a *nyelvet felismerő véges automata.*

Bizonyítás: Csak páratlan a-ra és $\mathbb{N}$ helyett $2\mathbb{Z}-1$-re. Legyenek $A_0, A_1, \cdots, A_{4a-1}$ az automata állapotai, kiinduló állapot A_0. Ha az automata az A_r állapotban van és d jelet kapja, akkor az új állapot legyen A_u, ahol u az $10r+d$ legkisebb nemnegatív maradéka modulo $4a$. Így a végén az automata, ha A_r-ben lesz, akkor $n \equiv r \pmod{4a}$. Kvadratikus reciprocitás tétele szerint (a és n páratlan): $\left(\frac{a}{n}\right) = (-1)^{\frac{a-1}{2}\frac{r-1}{2}}\left(\frac{n}{a}\right) = (-1)^{\frac{a-1}{2}\frac{r-1}{2}}\left(\frac{r}{a}\right)$, ami az a és r ismeretében számítható. Az elfogadó állapotok azok a páratlan r értékek legyenek, amelyekre $(-1)^{\frac{a-1}{2}\frac{r-1}{2}}\left(\frac{r}{a}\right) = 1$ teljesül.

2.4. Prímtesztelés NP ∩ co-NP-ben van

Legyen $PR = \{p : p \text{ prím}\}$, a prímek halmaza. Ekkor $PR \in co - NP$, hiszen tanú az n egy valódi osztója. De NP-ben is benne van:

14. Tétel. *NP-beli probléma, hogy adott n egészről eldöntsük, hogy prím-e.*

Bizonyítás: Ehhez először egy lemmát látok be: Legyen $n > 1$ egész, ekkor n prím, akkor és csak akkor, ha $\exists a : a^{n-1} \equiv 1 \pmod{n}$, de $a^d \not\equiv 1 \pmod{n}$, ha $0 < d < n-1$

Lemma bizonyítása: ha n prím, akkor ismert, hogy létezik primitív gyök modulo p, és ezt a-nak választva jó lesz, a primitív gyök tulajdonságai miatt. Megfordítása: Mivel $a^{n-1} \equiv 1 \pmod{n}$, ezért $lnko(a, n) = 1$, de akkor az Euler-Fermat tétel miatt $a^{\varphi(n)} \equiv 1 \pmod{n}$, lemma feltétele miatt viszont a legkisebb jó kitevő legalább $n-1$, így $\varphi(n) \geq n-1$, de ez utóbbi egyenlőtlenség csak a prímekre teljesül. Így a lemmát beláttuk.

Tétel bizonyítása: Tanúnak az a-t választva jó lesz, $a^{n-1} \pmod{n}$ polinom időben számítható. De az $a^d \equiv 1 \pmod{n}$ még exponenciálisan sok d ellenőrzését igényelné. De elegendő csupán $\forall p|n-1$-re megnézni, hogy $a^{\frac{n-1}{p}} \not\equiv 1 \pmod{n}$, mert ha ez igaz, akkor már minden $0 < d < n-1$-re $a^d \not\equiv 1 \pmod{n}$. Így, ha az a tanúhoz még azt is hozzáveszem, hogy $n-1 = {p_1}^{\alpha_1} \cdots {p_r}^{\alpha_r}$ az $n-1$ prímtényezős felbontása és rekurzíve minden p_i-ről belátom, hogy prímek, akkor n is prím. Az egész bizonyítást leírva egy fáként is ábrázolhatjuk a bizonyítást, minden nem levélben egy $m > 3$-nál nagyobb egész van, gyerekei az $m-1$ (különböző) prímosztói, amikről szintén belátom, hogy prímek. Mivel minden egyes szinten $p_i \leq \frac{m}{2}$ (mert m páratlan, így $m-1$ párosnak a 2 osztója), ezért minden szinten a számok legalább feleződnek azaz $O(\log(n))$ szint van. Továbbá minden nem levélre a gyerekeinek szorzata legfeljebb akkora, mint a szülő (prímtényezős felbontásból ez nyilvánvaló), így minden szinten a számok hosszainak összege $O(\log n)$. Így az algoritmus hossza $O(\log^2 n)$, a bizonyítás pedig szintén polinom időben ellenőrizhető.

Így a probléma $NP \cap co-NP$ eleme, az ilyen problémákról általában belátják, hogy valójában polinom időben eldönthetőek, az AKS teszt óta ez ismert is.

2.5. Prímtesztelés P-ben van, az AKS teszt

Az AKS algoritmus (lásd [3]): input egy $n > 1$ egész szám. Erről $O(\log^{12+\varepsilon} n)$ időben eldönti, hogy prím-e.

1. Ha $n = a^b$, ahol $a > 0, b > 1$, azaz n teljes hatvány, akkor output: n összetett.
2. Legkisebb r egész megkeresése, melyre $o_r(n) > {\log_2}^2 n$
3. Ha $1 <$lnko$(a, n) < n$ valamely $0 < a \leq r$ esetén, akkor output: n összetett.
4. Ha $n \leq r$, akkor output: n prím.
5. Minden $0 < a \leq \sqrt{\varphi(r)} \log_2(n)$-re: ha $(x+a)^n \not\equiv x^n + a \pmod{(x^r - 1, n)}$ akkor output: n összetett.
6. Output n prím.

Az algoritmus helyességének az igazolásában a könnyű eset amikor $n = p$ prím. Ugyanis az 1.,3. pontok prímekre nem teljesülhetnek. Továbbá az 5. sem, ugyanis binomiális tétel szerint $(x + a)^p = \sum_{k=0}^{p} \binom{p}{k} x^k a^{n-k}$, de legyen $0 < k < p$, ekkor $\binom{p}{k} = \frac{p}{k}\binom{p-1}{k-1}$, így átszorzással: $k\binom{p}{k}$ osztható p-vel, de k relatív prím p-hez, így $\binom{p}{k}$ osztható p-vel, azaz $(x+a)^p \equiv x^p + a^p \equiv x^p + a \pmod{(x^r - 1, n)}$ (kis Fermat tételt is használva). Így $n = p$ prím esetén az algoritmus a 4. vagy 6. pontban áll meg. Az algoritmus bizonyításának nehezebb része az amikor n összetett.

Még a 2009. évi 2. hackerverseny 3. feladatában (lásd [8]) éppen egy AKS tesztet kellett írni. Ez a Sage-ban szépen megcsinálható. De gmp-ben is: itt ugyan nincsenek polinomok, de az egészegyüttható polinomok szorzása visszavezethető nagy egészek szorzására: ehhez legyen $p(x) = \sum_{i=0}^{r-1} a_i x^i$, ahol $B > r * a_i \geq 0$ egész, a polinomhoz $c = \sum_{i=0}^{r-1} a_i B^i$ egész számot rendeljük. Másik ilyen polinom $q(x) = \sum_{j=0}^{r-1} b_j x^j$ és $d = \sum_{j=0}^{r-1} b_j B^j$, akkor $c * d$ szorzás elvégzése után a számrendszeres felírásban B^k együtthatója éppen $p(x)q(x)$-nek a k-adfokú együtthatója lesz, hiszen B választása miatt túlcsordulás sem lesz. Ez a trükk visszafelé is működik, egész számok szorzását vissza lehet vezetni polinomok szorzására.

Ha p prím és nem túl kicsi, akkor könnyen látható, hogy az AKS teszt a 6. pontban áll le. Míg összetett n esetén a költségesebb polinomszorzásokhoz nem is biztos, hogy eljut. Így az AKS teszt jellemzően a prímekre dolgozik sokat, arra érdemes tesztelni a programokat. Gmp $4-6$-szor gyorsabbnak bizonyult kisebb p prímekre, mint a Sage, 64 bitnél hosszabb p prímekre már több, mint hússzor lassabb.

Táblázat

e	$p = nextprime(2^e)$	r	$AKSgmp(p)$	$AKSsage(p)$
5	37	29	$< 1s$	$< 1s$
10	1031	107	$< 1s$	$2s$
15	32771	227	$< 1s$	$8s$
20	1048583	409	$3s$	$24s$
25	33554467	643	$7s$	$57s$
30	1073741827	911	$19s$	$128s$
35	34359738421	1229	$46s$	$261s$
40	1099511627791	1607	$97s$	$483s$
45	35184372088891	2039	$194s$	$821s$
50	1125899906842679	2521	$375s$	$1635s$
55	36028797018963971	3079	$661s$	$2667s$
60	1152921504606847009	3613	$1079s$	$4295s$
65	36893488147419103363	4253	$1807s$	$40043s$
70	1180591620717411303449	4931	$2844s$	$n.a.$
75	377789318629571617096 01	5651	$4265s$	$n.a.$
80	120892581961462917470 6189	6449	$6158s$	$n.a.$
85	386856262276681335905 97803	7253	$8885s$	$n.a.$
90	123794003928538027489 9124357	8111	$12602s$	$n.a.$
95	396140812571321687967 71975177	9029	$17528s$	$n.a.$
100	126765060022822940149 6703205653	10039	$24819s$	$n.a.$

2.6. Álprímek

2.6.1. Fermat-féle álprímek

3. Definíció (Fermat-féle álprím). *Az n páratlan összetett számot b alapra nézve Fermat-féle álprímnek nevezzük, ha $lnko(b, n) = 1$ és $b^{n-1} \equiv 1 \pmod{n}$ teljesül.*

15. Tétel. *Minden b alapra nézve végtelen sok álprím van.*

Bizonyítás: Legyen $p > 2$ prím és $n = \frac{b^{2p}-1}{b^2-1}$ és $b \not\equiv \pm 1 \pmod{p}$, ekkor n álprím b alapra nézve. Ugyanis $n = \frac{b^p-1}{b-1}\frac{b^p+1}{b+1} = (b^{p-1}+b^{p-2}+\cdots+b+1)(b^{p-1}-b^{p-2}\pm\cdots-b+1)$, így n páratlan és összetett. $b^{2p} \equiv 1 \pmod{n}$, hiszen $n = \frac{b^{2p}-1}{b^2-1} | b^{2p} - 1$, de kis-Fermat tétel szerint $b^{2p} \equiv b^2(b^{p-1})^2 \equiv b^2 \pmod{p}$. Mivel $n(b^2-1) = b^{2p} - 1$ ezért $n(b^2-1) \equiv b^{2p} - 1 \pmod{p}$, így $n(b^2-1) \equiv b^2 - 1 \pmod{p}$, mivel a feltételek szerint $lnko(b^2 - 1, p) = 1$, hiszen $b \pm 1$ nem osztható p-vel. A kongruenciát egyszerűsítve kapjuk: $n \equiv 1 \pmod{p}$, de n páratlan, ezért $n \equiv 1 \pmod{2p}$ is igaz, azaz $2p|n - 1$,

így $n|b^{2p}-1|b^{n-1}-1$, azaz $b^{n-1} \equiv 1 \pmod{n}$ is teljesül, azaz n álprím b alapra nézve. Ha $p > b+1$, akkor $b \not\equiv \pm 1 \pmod{p}$ triviálisan, így végtelen sok p prím található a tételhez, azaz végtelen sok álprím van minden b alapra. Ami kellett.

16. Tétel. *Ha* $lnko(b_1, n) = lnko(b_2, n) = 1$ *és* n *álprím* b_1, b_2 *alapokra, akkor* n *álprím* $b_1 b_2$ *és* $b_1 b_2^{-1}$ *alapokra is.*

Bizonyítás: $b_1^{n-1} \equiv 1 \pmod{n}$ és $b_2^{n-1} \equiv 1 \pmod{n}$ feltételek miatt, de akkor $(b_1 b_2)^{n-1} \equiv b_1^{n-1} b_2^{n-1} \equiv 1 \pmod{n}$, azaz $b_1 b_2$ alapra nézve is álprím. Továbbá $(b_1 b_2^{-1})^{n-1} \equiv b_1^{n-1} (b_2^{n-1})^{-1} \equiv 1 \pmod{n}$, azaz $b_1 b_2^{-1}$ alapra nézve álprím.

17. Tétel. *Ha* n *csak egyetlen* b*-re (*$lnko(b, n) = 1$ *mellett) is bukja* $b^{n-1} \equiv 1 \pmod{n}$ *Fermat tesztet, akkor a mod* n *maradékosztályok legalább felét bukja.*

Bizonyítás: Előző tétel miatt b alapok, melyekre $b^{n-1} \equiv 1 \pmod{n}$ teljesül $\mathbb{Z}_n$ multiplikatív csoportjában egy részcsoport, így, ha ez nem egyenlő $\mathbb{Z}_n^*$ csoporttal (azaz létezik b, melyre $lnko(b, n) = 1$ és $b^{n-1} \not\equiv 1 \pmod{n}$), akkor mivel részcsoport, így rendje legfeljebb $\frac{|\mathbb{Z}_n^*|}{2}$, ami pont azt jelenti, hogy a redukált maradékosztályok legalább felét bukja a teszt, ami kellett.

Azaz, ha az n páratlan összetett szám nem minden relatív prím b alapra nézve Fermat-féle álprím, akkor a redukált maradékosztályok legalább felére nem álprím, így $b^{n-1} \pmod{n}$ ellenőrzése jó valószínűségi prímtesztelést ad: először n-et teszteljük egy véletlen $1 < b < n$ alapra, ha $lnko(b, n) > 1$, akkor n összetett, ha nem, akkor $b^{n-1} \pmod{n}$-et kiszámoljuk $O(\log^3 n)$ bitoperációval ismételt hatványozással. Ha a maradék nem kongruens 1-gyel modulo n, akkor n megint nem prím, egyébként egy másik $1 < b' < n$-re teszteljük. Ha a két teszt független, akkor legfeljebb $\frac{1}{2}\frac{1}{2} = \frac{1}{4}$ valószínűséggel éli túl a 2 tesztet (ha van ilyen b alap). K darab b-vel tesztelve legfeljebb $\frac{1}{2^K}$ valószínűséggel éli túl.

Legendre szimbólum tulajdonsága (az Euler lemma következménye), hogy p páratlan prím és $lnko(b, p) = 1$ esetén $b^{\frac{p-1}{2}} \equiv (\frac{b}{p}) \pmod{p}$ teljesül.

2.6.2. Carmichael számok

4. Definíció (Carmichael szám (Univerzális álprím)). *Az* n *összetett számot Carhmichael számnak nevezzük, ha minden* n*-hez relatív prím alapra Fermat-féle álprím, azaz* $lnko(a, n) = 1$ *esetén* $a^{n-1} \equiv 1 \pmod{n}$.

Példa: 561 a legkisebb Carmichael szám.

18. Tétel (Korselt kritérium). *Az n összetett szám Carmichael szám akkor és csak akkor, ha n négyzetmentes és minden p prímosztójára $p-1|n-1$ teljesül.*

19. Tétel (Alford, Granville, Pomerance). *Végtelen sok Carmichael szám van. Ha n elég nagy, akkor legalább $n^{\frac{2}{7}}$ Carmichael szám van n-ig.*

2.6.3. Euler-féle álprímek

5. Definíció (Euler-féle álprím). *Ha n páratlan összetett szám, $lnko(b,n)=1$ és $b^{\frac{n-1}{2}} \equiv (\frac{b}{n}) \pmod{n}$ (ahol $(\frac{b}{n})$ Jacobi szimbólum), teljesül, akkor n-et Euler-féle álprímnek nevezzük b alapra nézve.*

20. Tétel. *Ha n Euler-féle álprím b alapra nézve, akkor ugyanerre az alapra nézve Fermat-féle álprím is. Visszafelé nem feltétlenül igaz.*

Bizonyítás: Feltétel szerint $b^{\frac{n-1}{2}} \equiv (\frac{b}{n}) \pmod{n}$, de $(\frac{b}{n}) \equiv \pm 1 \pmod{n}$,így az előbbi kongruenciát négyzetre emelve: $b^{n-1} \equiv 1 \pmod{n}$, azaz b-re nézve Fermat-féle álprím, ami kellett. Másik irányhoz $b=3, n=91$ ellenpélda.

