STRINGEK EGYENLŐSÉGÉNEK ELLENŐRZÉSE :
Legyen A és B két string:

A:	B:
$a_1,a_2,\ldots,a_n \in \{0,1\}^n$	$b_1,b_2,\ldots,b_n \in \{0,1\}^n$
$a=a_n2^{n-1}+a_{n-1}2^{n-2}+\ldots+a_1$	$b=b_n2^{n-1}+b_{n-1}2^{n-2}+\ldots+b_1$

\ /

a két sztring, akkor és csak akkor egyenlő, ha a=b, ehhez n bitet kell ellenőrizni, de kevesebb bitet szeretnénk küldeni ellenőrzésre.

Vegyünk egy p prímet! Legyen Fp(a) = a mod p az ujjlenyomat. Ha Fp(a) = Fp(b), akkor elfogadjuk az egyenlőséget. Viszont determinisztikusan választva a prímet $p < 2^n$ esetben : az ellenfél tud olyan példát adni, amikor hibázik az algoritmus. Az ötlet, hogy p legyen véletlenül generált prím (ilyet találni nehéz, de ezzel nem foglalkozunk). Tehát p-t véletlenül választjuk a τ -nál kisebb prímek közül, ahol $\tau < 2^n$. Ekkor a kommunikáció: O (log τ)-nyi bit. A hiba valószínűsége az Pr(Fp(a)=Fp(b) | a ≠ b) , ha ez kicsi, akkor jó az algoritmus. A prímszámtétel kimondja, hogy $\prod(\tau) \cong \tau / \ln(\tau)$, ahol $\prod(\tau)$ = a τ-nál kisebb prímek száma. Az algoritmus során $\prod(\tau)$ féle prímet választhatunk, és akkor van baj, ha a számok ugyanabba a maradékosztályba esnek, azaz p osztója a-b abszolút értékének, legyen c=|a-b|.
Másrészt ha $c \le 2^n$, akkor maximum n db prím osztója van. (Mivel ha x-nek több mint n db prím osztója van, akkor mind legalább 2, így $x \ge 2^m$.) Tehát legyen τ egy küszöbérték, ahol $\tau > n=\log c$. Ekkor a fentiek alapján $Pr(Fp(a) = Fp(b) \mid a \ne b) \le n / (\tau / \ln(\tau)) = (n / \tau) \ln(\tau)$ (mivel maximum n osztója lehet c-nek, és a véletlen prímet pedig $\tau / \ln(\tau)$ darab prím közül választhatjuk.
Ha $\tau = t\, n \ln(tn)$, valamely nagy t-re, akkor a fenti valószínűség $n / (t\, n \ln(tn)) \ln(t\, n \ln(tn)) \le 2 / t$ ($\approx O(1/t)$). Az átküldött bitek száma pedig O (ln(t) + ln(n)). Így ha t = n , akkor n bit helyett log(n) bit átküldése esetén a hiba valószínűsége legfeljebb O (1/n) .

MINTAILLESZTÉS

A mintaillesztés problémája a következő: adott két bináris sztring, S egy n bitből álló szöveg, amiben keresünk és egy M m db bitből álló mintát, amelynek az előfordulásait szeretnénk megtalálni. A feladat determinisztikus algoritmussal O(m+n) időben megoldható (ld.: Algoritmusok II.). O(mn) időben egy triviális kimerítő keresésen (brute force) alapuló algoritmus is megoldja. Itt is használhatjuk az ujjlenyomat technikát.
Legyen $S_j = s_j s_{j+1}, \ldots, s_{j+m-1}$ a j-edik helyen kezdődő m hosszú részsztring, $1 \le j \le m-n+1$, legyen
$F(S_j) = s_j * 2^{m-1} + s_{j+1} * 2^{m-2} + \ldots + s_{j+n-1}$. Ekkor $S_j = M$ esetén $F(S_j) = F(M)$. Az algoritmus ezen értékeknek a fentiekben definiált ujjlenyomatát hasonlítja össze az F(M) érték ujjlenyomatával.

Lényeges észrevétel, hogy $S_{j+1} = 2 [S_j - s_j 2^{m-1}] + s_{j+m}$, azaz a sorozat tagjai az előző tagból hatékonyan konstans időben számolhatóak, így a véletlenül generált prímre is $Fp(S_{j+1}) = 2 [Fp(S_j) - s_j * 2^{n-1}] + s_{j+m} \bmod p$.
Tehát ujjlenyomatként az Fp(Z)=Z mod p függvényt használjuk egy véletlenül generált p egy τ korlátnál nem nagyobb prímre. Ugyanúgy mint az első részben a sztringek összehasonlításánál kapjuk, hogy az algoritmus hibája legfeljebb $(n / \tau) \ln(\tau) = O((n \log \tau) / \tau)$ tehát kapunk egy Monte Carlo algoritmust. Legyen τ olyan, hogy a hiba O(1/n) –re egyszerűsödjön.

