Egyéni kutatómunka beszámoló

Andó Szabolcs

2021. Május

1. Bevezetés

Az egyéni kutatómunkámat a `https://cstheory.stackexchange.com` (a pontos címért lásd [1] forrás) címen található egyik kérdés, és annak válaszai alapján végeztem. Saját eredményt nem értem el, csak egy ennél hosszabb dokumentumban összefoglaltam magyarul, részletesen a kérdésre vonatkozó állításokat és azok bizonyításait. Ez az írás az említett dokumentumnak a rövid összefoglalása (a bizonyítások kihagyásával).

Tegyük fel, hogy tudunk generálni egyenletes eloszlásban véletlen egész számot $[1, 2^n]$-ből. A célunk az, hogy adjunk egyenletes eloszlásban egy egész számot $[1, 3]$-ból. Itt pontosan egyenletes eloszlásra gondolunk, nem csak $\frac{1}{2^n}$-es közelítéssel. Lehetséges-e ez polinomiális időben? Könnyen látható, hogy ha a várható időről szeretnénk, hogy polinomiális legyen, ez lehetséges, ezt biztosítja az alábbi

1.1. Állítás. A fenti probléma megoldható várható polinomiális időben.

Bizonyítás. Két eset van: ha n páros, akkor $2^n - 1$ osztható hárommal, tehát ekkor a következő a stratégia: legyen a generált számunk k. Ha $k < 2^n$, akkor az outputot úgy kapjuk meg, hogy redukáljuk k-t modulo 3, és hozzáadunk 1-et. Ha pedig $k = 2^n$, akkor elölről kezdjük az algoritmust, vagyis újra generálunk. Ha n páratlan, akkor ugyanezt csináljuk, azzal a különbséggel, hogy 2^n és $2^n - 1$ esetén egyaránt újrakezdjük az algoritmust. □

Várható polinomiális időben tehát könnyű megoldani ezt a feladatot. Mi a helyzet akkor, ha legrosszabb esetben is polinomiális időben szeretnénk generálni a véletlen számunkat? Persze ekkor a fenti algoritmus nem lesz segítségünkre, hiszen az szélsőséges esetben akár a végtelenségig is futhat. Sőt, az fog kiderülni, hogy a probléma oszthatósági megfontolásokból megoldhatatlanná válik ekkor. Ennek bizonyításában segít az alábbi triviális

1.2. Megjegyzés. Világos, hogy az alábbi feladatok ekvivalensek:

(i) Egyenletes eloszlásban generálunk egy véletlen egész számot $[1, 2^n]$-ből.

(ii) Egyenletes eloszlásban generálunk egy véletlen egész számot $[0, 2^n)$-ből.

(iii) $\{0, 1\}$-ből n db értéket egyenletes eloszlásban, egymástól függetlenül generálunk (azaz n db *bitet* generálunk).

1.3. Állítás. Oszthatósági megfontolások miatt nem tudunk a fenti orákulum segítségével egyenletes eloszlásban véletlen egész számot generálni 1 és 3 között úgy, hogy a futási időt egy véges számmal korlátozzuk.

Ezen a ponton úgy tűnik, hogy a kérdéssel kapcsolatban mindent megválaszoltunk. Ez szigorúan véve így is van, viszont felmerülhet bennünk egy másik gondolat:

1.4. Kérdés. Tegyük fel, hogy tudunk generálni egyenletes eloszlásban véletlen egész számot $[1, 2^n]$-ből, és kapunk egy nehéz problémát. A célunk az, hogy vagy oldjuk meg a nehéz problémát, vagy adjunk egyenletes eloszlásban egy számot $[1, 3]$-ból.

Ezzel kapcsolatban meg is fogalmazhatjuk az alábbi definíciót. A benne szereplő *függvényprobléma* kifejezés formális definíciója a következő fejezetben található, egyelőre lehet rá úgy gondolni, mint az előző 1.4 kérdésbeli nehéz probléma.