21. Tétel. *(konstrukció Gerbicz) Minden a-ra végtelen sok Euler-féle álprím van.*

Bizonyítás: 1. eset: b páratlan. Legyen ekkor $n=\frac{b^{2p}-1}{b^2-1}$, mint a Fermat-féle álprím konstrukciónál, azaz p páratlan prím, $lnko(b^2-1,p)=1$, de most $p \equiv 1 \pmod{4}$ is legyen igaz. $b^{2p} \equiv 1 \pmod{n}$ teljesül, $n=\frac{b^{2p}-1}{b^2-1}=b^{2p-2}+\cdots+b^2+1$, de b páratlan, így $b^{2k} \equiv 1 \pmod{8}$ (ahol k egész), hiszen egy páratlan szám négyzete 8-cal osztva 1 maradékot ad. Így n az p darab $8k+1$ alakú egész összege, azaz $n \equiv p \pmod{8}$, de $p \equiv 1 \pmod{4}$ miatt $n \equiv 1 \pmod{4}$.

$(b^2-1)n=b^{2p}-1 \equiv b^2-1 \pmod{p}$ a kis-Fermat tétel miatt, de $lnko(b^2-1,p)=1$, így $n \equiv 1 \pmod{p}$. Korábban láttuk $n \equiv 1 \pmod{4}$, de p páratlan, így $n \equiv 1 \pmod{4p}$ is teljesül, azaz $2p|\frac{n-1}{2}$. Így $b^{2p} \equiv 1 \pmod{n}$ miatt $b^{\frac{n-1}{2}} \equiv 1 \pmod{p}$ is igaz.

Ismeretes, hogy a kvadratikus reciprocitás törvénye igaz Jacobi szimbólumra is, így írható: $(\frac{b}{n})=(-1)^{\frac{b-1}{2}\frac{n-1}{2}}(\frac{n}{b})=1(\frac{n}{b})=1$, mert $2|\frac{n-1}{2}$ miatt $(-1)^{\frac{b-1}{2}\frac{n-1}{2}}=1$, továbbá $n \equiv 1 \pmod{b}$ miatt $(\frac{n}{b})=(\frac{1}{b})=1$

Az n páratlan és összetett, mert $\frac{b^{2p}-1}{b^2-1}=\frac{b^p-1}{b-1}\frac{b^p+1}{b+1}$ nemtrivi felbontás. Mivel végtelen sok $p \equiv 1 \pmod{4}$ prím van, ezért végtelen sok ilyen n páratlan összetett szám van, amire tehát $b^{\frac{n-1}{2}} \equiv (\frac{b}{n}) \pmod{n}$, azaz n Euler-féle álprím b alapra nézve, ami kellett.

2. eset: b páros. Az előző konstrukció itt nem működik. De $n=\frac{b^{4p}-1}{b^4-1}$ jó lesz, ahol $p>4$ prím és $lnko(p,b^4-1)=1$. $n=\frac{b^{4p}-1}{b^4-1}=b^{4p-4}+\cdots+b^4+1 \equiv 1 \pmod{16}$, mert b páros. Legyen $b=2^f(2g+1)$ (egyértelmű felírás), mivel b páros ezért $f>0$.

$n(b^4-1) = b^{4p}-1 \equiv b^4-1 \pmod{p}$ (a kis-Fermat tételt használva), de $lnko(p, b^4-1) = 1$ miatt $n \equiv 1 \pmod{p}$, de $8|(n-1)$ is igaz, így mivel p páratlan, ezért $8p|(n-1)$, azaz $4p|\frac{n-1}{2}$, mivel $b^{4p} \equiv 1 \pmod{n}$, ezért $b^{\frac{n-1}{2}} \equiv 1 \pmod{n}$ is teljesül.

$(\frac{b}{n}) = (\frac{2^f(2g+1)}{n}) = (\frac{2}{n})^f(\frac{2g+1}{n}) = (-1)^{\frac{n^2-1}{8}f}(\frac{2g+1}{n}) = (\frac{2g+1}{n})$, mivel $n \equiv 1 \pmod{8}$ miatt $\frac{n^2-1}{8}$ páros. Kvadratikus reciprocitás tétel miatt (n páratlan):
$(\frac{2g+1}{n}) = (-1)^{g\frac{n-1}{2}}(\frac{n}{2g+1}) = (\frac{n}{2g+1}) = 1$, mivel $\frac{n-1}{2}$ páros, továbbá $(2g+1)|b^4|(n-1)$, így $(2g+1)|(n-1)$, de akkor $(\frac{n}{2g+1}) = (\frac{1}{2g+1}) = 1$.

n persze páratlan és összetett is, mert $b^p - 1|b^{4p} - 1$ miatt $\frac{b^p-1}{lnko(b^p-1,b^4-1)}|\frac{b^{4p}-1}{b^4-1}$, és itt $1 < \frac{b^p-1}{lnko(b^p-1,b^4-1)} < n$, így n-nek van valódi osztója, azaz összetett.

1. Feladat. *Keressünk összetett számokat, melyek az első t prímre, mint alapra nézve Euler-féle álprímek!*

Konstrukció: Carmichael számok között érdemes keresni: legyen $N = Q_1Q_2Q_3 = (6k+1)(12k+1)(18k+1)$, ahol $Q_1 = 6k+1$,$Q_2 = 12k+1$,$Q_3 = 18k+1$ prímek és $\prod_{i=3}^{t} p_i|k$, ahol p_i az i-edik prím.

Könnyen látható, hogy $72k|(N-1)$, de $\text{lkkt}(Q_1-1, Q_2-1, Q_3-1) = 36k$, így kis-Fermat tétel miatt $p_i^{\frac{N-1}{2}} \equiv 1 \pmod{Q_j}$ minden j-re igaz, de akkor ez a kongruencia N modulusra is igaz, tehát $p_i^{\frac{N-1}{2}} \equiv 1 \pmod{N}$.

Jacobi szimbólum: $(\frac{p_i}{N}) = (-1)^{\frac{p_i-1}{2}\frac{N-1}{2}}(\frac{N}{p_i}) = (\frac{N}{p_i}) = (\frac{1}{p_i}) = 1$, itt használtuk, hogy $4|36k|\frac{N-1}{2}$ és minden i-re $p_i|6\prod_{j=3}^{t} p_j|6k|Q_j - 1$ minden j-re, így $N \equiv 1 \pmod{p_i}$ is igaz.

Tehát $p_i^{\frac{N-1}{2}} \equiv (\frac{p_i}{N}) \pmod{N}$ teljesül, azaz Euler-féle álprím p_i alapra nézve.

A feladat lényegében `https://www.spoj.pl/problems/SOLSTRAS/` az spoj online bírón.

2.6.4. Erős álprímek, Miller-Rabin tesztelés

6. Definíció (Erős álprím). *Ha n természetes egész páratlan összetett szám, $n - 1 = 2^s m$, ahol $s > 0$ és m páratlan, $lnko(b, n) = 1$, és $b^m \equiv 1 \pmod{n}$, vagy létezik $r : 0 \leq r < s$, hogy $b^{2^r m} \equiv -1 \pmod{n}$, akkor n-et erős álprímnek nevezzük b alapra nézve.*

22. Tétel (A.O.L. Atkin és R. Larson). *Ha n erős álprím adott a alapra, akkor Euler-féle álprím is az a alapra.*

Bizonyítás: Legyen n erős álprím az a alapra, $n = p_1p_2\cdots p_t$, ahol ugyanaz a prím többször is szerepelhet. Legyen k_j az az egész, amire 2^{k_j} pontos osztója $p_j - 1$-nek, tegyük fel, hogy $k_1 \leq k_2 \leq \cdots \leq k_t$ Mivel n erős álprím, ezért van $k \geq 0$, amelyre

2^k pontos osztója $ord_a(p^b)$-nek az n minden p^b pontos osztójára. Mivel $\frac{ord_a(p^b)}{ord_a(p)}$ az p hatvány mindig, és ezért páratlan így kapjuk, hogy 2^k pontos osztója $ord_a(p_j)$-nek minden j-re, ezért $k \leq k_1$. Legyen i azon k_j egészek száma amelyekre $k_j = k$, ekkor $n \equiv (2^k+1)^i \pmod{2^{k+1}}$, ezért 2^k pontos osztója $n-1$-nek vagy 2^{k+1} osztja $n-1$-et aszerint, hogy i páratlan vagy páros. Így, ha p^b pontos osztója n-nek, akkor $a^{\frac{n-1}{2}}$ az -1 illetve $+1 \pmod{p^b}$, aszerint, hogy 2^k pontos osztója $n-1$-nek vagy 2^{k+1} osztja $n-1$-et, azaz $a^{\frac{n-1}{2}}$ aszerint -1 illetve $+1 \pmod{n}$, hogy i páratlan vagy páros.

$(\frac{a}{p_j}) = -1$ vagy $+1$ aszerint, hogy $j \leq i$, vagy $j > i$, mert $(\frac{a}{p}) = -1$ akkor és csak akkor, ha $ord_a(p)$-nek és $p-1$-nek ugyanaz a pontos 2-hatvány osztója van, azaz $k = k_j$ teljesül, ha $p = p_j$ volt, ezen j indexek száma pedig éppen i. Ezért $(\frac{a}{n}) = \prod(\frac{a}{p_j}) = (-1)^i$, ami aszerint páratlan vagy páros, hogy i páratlan vagy páros. Ezzel beláttuk, hogy $a^{\frac{n-1}{2}} \equiv (\frac{a}{n}) \pmod{n}$, ami kellett.

23. Tétel (Malm). *Ha n $4k+3$ alakú Euler-féle álprím adott a alapra, akkor erős álprím is az a alapra, azaz $4k+3$ alakú számok körében a két fogalom megegyezik.*

Bizonyítás: n az $4k+3$ alakú egész, ezért $n-1 = 2d$, ahol d páratlan. Feltétel szerint Euler-féle álprím, ezért: $a^{\frac{n-1}{2}} \equiv a^d \equiv (\frac{a}{n}) \pmod{n}$, de Jacobi szimbólum az ± 1, így az n erős álprím az a alapra (definíciót teljesíti).

7. Definíció (Miller-Rabin tesztelés). *Tegyük fel, hogy az n természetes páratlan szám összetett voltát teszteljük. Legyen $n-1 = 2^s m$, ahol m páratlan és $s > 0$, $lnko(b,n) = 1$, Számoljuk ki $b^m \pmod{n}$-et, ha ez ± 1, akkor állta a tesztet, másik b-vel tesztelünk, ha ez nem ± 1, akkor négyzetre emeljük a maradékot, aztán újra, ha szükséges, azaz amíg -1-et nem kapunk vagy eljutunk $b^{n-1} \equiv b^{2^s m} \pmod{n}$-hez s darab négyzetre emelés után. Ha $b^{n-1} \not\equiv 1 \pmod{n}$ vagy az első 1-től különböző maradék nem -1, akkor n bukja a tesztet.*

24. Tétel. *Ha n páratlan összetett szám, $n \neq 9$, akkor n erős álprím a redukált maradékosztályok legfeljebb $\frac{1}{4}$-ére. Azaz legyen $n = 2^s * q + 1$, ahol $s \geq 1$, akkor $|R_n| = |\{1 \leq a \leq n | lnko(a,n) = 1$ és $a^q \equiv 1 \pmod{n}$ vagy $a^{2^j q} \equiv -1 \pmod{n} (0 \leq j \leq s-1)\}| \leq \frac{1}{4}\varphi(n)$*

Bizonyítás: Legyen n prímtényezős felbontása: $n = {n_1}^{e_1} {n_2}^{e_2} \cdots {n_r}^{e_r}$, ahol n_i prímek és $e_i \geq 1$ minden i-re. $p_i = n_i$ jelölést is használjuk a bizonyításban.

$\forall a \in R_n$ esetén $a^{n-1} \equiv 1 \pmod{n}$ igaz, ez ekvivalens $a^{n-1} \equiv 1 \pmod{{n_i}^{e_i}} (i = 1, \cdots, r)$ kongruencia rendszerrel a kínai maradéktétel szerint.

Ismeretes, hogy létezik primitív gyök modulo ${p_i}^{e_i}$, mivel p_i páratlan prím. Legyen g egy ilyen. Ekkor $g^y \equiv 1 \pmod{{p_i}^{e_i}}$ akkor és csak akkor teljesül a rend tulajdonságai

miatt, ha $\varphi({p_i}^{e_i})|y$ teljesül. Mivel $\text{lnko}(a,n)=1$ ezért $\text{lnko}(a,{p_i}^{e_i})=1$, így létezik k egész, hogy $g^k \equiv a \pmod{{p_i}^{e_i}}$. Nekünk $a^{n-1} \equiv g^{k(n-1)} \equiv 1 \pmod{{p_i}^{e_i}}$ kell, azaz $\varphi({p_i}^{e_i})|k(n-1)$, ami pontosan akkor igaz, ha $\frac{\varphi({p_i}^{e_i})}{lnko(\varphi({p_i}^{e^i}),n-1)}|k$, de $0 \le k < \varphi({p_i}^{e_i})$. Így ennek megoldásszáma: $\text{lnko}(\varphi({p_i}^{e_i}),n-1)$. Azaz a kínai maradéktétel szerint: $|R_n| = \prod_{i=1}^{r}\text{lnko}(n-1,\varphi({p_i}^{e_i})) = \prod_{i=1}^{r}\text{lnko}(n-1,{p_i}^{e_i-1}(p_i-1)) = \prod_{i=1}^{r}\text{lnko}(n-1,p_i-1) \le \prod_{i=1}^{r}(p_i-1)$, de $\varphi(n)=\prod_{i=1}^{r}({p_i}^{e_i-1}(p_i-1))$. Így $|R_n| \le \frac{\varphi(n)}{4}$, kivéve, ha: n négyzetmentes vagy $n=3^2n_2\cdots n_r$ (ahol $r\ge 2$, mert $n=9$ kivétel volt).

