Prímtesztelés

Diszkrét modellek alkalmazásai

Az eddigiekben sokat foglalkoztunk prímszámokkal. Például az RSA-nál vagy a Diffie–Hellman-kulcscserénél szükségünk volt nagy méretű (több ezer bites) prímszámokra. De hogyan tudunk ilyeneket előállítani?

Ez nem triviális feladat, hiszen például az RSA-nál éppen azért kellett a két titkos prímszámnak ilyen nagynak lennie, hogy a szorzatukat ne lehessen visszabontani prímtényezőkre – de akkor hogyan ellenőrizzük, hogy prímszámot generáltunk-e, ha nem tudunk ekkora számokat faktorizálni?

És egyáltalán, vannak-e olyan méretű prímszámok, amelyekre szükségünk van? Van-e annyi, hogy azokat legyen esélyünk megtalálni? Van-e annyi, hogy azokból legyen értelme titkos prímszámokat választani? (Hiszen ha túl kevés van, akkor azokat egy támadó egyenként végigpróbálgathatja.)

Ebben a jegyzetben ezekre a kérdésekre keressük a választ.

1. Prímszámok számossága

Hány prímszám van? Erre többféleképpen is válaszolhatunk. Egyrészt:

1.1. Tétel (Eukleidész tétele)**.** *Végtelen sok prímszám van.*

(Bizonyítás: tegyük fel, hogy csak véges sok prímszám van, például ezek: $p_1, p_2, p_3, \ldots, p_n$. Ekkor tekintsük a következő számot: $p_1 p_2 p_3 \ldots p_n + 1$. Ez nem lehet prímszám, mert az összes feltételezett prímszámnál nagyobb, de nem lehet összetett szám sem, mert semelyik prímszámunkkal sem osztható – tehát ellentmondásra jutottunk, vagyis valójában végtelen sok prímszámnak kell lennie.)

De a "végtelen sok van" még nem a teljes válasz. Például végtelen sokan vannak a természetes számok és a kettőhatványok is, de az utóbbiak mégis sokkal ritkábbak. A kérdést matematikailag úgy tehetjük fel, hogy 1-től n-ig hány adott tulajdonságú szám van. Például:

	1-től n-ig	1-től 1000000-ig
Egész számok $(1, 2, 3, 4, 5, \ldots)$	n	1000000
Páros számok $(2, 4, 6, 8, 10, \ldots)$	$\lfloor n/2 \rfloor$	500000
Négyzetszámok $(1, 4, 9, 16, 25, \ldots)$	$\lfloor \sqrt{n} \rfloor$	1000
Kettőhatványok $(1, 2, 4, 8, 16, \ldots)$	$\lfloor \log_2 n \rfloor + 1$	20
Prímszámok $(2, 3, 5, 7, 11, \ldots)$	?	78498

Prímszámok esetén erre a következő jelölést használjuk:

1.2. Definíció (prímszámláló függvény)**.** Jelölje $\boldsymbol{\pi(n)}$ a prímszámok számát 1-től n-ig.

Példa. $\pi(7) = 4$, mert 7-ig négy prímszám van: $2, 3, 5, 7$.

A $\pi(n)$ pontos értékét nem tudjuk képlettel felírni, csak közelítést tudunk rá adni:

1.3. Tétel (Prímszámtétel)**.** $\pi(n) \approx \dfrac{n}{\ln n}$, *pontosabban:* $\lim\limits_{n\to\infty} \dfrac{\pi(n)}{n/\ln n} = 1$.

Más szóval annak az esélye, hogy egy 1 és n közötti szám prímszám, kb. $1/\ln n$ (ahol $\ln n$ az n természetes logaritmusa, azaz $\ln n = \log_e n$, ahol $e \approx 2{,}718\ldots$).

Példa. 1-től 100-ig átlagosan minden negyedik szám prímszám: $\pi(100) = 25 = 100/4$, és $\ln 100 \approx 4{,}6$.

Példa. 1-től 1000000-ig kb. minden 13. szám prímszám: $\pi(1000000) = 78498 \approx 1000000/12{,}7$, és $\ln 1000000 \approx 13{,}8$.