1.5. Definíció (**GEN**-3)**.** Az R függvényprobléma **GEN**-3 osztályban van, ha az előző 1.4 kérdés igaz rá, azaz: létezik algoritmus, mely egy x input esetén

- vagy outputként ad egy y elemet, melyre $R(x, y)$ igaz;
- vagy egyenletes eloszlásban generál egy egész számot $[1, 3]$-ból.

Miféle nehéz problémára gondolunk itt? Ez sokminden lehet, például SAT eldöntése egy adott inputon, vagy akár egy egész szám prímekre bontása. A célunk az, hogy megfejtsük, hogy milyen fajta problémák esetén oldható meg a feladat, azaz mely feladatok vannak a **GEN**-3 osztályban.

1.6. Példa. Ha az 1.4-ben a nehéz probléma egy egyirányú permutáció dekódolása, és n páros, akkor a következő a megoldás. Legyen

$$f : \{1, 2, ..., 2^n\} \to \{1, 2, ..., 2^n\}$$

egyirányú premutáció, és inputként kapunk egy $f(x)$-et. Generáljunk egy r véletlen számot $[1, 2^n]$-ből. Ha $f(r) = f(x)$, akkor készen vagyunk: outputunk r, megoldottuk a nehéz problémát. Ha nem, akkor véletlen számot generálunk a következő módon:

$$\text{output} = \begin{cases} f(r) \pmod 3, & \text{ha } f(r) < f(x) \\ f(r) - 1 \pmod 3, & \text{ha } f(r) > f(x) \end{cases}$$

(itt a mod 3 műveletet úgy értjük, hogy az $\{1, 2, 3\}$ halmazból az a szám, melynek azonos a maradéka). Ekkor könnyen látható, hogy $f(r) \neq f(x)$ esetén $[1, 2^n - 1]$-ből fog egyenletes eloszlásban kikerülni az, amit utána mod 3 veszünk. Ez pedig megfelelő, hiszen $2^n - 1$ osztható 3-mal (feltettük, hogy n páros).

A dolgozatban az alábbi eredmények találhatóak.

Tétel. PPA-3 $\subseteq$ **GEN**-3. Ez a 3.5 következményként szerepel a 4. oldalon.

Tétel. Ha SAT-ot függvényproblémaként tekintjük (a tanú keresése a cél), akkor SAT $\notin$ **GEN**-3, feltéve, hogy **NP** $\neq$ **co-NP**. Ez a 4.1 számú állítás, és az 5. oldalon van.

Tétel. Ha $R \in$ **TFNP** olyan, hogy minden x input esetén a megoldások számát előre tudjuk, és 3-mal nem osztható számú megoldás van, akkor $R \in$ **GEN**-3. Ez az 5.1 számú állítás az 5. oldalon.

Tétel. Legyen $R \in$ **TFNP**, és $p(n)$ egy polinom. Tegyük fel, hogy minden x input esetén legfeljebb $p(|x|)$ db megoldás van x-re. Ekkor $R \in$ **GEN**-3. Ez a 5.5 állítás a dolgozatban, és a 5. oldalon található.

2. Függvényproblémák

Ebben a fejezetben teszünk egy kis kitérőt, hogy megismerkedjünk a függvényproblémákkal.

2.1. Definíció. Vegyük egy $R(x, y)$ bináris relációt, melyre minden x esetén y legfeljebb polinomiálisan hosszabb, mint x. Egy T Turing-gép *kiszámolja* R-et, ha teljesül minden x inputra a következő:

- ha $\exists y$, hogy $R(x, y)$ igaz, akkor T outputként ad egy ilyen y-t;
- ha nincs ilyen y, akkot T elutasít.