Az első esetet nézzük (második hasonlóan látható be). Tegyük fel, hogy $n=p_1\cdots p_r$ és $n_i=1+2^{s_i}q_i$, ahol q_i páratlan és $s_i>0$, mert n páratlan, így minden prímosztója páratlan. Rendezzük úgy a prímosztókat, hogy $1\le s_1\le s_2\cdots\le s_r$ legyen. Előzőekből: $x^q\equiv 1 \pmod n$ megoldásszáma: $\prod_{i=1}^{r}\text{lnko}(q,n_i-1)=\prod_{i=1}^{r}\text{lnko}(q,2^{s_i}q_i)=\prod_{i=1}^{r}\text{lnko}(q,q_i)=\prod_{i=1}^{r}q_{i'}$, ahol $q_{i'}=\text{lnko}(q,q_i)$. Nézzük az $x^{2^jq}\equiv -1 \pmod n$ megoldásszámát! Ez a kongruencia ekvivalens az $x^{2^jq}\equiv -1 \pmod{p_i}$ $(i=1,2,\cdots,r)$ rendszerrel. Legyen g primitív gyök modulo p_i, ekkor $g^{p_i-1}\equiv 1 \pmod{p_i}$, ebből $p_i|(g^{\frac{p_i-1}{2}}+1)(g^{\frac{p_i-1}{2}}-1)=g^{p_i-1}$, de g primitív gyök, ezért csak a szorzat első tagját oszthatja, azaz $g^{\frac{p_i-1}{2}}\equiv -1 \pmod{p_i}$ teljesül. Legyen $a^{2^jq}\equiv -1 \pmod{p_i}$, ekkor a és p_i relatív prímek, ezért létezik k, melyre $a\equiv g^k \pmod{p_i}$, így $a^{2^jq}\equiv g^{k2^jq}\equiv -1\equiv g^{\frac{p_i-1}{2}} \pmod{p_i}$ így a primitív gyök tulajdonságai miatt: $k2^jq\equiv\frac{p_i-1}{2} \pmod{p_i-1}$, ez megoldható pontosan akkor, ha $\text{lnko}(2^jq,p_i-1)|\frac{p_i-1}{2}$, és ekkor a megoldásszám: $\text{lnko}(2^jq,p_i-1)$. A rendszernek minden i-re megoldhatónak kell lennie, így $\text{lnko}(2^jq,p_i-1)=\text{lnko}(2^jq,2^{s_i}q_i)|2^{s_i-1}q_i=\frac{p_i-1}{2}$, azaz $2^{min(j,s_i)}\text{lnko}(q,q_i)=2^{min(j,s_i)}q_{i'}|2^{s_i-1}q_i$, mivel $s_1\le s_2\le\cdots\le s_r$, ezért $j\le s_1-1$ szükséges feltétel a megoldhatósághoz. Ekkor $\prod_{i=1}^{r}(2^jq_{i'})=2^{rj}\prod_{i=1}^{r}q_{i'}$ megoldás lesz. Ez minden $0\le j\le s_1-1$-re elmondható, ezért $|R_n|\le q_{1'}\cdots q_{r'}+\sum_{0\le j\le s_1-1}2^{rj}q_{1'}\cdots q_{r'}$. Nyilván $\varphi(n)=\prod_{i=1}^{r}(2^{s_i}q_i)=2^{s_1+\cdots s_r}q_1\cdots q_r$. Nézzük a következő hányadost:

$$\frac{|R_n|}{2^{s_1+\cdots+s_r}} \le \frac{q_{1'}\cdots q_{r'}(1+\sum_{0\le j\le s_1-1}2^{rj})}{2^{s_1+\cdots s_r}} = \frac{q_{1'}\cdots q_{r'}(1+\frac{2^{rs_1}-1}{2^r-1})}{2^{s_1+\cdots+s_r}} \le$$
$$\le q_{1'}\cdots q_{r'}\frac{2^{rs_1}+2^r-2}{(2^r-1)2^{rs_1}} \le q_{1'}\cdots q_{r'}(\frac{1}{2^r-1}+\frac{2^r-2}{(2^r-1)2^r}) =$$
$$= q_{1'}\cdots q_{r'}\frac{1}{2^{r-1}},$$

így $|R_n|\le\frac{\varphi(n)}{4}$, ha $r\ge 3$.

Marad az $r=2$ eset, ekkor $n=p_1p_2=1+2^sq$ és $p_1=1+2^{s_1}q_1, p_2=1+2^{s_2}q_2$. Ha $s_1<s_2$, akkor $\frac{|R_n|}{2^{s_1+s_2}}\le q_{1'}q_{2'}\frac{2^{2s_1}+2^2-2}{(2^2-1)2^{s_1+s_2}}\le q_{1'}q_{2'}\frac{2^{2s_1}+2}{3*2^{2s_1+1}}\le q_{1'}q_{2'}\frac{1}{4}$, amiből következik az állítás. Ha $s_1=s_2$, akkor $|R_n|=q_{1'}q_{2'}\frac{2^{2s_1}+2}{3}$, de $q_{i'}=\text{lnko}(q,q_i)$, mivel $n=(1+2^{s_1}q_1)(1+2^{s_2}q_2)=1+2^{s_1}(q_1+q_2+2^{s_1}q_1q_2)$, ezért $q=q_1+q_2+2^{s_1}q_1q_2$, így $q_{i'}=\text{lnko}(q,q_i)=\text{lnko}(q_1,q_2)$. Mivel $p_1\ne p_2$, ezért $q_1\ne q_2$, emiatt $q_{1'}q_{2'}=\text{lnko}(q_1,q_2)^2\le\frac{q_1q_2}{3}$. Így írható: $|R_n|\le\frac{q_1q_2}{3}\frac{2^{2s_1}+2}{3}\le\frac{1}{4}2^{2s_1}q_1q_2$, mert

$\frac{1}{3}\frac{2^{2s_1}+2}{3} \leq \frac{1}{4}2^{2s_1}$ könnyen belátható. Azaz $|R_n| \leq \frac{\varphi(n)}{4}$ ami kellett.

Következmény: Ha n páratlan szám és k darab véletlen $0 < b_1 < b_2 < \cdots < b_k < n, lnko(b_i, n) = 1$ számmal teszteljük n-et Miller-Rabin-nal, akkor legfeljebb $\frac{1}{4^k}$ valószínűséggel éli túl a tesztet.

2.6.5. Suyama trükk

25. Tétel (Suyama trükkje). *Ha d az n egy osztója, akkor szeretnénk gyors álprímtesztet $\frac{n}{d}$-re! Egy szokásos álprímteszt ehhez $O(\log(\frac{n}{d}))$ szorzást/osztást igényel (Fermat-,Euler-,erős álprímteszt), de van $O(\log d)$ szorzást/osztást használó algoritmus is.*

Bizonyítás: Számítsuk ki az $R = a^n \pmod n$-et. Ha $\frac{n}{d}$ prím, akkor kis-Fermat tétel szerint: $a^{\frac{n}{d}} \equiv a \pmod{\frac{n}{d}}$. Kongruenciát d-edik hatványra emelve kapjuk: $a^d \equiv a^n \equiv R \pmod{\frac{n}{d}}$ is igaz. Az R ismert, így csupán $a^d \pmod n$-et kell kiszámolni, ezt pedig $O(\log d)$ szorzással/osztással lehetséges.

Természetesen $O(\log n)$ szorzás itt is kellett az $R \equiv a^n \pmod n$ kiszámításához a trükknél, d minden egyes új osztó megtalálása esetén $\frac{n}{d}$-re a prímteszt csupán $O(\log d)$ szorzást használ az újabb $O(\log \frac{n}{d})$ helyett. Tipikusan még faktorizálatlan nagy Fermat számokra használják a trükköt.

2.7. Klasszikus prímtesztek

2.7.1. Lehmer tétele

26. Tétel. *Ha $n>1$ egész és létezik a egész, amelyre $a^{n-1} \equiv 1 \pmod n$ és minden $p|(n-1)$ prímre $a^{\frac{n-1}{p}} \not\equiv 1 \pmod n$, akkor n prím.*

Bizonyítás: $a^{n-1} \equiv 1 \pmod n$ miatt lnko(a,n)=1. Legyen p^k pontos osztója $n-1$-nek, ekkor $a^{\frac{n-1}{p}} \not\equiv 1 \pmod n$-ből következik, hogy $p^k|o_n(a)$. Mivel ez minden p-re igaz, ezért $n-1|o_n(a)$, lnko(a,n)=1 miatt az Euler-Fermat tétel szerint $a^{\varphi(n)} \equiv 1 \pmod n$, így a legkisebb jó kitevő az $\varphi(n)$ osztója, így $n-1|o_n(a)|\varphi(n) \leq n-1$ miatt $\varphi(n) = n-1$. Ekkor n prím.

2.7.2. Pepin tétele

27. Tétel. *Legyen $F_n = 2^{2^n}+1$ Fermat szám. Ekkor F_n prím ($n > 0$), akkor és csak akkor, ha $3^{\frac{F_n-1}{2}} \equiv -1 \pmod{F_n}$ teljesül.*

Bizonyítás: Ha $3^{\frac{F_n-1}{2}} \equiv -1 \pmod{F_n}$ akkor $3^{F_n-1} \equiv 1 \pmod{F_n}$ és $F_n - 1$-nek egyetlen prímosztója a 2, Lehmer tétel feltételei teljesülnek, így F_n prím.

Másik irány: tegyük fel, hogy F_n prím és $n > 0$, ekkor $3^{\frac{F_n-1}{2}} \equiv (\frac{3}{F_n}) \pmod{F_n}$, de $(\frac{3}{F_n}) = (-1)^{\frac{3-1}{2}\frac{F_n-1}{2}}(\frac{F_n}{3}) = (\frac{F_n}{3}) = (\frac{2}{3}) = -1$, hiszen $n > 0$-ra $F_n \equiv 1 \pmod 4$ és $F_n \equiv 2 \pmod 3$ teljesül.

Pepin tételét nagyon kis n-ekre lehet csak a gyakorlatban alkalmazni, hiszen még Schönhage–Strassen FFT algoritmusával is $O(4^n * n * \log n)$ időbe kerül a teszt. Valójában csak F_{20}, F_{24}-re éri meg alkalmazni, többi F_n-nek nemtrivi osztója ismert (vagy F_n prím) vagy nem futna le az algoritmus. F_{33} a legkisebb Fermat szám amiről jelenleg nem ismert, hogy prím vagy összetett.

fermat.c programommal Pepin tesztek különböző n-ekre, itt használtam, hogy F_n-nel lineáris időben lehet osztani, csak shiftelni kell. Így csak a (nagy számok) szorzásának hatékonyságán múlik a Pepin teszt. F_{n+1}-re a Pepin teszt FFT mellett legalább 4-szer több időbe kerül, mint F_n-re. Látható is, hogy a gmp valóban FFT-t használ a számok szorzásához:

Táblázat

n	< 15	15	16	17	18	19	20	21
$time$	$< 1s$	$3s$	$11s$	$59s$	$319s$	$1547s$	$7090s$	$31826s$

2.7.3. Lehmer tétel erősebb változata

28. Tétel. *Ha $n > 1$ egész, $n - 1 = R * F$, ahol $R < F$ és $a^{n-1} \equiv 1 \pmod n$ és minden $p|F$-re lnko($a^{\frac{n-1}{p}} - 1, n$)= 1 teljesül, akkor n prím.*

Bizonyítás: Legyen q az n tetszőleges prímosztója. $a^{n-1} \equiv 1 \pmod n$ miatt $a^{n-1} \equiv 1 \pmod q$ is igaz, de akkor lnko(a,q)=1. Feltétel szerint lnko($a^{\frac{n-1}{p}} - 1, n$)= 1 így lnko($a^{\frac{n-1}{p}} - 1, q$)= 1, azaz $a^{\frac{n-1}{p}} \not\equiv 1 \pmod q$ minden $p|F$-re. Így $F \leq o_q(a)$, de $o_q(a) \leq q - 1$ (kis Fermat tétel miatt). Így $F \leq q - 1$, de $R < F$ és $n = RF$ miatt $\sqrt{n} < F$, így $q > \sqrt{n}$ igaz az n minden q prímosztójára, de akkor n prím.

2.7.4. Proth tétele

29. Tétel. *Ha $n = t2^k + 1$, ahol $1 \leq t < 2^k$ és létezik a egész, amelyre $a^{\frac{n-1}{2}} \equiv -1 \pmod n$, akkor n prím.*

Bizonyítás: $F = 2^k$ legyen az erős Lehmer tételben.
Proth tételével találhatunk olyan nagy prímeket, melyekre p és $q = 2p - 1$ is prím, hiszen $p = t2^k + 1$ esetén $q = 2p + 1 = t2^{k+1} + 1$ vagy akár $r = 2q - 1 = 4p - 3 = t2^{k+2} + 1$ is prím. Ezeket (második típusú) Cunningham láncnak is nevezik. Ilyenre példa: $p = 387977793 * 2^{17864} + 1$, a lánc hossza 3, azaz $2p - 1$ és $4p - 3$ is prím.

2.7.5. Brillhart-Lehmer-Selfridge köbgyök tétele

30. Tétel. *Legyen* $F^2 < n = c_2F^2 + c_1F + 1 < F^3$, *ahol* $0 \leq c_1, c_2 < F$, *továbbá* d *olyan egész, amelyre* $d^{n-1} \equiv 1 \pmod{n}$ *és minden* $p|F$*-re* $lnko(d^{\frac{n-1}{p}} - 1, n) = 1$. *Ekkor* n *prím akkor és csak akkor, ha* $c_1^2 - 4c_2$ *nem négyzetszám.*

Bizonyítás: Lehmer tétel bizonyításához hasonlóan n minden prímosztója $kF+1$ alakú, de mivel $n < F^3$, azért n vagy prím vagy két prím szorzata. Tegyük fel, hogy $n = pq$, ahol $p = aF+1$ és $q = bF+1$. n nem osztható $F+1$-el, mert különben $F^2 < N$ miatt $F+1$ valódi osztó lenne, így n összetett, továbbá $n = c_2F^2+c_1F+1 \equiv c_2-c_1+1 \pmod{F+1}$, de $0 \leq c_1, c_2 < F$ miatt $-(F+1) < c_2 - c_1 + 1 < F + 1$, de $F + 1$-el osztva a maradék nulla, így $c_2-c_1+1 = 0$, ebből: $c_1^2-4c_2 = c_1^2-4(c_1-1) = (c_1-2)^2$ négyzetszám, ami kellett.

Így $2F + 1 \leq p, q$, de $n = pq < F^3$, ebből $p, q < \frac{F^2}{2}$, azaz $a, b < \frac{F}{2}$, írható $n = pq = (aF + 1)(bF + 1) = (ab)F^2 + (a + b)F + 1 = c_2F^2 + c_1F + 1 = n$, így $(ab)F + (a + b) = c_2F + c_1$, de $0 \leq a + b < F$, így $a + b = c_1$ és akkor $ab = c_2$ is igaz. Az a, b így megoldása az $X^2 - c_1X + c_2 = 0$ másodfokú egyenletnek, így mivel egész megoldásai vannak (és fő-polinom), ezért diszkriminánsa $c_1^2 - 4c_2$ négyzetszám, ami kellett.

Megfordítva: ha $c_1^2 - 4c_2 = L^2$ négyzetszám, akkor n nem prím. Valóban: $n = (\frac{c_1+L}{2}F + 1)(\frac{c_1-L}{2}F + 1)$ nemtriviális felbontás (c_1 és L paritása megegyezik).

2.7.6. Köbgyök tétel javítása

31. Tétel. $F^2 < N < KF^3$, *feltételek ugyanazok, mint a köbgyök tételben továbbá az is igaz, hogy* n*-nek nem osztója* $aF + 1$ *semmilyen* $1 \leq a \leq 2K - 1$*-re. Ekkor* n *prím akkor és csak akkor, ha* $c_1^2 - 4c_2$ *nem négyzetszám.*

Bizonyítás: Feltétel miatt prímosztók megint $aF + 1$ alakúak, $a < 2K$-ra nem osztó, így megint n prím vagy két prím szorzata, mert $(2KF)^3 > KF^3 > n$. Továbbá, ha $(aF+1)(bF+1) = n$ összetett, akkor $2KF+1 \leq bF+1$ miatt $aF+1 < \frac{KF^3}{2KF+1} < \frac{F^2}{2}$, így $a < \frac{F}{2}$ és hasonlóan $b < \frac{F}{2}$. Innen a bizonyítás ugyanaz. (Most persze $c_2 > F$ is lehet).

2.7.7. Bach tétele

32. Tétel. *Ha az általánosított Riemann sejtés (GRH) igaz, akkor, ha* n *páratlan összetett szám, akkor létezik olyan* b *alap, hogy* $lnko(b, n) = 1$ *és* $b < 2\log^2(n)$ *és bukja a Rabin-Miller tesztet.*

FFT-t használva a szorzáshoz ennek a műveletigénye $O(log^4(n))$, b alapokra elég csak a prímeket kipróbálni.

Sejtés (Bach, Huelsberger). Várhatóan $\frac{\log(n)\log\log(n)}{\log 2}$-ig is található ilyen b alap.