Sage-ben a $\pi(n)$-et a `prime_pi(n)` függvénnyel kérdezhetjük le. Például a következő kód ábrázolja $\pi(n)$-et és a közelítését 1-től 1000-ig (a felső, lépcsőzetes vonal a $\pi(n)$):

```
sage: var("n"); plot([n/ln(n), prime_pi], (1, 1000))
```

2. Prímszámok generálása

Tegyük fel, hogy szeretnénk egy véletlen prímszámot generálni adott a és b között. (Például ha háromjegyű prímszámot szeretnénk, akkor $a = 100$ és $b = 999$, ha pedig RSA-hoz generálunk egy 1024-bites prímszámot, akkor $a = 2^{1023}$ és $b = 2^{1024}-1$.)

Az algoritmus vázlata egyszerű: generáljunk véletlen számokat a és b között egészen addig, amíg prímszámot nem kapunk. Azaz:

Prímszám generálása:

Bemenet: $a, b \in \mathbb{N}$ $(a < b)$
ciklus
  $p :=$ véletlen szám a és b között
  ha prímszám(p)
    Kimenet: p

De vajon hány lépést kell tennünk a ciklussal, mire találunk egy prímszámot?

Ehhez azt kell tudni, hogy a véletlen szám milyen valószínűséggel lesz prímszám. Az n-bites prímszámok ($2^{n-1} \le p \le 2^n - 1$) számát a fenti prímszámtétellel tudjuk megbecsülni:

$$\pi(2^n) - \pi(2^{n-1}) \approx \frac{2^n}{\ln(2^n)} - \frac{2^{n-1}}{\ln(2^{n-1})} = \frac{2^n}{n \ln 2} - \frac{2^{n-1}}{(n-1)\ln 2} \approx \frac{2^n - 2^{n-1}}{n \ln 2} = \frac{2^{n-1}}{n \ln 2}.$$

Mivel n-bites számból összesen 2^{n-1} van, ezért egy véletlen szám $1/(n \ln 2)$ eséllyel lesz prímszám. Ez azt jelenti, hogy a ciklus várhatóan $n \ln 2 \approx 0{,}69\, n$ lépést tesz (pl. $n = 1024$ esetén kb. 700-at), azaz a prímszámok hosszában lineáris számú lépést – vagyis a prímszámok elég sokan vannak ahhoz, hogy találomra is viszonylag hamar találjunk egyet.

Gyorsíthatjuk az algoritmust, ha a véletlenszám-generálás során eleve csak páratlan számokat állítunk elő, hiszen úgyis csak azok lehetnek (ilyen nagy) prímszámok. (Ezt kettes számrendszerben úgy érhetjük el, hogy a véletlen szám utolsó bitjét 1-esre állítjuk, pl. Sage-ben/Python-ban: `p |= 1`.) Ekkor, mivel megfeleztük az alaphalmazt, ezért a prímszámok kétszer olyan gyakran fordulnak elő, azaz a lépésszám a felére csökken (pl. $n = 1024$ esetén csak kb. 350 lépés kell).

Megjegyzés: ismét hangsúlyozzuk, hogy ha a prímszám kriptográfiai alkalmazáshoz kell (pl. RSA-hoz), akkor a véletlen számokat biztonságos véletlenszám-generátorral állítsuk elő (lásd a Diszkrét logaritmus c. jegyzetet), hogy egy külső megfigyelő ne tudja őket megjósolni.

3. Prímtesztek fajtái

A továbbiakban azzal foglalkozunk, hogy a fenti prímszám-generálás során hogyan tudjuk eldönteni a véletlen számról, hogy prímszám-e.

