Kombinatorikus számítási modellek MSc hallgatók számára

7. Hét

Előadó: Hajnal Péter 2011. Március 14—25.

1. Korlátos szélességű elágazó programok

Az előző alkalommal láttuk, hogy minden m esetén a $\mod_m(x)$ függvény kiszámolható korlátos szélességű elágazó programmal. A nyilvánvaló megvalósítás növekvő m-hez növekvő szélességet igényelt. Felmerül a kérdés mi az ereje a korlátos szélességű elágazó programoknak. Nem nehéz látni, hogy három szélességben minden kiszámolható.

Példa. Legyen $f : \{0,1\} \to \{0,1\}$ egy tetszőleges Boole-függvény. Leírunk egy szintezett, három szélességű elágazó programot, ami kiszámítja f-et.

A szintek blokkokra lesznek osztva. Minden elfogadó p inputra lesz egy B_p blokk. Ezen blokk feladata annak tesztelése, hogy az input azonos-e p-vel. A szinteken lévő három csúcs neve/szerepe „Már tudom, hogy $f = 1$", „Az aktuális teszten elbukott", „Tesztelés folyik". Az „Tesztelés folyik" csúcsok egy blokkon belül sorba végigkérdezik az x_i-k értékét. Ha bármelyik válasz nem egyezik p megfelelő bitjével, akkor az „Az aktuális teszten elbukott" csúcsba vezet a program. Ha a blokk végéig ez nem történik meg, akkor tudjuk, hogy az input p-vel egyenlő, f értéke 1. Ekkor az ennek megfelelő csúcsba kerülünk. Ez csak azért nem outputcsúcs, mert szintezett programot tervezünk (ilyen csúcsba kerülő számítási utak végig itt maradnak, minden csúcs/teszt felesleges (csak a szintezettséget szolgálja), a teszt eredményétől függetlenül a következő szint ugyanilyen típusú csúcsába kerülünk). Az általános esetben azonban nincs ilyen szerencsénk. Ekkor a következő blokk következik egy másik, p' input tesztelésével. Ha az összes olyan inputot teszteltük ahol f értéke 1 és sose kerültünk „Már tudom, hogy $f = 1$" csúcsba, akkor egy outputcsúcsnak nevezhetjük helyünket, ahol $f = 0$ bejelentés történik.

A felírt program nyilván az f függvényt számolja ki.

A fenti példa azt is mutatja, hogy a kis szélességnek ára van: a csúcsok száma/a program hossza n-ben (az inputbitek számában) exponenciális sok. Egyszerű függvényeknél is szükségszerű ez? Mi a helyzet például a $\text{Többség}_n(x)$ függvénnyel, ha korlátos szélességű elágazó programmal szeretnénk kiszámolni? Milyen sok kapu kell? Szükséges-e exponenciális sok? Vagy akár tekintsük a $\mod_{2011}(x_1, \ldots, x_n)$ függvény kiszámolását 10 szélességben.

A válasz 1986-ban született. Barrington konstruált nagyon hatékony elágazó programokat a fenti függvényekre (sőt egy ezeket tartalmazó általános függvényosztályra), amelyek polinomiális méretűek/hosszúak voltak, közben szélességük akár csupán öt volt. Algoritmusai/elágazó programjai nagyon speciálisak voltak. Először azt a környezetet írjuk le, amiben ő tervezte algoritmusait.

Definíció. Programunk csúcsai szintekre vannak bontva. Mindegyik szinten w csúcs van, amelyek indexeltek az $1, 2, \ldots, w$ indexszel. Az s-edik szinten ugynazt az x_{i_s}

változót teszteljük. A 0 válasznak megfelelő w darab (következő szinthez vezető) élt nevezzük 0-éleknek. Az 1 válasznak megfelelő w darab (következő szinthez vezető) élt nevezzük 1-éleknek. Feltételünk az, hogy bármely két szint között a 0-élek párosítást alkossanak és ugyanez teljesüljön az 1-élekre is. A fenti irányított gráfot *w-permutációs elágazó programnak* nevezzük. Az elnevezés oka kiderül abból, hogy hogyan is számol ki ez a program egy Boole-függvényt. Az elnevezésben szereplő w konstans. Ezt a paramétert általában — ha nem szükséges a hangsúlyozása — nem említjük.