Előre ki is lehet számolni, hogy meddig kell elmenni a Rabin Miller tesztelésnél, hogy bebizonyítsuk, hogy n prím. [6] szerint (páratlan $n > 1$-re):

Ha $n < 2047$ és n az 2 alapú erős álprím, akkor n prím.
Ha $n < 1373653$ és n az $2, 3$ alapú erős álprím, akkor n prím.
Ha $n < 25326001$ és n az $2, 3, 5$ alapú erős álprím, akkor n prím.
Ha $n < 118670087467$ és n az $2, 3, 5, 7$ alapú erős álprím, akkor $n = 3215031751$ vagy n prím.
Ha $n < 2152302898747$ és n az $2, 3, 5, 7, 11$ alapú erős álprím, akkor n prím.
Ha $n < 3474749660383$ és n az $2, 3, 5, 7, 11, 13$ alapú erős álprím, akkor n prím.
Ha $n < 341550071728321$ és n az $2, 3, 5, 7, 11, 13, 17$ alapú erős álprím, akkor n prím.

Továbbá, de ez még nem publikált eredmény:
Ha $n < 3825123056546413051$ és n az $2, 3, 5, 7, 11, 13, 17, 19, 23$ alapú erős álprím, akkor n prím.
Ha $n < 2^{64}$ és n az $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37$ alapú erős álprím, akkor n prím.

3. fejezet

Prímfaktorizáció

Prímfaktorizáció során a feladat egy adott $n > 1$ egész prímtényezős felbontásának meghatározása. Az $n = p$ prím esete mutatja, hogy ez nem könnyebb probléma, mint a prímtesztelés. Turing gépen nem is ismert polinom idejű algoritmus rá, sőt jelenleg csupán Dixon algoritmusa bizonyítottan szubexponenciális futásidejű.

3.1. Kis számok faktorizációja, a próbaosztás

Faktorizálandó szám $n > 1$, ekkor
próbaosztás(n):
$N \leftarrow n, k \leftarrow 2$
Amíg $k * k \leq N$:
Ha $k|N$:
Amíg $k|N : N \leftarrow \frac{N}{k}$, kiír k
$k \leftarrow k + 1$
Ha $N > 1$, akkor kiír N

A kiírt számok az n prímosztói (multiplicitással), hiszen végig $N|n$, és k prím, mert a k-nál kisebb prímosztókat már kiírtuk így k nem lehet összetett.

Az eljárás végén, ha $N > 1$, akkor N prímosztója n-nek: osztó az eljárás miatt, de prím is, mert $\sqrt{N}$-ig nincs valódi osztója.

Algoritmus bonyolultsága: legyen $p_k(n) = n$-nek a k-adik legnagyobb prímosztója (multiplicitással), ha ilyen van egyébként legyen 1. Ekkor a fenti algoritmus $g(n) = max(p_2(n), \sqrt{p_1(n)})$ ideig fut. Ez javítható egyébként $f(n) = p_2(n) + O(\log^c(n))$-re, ha prímosztó találása esetén az osztások után kapott számról mindig eldöntjük, hogy prím-e egy polinom futásidejű algoritmussal.

$g(n) = max(p_2(n), \sqrt{p_1(n)}) \leq max(\sqrt{n}, \sqrt{n}) = \sqrt{n}$, hiszen $p_1(n) \leq n$ és $p_2(n) \leq$

$\sqrt{n}$. A javítás $f(n)$-re nem jelent lényeges gyorsítást, mert érvényes: legyen $E_n = \frac{1}{n}\sum_{k=1}^{n} f(k)$ várható futásideje az algoritmusnak, ekkor $E_n \leq \frac{1}{n}\sum_{k=1}^{n}\sqrt{k} \leq \frac{1}{n}n\sqrt{n} = \sqrt{n}$. Továbbá nagy n-re $E_n > \frac{1}{8}\frac{\sqrt{n}}{\log^2 n}$.

Bizonyítás: Tekintsük az $m = pq$ alakú számokat, ahol p, q (különböző) prímek és $\frac{1}{2}\sqrt{n} < p, q \leq \sqrt{n}$, ekkor persze $f(m) = f(pq) > \frac{1}{2}\sqrt{n}$ és $m = pq < n$. Az ilyen tulajdonságú m egészek száma $\binom{\pi(\sqrt{n})-\pi(\frac{1}{2}\sqrt{n})}{2} > \frac{n}{4\log^2 n}$ (prímszámtételt használva, elég nagy n-re). Így $E_n \geq \frac{1}{n}\frac{n}{4\log^2 n}\frac{1}{2}\sqrt{n} = \frac{1}{8}\frac{\sqrt{n}}{\log^2 n}$, ami kellett.

3.2. Pollard-rho módszer, Brent javításával

A módszer lényege, hogy, ha $f : \mathbb{Z} \to \mathbb{Z}$ egy közel random függvény, akkor az $x \to f(x) \pmod p$ leképezés $\frac{1}{2}$ valószínűséggel $O(\sqrt{p})$ lépésen belül ciklizálni fog a születésnap paradoxon miatt. Így, ha $p|n$ és $f^{(s)}(a) \equiv f^{(t)}(a) \pmod p$ akkor $p|f^{(s)}(a) - f^{(t)}(a)$, azaz $p|\text{lnko}(n, f^{(s)}(a) - f^{(t)}(a))$, ahol $f^{(0)}(a) = a$ és $f^{(l)}(a) = f(f^{(l-1)}(a))$. Ha az $\text{lnko}(n, f^{(s)}(a) - f^{(t)}(a)) \neq n$, akkor nemtriviális osztót találtunk, ami az esetek nagy részén így is van, különben n összes prímhatvány osztóját ugyanabban az iterációban találtuk volna meg. Ha az lnko$= n$ lenne mégis, akkor az algoritmust egy másik kezdőértékkel, vagy egy másik f-fel újraindítható. Floyd cikluskereső algoritmusát használta Pollard: ciklizálás miatt lesz olyan s is, amelyre $f^{(2s)}(a) \equiv f^{(s)}(a) \pmod p$, így egyszerre két sorozatot futtatva a ciklus megtalálható:

$Floyd(n, f, a)$

$x \leftarrow a, y \leftarrow a, d \leftarrow 1$

Amíg $d = 1$:

$x \leftarrow f(x)$

$y \leftarrow f(f(y))$

$d \leftarrow$lnko$(x - y, n)$

Ha $d = n$ lesz az algoritmus végén, akkor nem talált valódi osztót, egyébként igen. Brent módszere már csak egy sorozatot használ:

$Brent(n, f, a)$

$x \leftarrow a, y \leftarrow a, d \leftarrow 1, k \leftarrow 1, l \leftarrow 0$

Amíg $d = 1$:

Ha $k = l$, akkor $k \leftarrow 2k, l \leftarrow 0, y \leftarrow x$

$x \leftarrow f(x)$

$d \leftarrow$lnko$(x - y, n)$

$l \leftarrow l + 1$

Ha $d \neq n$, akkor nemtriviális osztót talált.

Brent magasabb indexű tagnál találja meg a ciklizálást, de összességében kevesebbet számol, mint Floyd. Brent az $f^{(s)}(a) \equiv f^{(t)}(a) \pmod{p}$ ciklizálást az $2^w + |t-s|$ tagnál veszi észre, ahol $2^w > s, t$ és w a legkisebb ilyen egész. Általában $f(x) = x^2 + b$ függvényt választják a Pollard módszerhez.

33. Tétel. *Az $f(x) = bx + c$ gyenge választás a Pollard módszerben.*

Bizonyítás: Legyen $T(k) = f^{(k)}(a)$ és $U(k) = T(k) + w$, ahol $w \equiv c(b-1)^{-1} \pmod{p}$ (ha ilyen nincs, akkor $\text{lnko}(b-1, n) > 1$ lesz). Ekkor $U(k+1) = T(k+1) + w = bT(b) + c + w = b(U(k) - w) + c + w = bU(k) + w(1-b) + c = bU(k)$, azaz $U(k) = b^k U(0)$, így $T(k) \equiv b^k U(0) + w = b^k(T(0) + w) + w \pmod{n}$, amit, ha $\pmod{p}$ nézünk ($p|n$ legyen), akkor látható, hogy $o_b(p)$ lesz a ciklus hossza, ami $\sqrt{p}$-nél általában sokkal nagyobb.

34. Tétel. *Ha n minden p prímosztójára $d|p-1$, akkor $f(x) = x^d + c$ gyorsabb módszer a Brenthez, mint $g(x) = x^2 + c$, ha d nagy.*

Bizonyítás: $f(x) = x^d + c$ kiszámolása $O(\log d)$ osztást/szorzást igényel, de d-edik hatványmaradékból csupán $\frac{p-1}{d}$ van csak, mert $d|p-1$, így g-hez képest ez $O(\frac{\sqrt{d}}{\log d})$-szer gyorsabb módszert ad. Lehmer tétele lehetőséget ad arra, hogy ilyen d értéket találjunk.

Brent módszerében az lnko kiszámolása a leglassabb, ennek javítása: $v \leftarrow v(x-y) \pmod{n}$ legyen ($v = 1$ az algoritmus elején) és csak minden például 40-edik iterációban ellenőrizzük, hogy $\text{lnko}(n, w) > 1$. Valójában a 40 helyett egy nem konstanst kell választani, mert az lnko számolása nem konstansszor lassabb, mint egy szorzás/osztás elvégzése.

3.3. Pollard P-1 módszere, nagy prímvariánssal

Módszere a kis Fermat tételt használja: legyen $p|n$ prím és $\text{lnko}(b, p) = 1$. Ekkor $p-1|e$ esetén $b^e \equiv 1 \pmod{p}$, így $p|\text{lnko}(n, b^e - 1)$ teljesül.

Ha $p|n$ és $(p-1)$ minden prímhatvány osztója kicsi, akkor gyorsan tudunk találni ilyen e számot. Válasszunk egy B egészet és egy a egész számot a Pollard módszerhez:

Pollard(n, B, a)
$\forall q \leq B$ prímre
  q^k legyen a legnagyobb B-nél nem nagyobb q hatvány
  $a \leftarrow q^k \pmod{n}$
$g \leftarrow \text{lnko}(a-1, n)$
Ha $1 < g < n$, akkor valódi osztót talált.

Ha $g = n$, akkor kisebb B értékkel próbálkozzunk.
Ha $g = 1$, akkor nagyobb B értékkel.

Így Pollard módszere akkor eredményes, ha $(p-1)$ minden prímhatvány osztója legfeljebb B, mert a kitevője az eredeti a számnak pontosan a B-nél nem nagyobb prímhatványok szorzata, így $p-1$ osztja a kitevőt. Speciális alakú számoknál eleve tudunk mondani $(p-1)$-nek valódi osztóját, így ezeket érdemes külön belevenni, ha nagyobbak B-nél. Például, ha r prím, akkor $2^r - 1$ Mersenne szám minden prímosztójára $r|p-1$ teljesül. Ha F_n az n-edik Fermat szám ($n > 1$), akkor minden prímosztójára $2^{n+2}|(p-1)$.

Pollard módszer nagy prím variánssal: akkor működik, ha $(p-1)$ minden prímhatvány osztója egy kivételével kicsi, a legnagyobb prímosztó sem túl nagy. Az előző algoritmust futtassuk le B_1 korláttal. Ha az lnko= 1 lenne, akkor válasszunk egy B_2 korlátot. Egyszerűség kedvéért B_1-et vegyük az algoritmusban prímnek.

Pollard nagy prímmel(n, B_1, B_2, a)
Pollard(n,B1,a) nem talált valódi osztót.
$S \leftarrow 1, T \leftarrow a^{B_1} \pmod n$, itt az a már a Pollard(n, B_1, a) algoritmussal kapott a
$\forall B_1 < q \leq B_2$ prímre:
$T \leftarrow T * a^{d_k} \pmod n$, ahol $d_k = p_k - p_{k-1}$
$S \leftarrow S * (T - 1)$
$g \leftarrow \text{lnko}(S, n)$
Ha $1 < g < n$, akkor valódi osztót talált.
Ha $g = n$, akkor kisebb B_2 értékkel próbálkozzunk.
Ha $g = 1$, akkor nagyobb B_2 vagy B_1értékkel.

Az $a^{d_k} \pmod n$ értékeket persze nem számoljuk ki mindig újra, $d_k = O(\log^2 k)$ sejtés szerint kevés lehetőség van rá, így memóriában tárolható.

Nagy prímvariáns nélkül a Pollard $O(B_1)$ szorzást/osztást használ, míg nagy prímvariánsnál $O(\frac{B_2}{\log B_2})$ szorzást/osztást. Így, ha azt szeretnénk, hogy a 2 lépés ugyanannyi ideig tartson, akkor az optimális választás $B_2 = O(B_1 \log B_1)$. Leggyakrabban azonban $B_2 = 100B_1$ a választás.

pollard.c programmal például a (random) $n = 44760975078749393$-re $B1 = 100$, $B2 = 10000$ választással megtalálja $p = 84417343$ prímosztót, valóban itt $p - 1 = 2*3*17*19*43*1013$, ezzel egyébként faktorizáltunk is, mert $\frac{n}{p}$ is prím. pollardgmp.c programmal $6^{97} - 1$ kofaktora:
$n = 561119822949401309240341400846362000627333962829157368777$, erre $B1 = 10^6$, $B2 = 10^8$ választással kb. 4 másodperc alatt megtalálja

$p = 189696531812993971752 71$-et, hiszen $p - 1 = 2 * 5 * 7 * 17 * 97 * 439 * 2531 * 3491 * 42367631$ (ezt először Atkin és Rickert találta meg 1984-ben).

3.4. Polinom idejű faktorizációja egyszerre sok (kis) számnak (Bernstein)

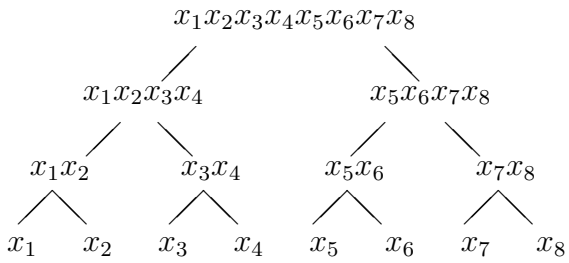
Először nézzünk meg néhány, nem annyira ismert módszert, hogyan lehet felgyorsítani klasszikus algoritmusokat amikor sok számunk van. Az algoritmusokhoz lásd [4]-et.

Legfeljebb b bites számokra szorzás/osztás legyen elvégezhető legfeljebb $b\mu(b)$ időben. Jelenlegi leggyorsabb módszerrel (FFT-vel) ez $\mu(b) = O(\log b \log(\log b))$.

Sok szám gyors összeszorzásához egy úgynevezett szorzatfát használhatunk. Ennek felépítése és műveletigénye: $p_1, p_2, \cdots, p_m$ legyenek az összeszorzandó számok, amik összesen b bittel írhatóak le.

Algoritmus:
Szorzatfa$(p_1, p_2, \cdots, p_m)$:
Ha $m = 1$, akkor Kiír p_1
Egyébként: Szorzatfa$(p_1, p_2, \cdots, p_{\lfloor \frac{m}{2} \rfloor})$
Szorzatfa$(p_{\lfloor \frac{m}{2} \rfloor + 1}, \cdots, p_m)$
Gyökér← két gyerek szorzata

$m = 8$-ra például a szorzatfa:



Állítás: Ha $m \leq 2^k$ és a számok szorzata b bites, akkor a szorzatfa elkészítése legfeljebb $kb\mu(b)$ időbe kerül.