A legegyszerűbb módszer a *próbaosztás* (lásd: Osztók, prímszámok), azaz hogy végigpróbálgatjuk a lehetséges számokat, hogy osztja-e valamelyik. Ez azonban nagyobb számok esetén használhatatlanul lassú: még a próbaosztás optimalizált változata is a számnak a négyzetgyökéig megy, azaz pl. egy 1024-bites szám esetén kb. 2^{512} számot vizsgál meg – ami már maga is egy 155-jegyű szám! De ez nem is meglepő, hiszen a próbaosztással egyúttal prímtényezőkre is tudjuk bontani a számot, márpedig az RSA éppen azon alapul, hogy ekkora számokat nem tudunk faktorizálni.

A próbaosztás ettől függetlenül még – korlátozott formában – hasznos lehet arra, hogy kiszűrjük a kisebb prímtényezőket, mielőtt más módszerekhez nyúlunk. Például ha a számunk osztható 3-mal, 5-tel vagy 7-tel – ami a számok majdnem felére igaz – akkor ez próbaosztással elég hamar kiderül, és akkor felesleges a lent tárgyalt bonyolultabb módszereket használni.

Szerencsére eldönteni egy számról, hogy prímszám-e, sokkal könnyebb feladat, mint prímtényezőkre bontani. Az ilyen módszereket **prímteszt**eknek nevezzük. Kétféle prímteszttel foglalkozunk:

1. A *determinisztikus prímteszt*ek biztosan meg tudják mondani, hogy egy szám prímszám-e.
2. A *valószínűségi prímteszt*ek nem 100%-ban adnak helyes választ, néha hibáznak. A tipikus tesztek mindig csak az egyik irányban tévednek: prímszámokra biztosan helyes választ adnak, de összetett számokat néha tévesen prímszámnak jeleznek.

De miért használnánk olyan teszteket, amelyek nem mindig mondanak igazat? Azért, mert a valószínűségi prímtesztek általában sokkal gyorsabbak. Például a próbaosztás egy determinisztikus prímteszt, és fent láttuk, hogy mennyire lassú. Vannak ennél hatékonyabb determinisztikus módszerek is (ezeket itt nem tárgyaljuk), de a gyakran használt valószínűségi módszerek ezeknél is sokkal hatékonyabbak. A kérdés persze az, hogy az utóbbiak milyen valószínűséggel tévednek, és ez az adott esetben még elfogadható-e.

A Sage a prímtesztek mindkét típusát támogatja: az `is_prime()` függvény alapértelmezésként determinisztikus prímtesztet használ, de átállíthatjuk valószínűségi prímtesztre is, hogy gyorsabb legyen, megengedve a tévedés kockázatát:

```
sage: n = 2^2048 + 981
sage: is_prime(n)    # determinisztikus: LASSÚ!
True
sage: proof.arithmetic(False)
sage: is_prime(n)    # valószínűségi: gyors, de megbízhatatlan
True
```

4. Fermat-prímteszt

Idézzük fel a kis Fermat-tételt, amelyről a Diszkrét logaritmus c. jegyzetben volt szó: ha p prímszám és $a \not\equiv 0 \pmod{p}$, akkor

$$a^{p-1} \equiv 1 \pmod{p}.$$

Ez a tétel csak prímszámokra vonatkozik, más számokról nem állít semmit. Ezért feltehetjük a kérdést: mi a helyzet akkor, ha p nem prímszám? Igaz lehet-e ez az állítás akkor is?

Hogy erre választ kapjunk, írassuk ki az $a^{n-1} \bmod n$ értékeket különböző n-ekre és különböző a-kra (ahol $1 \leq a \leq n-1$):

```
sage: for n in range(5, 16):
....:     print(n, [power_mod(a, n-1, n) for a in range(1, n)])
5 [1, 1, 1, 1]
6 [1, 2, 3, 4, 5]
7 [1, 1, 1, 1, 1, 1]
8 [1, 0, 3, 0, 5, 0, 7]
9 [1, 4, 0, 7, 7, 0, 4, 1]
10 [1, 2, 3, 4, 5, 6, 7, 8, 9]
11 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
12 [1, 8, 3, 4, 5, 0, 7, 8, 9, 4, 11]
13 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
14 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
15 [1, 4, 9, 1, 10, 6, 4, 4, 6, 10, 1, 9, 4, 1]
```

Vegyük észre a következőket:

1. Ha n prímszám, akkor minden érték 1, ahogy azt a kis Fermat-tétel is garantálja.
2. Ha $a = 1$ (első értékek), akkor annak nyilván minden hatványa 1.
3. Ha $a = n-1$ (utolsó értékek), akkor is gyakran kapunk 1-et, hiszen $n-1 \equiv -1 \pmod{n}$, és -1 minden második hatványa 1.
4. Az összes többi esetben azonban – azaz ha n összetett szám és $2 \leq a \leq n-2$ – szinte soha nem kapunk 1-et.
5. De néha igen: például $n = 15$-re és $a = 4$-re $4^{14} \equiv 1 \pmod{15}$.

Adja magát az ötlet, hogy ha n-ről szeretnénk eldönteni, hogy prímszám-e, akkor valamilyen a-ra számítsuk ki $a^{n-1} \bmod n$-et, és ha nem 1-et kapunk, akkor n biztosan nem prímszám, ha pedig 1-et, akkor *valószínűleg* prímszám. Ezzel tehát kaptunk egy valószínűségi prímtesztet. Ezt úgy hívják, hogy **Fermat-prímteszt**, és a következőképpen működik:

Fermat-prímteszt(n):

> **Bemenet:** $n \in \mathbb{N}, n \geq 4$, a tesztelendő szám
> Válasszunk egy véletlen $a \in \{2, 3, \ldots, n-2\}$ alapot.
> $x := a^{n-1} \bmod n$
> **ha** $x \neq 1 \rightarrow$ *hamis* (= "nem prímszám")
> **ha** $x = 1 \rightarrow$ *igaz* (= "valószínűleg prímszám")

Ez az algoritmus nagyon hatékony, hiszen csak egy moduláris hatványozást kell elvégezni, amely gyorshatványozással megoldható (lásd a Diszkrét logaritmus c. jegyzetet). Összehasonlítva a próbaosztással, ahhoz kb. $\sqrt{n}$ osztás kell, míg a Fermat-prímteszthez legfeljebb $2 \log_2 n$ szorzás (lásd a gyorshatványozásnál). Például ha $n \approx 2^{1024}$, akkor az előbbi kb. 2^{512}, míg az utóbbi $\leq 2048 = 2^{11}$.

A tévedés valószínűségét csökkenthetjük, ha többször futtatjuk le a tesztet (különböző véletlen a-kra), és csak akkor fogadjuk el a számot prímszámnak, ha minden teszt igazat ad. Például ha az $n = 15$-öt csak $a = 4$-gyel tesztelnénk, akkor n átmenne a teszten, hogy "prímszám". De ha megismételjük a tesztet pl. $a = 2$-vel, akkor n már "lebukik", hogy összetett szám.

4.1. Definíció. Egy adott $a \in \mathbb{N}^+$ számra az n összetett számot **Fermat-álprím**nek nevezzük, ha $a^{n-1} \equiv 1 \pmod{n}$. (Néha egyszerűen jelző nélkül **álprím**nek [*pseudoprime*] is hívják.)

Példa. $n = 15$ egy Fermat-álprím $a = 4$-re, mert $4^{14} \equiv 1 \pmod{15}$.

Vagyis a Fermat-álprímek azok, amelyek "becsapják" a Fermat-prímtesztet.

Megjegyzés: ez a fogalom függ az alaptól, a-tól, azaz ugyanaz az n lehet az egyik a-ra álprím, de a másik a-ra nem. Ennek hangsúlyozására használhatjuk a következő fogalmat:

4.2. Definíció. Egy n összetett számra az $a \in \mathbb{N}^+$ alapot **Fermat-tanú**nak nevezzük, ha n nem Fermat-álprím az a alapra.

Példa. $a = 2$ Fermat-tanú $n = 15$-re, mert $2^{14} \equiv 4 \not\equiv 1$ (15), $a = 4$ viszont nem (lásd fent).

Szerencsére azonban a Fermat-álprímek viszonlag ritkák. Például $a = 2$-re 1-től 10000-ig csak 22 Fermat-álprím van (a legkisebb a 341) – miközben ugyanezen intervallumban 1229 prímszám van.