Az első szinten is w csúcs van. Azaz nincs kitüntetett START-csúcs. A kezdeten az első szintre w kavicsot rakunk. Azonosításuk miatt kiszínezzük őket. Így egy az 1-től w-ig indexelt csúcsok egy színsorrendet alakítnak ki. Ez egy strandard kiinduló színsorrend lesz. Az első teszt eredménye 0 vagy 1 és ennek megfelelően éleink permutálják a kiinduló sorrendet. A számítás végén az utolsó szintre jutnak el a kavicsaink egy az inputbitektől függő sorrendben.

Akkor mondjuk, hogy a program kiszámol valamit, ha minden inputra a végső szinten kialakuló sorrend kétféle lehet. Nem megszorítás, ha az egyik sorrendről feltesszük, hogy az eredeti sorrend. Azaz kétféle permutáció történhet id bvagy π. A kiszámolt függvény f, ha $f(x) = 1$ esetén az elvégzett permutáció id és $f(x) = 0$ esetén az elvégzett permutáció π.

Első észrevételünk, hogy id és π kétféle sorrend különbözőségének egy szín pozíciójában is meg kell felelni. Ha ez például a „piros”, akkor a fenti programot kiritkíthatjuk: A kezdő szint piros csúcsa lesz a START-csúcs és az utolsó szint két pemutációjának két piros pozíciója lesz a két outputcsúcs. Így egy f-et kiszámító permutációs elágazó programból egy standard elágazó programhoz jutunk, amely ugyanazt a függvényt számolja ki, mérete (csúcsszáma) pedig nem nagyobb, közben szélessége w.

Megjegyzés. Egy kissé általánosabban is fogalmazhattunk volna. Bevezethettük volna a $\mathbb{G}$-elágazó programot, ahol $\mathbb{G}$ egy csoport. A számítás menetét egyetlen úttal is leírhattuk volna. Az aktuális helyen lévő részeredményt egy $\mathbb{G}$ csoportelem írná le. Az aktuális teszt végeredményétől (0 vagy 1) függően a részeredményt vagy g_0 vagy g_1 csoportelemmel szoroznánk és a következő csúcsba érnénk. Az utolsó csúcsban elért csoportelem lenne a kiszámolt érték, amiről feltennénk, hogy csak kétféle lehet (1 és h). 1 kódolná az elfogadást, h az elvetést. A w-permutációs elágazó programok fenti definiciója ennek egy kissé vizualizáltabb formája, amikor $\mathbb{G} = \mathbb{S}_w$, a w elemű alaphalmazra épülő szimmetrikuscsoport.

A fentiek után nyilvánvaló, hogy ha egy f függvényre tervezünk egy„kicsi” 5-permutációs elágazó programot, akkor f kiszámolható kicsi elágazó programmal öt szélességben.

1. Lemma. *Tegyük fel, hogy f-et kiszámoljuk egy permutációs elágazó programmal amelyben az elvetés kódja $\pi \in \mathbb{S}_w$. Ekkor $\neg f$is kiszámolható egy ugyanilyen paraméterű (hossz, szélesség, méret) permutációs elágazó programmal amelyben az elvetés kódja $\pi^{-1} \in \mathbb{S}_w$.*

2. Lemma. *Tegyük fel, hogy f-et kiszámoljuk egy permutációs elágazó programmal amelyben az elvetés kódja $\pi \in \mathbb{S}_w$. Legyen $\rho \in \mathbb{S}_w$ tetszőleges. Ekkor f-re adható egy alternatív, de ugyanilyen paraméterű (hossz, szélesség, méret) permutációs elágazó program, amelyben az elvetés kódja $\rho^{-1}\pi\rho \in \mathbb{S}_w$.*

3. Következmény. *Tegyük fel, hogy f-et kiszámoljuk egy permutációs elágazó programmal amelyben az elvetés kódja $\pi \in \mathbb{S}_w$. Ekkor f-re adható egy alternatív, de ugyanilyen paraméterű (hossz, szélesség, méret) permutációs elágazó program, amelyben az elvetés kódja π^{-1}.*