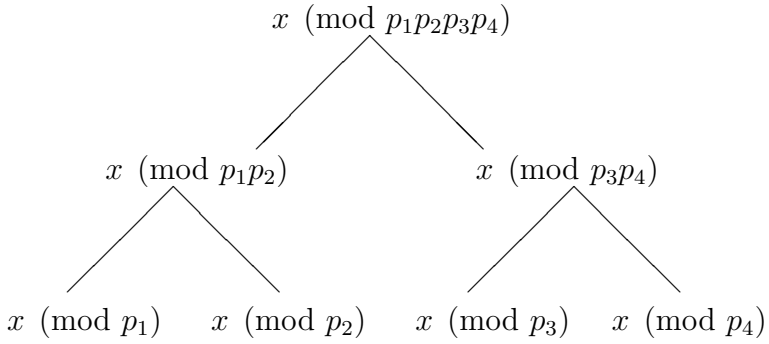
Bizonyítás: $k = 1$-re triviális, hiszen csak két legfeljebb b bites szám szorzatát kell kiszámítani, ami $b\mu(b)$ időbe kerül. Ha $k > 1$, akkor indukcióval és az algoritmussal: legyen a bites a bal részfa gyökerébe írt szám, akkor a jobb részfa gyökerében legfeljebb $b - a$ bites szám van, így indukcióval a műveletigény (használjuk azt is, hogy a két részfában már csupán legfeljebb 2^{k-1} darab számot kell összeszorozni), így az indukció működik: $(k-1)a\mu(a) + (k-1)(b-a)\mu(b-a) + b\mu(b) \leq$

$(k-1)a\mu(b)+(k-1)(b-a)\mu(b)+b\mu b = kb\mu(b)$, ami kellett.

(Osztófa algoritmus.) Adott x szám és P pozitív egészek halmazához egy gyors algoritmus $\{p \in P : x \equiv 0 \pmod{p}\}$ kiszámításához. T a P-hez tartozó szorzatfa.

1. $u \leftarrow T$ gyökere
2. $r \leftarrow x \pmod{u}$
3. Ha T levél, akkor kiír u, ha $r \equiv 0$. Stop.
4. T bal részfájára alkalmazzuk az osztófa algoritmust.
5. T jobb részfájára alkalmazzuk az osztófa algoritmust.

Példa, ha $\#P = 4$, akkor az osztófa:



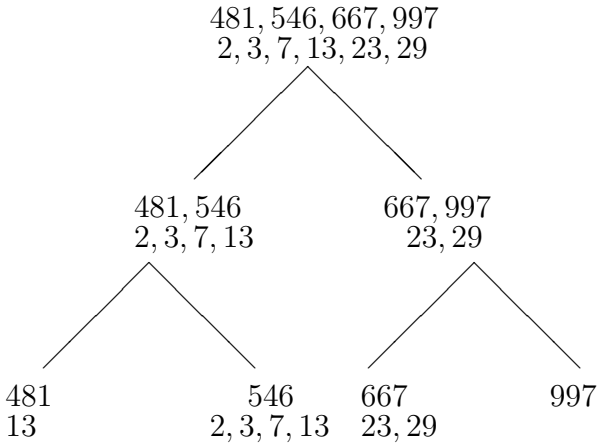
35. Tétel. *Az osztófa algoritmus legfeljebb* $(max(e,b)+2kb)\mu(max(e,b))$ *időben elvégezhető. Ahol* P *legfeljebb* 2^k *elemű* ($k \geq 0$), $x < 2^e$ ($e \geq 0$).

Bizonyítás: Ha $k = 0$, akkor P egy elemű, csak egy osztást kell végeznünk, ez $max(e,b)\mu(max(e,b))$ időbe kerül. Ha $k > 0$, akkor indukcióval 4. lépés legfeljebb $(max(b_1,b)+2(k-1)b_1)\mu(max(b,b_1)) = (b+2(k-1)b_1)\mu(b)$ idő, hasonló igaz a jobb oldali részfára. Az össz műveletigény így: $max(e,b)\mu(max(e,b)) + (2b+2(k-1)b)\mu(b) \leq (max(e,b)+2kb)\mu(max(e,b))$.

Algoritmus. Legyen N pozitív egészek egy halmaza, és P különböző prímek egy halmaza. Feladat: minden egyes $n \in N$-re $\{p \in P : n \equiv 0 \pmod{p}\}$ meghatározása, azaz n-nek a P-beli prímosztóinak megkeresése.

1. Ha P üres, akkor stop.
2. $x \leftarrow \prod_{n \in N} n$ (szorzatfával).
3. P-nek a szorzatfájának a kiszámítása.
4. $Q \leftarrow \{p \in P : x \equiv 0 \pmod{p}\}$ az osztófa algoritmussal.
5. Ha N egy elemű, akkor Q már az $n \in N$ prímosztóiból áll. Stop.
6. Legyen M az N részhalmaza, $\#M = \frac{\#N}{2}$
7. (M, Q)-ra az algoritmus alkalmazása.
8. $(N-M, Q)$-ra az algoritmus alkalmazása.

Példa: legyen $P = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}$, míg $N = \{841, 546, 741, 997\}$-re az algoritmus, (adott számok alá mindig a P halmaz összes osztója kerül), a példa egyébként olyan, hogy ezzel már faktorizálunk is, mivel a legnagyobb N-beli szám négyzetgyökéig az összes prímet tartalmazó halmaz a P.



36. Tétel. *Legyen adott $O(\frac{\sqrt{n}}{\log^{O(1)} n})$ darab x_i szám ($x_i < n$), ekkor ezek $O(\sqrt{n} \log^{O(1)} n)$ időben faktorizálhatóak, azaz egy számra lebontva ez polinom időt jelent.*

Bizonyítás: Nyilván elég megtalálni $\sqrt{n}$-nél nem nagyobb prímosztókat minden x_i számhoz. Az előbbi algoritmusban legyenek p_i számok a $\sqrt{n}$-nél nem nagyobb prímek. Ekkor $k = O(\log n)$, továbbá $b, e = O(\sqrt{n} \log^{O(1)} n)$.

bernstein.c programommal a sok (kis) szám faktorizációjára egyetlen példa: 1 millió darab 50 bites (random) számot 63 másodperc alatt faktorizál, ugyan $750MB$ Ram kellett hozzá.

3.4.1. Faktoriális, binomiális együttható gyors kiszámítása

Computeralgebrai programoknál az egyik legnépszerűbb teszt, hogy milyen gyorsan tudják kiszámítani $n!$ értékét, nagy n-re, például $n = 10^7$-re. Definícióból számolva $n! = 1 * 2 * \cdots * n$ értékét $n - 1$ szorzást igényel, ami meglehetősen lassú: ha n még elfér egy ábrázolt szóban, mondjuk $n < 2^{32}$, akkor $k \leq n$-nel szorozva egy egészt megy lineáris időben, így $n!$ kiszámításánál egy szorzás $\log(n!) = O(n \log(n))$ miatt $O(n \log(n))$ művelet, így összesen $O(n^2 \log(n))$ időbe kerül ez a módszer. Ennél lényegesen gyorsabb szorzatfát használni, azaz $n! = Szorzatfa(1, 2, \cdots, n-1, n)$-nel, ez $O(n(\log(n))^2 \mu(n \log(n)))$ műveletbe kerül.

Ennél is gyorsabb algoritmust adott még Schönhage, szintén szorzatfával, de négyzetreemelések is vannak benne. Ez nem csak $n!$ kiszámítására működik, hanem tetszőleges N-re, legyen $N = \prod_{i=1}^{r} {p_i}^{e_i}$ (ez nem szükségképpen az N prímtényezős

felbontása). Az algoritmus:
$e = max(e_1, e_2, \cdots, e_r)$ ennyi bites
$R \leftarrow 1$
Amíg $e \geq 0$:
    $H = \{i : e_i$-nek az e-dik bitje $1\}$
    $S \leftarrow$ Szorzatfa$(p_i : i \in H)$
    $R \leftarrow R^2 * S$
    $e \leftarrow e - 1$
return R

Faktoriális kiszámításához: $n! = \prod_{p \leq n} p^{e_{n,p}}$, ahol Legendre formulája szerint: $e_{n,p} = \sum_{i>0} \lfloor \frac{n}{p^i} \rfloor$. Tehát először n-ig meghatározom a prímeket $O(n \log\log(n))$ időben Eratoszthenész-i szitával, majd az $e_{n.p}$ kitevőket. Végül alkalmazom Schönhage algoritmusát, itt vegyük észre, hogy S nem lehet túl nagy, ugyanis $S \leq 4^n$, mert $S \leq \prod_{p \leq n} p \leq 4^n$ ismert egyenlőtlenség igaz. Továbbá az $R \leftarrow R^2 * S$ szorzás elvégzése előtt még összesen $e+1$-szer fogjuk R-et négyzetre emelni, de mivel a végén $n!$ értékét kapjuk meg, így $R \leq (n!)^{\frac{1}{2^{e+1}}}$. Ezzel belátható, hogy $O(n \log(n) \mu(n \log(n)))$ időbe kerül ez az algoritmus $n!$ kiszámításához. Igazából ennél gyorsabbat nem is nagyon remélünk, hogy létezik, mert ugyanennyi időbe kerül két $n!$ nagyságú szám összeszorzása. Ezt az algoritmust írtam meg, amit az Mpir 1.3.0 verziótól használ, nem túl kicsi n esetén.

Binomiális együtthatók gyors kiszámítása is izgalmas, de ott Schönhage módszere nem hasznos, ugyanis nincsenek egy binomiális együtthatónak nagy prímhatvány osztói, mert ismert, hogy $\binom{n}{k}$-nak n-nél nem nagyobbak a prímhatvány osztói. Rekurzív algoritmus $\binom{n}{k}$ kiszámítására: $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ itt is elég lassú, habár például a gmp sokáig ezt a primitív algoritmust használta. A szorzatfa algoritmus itt is hatékony, de ehhez először szorzat alakba írom $\binom{n}{k}$-t.
GyorsBinom(n, k):
$i = 0$-tól $(k-1)$-ig:
    $A[i] \leftarrow (n - i)$
$q = p^e \leq k$ prímhatványokra:
    $n \equiv r \pmod{q}$, ahol $0 \leq r < q$
    $t \leftarrow \lfloor \frac{k}{q} \rfloor$
    $i = 1$-től t-ig:
        $A[r] \leftarrow \frac{A[r]}{p}$
        $r \leftarrow r + q$
return Szorzatfa(A)

Gyakorlatilag az működik itt, hogy a binomiális együttható egész, és így az $\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{1 2 \cdots k}$-ban tudunk egyszerűsíteni minden $q \leq n$ prímhatvánnyal, különben nem kapnánk egész számot. És amikor egyszerűsítünk a számlálóban, akkor egy számtani sorozat mentén megyünk, mert minden q-adik osztható q-val.

Binomiális együttható nem csak egész n esetén van értelmezve. PARI-GP, Sage is meglehetősen lassú volt, amikor lebegőpontos valós/komplex r esetén kellett $\binom{r}{k}$-t számolni. Ötletem alapján már: $\binom{r}{k} = \frac{\Gamma(r+1)}{\Gamma(k+1)\Gamma(r-k+1)}$ alapján számolják, lásd `http://trac.sagemath.org/sage_trac/ticket/3309`. (Itt Γ az úgynevezett gamma függvény.)

Faktoriális és binomiális együtthatók közös általánosítása a mutlinomiális együtthatók: $multinomial(a_1, a_2, \cdots, a_k) = \frac{(\sum_{i=1}^{k} a_i)!}{\prod_{i=1}^{k}(a_i)!}$, ez is számolható lenne gyorsan szorzatfával. Ezekből is látható, hogy egy hatékony dolog a szorzatfa, ami tulajdonképpen az oszd meg és uralkodj elvet használja. Utolsó alkalmazás: ha a kínai maradéktételnél sok kongruenciánk van, akkor hatékonyabb oszd meg és uralkodj elvével megoldani, mint sorban egymás után.

newbinuiui.c programmal például $\binom{10^8}{5*10^7}$ kiszámítása (szorzatfával) kb. 13 másodperc.

newfacui.c programmal 10000000! kiszámítása (fenti módon) kb. 7 másodperc. Ez utóbbi programom az mpir könyvtár része lett.

Osztófa algoritmus nagyon hasznos nextprime(n) kiszámításához, mert ekkor egyszerre sok (kis) p prímre tudjuk gyorsan kiszámítani $r_p = n \pmod{p}$ értékét és így ki tudjuk szitálni az n-nél nagyobb számokat, ha p-vel osztható. Az ötlet régi, gmp program azonban nem használ osztófát. newnextprime.c programmal például nextprime(10^{2000}) kiszámítása kb. 24 másodperc, míg a gmp beépített algoritmusa kb. 53 másodperc alatt határozza meg. A program $O(\log^2(n))$-ig veszi a prímeket. Az RSA módszerhez is szokták használni a nextprime() függvényt: véletlen t_1, t_2 egészekhez $p = nextprime(t_1)$, $q = nextprime(t_2)$ választással.

Utolsó alkalmazás: (sok) kongruenciából álló szimultán kongruenciarendszer megoldása oszd meg és uralkodj elvével. Két külön eset van, ha a modulusok relatív prímek: crtcoprime.c (ekkor biztosan van megoldás a kínai maradéktétel miatt). Míg az általános esetre: crt.c (nem biztos, hogy van megoldás).

3.5. Dixon faktorizációs algoritmusa

Jelenleg Dixon algoritmusa az egyetlen bizonyítottan szubexponenciális futásidejű algoritmus egész számok faktorizációjára. Tulajdonképpen a faktorbázis algoritmus

prototípusa. Ötlete a más algoritmusokban is használt Fermat faktorizációs trükkjét használja: olyan x, y egészeket keres, amelyekre $n|x^2-y^2$, de $n \not| x \pm y$, ugyanis ekkor a (polinom futásidejű) Euklidész algoritmussal d=lnko$(x+y, n)$ egy nemtriviális faktora n-nek, és ekkor d-re, $\frac{n}{d}$-re iterative megismételve az eljárást (feltéve, hogy nem prímek) n-et faktorizáljuk.

8. Definíció (Faktorbázis). *$B = \{p_1, \cdots, p_k\}$ (ahol $p_1 < p_2 < \cdots < p_k$) halmazt faktorbázisnak hívunk, ha $p_1, \cdots, p_k$ prímek, de $p_1 = -1$ is lehet.*

Legyen $n \in \mathbb{N}$ rögzített, $a \in \mathbb{N}$ esetén azt mondjuk, hogy a az B szám az adott n-re nézve, ha $a^2 \pmod n$ legkisebb abszolútértékű maradéka B-beli számok szorzataként írható. Például legyen $n = 4633$ és $B = \{-1, 2, 3\}$, ekkor $a = 67, 68, 69$ B számok, ugyanis:
$67^2 \equiv -2^4 * 3^2 \pmod n$
$68^2 \equiv -3^2 \pmod n$
$69^2 \equiv 2^7 \pmod n$

37. Tétel (Faktorbázis algoritmus). *Legyen n és $B = \{b_1, \cdots, b_k\}$ faktorbázis adottak. Keressünk $k+1$ darab B számot: $a_1, \cdots, a_{k+1}$. Legyen ${a_i}^2 \equiv {b_1}^{\alpha_{i,1}} \cdots {b_k}^{\alpha_{i,k}} \pmod n$ $(1 \leq i \leq k+1)$. Tekintsük a következő $k+1$ vektort:*
$u_i = (\beta_{i,1}, \cdots, \beta_{i,k})$, ahol $\beta_{i,j} = \alpha_{i,j} \pmod 2$, ahol $0 \leq \beta_{i,j} < 2$, ez összesen $k+1$ vektor az $\mathbb{F}_2$ feletti k-asok k dimenziós vektorteréből, ezért lineárisan összefüggnek, azaz a nullvektor előáll a vektorok nemtriviális lineáris kombinációjaként, azaz léteznek $\epsilon_1, \cdots, \epsilon_{k+1} \in \{0, 1\}$, hogy $\epsilon_1 u_1 + \cdots + \epsilon_{k+1} u_{k+1} = 0$, de
$\prod_{i=1}^{k+1} {a_i}^{2\epsilon_i} \equiv \prod_{i=1}^{k+1} (\prod_{j=1}^{k} {b_j}^{\alpha_{i,j}})^{\epsilon_i} \equiv \prod_{j=1}^{k} \prod_{i=1}^{k+1} {b_j}^{\epsilon_i \alpha_{i,j}} \equiv \prod_{j=1}^{k} {b_j}^{\sum_{i=1}^{k+1} \epsilon_i \alpha_{i,j}} \pmod n$, itt $\epsilon_1 u_1 + \cdots + \epsilon_{k+1} u_{k+1} = 0$ miatt $\sum_{i=1}^{k+1} \epsilon_i \alpha_{i,j} = 2k_j$ páros szám. Azaz: $(\prod_{i=1}^{k+1} {a_i}^{\epsilon_i})^2 \equiv (\prod_{j=1}^{k} {b_j}^{k_j})^2 \pmod n$, tehát találtunk olyan x, y számokat, amelyre $x^2 \equiv y^2 \pmod n$. Ha még $x \not\equiv \pm y \pmod n$ is teljesül, akkor készen vagyunk. Egyébként további B számokat keresünk.

