A $s(n)$ számelméleti függvény alkalmazhatósága a kódolás, titkosítás területén

Pap Máté

December 6, 2010

A Programozási feladat 3 tárgyamat szeretném az Önálló kutatási feladat 2 tárggyal társítani, ezért ennek témáját röviden ismertetném. Azért választottam a $s(n)$ számelméleti függvényt a kutatásom tárgyának, mert annak ellenére, hogy a definíció megértéséhez kevés alapismeret szükséges, számos érdekes tulajdonsága van és mai napig sok nyitott kérdés létezik a függvénnyel kapcsolatban. Továbbá a kutatás elméleti részében szerzett ismereteket gyakorlati problémák megoldására tudjuk használni.

Az általam vizsgálni kívánt területen a természetes számok esetén kétféle összegfüggvényt szokás vizsgálni. Az egyik esetben az n szám összes pozitív osztóját adjuk össze, ezt az összeget a szakirodalomban $\sigma(n)$ jelölik, míg az n szám nála kisebb osztóinak összegét jelöljük $s(n)$-nel. Nyilván $s(n) = \sigma(n)–n$, (pl. $s(18) = 1 + 2 + 3 + 6 + 9 = 21$).

A $\sigma(n)$ és $s(n)$ függvények vizsgálatakor felmerülő érdekes kérdés, hogy $\sigma(n)$ és $s(n)$ értékének ismeretében hogyan, milyen módszerek segítségével tudjuk visszakeresni n-t.

Az elsőre viszonylag egyszerű, átlátható módszer adható, míg a második probléma igen hosszadalmasnak tűnik. Egy n szám osztóinak összege megadható a következő zárt alakban:

$$\sigma(n) = \frac{(p_1^{k_1+1} - 1) \cdot (p_2^{k_2+1} - 1) \cdots (p_m^{k_m+1} - 1)}{(p_1 - 1) \cdot (p_2 - 1) \cdots (p_m - 1)}.$$

Ezzel n ismeretében megadható $\sigma(n)$ és $s(n)$, továbbá a képlet segítségével $\sigma(n)$-ből meg tudjuk határozni n-t. Mivel $\sigma(n) > n$, így a fenti képletbe behelyettesítve a prímek különböző variációit, véges számú lépésben visszakereshető n.

Az $s(n)$ függvénynél nincsenek belátható időn belül elvégezhető eljárások. Nem tudjuk ilyen egyszerűen eldönteni $s(n)$ értékének ismeretében, hogy egy prím szerepel-e n-ben vagy nem. Tulajdonképp nincs más lehetőségünk, mint végig próbálunk minden lehetséges n-t, hogy az adott $s(n)$ tartozik-e hozzá. Természetesen a próbálkozások száma csökkenthető különböző elméleti meggondolások alapján. Tudva azt, hogy $\sigma(n)$ akkor és csak akkor páratlan, ha n négyzetszám, adódik, hogy

- ha $s(n)$ páros, akkor n páros és nem négyzetszám, vagy páratlan négyzetszám;
- ha $s(n)$ páratlan, akkor n páros négyzetszám, vagy páratlan és nem négyzetszám.

A próbálkozásokból ki lehet hagyni a prímeket; illetve ha n nem prím ($s(n) \neq 1$), akkor egyszerű belátni, hogy egy adott $s(n)$ érték esetén $n \leq (s(n) - 1)^2$.

Ez pl. azt jelenti, hogy egy 20 jegyű $s(n)$ esetén n nem lehet nagyobb 40 jegyűnél. Ha a 40 jegyűekig minden n-t végig kellene próbálni, hogy az adott $s(n)$ érték tartozik-e hozzá, akkor ez igen sokáig tartana. Pl. ha egy számítógép minden n esetén átlagosan 0,00001 s alatt döntené el, hogy hozzá az adott $s(n)$ tartozik-e, akkor ez kb. 10^{28} (!) évig tartana.