Egy *függvényprobléma* egy ilyen R kiszámolása. Ha van egy polinomiális futási idejű T Turing-gép, mely kiszámolja $R(x, y)$-t, akkor R az **FP** függvényosztályban van. Ha ez nem feltétlenül teljesül, viszont egy Turing gép minden x, y pár esetén eldönti, hogy $R(x, y)$ igaz-e, akkor R *polinomiális reláció*, és az **FNP** függvényosztályban van.

2.2. Definíció (**TFNP**)**.** Egy $R \in \mathbf{FNP}$ függvényprobléma *teljes*, ha minden x esetén létezik y, melyre $R(x, y)$ teljesül. Az ilyen függvények osztályát **TFNP**-nek nevezzük.

2.3. Definíció (Polinomiális visszavezetés)**.** Legyen $R, Q \in \mathbf{TFNP}$. Ekkor R *visszavezethető* Q-ra (jel: $R \prec Q$), ha létezik $f : \Sigma_0^* \to \Sigma_0^*$ és $g : \Sigma_0^* \times \Sigma_0^* \to \Sigma_0^*$ polinomiálisan kiszámítható függvények, továbbá minden x, y esetén:

$$Q(f(x), y) \Longrightarrow R(x, g(x, y)).$$

2.4. Megjegyzés. Könnyű meggondolni, hogy **GEN**-3 zárt a polinomiális visszavezetésre a következő értelemben: ha $Q \in \mathbf{GEN}$-3, és $R \prec Q$, akkor $R \in \mathbf{GEN}$-3.

2.5. Definíció. Két **TFNP**-beli függvényprobléma polinomiálisan ekvivalens egymással, ha egymásra visszavezethetőek.

3. PPA-3

PPA-3 egy bonyolultsági osztály, alább leírjuk a klasszikus definícióját ([3] cikk szerint).

Tekintsük a következő A problémát. Adott egy T polinomiális (determinisztikus) Turing-gép, és egy $p(y)$ polinom. Legyen x egy input A-ra. A *konfigurációk tere* legyen $C(x) = \Sigma_0^{\leq p(|x|)}$, vagyis a legfeljebb $p(|x|)$ hosszúságú szavak halmaza. Ha $c \in C(x)$ egy konfiguráció, akkor T kiszámol $O(p(n))$ időben egy $T(x, c)$ halmazt, melyben polinomiálisan sok konfiguráció található. Definiáljunk egy gráfot a $C(x)$ halmazon. A $[c, c']$ élet húzzuk be pontosan akkor, ha $c \in T(x, c')$ és $c' \in T(x, c)$ egyszerre teljesül. Szeretnénk, hogy a gráfunk páros legyen. Ezt úgy hajtjuk végre formálisan, hogy a 0-val kezdődő csúcsok, és az 1-gyel kezdődő csúcsok legyenek a két csúcshalmaz. Vagyis azt akarjuk, hogy ha c 0-val kezdődik, akkor a $T(x, c)$-beli csúcsok 1-gyel kezdődjenek, és fordítva. Tehát, ha találunk c, c' párt, hogy mindkettő 0-val kezdődik (vagy mindkettő 1-gyel kezdődik), és $[c, c']$ él be van húzva, akkor ezt a párt is elfogadjuk az A probléma megoldásaként (hiszen megsérti azt a várakozásunkat, hogy a gráf páros legyen). Most tegyük fel, hogy $T(x, 00...0) = \{11...1\}$, és $00...0 \in T(x, 11...1)$, így $00...0$ egy levél. A feladatunk, hogy találjunk egy másik csúcsot, melynek foka nem osztható 3-mal.

3.1. Definíció (**PPA**-3)**.** A **PPA**-3 azon problémák halmaza, melyek a fenti alakra hozhatóak, arra visszavezethetőek.

Röviden a fenti problémát a következőképpen lehet megfogalmazni: adott egy polinomiális futási idejű Turing-gép, mely egy páros gráfot reprezentál, és a gráfban adott egy csúcs, melynek foka nem osztható 3-mal. Feladatunk, hogy találjunk egy másik ilyen csúcsot.