Előző példában: $k = 3$ (faktorbázis 3 elemű). Továbbá a megfelelő vektorok:
$(1, 4, 2) \to u_1 = (1, 0, 0)$
$(1, 0, 2) \to u_2 = (1, 0, 0)$
$(0, 7, 0) \to u_3 = (0, 1, 0)$

Nekünk elvileg $k + 1 = 4$ darab B szám kellene, de nemtriviális lineáris kombinációt kevesebb vektor között is találhatunk, mint most is: $u_1 + u_2 = 0$ a kételemű test felett. Azaz $\epsilon_1 = \epsilon_2 = 1, \epsilon_3 = 0$, továbbá

${a_1}^2{a_2}^2 = 67^2 * 68^2 \equiv (-1) * 2^4 * 3^2 * (-1) * 3^2 \equiv 2^4 * 3^4 \pmod{n}$
$(67 * 68)^2 \equiv (2^2 * 3^2)^2 \pmod{n}$, azaz $4556^2 \equiv 36^2 \pmod{n}$, de $4556 \not\equiv \pm 36 \pmod{n}$, így $\text{lnko}(4556+36, n) = 41$ nemtriviális osztója $n = 4633$-nak. (Ezzel egyébként már faktorizáltunk is: $4633 = 41 * 113$).

A nemtriviális lineáris kombináció megtalálásához Gauss eliminációt használhatunk. Tulajdonképpen a Gauss elimináció itt nem is olyan lassú, mert 2 elemű test fölött dolgozva 4 vagy 8 byte-os változóba 32, illetve 64 mátrixelemet tudunk belepakolni, továbbá a Gauss elimináció csak sorokat ad egymáshoz (illetve sorokat/oszlopokat cserél fel, ha szükséges), ami a kételemű test felett a kizáró vagy műveletének felel meg.

3.6. Lánctört faktorizáció (CFRAC)

Az algoritmus ötlete Kraitchik-től származik (1920-as évek), vagy még korábban Legendre-től. Lehmer és Powers írta le az algoritmust (1930-as évek) Számítógépre 40 évvel később írta Morrsion és Brillhart, sikeresen faktorizálták vele $F_7 = 2^{128} + 1$-et 1970-ben. Kvadratikus szita megjelenéséig a leggyorsabb faktorizációs algoritmus volt nagy számokra, még 60 jegyű számokat is faktorizáltak vele. Még ma is a leggyorsabb 10-20 jegy között.

CFRAC az $N > 1$ szám faktorizálásánál olyan kis abszolútértékű w számokat keres, amelyekre $x^2 \equiv w \pmod{N}$-nek egy megoldását találja. Mivel w kicsi, itt $O(\sqrt{N})$ lesz, ezért nem kis valószínűséggel w a faktorbázisban levő prímek szorzata, így a faktorbázis algoritmus működik. Mivel $x^2 \equiv w \pmod{N}$, ezért $x^2 = w + kNd^2$, ahol k, d egész. Ebből $(\frac{x}{d})^2 - kN = \frac{w}{d^2}$, így $\frac{x}{d}$ éppen $\sqrt{kN}$ egy jó approximációját adja, ezeket pedig így lánctörtekkel keresünk: legyen kN nem négyzetszám, akkor ismeretes, hogy végtelen sok p, q pár van, amelyre: $|\sqrt{kN} - \frac{p}{q}| \leq \frac{1}{q^2}$, ebből:
$w = p^2 - q^2kN = (p - q\sqrt{kN})(p + q\sqrt{kN})$, így
$|w| = |p - q\sqrt{kN}||p + q\sqrt{kN}| \leq \frac{1}{q}(|p - q\sqrt{kN}| + 2q\sqrt{kN}) \leq \frac{1}{q}(1 + 2q\sqrt{kN}) \leq 1 + 2\sqrt{kN}$, tehát kicsi, $O(\sqrt{kN})$ nagyságrendű.

Egy trükk a faktorbázis méretének a felezéséhez: $p^2 = w + kNq^2$, ezért minden $r|w$ (r prím) esetén $p^2 \equiv kNq^2 \pmod{r}$, így $(\frac{p}{q})^2 \equiv kN \pmod{r}$, azaz kN kvadratikus maradék mod r, ezáltal adott korlátig kb. a prímek fele lesz a bázisban, továbbá a -1 is, hiszen a lánctörtek alsó-felső közelítést adnak, emiatt w felváltva lesz pozitív/negatív.

Lánctört előállítása az $S = \frac{a+\sqrt{b}}{c}$ irracionális számnak (a, b, c egészek, $S > 0$). Legyen $\frac{h_i}{k_i}$ az i-edik tört. Ismert, hogy ekkor az $S > 0$-hoz tartozó lánctört előállításánál a rekurzió:

$h_{-2} = 0, h_{-1} = 1, h_n = a_n h_{n-1} + h_{n-2}$
$k_{-2} = 1, k_{-1} = 0, k_n = a_n k_{n-1} + k_{n-2}$
Továbbá
$m_0 = 0$
$d_0 = 1$
$a_0 = \lfloor \sqrt{S} \rfloor$
$m_{n+1} = d_n a_n - m_n$
$d_{n+1} = \frac{S - m_{n+1}^2}{d_n}$
$a_{n+1} = \lfloor \frac{\sqrt{S} + m_{n+1}}{d_{n+1}} \rfloor = \lfloor \frac{a_0 + m_{n+1}}{d_{n+1}} \rfloor$

Minket csak $\pmod{kN}$ érdekelnek a számok, ami a fentiek alapján számítható. Mivel a kvadratikus irracionális szám lánctört kifejtése periodikus, így a periódus elérése után új w értéket nem kapunk, így, ha túlságosan kicsi lenne $\sqrt{N}$ periódusa, akkor kN-et próbáljuk faktorizálni (például F_n periódusa csak 1, ha $n > 0$). Ilyenkor az a kedvező dolog is történhet, hogy sűrűbben kapjuk a w értékeket, mert kN gyakrabban lesz kvadratikus maradék.

CFRAC futási ideje (sejtés): $L_n[\frac{1}{2}, \sqrt{2}]$, de csak akkor, ha a faktorbázis algoritmusban a Gauss eliminációt lecseréljük.

cfrac.c programmal néhány faktorizáció: $F_7 = 59649589127497217 * p22$ felbontás megtalálása kb. 14 másodperc alatt. $7^{91} - 1$ egy kofaktorának:
$c51 = 825172026552223998772354571149416627928134660979273$ faktorizációja kb. 404 másodperc alatt: $c51 = 231410451435538144122809 * p28$ ezt még Wagstaff találta meg először 1983-ban. Míg a jelenlegi cfrac rekord a Cunningham tálázatban $5^{171} + 1$ egy kofaktorának faktorizációja:
$c62 = 515351290468951560075798538686416202987841688486600693903948 47 =$
$= 626747642757850246145380 9 * p37$ kb. 8494 másodpercben, ezt először McCurdy és Wunderlich találta meg 1986-ban.

3.7. RSA és törései

Az RSA nyílt kulcsú titkosítást R. L. Rivest, A. Shamir és L. Adleman 1978-ban hozták nyilvánosságra. Módszer alapja: Aliz és Bob szeretne kommunikálni egymással, ehhez választanak két (nagy) p, q prímet. Legyen $n = pq$, választanak továbbá egy e egészt, melyre $lnko(e, \varphi(n)) = 1$ teljesül. A kibővített Eukleidészi algoritmussal így találnak egy d egészt, melyre $de \equiv 1 \pmod{\varphi(n)}$. Titkos $0 \leq m < n$ üzenetet szeretné Aliz elküldeni Bobhoz, ehhez kiszámítja $c \equiv m^e \pmod{n}$ értéket, és ezt küldi el Bobnak. Aki a fejtéshez d-edik hatványra emelve megkapja az m üzenetet, ugyanis: $c^d \equiv (m^e)^d \equiv m^{de} \equiv m^{k\varphi(n)+1} \equiv m \pmod{n}$ teljesül, hiszen $de \equiv 1$

(mod $\varphi(n)$), így $de = k\varphi(n) + 1$ alkalmas k egészre, és használtuk az Euler-Fermat tételt.

Az RSA-nál nyilvános kulcs az (n, e) pár, titkos: p, q, d. A módszernél mindent gyorsan, polinom időben tudnak kiszámolni: p, q prímeket választják véletlenül b bites számok közül, ezt polinom időben megtehetik az AKS teszt miatt és amiatt hogy a prímek sűrűn vannak, várhatóan $O(b)$ teszt kell egy prím megtalálásához a prímszámtétel miatt. $n = pq$ és $\varphi(n) = (p-1)(q-1)$ (gyorsan) kiszámítható. Fejtéshez a d egészt a kibővített Eukleidészi algoritmus polinom időben meghatározza vagy alternatív út: $d \equiv e^{\varphi(n)-1} \pmod{n}$ az Euler-Fermat tétel miatt. Továbbá a titkosítás és fejtés is polinom időben megy a gyors hatványozás miatt. Ha az m üzenetre $m < n$ nem teljesül, akkor fel kell darabolni az üzenetet, hogy az egyes darabokra már igaz legyen.

Mit tud egy támadó? A nyilvános (n, e) párt, illetve a kódolt $c \equiv m^e \pmod{n}$ üzenetet, de nem ismeri p, q, d értékét. Szeretné megkapni az m üzenetet, azaz egy összetett modulus mellett kéne e-edik gyököt vonni. Ez az RSA probléma, és jelenleg nem tudunk rá gyorsabb módszert, mint az n faktorizálását, ekkor természetesen $n = pq$ miatt a támadó $\varphi(n)$-et, majd a titkos d értékét is meg tudja határozni, azaz tud dekódolni. Elvileg elképzelhető lenne egy faktorizálást nem használó módszer is m meghatározásához.

Ahhoz, hogy a támadónak ne legyen könnyű faktorizálni olyan prímeket célszerű választani amelyekkel az ismert $P \pm 1$ módszerekkel nem kaphatóak meg könnyen, azaz $p \pm 1, q \pm 1$ számoknak legyen nagy prímosztójuk. Továbbá $|p - q|$ ne legyen túl kicsi, különben Fermat faktorizációs módszerével p, q könnyen megkapható.

38. Tétel. *Az RSA-nál az, hogy n-et faktorizáljuk, vagy $\varphi(n)$-et határozzuk meg polinomiálisan ekvivalens probléma.*

Bizonyítás: Ha $n = pq$ faktorizálást ismerjük, akkor $\varphi(n) = (p-1)(q-1)$ kiszámítható polinom időben. Megfordítva: ha $\varphi(n)$-et ismerjük, és természetesen n-et, akkor tehát ismerjük: $\beta = pq$ és $\alpha = p + q = n - 1 - \varphi(n)$-et. Azaz két szám összegét és szorzatát, így a Vieta formulák miatt p, q gyöke az $x^2 - \alpha x + \beta = 0$ egyenletnek, ahonnan p, q értéke a másodfokú egyenlet megoldóképletéből polinom időben megkapható.

2. Feladat (Gerbicz). *$n, \varphi(n), \sigma(n)$ ismeretében n faktorizálása (polinom időben), ha $n = pqr$.*

Még gimnazistáknak adtam fel ezt a példát, lásd:
`http://www.spoj.pl/problems/HS08CODE/`. Előző tételhez hasonlóan:

$\varphi(n) = (p-1)(q-1)(r-1) = pqr - pq - qr - rp + p + q + r - 1$ és
$\sigma(n) = (p+1)(q+1)(r+1) = pqr+pq+qr+rp+p+q+r+1$, továbbá $n = pqr$ miatt az elemi szimmetrikus polinomok kifejezhetőek: $\alpha = p+q+r = \frac{\sigma(n)+\varphi(n)-2n}{2}; \beta = pq + qr+rp = \frac{\sigma(n)-\varphi(n)-2}{2}; \gamma = pqr = n$, ezért p, q, r gyöke az $f(x) = x^3 - \alpha x^2 + \beta x - \gamma = 0$ harmadfokú egyenletnek. Ez a Cardano képlettel megoldható, de ez sem kell hozzá, mert páratlan fokú és $0 < p, q, r < n$ teljesül a gyökökre és a főegyüttható pozitív így $f(0) < 0$ és $f(n) > 0$, emiatt intervallumfelezéssel egy gyök megkapható, majd $x - p$-vel leosztva a másodfokú egyenletből a másik két gyök is.

Sőt ennél több is igaz, de már nem determinisztikusan:

39. Tétel. *$\varphi(n)$ egy kis (nemnulla) többszörösének ismeretében n várhatóan polinom időben faktorizálható.*

Még erősebbet látok be, legyen $n = \prod_{i=1}^{r} p_i^{e_i}$ az n prímtényezős felbontása, és $\lambda(n) = lkkt(\varphi(p_1^{e_1}), \cdots, \varphi(p_r^{e_r}))$.

40. Tétel. *$\lambda(n)$ egy kis (nemnulla) többszörösének ismeretében n várhatóan polinom időben faktorizálható.*

Bizonyítás (Gerbicz): Ez valóban erősebb, hiszen $\lambda(n)|\varphi(n)$ teljesül. Vegyük észre, hogy $m|n$ esetén $\lambda(m)|\lambda(n)$ teljesül, mert ez az Euler-féle φ függvényre is igaz. Így, ha n egy osztója d, akkor $\lambda(n)$ többszöröse a $\lambda(d)$-nek is egy többszöröse. Így elég csupán egy valódi osztót találni polinom időben, mert akkor d-re és $\frac{n}{d}$-re rekurzíve meghívva az algoritmust polinom időben faktorizálunk.

Ha n páros, akkor $d = 2$ triviális osztó. Az AKS teszttel polinom időben el tudjuk dönteni, hogy n prímhatvány-e. Így feltehető, hogy $n > 2$ páratlan és nem prímhatvány, ekkor $\lambda(n)$ páros. Legyen n prímtényezős felbontása $n = \prod_{i=1}^{r} p_i^{e_i}$ úgy sorszámozva a prímosztókat, hogy a p_i-k közül $p = p_1$-re a $p-1$ pontos 2-hatvány osztója a legnagyobb, ha több ilyen is van, akkor azok egyike. Legyen továbbá $e_1 = k$ és n-nek egy p-től különböző tetszőleges prímosztója q, míg q^l az n pontos q-hatvány osztója. Az adott $\lambda(n)$ többszörös legyen $C = 2^K(2L+1)\lambda(n)$, itt C páros, mert $\lambda(n)$ páros. Továbbá $p-1$ pontos 2-hatvány osztója 2^f, míg $q-1$-nek 2^g, választásunk miatt $f \geq g$.