A Monte Carlo algoritmus: Visszaadja az első j-t, amire $Fp(S_j) = Fp(M)$
Adódik, hogy a futásidő: O(m+n).
$P(\text{hiba}) \le 1/n$, hiba: ha mégsem egyezik valójában: $Fp(S_j) = Fp(M)$ de $S_j \ne M$

Las Vegas algoritmussá alakíthatjuk át a következőképpen:

1. MC futtatás O(m+n) idő
2. valóban illeszkedik-e a minta (ellenőrzés) O(m) idő
3. ha igen, akkor vége O(1) idő
4. ha nem, akkor Brute Force

A 4. lépés a Brute Force algoritmus legfeljebb 1/n valószínűséggel hajtódik végre, a futási ideje O(mn) így ezen lépés várható futási ideje is O(m+n). Következésképpen a Las Vegas algoritmus futási ideje O(m+n).

Összefoglalás

Összefoglalva, a következő 3 módszert használjuk ujjlenyomatok definiálására.
1. módszer: Legyen a kérdés, hogy $Q_1(x) = Q_2(x)$ igaz -e, ahol Q egy F test feletti polinom. A módszer egyenletes eloszlás alapján egy r kiértékelési hely választása, és $Q_1(r) = Q_2(r)$ tesztelése. Ez a sztring egyezésnél is használható. Legyen $Q_1 = a_1 x^{n-1} + a_2x^{n-2} + \ldots + a_1$ és $Q_2 = b_1 * x^{n-1} + b_2 * x^{n-2} + \ldots + b_2$ a két sztring akkor és csak akkor egyezik meg, ha $Q_1(x) = Q_2(x)$.
2. módszer: A prímet választjuk véletlennek és adott x pontban értékelünk. A fenti példákban x=2 volt az érték ahol értékeltünk.
3. módszer: Ha a kérdés a = b , akkor megtehetjük azt is, hogy rögzítünk egy p prímet és egy véletlenül generált Q polinomra az F_p test felett vizsgáljuk, hogy Q(a) = Q(b) teljesül -e.

INTERAKTÍV BIZONYÍTÁSI RENDSZEREK

Az interaktív bizonyítási rendszerben két játékos van.

1. Ellenőrző személy V, aki egy állításról szeretné eldönteni, hogy igaz vagy hamis. (Bonyolultságelméleti definíciókkal egy adott L nyelv esetén egy x szóról akarja eldönteni $x \in L$ teljesül-e.) Az ellenőrző polinomiális idejű véletlenített algoritmusokat használhat és kérdést tehet fel a bizonyítónak.
2. Bizonyító P, az ő esetében nincs időkorlát (mindent ki tud számolni). De lehet rosszindulatú bizonyító is, aki hazudhat (válaszra és a bizonyításra is) és V nem tudja leellenőrizni polinomiális időben ezt (mert pl exponenciális idejű a bizonyítás).

Def: Interaktív bizonyítási rendszer van egy eldöntendő L kérdésre, azaz a kérdés az IP bonyolultságelméleti osztályba tartozik (interaktív polinom idejű osztály, ha $\exists$ V ellenőrző és P bizonyító, úgy hogy ha:

- igaz a válasz ($x \in L$), akkor Pr(V(x,P) = elfogad) = 1
- ha nem a válasz ($x \notin L$), akkor $\forall$ P' esetleg rosszindulatú bizonyítóra is:Pr(V(x,P')= elfogad) $\leq$ ½ .

GRÁFIZOMORFIZMUS

Def.: $G_1(V_1,E_1)$ izomorf $G_2(V_2,E_2)$ akkor és csak akkor, ha létezik egy π permutációja az 1,2,…,|V| számoknak (csúcsok), hogy $(i,j) \in E_1$ akkor és csak akkor, ha $(\pi(i), \pi(j)) \in E_2$.
A gráfizomorfizmus problémája NP -ben van, ha izomorf a két gráf könnyű bizonyítást ellenőrizni (az adott permutációt kell csak ellenőrizni izomorfizmus -e.) A lenti ábrán két izomorf gráf van.