Vannak viszont olyan álprímek, amelyek nagyon makacsul ellenállnak a Fermat-prímtesztnek:

4.3. Definíció. Egy n összetett szám **Carmichael-szám**, ha minden $a \in \mathbb{N}$ alapra, amely relatív prím az n-hez, $a^{n-1} \equiv 1 \pmod{n}$.

Vagyis a Carmichael-számok azok, amelyek a lehető legtöbb alapra Fermat-álprímek. (Ennél több alapra már nem is lehetnének, hiszen ha a és n nem relatív prímek, akkor a hatvány biztosan nem lehet 1, ahogy azt az előző jegyzetben megállapítottuk.) A legkisebb Carmichael-szám az 561. Ezek szerencsére még ritkábbak, mint a közönséges Fermat-álprímek (például 1-től 10000-ig csak 7 Carmichael-szám van).

A jó hír az, hogy ha egy álprím nem Carmichael-szám, akkor viszonylag könnyű "lebuktatni":

4.4. Tétel. *Ha n összetett szám, de nem Carmichael-szám, akkor az 1 és $n-1$ közötti a-k több mint felére $a^{n-1} \not\equiv 1 \pmod{n}$.*

Ezek szerint tehát a Fermat-prímteszttel kapcsolatban három eset lehetséges:

1. Ha n prímszám, akkor a teszt biztosan helyes eredményt ad.
2. Ha n Carmichael-szám (ami ritka), akkor a teszt nagyon megbízhatatlan.
3. Egyébként viszont a teszt több mint 50% valószínűséggel helyes eredményt ad. Ez így első ránézésre kevésnek tűnik, de ez csak egy pesszimista alsó korlát, és ha megismételjük a tesztet különböző a-kra, akkor ez a korlát is tovább csökkenthető: k futtatás után a hiba valószínűsége kevesebb, mint $1/2^k$, pl. $k = 10$ esetén $< 0{,}001$ ($= 0{,}1\%$).

5. Miller–Rabin-prímteszt

Az alábbiakban bemutatunk egy hatékonyabb és pontosabb valószínűségi prímtesztet, mint a fenti Fermat-prímteszt. Ez szinten a kis Fermat-tételen alapul, valamint a következő állításon.

Valós számokra tudjuk, hogy az $x^2 = 1$ egyenletnek csak két megoldása van: $x = 1$ és $x = -1$. A következő tétel azt mondja, hogy ez bizonyos esetekben moduláris aritmetikára is igaz:

5.1. Tétel. *Ha p prímszám és $x^2 \equiv 1 \pmod{p}$, akkor $x \equiv 1 \pmod{p}$ vagy $x \equiv -1 \pmod{p}$.*

Példa. Ha a modulus $p = 5$, akkor a $0, 1, 2, 3, 4$ számok moduláris négyzetei rendre $0, 1, 4, 4, 1$, azaz az $x^2 \equiv 1$ (5) kongruenciának valóban csak az $x \equiv 1$ (5) és az $x \equiv 4 \equiv -1$ (5) megoldásai vannak. Összetett számokra viszont nem feltétlenül igaz a tétel, például $3^2 \equiv 1$ (8).

(Bizonyítás: a feltétel szerint $p \mid x^2-1 = (x-1)(x+1)$, de prímszám egy szorzatot csak úgy oszthat, ha valamelyik tényezőjét is osztja, azaz $p \mid (x-1)$ vagy $p \mid (x+1)$.)

Induljunk ki ismét a kis Fermat-tételből, azaz hogy egy p prímszámra és egy a nemnulla alapra $a^{p-1} \equiv 1 \pmod{p}$. Például $p = 29$-re:

$$a^{28} \equiv 1 \pmod{29}.$$

Ha alkalmazzuk erre a fenti tételt $x = a^{14}$-re (azaz $x^2 = a^{28}$), vagyis megfelezzük a kitevőt, akkor azt kapjuk, hogy

$$a^{14} \equiv 1 \quad \text{vagy} \quad a^{14} \equiv -1 \pmod{29}.$$

Ha $a^{14} \equiv 1 \pmod{29}$, akkor még egyszer megfelezhetjük a kitevőt, és a tételből azt kapjuk, hogy

$$a^{7} \equiv 1 \quad \text{vagy} \quad a^{7} \equiv -1 \pmod{29}.$$

Itt viszont a kitevő már páratlan, ezért nem tudunk tovább felezni.