Az Euler-Fermat tételből következik, hogy tetszőleges a-ra (ahol $lnko(a, n) = 1$), teljesül $a^{\lambda(n)} \equiv 1 \pmod{n}$, így $a^C \equiv 1 \pmod{n}$. Ezt fogjuk faktorizálásra használni, számítsuk ki sorban az $a^C \pmod{n}, a^{\frac{C}{2}} \pmod{n}, \cdots a^{\frac{C}{2^s}} \pmod{n}$ maradékokat, ahol 2^s a C pontos 2-hatvány osztója. Ha az első 1-gyel inkongruens maradék a sorban inkongruens még (-1)-gyel is, akkor egy nemtrivi osztót találtunk, ugyanis legyen ez a maradék w, akkor $w \not\equiv \pm 1 \pmod{n}$, de $w^2 \equiv 1 \pmod{n}$, így $n|(w-1)(w+1)$.

Az nem lehetséges, hogy egy prím mindkét tagot ossza, ellenkező esetben a különbségüket is osztaná, de az 2 és n páratlan, ellentmondás. Így n prímhatványosztói $w \pm 1$-et osztják, az nem lehetséges, hogy mindegyik prímhatvány ugyanazt a tagot, mert különben $w \equiv \pm 1 \pmod{n}$ lenne. Így $d = lnko(w+1, n)$ egy nemtrivi osztó.

Azt állítom, hogy a mod n redukált maradékrendszer elemeinek legalább a fele jó választás az a-ra, amikor tehát tudunk faktorizálni. $C = 2^K(2L+1)\lambda(n)$ volt, így még $a^{\frac{C}{2^K}} \equiv 1 \pmod{n}$ még biztosan igaz. p-t úgy választottuk n prímosztói közül, hogy $p-1$-nek volt a legnagyobb 2^f osztója, ezért $\lambda(n)$ pontos 2-hatvány osztója is 2^f (használjuk, hogy n páratlan és $\varphi(p^k) = (p-1)p^{k-1}$-nek a pontos 2-hatvány osztója ugyanaz, mint $p-1$ pontos 2-hatvány osztója).

Innentől már csak két eset lesz: 1. eset: $f > g$ (emlékeztetőül: $q-1$ pontos 2-hatvány osztója 2^g). Ekkor $\frac{C}{2^{K+1}}$ még osztható lesz $\varphi(q^l)$-el, így $a^{\frac{C}{2^{K+1}}} \equiv 1 \pmod{q^l}$ az Euler-Fermat tétel miatt. Ugyanakkor $a^{\frac{C}{2^{K+1}}} \equiv a^{\frac{(2L+1)\lambda(n)}{2}} \equiv a^{\frac{(2L+1)(2M+1)\varphi(p^k)}{2}} \equiv a^{\frac{\varphi(p^k)}{2}} \pmod{p^k}$ (itt $\lambda(n) = (2M+1)\varphi(p^k)$). Ismert, hogy, ha a kvadratikus maradék modulo p^k, akkor $a^{\frac{\varphi(p^k)}{2}} \equiv 1 \pmod{p^k}$, ha kvadratikus nemmaradék, akkor $a^{\frac{\varphi(p^k)}{2}} \equiv -1 \pmod{p^k}$, továbbá a redukált maradékosztály felére lesz kvadratikus maradék, illetve nemmaradék. Így amikor kvadratikus nemmaradék ott: $a^{\frac{C}{2^{K+1}}} \equiv 1 \pmod{q^l}$ és $a^{\frac{C}{2^{K+1}}} \equiv -1 \pmod{p^k}$, emiatt $a^{\frac{C}{2^{K+1}}} \not\equiv \pm 1 \pmod{n}$ (használjuk, hogy n páratlan, így p, q páratlan). Azaz az esetek (legalább) felében itt találunk egy nemtrivi osztót. 2. eset: $f = g$, az előző gondolatmenethez hasonlóan: az esetek felében: ha a kvadratikus maradék modulo q^l, akkor $a^{\frac{\varphi(q^l)}{2}} \equiv 1 \pmod{q^l}$, ha kvadratikus nemmaradék, akkor $a^{\frac{\varphi(q^l)}{2}} \equiv -1 \pmod{q^l}$. Ebből: ha a kvadratikus maradék modulo p^k és kvadratikus nemmaradék modulo q^l vagy fordítva (ez a $p^k q^l$ redukált maradékrendszer éppen felére teljesül), akkor $a^{\frac{C}{2^{K+1}}} \not\equiv \pm 1 \pmod{n}$, ami kellett.

Megjegyzés: valójában nem kell az AKS teszt sem, hiszen (legalább) $\frac{1}{2}$ valószínűséggel találunk nemtrivi osztót, akkor várhatóan 2 próba elég, vagy másképpen mondjuk 100 sikertelen próba esetén az n legalább $1 - \frac{1}{2^{100}}$ valószínűséggel prím. Természetesen $\lambda(n)$ egy triviális többszöröse n! ez azonban túl nagy többszörös, n hosszában nem polinomiális, így nem használható. Önmagában $\lambda(n)$ is fontos az RSA-nál, hiszen $te \equiv 1 \pmod{\lambda(n)}$ esetén d helyett már t-vel is lehet fejteni, ugyanis $c^t \equiv (m^e)^t \equiv m^{te} \equiv m^{k\lambda(n)+1} \equiv m \pmod{n}$ igaz. Ez azt is jelenti, hogy RSA-ban nem szabad Carmichael számot választani, mert azokra (a Korselt kritériumból) $\lambda(n)|(n-1)$ így t megkapható. (Persze ez egy nem hagyományos RSA lenne, mert a Carmichael számoknak legalább 3 prímosztójuk van.)

3. Feladat. *2010. decemberben tartott IV. hackerverseny 2. feladata (lásd [8]), melyben RSA-t kellett törni.*

Röviden a feladatról: byte-onként kódolt angol nyelvű szövegek az üzenetek, ismert,

hogy p, q prímek legfeljebb N számjegyből állnak, titkosított üzenetben a mod n maradékokat, ha kevesebb, mint $2N$ jegyből áll, akkor nullákkal töltik fel, ezért van meglehetősen sok nulla c-ben. PARI-GP-ben triviális a megoldás, először a titkos d értéket határozom meg $de \equiv 1 \pmod{\varphi(n)}$-ből majd c-t dekódolva byte-onként megkapom a titkosított üzenetet:

```
rsa(N,n,e,c)=d=lift(Mod(1/e,eulerphi(n)));cc=c;T=[];\
while(cc,r=cc%(10^(2*N));cc\=10^(2*N);T=concat(lift(Mod(r,n)^d),T));\
return([factor(n),d,Strchr(T)])
```

(Program a feladat input/output követelményeinek nem felel meg, de most csak az RSA/matek szempontjából érdekel a feladat.)

Például az N10.rsa fileban:

N=10

n=1557769436977951

e=877

c=000000660061853600820000153538200029472900000774539304012236

000009948747556303990000010998048093900800001535382000294729

000000651637023002950000010998048093900800001306262887725215

000013852181463646440000115868620783448300001357229380580691

000008938431115372370000010998048093900800000651472665632618

00001545327422011293

Előző programmal:

p=524287

q=2971215073

d=1403233189895653

```
m=Life is hard! ;)
```

Többi üzenet megfejtése, faktorizáció és d értéke nélkül:

```
N2.rsa: m=My momma always said, "Life was like a box of chocolates.
You never know what you're gonna get." /Forrest Gump/
```

```
N3.rsa: m=Life is not about waiting for the storm to pass,
its about learning how to dance in the rain. /Anonymous/
```

```
N4.rsa: m=Life is full of misery, loneliness, and suffering -
```

```
and it's all over much too soon.
/Woody Allen/

N5.rsa: m=A life without cause is a life without effect.
/Barbarella/

N8.rsa: m=Life is something that happens when you can't get to sleep.
/Fran Lebowitz/
```

A feladatról: habár ez valóban RSA titkosítást jelent, de ez egy meglehetősen lebutított verziója. A baj ott van vele, hogy túl hamar darabolja a szöveget, jelen esetben byte-onként, n nagyságától függetlenül. Holott az RSA-ban $hossz(n) - 1$ byte-ot lehet egy darabban kódolni. A byte-onkénti kódolás eredménye, hogy ez valójában egy egyszerű helyettesítésnek felel meg, így könnyen törhető megfelelő hosszú üzenet esetén nyelvstatisztika alapján mindenféle faktorizáció nélkül, azaz az RSA szokásos törését megkerülve. 256 féle lehet egy byte, de az ASCII-ben a felső 128 jelet az üzenet nem használja, így feltehető, hogy $|A| = 128$, az ábécé mérete, legyen a nyelv entrópiája mondjuk $H = 3$ (ez azt jelenti, hogy egy betű takarékos kódolással átlagosan hány bittel ábrázolható). $H(K) = \log_2(128!)$ a kulcs entrópiája, hiszen 128! féle permutációja van a betűknek. Az ismert képlet szerint az egyértelműségi pontra: $U = \frac{H(K)}{\log_2(|A|)-H}$, ez a mi esetünkben: $U = \frac{\log_2(128!)}{log_2(128)-3} \approx 179.04$-et jelent. Azaz várhatóan 180 hosszú szöveg már egyértelműen fejthető.

4. Feladat. *Tegyük fel, hogy van rengeteg RSA számunk és prímünk, de nem tudjuk, hogy melyik prím melyik RSA számot osztja. Szeretnénk a számokat minél gyorsabban faktorizálni a prímlista alapján (amelyeket persze lehet).*

Osztófa algoritmus gyorsan megoldja ezt a problémát. Ezért sem érdemes publikus prímeket használni az RSA módszerhez.

3.8. Teljes hatványok felismerése

9. Definíció (Teljes hatvány). $n > 0$ *teljes hatvány, ha* $n = b^k$ *valamely* $b, k \geq 2$ *egészekre.*

Különféle modern faktorizációs algoritmusokban kell eldöntenünk (gyorsan), hogy adott n szám négyzetszám, illetve teljes hatvány-e.

41. Tétel (D. J. Bernstein). *Majdnem lineáris időben eldönthető, hogy* n *teljes hatvány-e. (Lásd [5]).*

Más módszerek:

1. Elég minden k helyett csupán a prímkitevőket ellenőrizni, azt is csupán $p \leq \lfloor \log_2 n \rfloor$-ig, hiszen $p > \lfloor \log_2 n \rfloor$-re $b^p \geq 2^p > n$ teljesül.

2. Adott L korlátig a kis prímekkel osztható-e n? Akár egyetlen prímosztót is találva jelentősen csökkenthető a keresési tér: tegyük fel ugyanis, hogy ${p_1}^{e_1}, \cdots, {p_k}^{e_k}$ az n pontos prím(hatvány) osztói, ekkor $n = b^p$ esetén $p | \text{lnko}(e_1, \cdots, e_k)$ triviálisan teljesül. Történetesen, ha ez 1, akkor megállhatunk, n nem teljes hatvány, egyébként az lnko prímosztóit kell csak végignézni.

3. Ha nem találtunk prímosztót, akkor is kevesebb prímet kell ellenőrizni: legyen $n = b^p$, ekkor $b \leq L$ nem lehet, mert n-nek, így b-nek sincs L-nél kisebb prímosztója. Így $n = b^p > L^p$, amiből $p \leq \lfloor \log_L n \rfloor$

4. Az eddigieknél bonyolultabb módszer: hatványmaradékok segítségével zárunk ki lehetséges p értékeket! Legyen $q = 2kp + 1$ alakú prím, ha $n = b^p$, akkor $n^{\frac{q-1}{p}} \equiv b^{q-1} \equiv \{0, 1\} \pmod q$ teljesül a kis Fermat tétel miatt, így $n^{\frac{q-1}{p}} \not\equiv \{0, 1\} \pmod q$ esetén n egy egésznek sem p-edik hatványa. Itt $p \leq \lfloor \log_L n \rfloor$ miatt q is aránylag kicsinek választható. Sőt a Lehmer tétel miatt ilyen q-t gyorsan is találhatunk. Sok q értékre szeretnénk kiszámolni $n^{\frac{q-1}{p}} \pmod q$-t, itt a költséges számolás $n \pmod q$ meghatározása, mert $q << n$. Maradékfát építve tehetjük meg ezt gyorsan egyszerre sok q értékre.

5. Megmaradó p értékekre $b = \lfloor n^{\frac{1}{p}} \rfloor$ kiszámolása után ellenőrizzük, hogy $n = b^p$ teljesül-e. Ezt Newton iterációval tehetjük meg: legyen x az $n^{\frac{1}{p}}$ egy közelítése, majd az iteráció:

$x \leftarrow \frac{1}{p}((p-1)x + \frac{n}{x^{p-1}})$

itt az értékes jegyek száma iterációnként duplázódik. Kezdeti x érték választása: $2^k \leq n < 2^{k+1}$ esetén legyen $x = 2^{\lfloor \frac{k}{p} \rfloor}$.

newperfpow.c (illetve newperfpow2.c) programmal például $n = 10^{10000000}+13$-ról kb. 8 másodperc alatt dönti el, hogy nem teljes hatvány. gmp-nek az új kódja kb. 15 másodperc alatt dönti el ugyanezt.

3.9. Kvadratikus szita (QS)

1981-ben fedezte fel Carl Pomerance a kvadratikus szitát, ami szitával Dixon faktorizációs módszerénél gyorsan talál B-számokat. Jelenleg is a kvadratikus szita a leggyorsabb faktorizációs algoritmus 20-100 jegyű számok között.

Ha y az B-szám és négyzetszám, akkor r minden prímtényezője bázisbeli, ahol $y \equiv r \pmod n$ és r a legkisebb abszolútértékű ilyen szám. Szitáláshoz az y számokat egy másodfokú polinom helyettesítési értékeiként keresi. A legegyszerűbb formája:

$y = f(x) = (x + \lfloor\sqrt{n}\rfloor)^2 - n$, ahol $|x| << \sqrt{n}$ teljesül, ekkor ugyanis $y \approx 2x\sqrt{n}$. Azaz kicsi, hiszen, így sokkal nagyobb valószínűséggel lesz B-szám, mintha mondjuk egy random y számot vettünk volna. QS az $x = 1, \cdots, L$ értékekre ellenőrzi, hogy $f(x)$ az B-szám-e. Ekkor tulajdonképpen faktorizálni kell az L egymásutáni értékre a másodfokú polinom helyettesítési értékeit. Ez szitálással megy: tegyük fel, hogy $f(x) \equiv 0 \pmod{p}$, ahol p prím. Azaz:
$x^2 \equiv n \pmod{p}$, ez pedig Shanks-Tonelli algoritmusával (várhatóan polinom időben) oldható meg. Ha $\text{lnko}(n, p) = 1$, akkor két inkongruens megoldás lesz $\pmod{p}$: x_1, x_2, ahol persze $x_1 + x_2 \equiv 0 \pmod{p}$. p-nek magasabb hatványai is oszthatják $f(x)$-et, de azokat az előbbi megoldásokból már egyszerűen kapjuk.

Szitánál egy gyorsítást kapunk, ha $f(x)$-et nem osztjuk be a megtalált prímekkel, hanem az algoritmus elején az x-eknek megfelelő L hosszú tömbben a tömb minden elemét $A[x] \leftarrow \lfloor \log_2 f(L) \rfloor$ értékkel inicializáljuk, és amikor p osztja $f(x)$-et akkor az osztás elvégzése helyett $A[x] \leftarrow A[x] - \lfloor \log_2 p \rfloor$-et állítjuk be. A szitálás végén, ha $A[x]$ közel van a nullához, akkor nagy valószínűséggel $f(x)$-et teljesen faktorizáltuk, azaz $f(x)$ az B-szám. Ha nagy érték az $f(x)$ akkor nem faktorizáltunk, azaz $f(x)$ nem B szám. További gyorsítás, ha a kis prímekkel, illetve az összes prímhatvánnyal nem szitálunk.