Persze a definícióban nem pontosan ezt mondtuk, hiszen ott egy levelet adtunk meg, mely speciálisabb, mint egy csúcs, melynek foka nem osztható 3-mal, azonban könnyű érvelés bizonyítja, hogy a kettő ekvivalens.

3.2. Megjegyzés. Fennáll **PPA**-3 $\subseteq$ **TFNP**, a következő okból. Ha a létrejövő gráf páros, akkor van benne egy levél (a 00...0), aminek a foka nem osztható 3-mal. Páros gráfban ekkor mindig van legalább egy másik ilyen csúcs, tehát ekkor van megoldás. Ha a létrejövő gráf nem páros, akkor van egy c, c' pár, mely ezt tanúsítja, ez szintén megoldás.

Az fog kiderülni, hogy a **PPA**-3-beli problémák mind olyanok lesznek, melyek esetén az 1.4 Kérdésre igenlő választ kapunk, azaz **PPA**-3 $\subseteq$ **GEN**-3. Ezt azonban nem közvetlenül bizonyítjuk be, hanem bevezetünk egy ekvivalens definíciót **PPA**-3-ra.

3.3. Állítás. Az alábbi 3 probléma páronként polinomiálisan ekvivalens egymással:

(i) Adott egy polinomiális futási idejű Turing-gép, mely egy páros gráfot ír le, melyben adva van egy csúcs, melynek foka nem osztható 3-mal. Keressünk egy másik ilyen csúcsot! (Ez a fentiek alapján ekvivalens a definiált **PPA**-3-teljes problémával).

(ii) Adott egy polinomiális Turing-gép, mely egy irányított gráfot reprezentál. Ebben a gráfban adott egy csúcs, melynek kifokából a befokát kivonva (ezt a csúcs *egyensúlyának* nevezzük) 3-mal nem osztható számot kapunk. Feladatunk, hogy találjunk egy másik ilyen csúcsot.

(iii) Adott egy (polinomiális) Turing-gép, mely kiszámol egy $f : \{1, 2, ..., 2^n\} \to \{1, 2, ..., 2^n\}$ függvényt, melyre $f^3 = \mathrm{id}$. Feladatunk f egy fixpontjának megtalálása.

3.4. Következmény. A következő problémára polinomiálisan visszavezethető feladatok osztálya megegyezik **PPA**-3-mal:

Legyen az abc bináris: $\Sigma = \{0, 1\}$. Így minden szó tekinthető 2-es számrendszerbeli számnak. Legyen x az input, hossza n. Adott egy $f(x, y)$ polinomiálisan kiszámítható függvény, és egy p polinom úgy, hogy ($m = p(n)$ jelöléssel) az

$$\begin{aligned} f_x : \Sigma_0^m &\to \Sigma_0^* \\ y &\mapsto f(x, y) \end{aligned}$$

függvényre teljesül, hogy Σ_0^m-be képez (vagyis igazából $f_x : \Sigma_0^m \to \Sigma_0^m$, a fenti megfeleltetéssel $f_x : [0, 2^m) \to [0, 2^m)$ írható), valamint $f_x(f_x(f_x(y))) = y$ minden $y \in [0, 2^m)$ esetén.

Az output:

- vagy egy y fixpont, melyre tehát $f_x(y) = y$;
- vagy egy $y \in [0, 2^m)$, mely tanúsítja, hogy f_x nem megfelelő: $f_x^3(y) \neq y$, vagy $f_x(y) \geq 2^m$ valamelyike teljesül.

3.5. Következmény. **PPA**-3 $\subseteq$ **GEN**-3.

4. NP-teljes feladatok

Eddig olyan problémákat láttunk, melyekről beláttuk hogy **GEN**-3-ban vannak. Most fordítva futunk neki: megmutatjuk, hogy nem minden **FNP**-problémára igaz ez, feltéve, hogy **NP** $\neq$ **co-NP** (amely nyitott kérdés).