4. Lemma. *Tegyük fel, hogy f-et kiszámoljuk egy permutációs elágazó programmal amelyben az elvetés kódja $\pi \in \mathbb{S}_w$. Legyen a program hossza ℓ_f. Tegyük fel, hogy g-t kiszámoljuk egy permutációs elágazó programmal amelyben az elvetés kódja $\rho \in \mathbb{S}_w$. Legyen a program hossza ℓ_g. Tegyük fel, hogy $\rho^{-1}\pi\rho\pi^{-1}$ nem az egységelem. Ekkor $f \vee g$ kiszámítható egy permutációs elágazó programmal amelyben az elvetés kódja $\rho^{-1}\pi\rho\pi^{-1}$. Továbbá a program hossza $2(\ell_f + \ell_g)$.*

5. Feladat. *Igazoljuk a fenti három állításokat.*

A fenti három lemma kombinációja nagyon sok függvényre ad hatékony kiszámítási módot. Az alkalmazás előtt leírunk egy tág függvényosztályt, igazából függvénysorozatok egy osztályát.

Ehhez definíciók egy sorozata vezet.

Definíció. Egy irányított kör nélküli irányított gráf egy *hálózat*, ha minden csúcsának a befoka 0, 1 vagy 2. A csúcsok a következő szabály szerint címkézve vannak: A 0 befokú csúcsok (inputcsúcsok) címkéje az $\{x_1, x_2, \ldots, x_n\}$ halmaz egy eleme. Az 1 befokú csúcsok címkéje $\neg$. A 2 befokú csúcsok címkéje $\vee$ vagy $\wedge$.

Egy adott input mellett minden csúcs kiszámol egy bitet. Ezt rekurzív módon definiáljuk az inputcsúcsokkal kezdve azon csúcsokon keresztül, amelyekbe vezető élek kezdetén már megtörtént a kiszámított érték meghatározása (kiértékelése). A rekurzió indítása és a rekurzív lépések is természetesek.

A hálózat által kiszámított érték egy speciális csúcs (outputcsúcs) által kiszámított érték. Egy $\mathcal{C}$ hálózat $d(\mathcal{C})$ mélysége az outputcsúcsba vezető leghosszabb irányított út hossza.

Definíció. Egy $\{f_n(x_1, x_2, \ldots, x_n)\}_{n=1}^{\infty}$ függvénysorozat az $\mathcal{NC}^1_{nem-uniform}$ osztályba esik, ha van olyan $\{\mathcal{C}_n\}_{n=1}^{\infty}$ hálózatsorozat, amely $\mathcal{C}_n$ eleme f_n-et számolja ki és mélysége $\mathcal{O}(\log n)$.

Megjegyzés. Az osztály jelölésében szereplő nem-uniformitás lényege, hogy a hálózatok sorozatáról nem követeljük meg, hogy egy egyszerű algoritmussal legyárthatók legyenek (n ismeretében). Azaz megengedünk olyan hálózatsorozatokat, amelyek „nagyon hektikusan” változnak, ahogy n-et növeljük.

Ezekután egy tetszőLeges $\mathcal{NC}^1_{nem-uniform}$-beli függvényre a korábbi lemmák iterálásával konstruálunk hatékony permutációs elágazó programot. A főtételt két lemma vezeti fel.

6. Lemma. *Legyen $\mathcal{C}$ egy hálózat. Ekkor létezik olyan hálózat, amely ugyanezt a függvényt számolja ki, de csúcsain nem használjuk az $\wedge$ címkét. Továbbá mélysége legfeljebb $3d$, ahol d a $\mathcal{C}$ hálózat mélysége.*

7. Feladat. *Igazoljuk az előző lemmát.*

8. Lemma. *$\mathbb{S}_5$-ben létezik két, π és ρ öt hosszú ciklus, hogy kommutátoruk ($\pi^{-1}\rho\pi\rho^{-1}$) szintén öt hosszú ciklus.*

Bizonyítás.

$$(24531)(54321)(13542)(12345) = (13254).$$

■

Emlékeztetünk, hogy két permutáció akkor és csak akkor konjugált, ha ciklusstruktúrájuk megegyező. Azaz két öt hosszú ciklus mindig konjugált.