Nagy prímvariáns itt is használható: legyen $f(x_1)$ és $f(x_2)$ ugyan nem B-számok, de felírhatóak: $f(x_1) = b_1 p$, illetve $f(x_2) = b_2 p$ alakban, ahol b_1, b_2 B-szám és p prím. Ekkor viszont B-szám lesz $\frac{f(x_1)f(x_2)}{p^2}$. Általánosan, ha t darab ilyen relációt találunk adott p prímre, akkor ez $t - 1$ darab B számot ad. Kicsivel bonyolultabb a helyzet 2 nagy prímre: építsünk fel egy (irányítatlan) gráfot, melynek csúcsai a relációkban szereplő nagy prímeknek felelnek meg. Egyetlen nagy prímet tartalmazó relációnak az adott prímnek megfelelő csúcsra illeszkedő hurokélt rakjunk. Míg két nagy prímnél a prímeknek megfelelő csúcsok közti élt. Ekkor újabb relációk megtalálásához először tekintsük a hurokél(ek) nélküli gráf összefüggő komponenseit, ha bármelyikben van kör, az ad egy újabb relációt. Vegyük vissza a hurokéleket, ha egy komponensben két hurokél lenne, akkor a köztük lévő út, a két hurokéllel együtt ad egy újabb relációt. Ezt iterálhatjuk. Kettőnél több nagy prímet nem éri meg használni 100 jegyű számokra sem.

QS futási ideje az n inputra, (sejtés): $L_n[\frac{1}{2}, 1]$.

3.10. GNFS/SNFS faktorizációnál a polinom kiválasztása

Nagy számok faktorizációjánál a leggyorsabb ismert módszer a GNFS (general number field sieve) minden számra, míg speciális alakú számokra az SNFS (special number field sieve). Programtól függően $90-95$ jegy felett már gyorsabbak, mint a kvadratikus szita. Sejtés szerint futásidejük $L_n[\frac{1}{3}, c]$, GNFS-nél nagyobb c értékre, mint az SNFS-nél. A faktorizáció a kvadratikus szita egy javításának is tekinthető, csak itt jóval kisebb B-számokat kell hozzá faktorizálni, n-hez képest szubexponenciális nagyságú számokat, míg a kvadratikus szitánál $\sqrt{n}$ nagyságrendűeket. John Pollard fedezte fel az SNFS módszert 1988-ban, majd Pomerance 1990-ben a GNFS-t, ami már minden számra működik. Részletekbe nem megyek bele, de az egyik önmagában is érdekes része a polinom szelekció néhány gyakorlati módszerébe igen.

GNFS/SNFS módszerhez két (kis fokszámú) $\mathbb{Q}$ felett irreducibilis egészegyütthatható f, g polinomot kell választanunk úgy, hogy legyen közös m gyöke a két polinomnak $\mathbb{Z}_n$ felett. Általában az egyik polinom elsőfokú. Az SNFS módszernél a polinomok együtthatói kisebbek, mint a GNFS-nél.

Minden (egész) számra működő módszer: $g(x) = x - b$ legyen, továbbá $f(x) = \sum_{i=0}^{d} a_i x^i$, ahol $b = 1+\lceil n^{\frac{1}{d+1}} \rceil$ és $n = \sum_{i=0}^{d} a_i b^i$ az n felírása b alapú számrendszerben. Így $0 \leq a_i < b$, továbbá $f(x), g(x)$-nek közös gyöke $\mathbb{Z}_n$ felett $m = b$. Az n nagyságától függően választjuk d értékét. Ma már azért sokkal ravaszabb eljárások vannak a polinom kiválasztására.

SNFS-nél, tehát speciális alakú számoknál a polinom kiválasztása sokkal könnyebb. Tipikusan $b^n \pm 1$ alakú számokra alkalmazzák, ahol $1 < b \leq 12$ és b nem teljes hatvány. Ezeket a Cunningham project faktorizálja, valószínűleg ez a leghosszabb folyamatosan tartó számítási project a világon, amit még Allan Cunningham kezdett el 1925-ben. Nem kell az eredeti $b^n - 1$ számot faktorizálni, hiszen ismeretes, hogy $x^n - 1 = \prod_{d|n} \phi_d(x)$, itt $\phi_d(x)|(x^d - 1)$, így elég csupán $\phi_n(b)$-t faktorizálni, az eredeti $b^n - 1$ helyett. Ismeretes az is, hogy a körosztási polinomok $\mathbb{Q}$ felett irreducibilisek, de konkrét b, n párra még lehet további algebrai osztó, ezeket Aurifeuillian osztóknak nevezik és belátható, hogy más algebrai osztó ezeken kívül nincsen. Ilyen Aurifeuillian osztóra példa: $2^{4k-2} + 1 = (2^{2k-1} - 2^k + 1)(2^{2k-1} + 2^k + 1)$

Tegyük fel, hogy $2^{1533}-1$-et szeretnénk faktorizálni, pontosabban csak a primitív részét ($1533 = 3 * 7 * 73$ a kitevő faktorizálása). A körosztási polinom ötletet is ad hozzá, hogyan válasszuk a polinomot: $f(x) = \phi_{21}(x) = x^{12} - x^{11} + x^9 - x^8 + x^6 - x^4 + x^3 - x + 1$ és $g(x) = x - m$, ahol $m = 2^{73}$, akkor m közös gyöke a két polinomnak $\mathbb{Z}_N$ felett, ahol $N = \phi_{21}(2^{73})$. De ez túl nagy fokú polinomot jelent az f-

re, gyakorlatban 7-nél magasabb fokú polinomokat nem használnak. De szerencsére f egy úgynevezett reciprokpolinom, így $\frac{f(x)}{x^6}$ az $y = x + \frac{1}{x}$ változóban már csupán 6-odfokú, kiszámítani ujjgyakorlat: $\frac{f(x)}{x^6} = x^6 - x^5 + x^3 - x^2 + 1 - \frac{1}{x^2} + \frac{1}{x^3} - \frac{1}{x^5} + \frac{1}{x^6} = y^6 - y^5 - 6y^4 + 6y^3 + 8y^2 - 8y + 1 = F(y)$, ezzel $F(M) = 0$, ahol $M = 2^{73} + \frac{1}{2^{73}}$ és a másik polinom $G(y) = 2^{73}y - (2^{146} + 1)$. Ez már a gyakorlatban is működő választás a 6-odfokú, kis együtthatójú f polinommal, g elsőfokú. Kisebb algebrai osztókat nem használtam, túl nagy lett volna f foka. Természetesen $\phi_{1533}(2)|N$ teljesül.

Tegyük fel, hogy $n = 12^{293} + 1$-et szeretnénk faktorizálni, az egyetlen algebrai osztó $12 + 1 = 13$, ami persze semmit sem segít, a hányadospolinom $x^{292} - x^{291} \pm \cdots - x + 1$ 292-edfokú lenne. Aurifeuillian osztók nincsenek. Így a standard eljárás megy csupán: általánosan $N = b^n + c$ faktorizálása (ahol $|c|$ kicsi, Cunningham számoknál természetesen $c = \pm 1$), d legyen kicsi (jellemzően $d = 3, 4, 5, 6, 7$) $f(x) = b^r x^d + c$ és $g(x) = x - b^e$, ahol $e = \lfloor \frac{n}{d} \rfloor$ és $r = n \pmod d$ választással f, g közös gyöke $\mathbb{Z}_N$ felett $m = b^e$. Visszatérve az eredeti számunkra: $n = 12^{293} + 1$-re $d = 6$ választással: $f(x) = 12^5x^6 + 1$ és $g(x) = x - 12^{48}$ közös gyöke $\mathbb{Z}_N$ felett $m = 12^{48}$, egy szokásos trükk, hogy egy még nagyobb számot faktorizálunk, N-et 12-vel szorozva $N_2 = 12n = 12^{294} + 12$, $d = 6$-odfokú polinommal $F(x) = x^6 + 12$ és $G(x) = x - 12^{49}$ közös gyöke $\mathbb{Z}_{N_2}$ felett $m_2 = 12^{49}$.

Jelenlegi faktorizációs világrekord speciális alakú számokra az 1039 bites: $n = 2^{1039} - 1$ amit még 2007-ben faktorizáltak. Itt nincsen algebrai osztó. 6-odfokú polinomot használtak: $f(x) = 2x^6 - 1$ és $g(x) = x - 2^{173}$, ekkor a közös gyök $\mathbb{Z}_n$ felett: $m = 2^{173}$.

Tegyük fel, hogy $Fib(1337)$-t szeretnénk faktorizálni, pontosabban $N = \frac{fib(1337)}{fib(191)}$-et, hiszen ismeretes, hogy $Fib(m)|Fib(n)$ teljesül, ha $m|n$. Így $Fib(1337)$-nek $Fib(191)$ egy (algebrai) osztója. Itt már az sem világos, hogy találunk jó polinomot az SNFS módszerhez a Fibonacci számhoz. Ismeretes a Fibonacci számokra vonatkozó azonosság: $Fib(n+1)Fib(m) + Fib(n)Fib(m-1) = Fib(m+n)$, ezt felhasználva igazolható, hogy $Fib(kn)$ felírható $Fib(n)$ és $Fib(n+1)$ kétváltozós polinomjaként, például: $Fib(2n) = 2Fib(n)Fib(n+1) - Fib(n)^2$, hasonlóan megkapható:
$Fib(7n) = 7ab^6 - 21a^2b^5 + 70a^3b^4 - 105a^4b^3 + 105a^5b^2 - 56a^6b + 13a^7$, ahol $a = Fib(n)$ és $b = Fib(n+1)$. (Itt az együtthatókat akár egy lineáris egyenletrendszerből is megkaphatjuk, ha már tudjuk, hogy létezik ilyen előállítás.) Az előbbi homogén 7-edfokú polinomot osszuk el ab^6-nal, így $\frac{a}{b}$-ben egy hatodfokú polinomot kapunk: $\frac{1}{b^6}\frac{Fib(7n)}{Fib(n)} = 7 - 21\frac{a}{b} + 70(\frac{a}{b})^2 - 105(\frac{a}{b})^3 + 105(\frac{a}{b})^4 - 56(\frac{a}{b})^5 + 13(\frac{a}{b})^6$, így $f(x) = 7 - 21x + 70x^2 - 105x^3 + 105x^4 - 56x^5 + 13x^6$ és $g(x) = Fib(n+1)x - Fib(n)$ választással f, g közös gyöke $\mathbb{Z}_N$ felett $m = \frac{Fib(n)}{Fib(n+1)}$, ahol $N = \frac{Fib(7n)}{Fib(n)}$ (példánkban $n = 191$). Amire vigyázni kell, hogy itt m létezéséhez $b = Fib(n+1)$-nek inver-

tálható elemnek kell lennie Z_N-ben, ehhez: $lnko(Fib(n+1), N) = lnko(Fib(n+1), \frac{Fib(7n)}{Fib(n)}) = lnko(Fib(n+1), Fib(7n)) = Fib(lnko(n+1, 7n)) = Fib(lnko(n+1, 7))$, azaz $7 \not| (n+1)$ esetén működik a módszer. (Itt ismételten használtuk az ismert $lnko(Fib(n), Fib(m)) = Fib(lnko(n, m))$ azonosságot.)

Utolsó példa: $n = RevSmarandache(94)$ faktorizálása, ahol $RevSmarandache(n) = n \cdots 321$, azaz úgy kapjuk, hogy n-től kezdve visszafele leírjuk a pozitív egész számokat tízes számrendszerben. Ez is egy speciális szám szorozzuk meg 100-al és vegyük észre, hogy ez olyan, mint n, csak a felírásban a kétjegyű egészek el vannak benne csúsztatva:

$n = \quad 94939291 \cdots 121110987654321$

$100n = 9493929190 \cdots 111098765432100$, így kivonva egymásból a kettőt:

$99n = (94 * 10^{168} - 1010101 \cdots 0101) * 10^{11} + 98765432100 - 10987654321$. Itt a $1010101 \cdots 0101$ egy olyan szám amiben a jegyek periodikusan ismétlődnek, így felírható egy mértani sor összegeként, $\sum_{k=0}^{83} 10^{2k} = \frac{10^{2*84}-1}{10^2-1} = \frac{10^{168}-1}{99}$ Azaz $99n = (94 * 10^{168} - \frac{10^{168}-1}{99})10^{11} + 87777777779$, innen 99-cel osztva és közös nevezőre hozva: $n = \frac{9305*10^{179}+8790000000121}{9801}$, így $19602n = 1861 * 10^{180} + 17580000000242$. $d = 5$-ödfokú polinomot választva így $N = 19602n$-nel $f(x) = 1861x^5 + 17580000000242$ és $g(x) = x - 10^{36}$ választással f, g közös gyöke $\mathbb{Z}_N$ felett $m = 10^{36}$.

4. fejezet

Összefoglalás

Egy kitekintést adtam a manapság is használatos prímtesztelési, prímfaktorizációs módszerekhez. Az AKS volt az első polinom idejű prímtesztelési módszer, bár mint láttuk ez a gyakorlatban meglehetősen lassú. Sokkal gyorsabb módszerek vannak speciális alakú számok teszteléséhez, lásd Proth/Pepin teszt, vagy, ha $n-1$-nek ismerjük "sok" prímosztóját (Lehmer tételek). Tetszőleges egészre ott van a (gyors), de csak valószínűségi Miller-Rabin teszt.

Számokat tudunk faktorizálni Pollard születésnap-paradoxont használó módszerével, illetve Brent javításával. Pollard p-1 algoritmusával, amikor $p|n$ és $p-1$ minden prímosztója kicsi. Lánctörteket használó eljárással is lehet faktorizálni (CFRAC), tipikusan 10-20 jegy között, kvadratikus szitát 20-100 jegy között, míg nagyobb számokat GNFS/SNFS módszerrel faktorizálnak. Ha sok (kis) számunk van, akkor használhatjuk Bernstein algoritmusát, ami egy számra lebontva a futási időt polinom időben faktorizál! De nem ismert, hogy a faktorizáció P-ben van-e, sőt az egyetlen bizonyított szubexponenciális faktorizációs algoritmus Dixon módszere. Pedig mint láttuk az RSA miatt a faktorizáció komplexitását nagyon fontos lenne tudni.

Köszönöttel tartozom témavezetőmnek Gyarmati Katalinnak a szakmai tanácsaiért. Továbbá szüleim támogatásáért.

Hivatkozások:

[1] Junho Peter Whang, "Another Proof of the Infinitude of the Prime Numbers", American Mathematical Monthly, volume 117, number 2, February 2010, pp. 181

[2] Marek Wolf: "Search for primes of the form m^2+1" (2008)
`http://xxx.lanl.gov/abs/0803.1456`

[3] Manindra Agrawal, Neeraj Kayal, Nitin Saxena, "PRIMES is in P", Annals of Mathematics 160 (2004), no. 2, pp. 781–793

http://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf
[4] D. J. Bernstein. "How to find small factors of integers" (2000)
http://cr.yp.to/papers/sf-20000807.pdf
[5] D. J. Bernstein. "Detecting perfect powers in essentially linear time." Mathematics of Computation 67 (1998), 1253-1283.
http://cr.yp.to/papers/powers.pdf
[6] http://primes.utm.edu/prove/prove2_3.html
[7] Carl Pomerance, J. L. Selfridge, Samuel S. Wagstaff, Jr "The Pseudoprimes to $25 * 10^9$" Mathematics of Computation, volume 35, number 151 (1980), pp. 1009
http://mpqs.free.fr/ThePseudoprimesTo25e9.pdf
[8] http://hackerverseny.hu/
[9] https://sites.google.com/site/robertgerbicz/diploma