A fenti műveletsor minden lépéséhez felhasználtuk, hogy p prímszám. Ha nem az, akkor jó eséllyel valamelyik lépés elromlik (de nem biztos). Ezzel tehát kaptunk egy újabb valószínűségi prímtesztet. **Példa.** Ha $n = 15$ és $a = 4$, akkor $a^{n-1} \equiv 1\ (n)$, azaz $4^{14} \equiv 1\ (15)$ még teljesül, viszont ha felezzük a kitevőt, akkor $4^7 \equiv 4\ (15)$, ami nem 1 és nem is -1, vagyis ez már megsérti a fenti "kitevőfelezős" tételt, tehát 15 nem lehet prímszám.

A **Miller–Rabin-prímteszt** ezen az elven működik, csak a műveleteket fordított sorrendben hajtja végre: nem az a^{n-1}-t számítja ki először, és felezgeti a kitevőt, hanem először kiszámítja az utolsó, már nem felezhető kitevőre a hatványt (pl. a fenti $p = 29$-re a^7-t), és onnan duplázgatja vissza a kitevőt $n-1$-ig (pl. $a^7 \to a^{14} \to a^{28}$). Ennek az az értelme, hogy a kisebb hatványt könnyebb kiszámítani, és nem is kell mindig visszamenni egészen n–1-ig, ha már hamarabb is eldőlt az eredmény (például mert megsértette a fenti "kitevőfelezős" tételt).

Miller–Rabin-prímteszt(n):

Bemenet: $n \in \mathbb{N}, n \geq 4$, a tesztelendő szám
ha n *páros* $\to$ *hamis* (= "nem prímszám")
Válasszunk egy véletlen $a \in \{2, 3, \ldots, n-2\}$ alapot.
Felezgessük $d := n-1$-et addig, amíg páratlan nem lesz, és legyen k a felezések száma.
$x := a^d \bmod n$
ha $x = 1 \to$ *igaz* (= "valószínűleg prímszám")
ciklus k*-szor*
  ha $x = n-1 \to$ *igaz* (= "valószínűleg prímszám")
  $x := x^2 \bmod n$
  ha $x = 1 \to$ *hamis* (= "nem prímszám")
$\to$ *hamis* (= "nem prímszám")

Az algoritmus addig duplázza a kitevőt, amíg a következők valamelyike nem teljesül:

1. A hatvány $x = -1$ lesz (pontosabban $n-1$, mert $n-1 \equiv -1\ (n)$): ekkor n átmegy a teszten ("valószínűleg prím"), mert -1 négyzete 1, és onnantól kezdve mindig 1-et kapunk.
2. A hatvány $x = 1$ lesz: ekkor n nem lehet prímszám, mert ha az lenne, akkor az előző lépésben 1-et vagy -1-et kellett volna kapnunk, és akkor amiatt már megálltunk volna.
3. A kitevő eléri az $n-1$-et k duplázás után ($n-1 = 2^k d$): ekkor n nem lehet prímszám, mert ha az lenne, akkor a hatvány 1 lenne (kis Fermat-tétel), és akkor amiatt már megálltunk volna.