9. Tétel (Barrington-tétel). *Legyen* $\{f_n(x_1, x_2, \ldots, x_n)\}_{n=1}^{\infty} \in \mathcal{NC}^1_{nem-uniform}$ *tetszőleges. Részletezve: van olyan* $\{\mathcal{C}_n\}_{n=1}^{\infty}$ $\neg$-$\vee$ *kapukat használó hálózatsorozat, amelyre* $\mathcal{C}_n$ *az* f_n*-et számolja ki és mélysége* $\alpha_f \cdot \log n$*. Ekkor* f_n *kiszámolható öt szélességű permutációs elágazó programmal, amely hossza* $4^{\alpha_f \cdot \log n} = \mathcal{O}(n^{\beta_f})$*.*

Bizonyítás. Egy kicsit erősebbet bizonyítunk. Mindegyik programunk az elvetést egy öt hosszú ciklussal fogja jelezni.

A $\mathcal{C}_n$ hálózat mélysége szerinti indukcióval bizonyítunk.

Ha a mélység 0, akkor a függvény $f(x) = x_i$ (az egyik inputcsúcs egyben az outputcsúcs is). Az állítás nyilvánvaló.

Az indukciós lépésnél eseteket különböztetünk meg az outputcsúcs címkéje szerint.

1. eset: $f_n = \neg f$.

2. eset: $f_n = g \vee g'$.

Mindkét eset egyszerűen adódik az indukciós feltevésből és lemmáinkból. ■

10. Feladat. *Dolgozzuk ki a részleteket (rakjuk össze az összerakós játék darabjait).*

11. Lemma. mod-$m(x_1, \ldots, x_n)$*,* $Többség_n(x)$ *függvények* $\mathcal{NC}^1_{nem-uniform}$*-beli függvények.*

12. Feladat. *Igazoljuk a fenti lemmát.*

2. Egy alsó becslés permutációs elágazó programokra

Barrington fenti eredménye mutatja, hogy permutációs elágazó programok ereje milyen nagy. Ezt az érzést finomítja az alábbi eredmény.

13. Feladat. *Igazoljuk, hogy a* $ÉS(x_1, \ldots, x_n)$ *Boole-függvény kiszámítható kettő szélességű, n hosszú elágazó programmal.*

Belátjuk, hogy ez a nagyon egyszerű függvényt kiszámító tetszőleges permutációs elágazó program szuperlineáris hosszú.

14. Tétel (Cai—Lipton-tétel, 1994). *Minden w-permutációs elágazó program, amely kiszámolja a* $ÉS(x_1, \ldots, x_n)$ *függvényt* $\Omega(n \log \log n)$ *hosszú.*

Bizonyítás. Általánosan $\mathbb{G}$-elágazó programokkal dolgozunk. Tegyük fel, hogy az i-edik tesztre a 0 választ a $g^{(i)}$ elem kódolja és a 1 válasz kódja id (ez feltehető, a szükséges átlakítás a hosszt nem befolyásolja). Legyen L a program hossza. Indirekten tegyük fel, hogy $L < n \log_3 \log_3 n/4$. Legyen $\nu = n/\log_3 \log_3 n$. Felteszük, hogy n elég nagy (n kicsi értékeire az állítást könnyen igazzá tehetjük a rejtett konstans alkalmas váltasztásával).

15. Feladat. *Igazoljuk van olyan $\tau \leq \log_3 \log_3 n/2$, hogy a pontosan τ-szor tesztelt változók száma legalább ν.*

A fenti feladat garantál legalább ν változót, amelyek pontosan τ-szor teszteltek. Feltehető, hogy ezek $x_1, x_2, \ldots, x_\nu$ és indexelésük első tesztjük sorrendjét követi. Az összes többi változó értékét rögzítsük 1-nek. Így egy ν változós ÉS-t számol ki (megszorított) programunk.

A változók indexelése (első előfordulása) alapján lehet egy sorbarendezett változó halmazról azt mondani, hogy csökkenő vagy növekvő. Célunk, hogy kiemeljünk ℓ megmaradt változót úgy, hogy az i-edik előfordulásaik ($i = 1, 2, \ldots, \tau$) sorrendje „szabályos" legyen. Pontosan fogalmazva: Az előfordulásaik sorozata diszjunkt blokkokra osztható legyen úgy hogy minden blokkban mindegyik kiemelt változó ugyanannyiszor szerepljen, a különböző változók egy blokkban részblokkokban következzenek és a különböző változók vagy növekvő vagy csökkenő sorrendben következzenek. Azaz egy blokkban a kiemelt változók sorozata

$$x_{i_1}, x_{i_1}, \ldots, x_{i_1}, x_{i_2}, x_{i_1}, \ldots, x_{i_2}, \ldots, x_{i_\ell}, x_{i_\ell}, \ldots, x_{i_\ell}$$

legyen, ahol $i_1, i_2, \ldots, i_\ell$ csökkenő vagy növekvő sorozat. Természetesen minél nagyobb ℓ értékre szeretnénk ezt elérni.