Az n és $s(n)$ olyan párt alkot, ahol az egyikből (n) gyorsan megadható a másik $s(n)$, vissza azonban nagyon lassan működik a dolog. Minden olyan művelet, ami egyik irányban viszonylag gyorsan kiszámolható, de ez a számítás visszafele nagyon sok ideig tartana alkalmas lehet titkosításra. A számpárt olyan üzenetek titkosítására használjuk, ahol az elküldés pillanatában még nem szeretnénk, hogy a címzett el tudja olvasni az üzenetet (pl. egy sakklépés vagy árajánlat esetén), továbbá a borítékbontásig mi sem tudunk változtatni az üzeneten.

A kódolás lépései a következők:

1. A kódolandó üzenetet ASCII kód segítségével átírjuk egy számmá, mivel az ASCII kód minden karakterhez egy számot rendel hozzá. Legyen ez a szám n.

2. n számhoz ezután a Mathematica 6.0 program segítségével rendeljük hozzá az $s(n)$-jét. Ezt az $s(n)$-t küldjük el, mint kódolt üzenetet.

A kódolás folyamata egy szemléletes példa segítségével: Képzeljük el, hogy két személy interneten keresztül sakkozik egymással. A játékot, amit természetesen folytatni szeretnének valamilyen oknál fogva fel kell, hogy függesszék. Ekkor a soron következő játékos, hogy hosszabb idejű gondolkodásra ne legyen módja, illetve lépési szándékán változtatni ne tudjon, következő lépését borítékolja. Így a játék ezzel a borítékolt lépéssel fog folytatódni. A borítékolás menete a következő:

1. Kódolandó üzenet: A2-ről lépek B3-ra

   Ha rövid vagy véges eshetőségű üzenetet kódolunk (például egy sakklépést) szükséges töltelékszavakat is használnunk, mert ugyan $s(n)$-ből n nehezen határozható meg, de véges eshetőség esetén n-ből $s(n)$ már kitalálható. Az üzenetként elküldött $s(n)$ egy részét megfejteni pedig lehetetlen (ezzel szemben, ha mondjuk csak 100-féle lehetséges sakklépést kell megvizsgálnunk, akkor nem túl hosszú idő alatt kitalálhatjuk ezeket a rövidebb üzeneteket).

2. Torzított üzenet: x57sD A2-ről lépek B3-ra xxtq

3. A torzított üzenet Ascii-kóddal kódolt alakja: 120 535 511 568 327 627 911 227 911 510 110 958 975 098 513 212 089 116 113

4. A fenti szám $s(n)$-jének értéke: 76 260 312 761 590 777 929 773 580 454 064 964 591 576 104 277 167

Ezt a számsort fogjuk borítékolni és elküldeni ellenfelünknek. A játék folytatásakor elküldjük az üzenetet. Ezután ellenfelünk könnyen leellenőrizheti, hogy ehhez az n-hez valóban az előzőleg elküldött $s(n)$ tartozik-e (ezáltal időközben nem tudunk változtatni az üzeneten). Az n számból az üzenet visszafejthető, hiszen csak az Ascii-kódot kell megfejtenie. A titkosítás során felmerülhet néhány probléma. A legfontosabb közülük, hogy elméletileg több n-hez is tartozhat ugyanaz az $s(n)$, így egy $s(n)$ szám több üzenetet is kódolhat. Annak az esélye azonban, hogy ezek az üzenetek értelmesek lesznek, szinte minimális. Az önálló kutatási feladat tantárgy keretében arra szeretném keresni a választ, hogy milyen n-ekhez tartozhat ugyanaz a $s(n)$, illetve, hogy egy adott titkosítás során mekkora a valószínűsége, hogy ugyanaz a $s(n)$ több üzenetet is kódol.

A titkosítás akkor lehet tökéletes, ha a kódolás folyamatát a kódfejtők szemszögéből is vizsgáljuk, kutatásom folytatása során ezért azt fogom vizsgálni, hogy létezik-e, megadható-e a mai napig még ismeretlen módszer, amellyel az üzenet visszafejtése során a próbálkozások száma oly mértékben csökkenthető legyen, hogy látható időn belül meg tudjuk fejteni az üzenetet.