Példa. Az $n = 29$ prímszám, és valóban, bármilyen alapra is próbáljuk a tesztet, mindig elfogadja (mert az 1-esek előtt mindig -1 van, vagy kezdettől fogva csak 1-esek vannak):

$$2^7 \equiv 12, \quad 2^{14} \equiv -1, \quad 2^{28} \equiv 1 \pmod{29} \quad \checkmark$$
$$5^7 \equiv -1, \quad 5^{14} \equiv 1, \quad 5^{28} \equiv 1 \pmod{29} \quad \checkmark$$
$$7^7 \equiv 1, \quad 7^{14} \equiv 1, \quad 7^{28} \equiv 1 \pmod{29} \quad \checkmark$$

Példa. Az $n = 45$ összetett szám, és bármilyen alapra próbáljuk a tesztet, valahol mindig elromlik (vagy azért, mert -1 nélkül kapunk 1-et, vagy mert soha nem kapunk 1-et):

$$2^{11} \equiv 23, \quad 2^{22} \equiv 34, \quad 2^{44} \equiv 31 \pmod{45} \quad \times$$
$$8^{11} \equiv 17, \quad 8^{22} \equiv 19, \quad 8^{44} \equiv 1 \pmod{45} \quad \times$$
$$19^{11} \equiv 19, \quad 19^{22} \equiv 1, \quad 19^{44} \equiv 1 \pmod{45} \quad \times$$

Ebből is látszik, hogy a Miller–Rabin-teszt erősebb, mint a Fermat-teszt, amely 8-ra és 19-re is prímszámnak fogadta volna el a 45-öt (mert csak az utolsó 1-est nézi).

Példa. Az $n = 25$ ugyancsak összetett szám, de ez nem mindig derül ki: egyes alapokra "lebukik", másokra viszont prímszámnak "tetteti magát":

$$6^3 \equiv 16, \quad 6^6 \equiv 6, \quad 6^{12} \equiv 11, \quad 6^{24} \equiv 21 \pmod{25} \quad \times$$
$$7^3 \equiv 18, \quad 7^6 \equiv -1, \quad 7^{12} \equiv 1, \quad 7^{24} \equiv 1 \pmod{25} \quad \checkmark \text{ (!)}$$

Tehát a Miller–Rabin-prímteszt sem mentes az álprímektől.

5.2. Definíció. Egy adott $a \in \mathbb{N}^+$ számra az n összetett számot **erős álprím**nek nevezzük, ha a Miller–Rabin-prímteszt elfogadja n-et prímszámnak az adott a-ra.

Példa. Az $n = 25$ erős álprím $a = 7$-re (és Fermat-álprím is), ahogy az a fenti példából láthattuk.

Példa. Az $n = 45$ nem erős álprím semmilyen alapra (viszont Fermat-álprím pl. $a = 8$-ra).

Az erős álprímek egyúttal mindig Fermat-álprímek is (hiszen a Miller–Rabin-teszt a Fermat-teszt továbbfejlesztése), de ez fordítva nem feltétlenül igaz.

A Miller–Rabin-prímteszt több szempontból is jobb, mint a Fermat-prímteszt:

1. Itt nincsenek olyan számok, mint a Fermat-prímtesztnél a Carmichael-számok, azaz amelyekre a Miller–Rabin-prímteszt majdnem mindig rossz eredményt adna.
2. A Miller–Rabin-teszt az esetek 75%-ában ad helyes eredményt (szemben a Fermat-prímteszt 50%-ával), vagyis k-szori ismétléssel a hiba valószínűsége $< 1/4^k$ lesz (szemben az $1/2^k$-nal), tehát ugyanazt a pontosságot feleannyi lépéssel érhetjük el ($1/4^k = 1/2^{2k}$).
3. Ráadásul a teszt egy lépése is lehet gyorsabb, mint a Fermat-prímteszté, mert nem mindig számítja ki a teljes a^{n-1} hatványt, hanem gyakran hamarabb is megállhat.

6. Szerző

Ezt a jegyzetet Uray M. János írta, részben felhasználva az alábbi forrásokat:

[1] https://compalg.elte.gitlab-pages.hu/dimoa-web/tematika/dimoa

[2] https://en.wikipedia.org/wiki/Fermat_primality_test

[3] https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test

[4] J. Katz and Y. Lindell: Introduction to Modern Cryptography, Second Edition; Chapter 8.2: Primes, Factoring, and RSA

Hibákat, észrevételeket itt lehet jelezni: uray.janos@inf.elte.hu.