Na de mit is jelent jelen helyzetben az, hogy **NP**-teljes probléma? Most függvényproblémákkal foglalkozunk, az **NP** pedig nyelvosztály. A következő módon lehet ezt érteni. Mint ismert, egy nyelv **NP**-beli volta megfogalmazható polinomiális *tanúk* segítségével: ha van egy $L \in$ **NP** nyelvünk, akkor létezik egy $R(x, y)$ polinomiális reláció (lásd 2.1 definíció), hogy $x \in L \iff \exists y : R(x, y)$. Ez a leírás már látszik, hogy egy függvényproblémát rejt magában: feladatunk, hogy minden x esetén keressük meg azt az y tanút, mely mutatja, hogy $x \in L$, vagy pedig írjuk ki, hogy $x \notin L$.

4.1. Állítás. Legyen $R(x, y)$ a SAT-ot leíró polinomiális reláció: egy x Boole-formulára és egy y kiértékelésre akkor és csak akkor igaz $R(x, y)$, ha az x formula az y kiértékelés mellett "igaz" értéket vesz fel. Tegyük fel, hogy $R \in$ **GEN**-3. Ha ez igaz, akkor **NP** = **co-NP**.

4.2. Megjegyzés. Hasonló érveléssel az állítás megfogalmazható minden **NP**-teljes problémára.

5. Komplexebb TFNP-beli problémák

Ebben a fejezetben a megoldások számáról teszünk fel különböző dolgokat, és ezekből próbáljuk levezetni egyes problémák **GEN**-3-beli voltát.

5.1. Állítás. Ha $R \in$ **TFNP**, és minden x esetén *előre tudjuk*, hogy hány y esetén teljesül $R(x, y)$, továbbá az ilyen y-ok száma nem osztható 3-mal, akkor $R \in$ **GEN**-3.

Nagy megkötés ennél az állításnál, hogy előre tudnunk kell a megoldások számát. Kijön viszont ebből egy másik eredmény, melynél egy másfajta korlát érvényes a megoldások számát illetően. Előbb azonban ehhez kimondunk egy lemmát.

5.2. Lemma. Legyen $R \in$ **TFNP**, és ehhez x egy input. Jelöljük ehhez a megoldások számát s_x-szel. Tegyük fel, hogy s_x felülről becsülhető x hosszának egy polinomjával (legyen ez a polinom q), továbbá mindig ismerjük s_x-et előre. Ekkor $R \in$ **GEN**-3.

5.3. Megjegyzés. A fenti lemma bizonyítása hasonlít a [2] cikkbeli 3. tétel bizonyításához.

5.4. Megjegyzés. Ez a lemma annyival tud többet az előző állításnál, hogy nem követeljük meg, hogy a megoldások száma ne legyen osztható 3-mal, viszont ennek ára az, hogy csak polinomiális lehet a megoldások száma.

Az alábbi állításban a lemmával ellentétben már nem kell tudnunk, hogy hány megoldás van, csak annyit, hogy legfeljebb polinomiálisan sok.

5.5. Állítás. Ha $R \in$ **TFNP** olyan, hogy minden x esetén a megoldások száma felülről becsülhető x hosszának egy polinomjával (legyen ez a polinom q), akkor $R \in$ **GEN**-3.

Hivatkozások

[1] URL: https://cstheory.stackexchange.com/questions/37773/can-we-fast-generate-perfectly-uniformly-mod-3-or-solve-np-problem.

[2] Paul W. Goldberg Alexandros Hollender. "The Complexity of Multi-source Variants of the End-of-Line Problem, and the Concise Mutilated Chessboard". *Electronic Colloquium on Computational Complexity* (2018).

[3] Christos H. Papadimitriou. "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence". *Journal of Computer and System Sciences* 48 (1994), 498–532. old.