Az ehhez vezető út mozgató ereje Erdős Pál és Szekeres György következő lemmája.

16. Lemma. *Számok egy legalább $s \cdot t + 1$ hosszú sorozatában biztos lesz $s + 1$ hosszú monoton növő vagy $t + 1$ hosszú monoton csökkenő részsorozat. (Egy sorozat részsorozata olyan sorozat, amely az eredeti sorozatból elemek kihúzásával keletkezik.)*

17. Feladat. *Igazoljuk a fenti állítást.*

Vegyük a megmaradt ν változó második tesztelését/előfordulását. Ez bizonyos sorozatban történik. Ebből kiválasztható $\nu^{1/3}$ hosszú csökkenő sorozat vagy $\nu^{2/3}$ hosszú növekvő sorozat. Ha csökkenő részsorozatot találunk, akkor a programban ezek olyan blokkban fordulnak elő, ami diszjunkt (sőt későbbi) mint első előfordulásaik blokkja. Növekvő részsorozat esetén ez nem igaz.

18. Lemma. *Az $1, 2, \ldots, s$ sorozat két példányát tetszőlegesen fésüljük össze. Ekkor lesz $i_1, i_2, \ldots, i_t, i_1, i_2, \ldots, i_t$ vagy $i_1, i_1, i_2, i_2, \ldots, i_t, i_t$ részsorozat, ahol $t \geq \sqrt{s}$.*

19. Feladat. *Igazoljuk ezt a lemmát.*

Ezekután a lemmáink alkalmazása elvezet oda, hogy $\nu^{1/3}$ változót kiemeljünk (a többi értékét 1-nek rögzítsük) és kialakítsunk egy vagy két blokkot (az első esetben ebben az egy blokkban minden változó kétszer tesztelt), ahogy megígértük. Vesszük a megmaradt változók harmadik tesztelését és a fenti gondolatmenetet megismételjük, majd iteráljuk. τ iteráció után megmaradt $n_0 = n^{(1/3)^\tau}$ változónk és lesz legfeljebb τ blokkunk. Ha egy blokkban többször teszteljük túlélő változóinkat, akkor ezeket egymásután teszteljük és ezeket a teszteket egybeolvaszthatjuk. Ezek után blokkjaink kétfélék lesznek: egyikben az n_0 megmaradt változót sorban (index szerint növekedve) teszteljük a másik típusban az n_0 megmaradt változót fordított sorban teszteljük.

Egy B blokkok hatása a csupa 0 inputon egy g_B elemmel való szorzás lesz. Ez n_0 csoportelem szorzataként adódik. Most az első, majd második, majd a harmadik

bitet cseréljük át 1-esre, és így tovább. Minden cserénél egyre rövidebb szorzat adja a blokkhatását a számításra. Végezzük el ugyanezt a „kisérletet" az összes blokkra. Így mindig $\mathbb{G}^\tau$ egy elemét kapjuk, ha az egyes blokkok hatását külön vizsgáljuk. Ha n_0 olyan nagy, hogy meghaladja $|\mathbb{G}|^\tau$-t, akkor a skatulya-elv garantálja, hogy ahogy változik az 1-esek száma két vizsgált inputon az összes blokk ugyanúgy hat. Ez azt jelenti, hogy egy $1^i0^j1^k$ alakú input esetén minden blokk hatása id lesz. Ez azonban ellentmond annak, hogy programunk az ÉS függvényt számolja ki.

Ami hátra van, az annak ellenőrzése, hogy a fent megadott paraméterek korrektek. Azaz a kezdeti $\tau \leq \log_3 \log n/2$ választás olyan, hogy

$$n_0 = n^{(1/3)^\tau} > |\mathbb{G}|^\tau.$$

Ez könnyen ellenőrizhető nagy n esetén. ■