GRÁF NEM IZOMORFIZMUS

Válasz: igen, ha G_1 nem izomorf G_2-vel.
Interaktív bizonyítási séma:

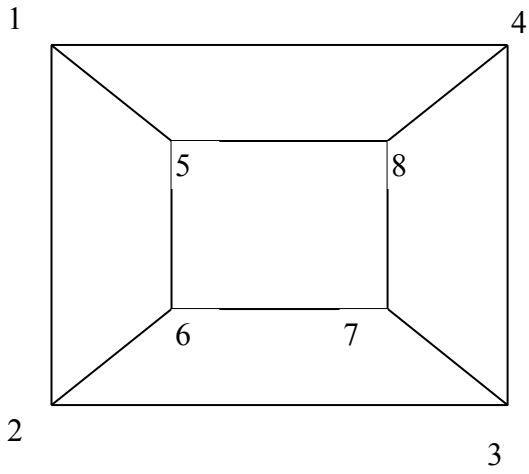
1. σ egy véletlen permutáció
2. $i \in \{1,2\}$ véletlen értéket generálunk egyenletes eloszlás alapján
3. G_i gráfon σ permutációt végrehajtjuk, így kapjuk a H gráfot
4. H-t odaadjuk a bizonyítónak, és megkérdezzük, hogy G_1 -el vagy G_2-vel izomorf.
5. Ha válaszként G_i-t kapjuk, akkor elfogadjuk, hogy G_1 és G_2 nem izomorf.

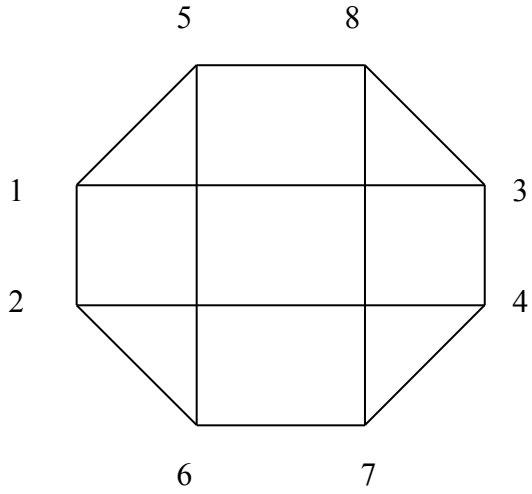
Állítás: Ez egy interaktív bizonyítási rendszer.
Bizonyítás:

- ha G_1 és G_2 nem izomorfak (tehát a válasz igen), akkor P meg tudja mondani melyiket választottuk, ezért a séma a választ elfogadja.
- ha G_1 és G_2 izomorfak (tehát a válasz hamis), akkor H alapján a bizonyító nem tudja eldönteni, melyik gráfot választottuk. Annyit tehet, hogy x valószínűséggel 1-et mond, és 1-x valószínűséggel 2-t mond. Viszont ebben az esetben a válasz elfogadásának valószínűsége

$$P(\text{elfogad}) = ½ x + ½ (1-x) = ½$$

Példa izomorf gráfokra:





H3SAT PROBLÉMA

Def: 3CNF egy konjuktív normálforma, ahol minden tag 3 darab $\vee$ jellel összekapcsolt esetleg negált változót tartalmaz, a formula a tagok konjukciója. (példák 3CNF alakra: $(x \vee y \vee z) \wedge (a \vee b \vee c) \wedge \ldots \wedge (e \vee f \vee g)$, $(x \vee !y \vee z) \wedge (y \vee z \vee c) \wedge \ldots \wedge (e \vee f \vee g)$)
Def: 3SAT probléma: Adott 3CNF kielégíthető -e?
Def: H3SAT probléma: Adott 3CNF hányféleképpen elégíthető ki?

Tétel: Adott S érték esetén a H3SAT = S probléma az IP osztályban van.
Bizonyítás: Elsőként aritmetizáljuk a formulát, ami egy gyakran használható technika. Rendeljük az x_i logikai változóhoz az $f(x_i)=1-x_i$ aritmetikai függvényt, a $!x_i$ logikai változóhoz az x_i aritmetikai függvényt. Továbbá a $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$ logikai kifejezéshez az $f(c_i)= 1 - f(l_{i1})f(l_{i2})f(l_{i3})$ aritmetikai kifejezést. Például:
Az $x_1 \vee !x_2 \vee x_3$ logikai kifejezés aritmetizálása az $1-(1-x_1)x_2(1-x_3)$ kifejezést adja. Könnyen látható, hogy egy c klóz akkor és csak akkor igaz, ha f(c)=1. A teljes CNF aritmetizálását úgy kapjuk, hogy összeszorozzuk a benne szereplő klózok aritmetizáltjait. Például, ha $x = (x_1 \vee ! x_2 \vee x_3) \wedge (!x_2 \vee x_3 \vee x_4)$, akkor
$f(x) = (1-(1- x_1) x_2 (1- x_3))(1- x_2 (1- x_3)(1- x_4))$
Könnyen belátható, hogy egy x 3CNF kifejezés akkor és csak akkor igaz, ha f(x)=1.
Most legyen $Hf(x) = \sum^1_{x1=0}\sum^1_{x2=0}\ldots\sum^1_{xn=0} f(x_1,\ldots,x_n)$, ez az összeg pontosan a H3SAT probléma megoldását adja, mivel az összegben azon x_i értékek esetén kapunk 1-et, amelyek kielégítik a logikai formulát. Használjuk a következő jelöléseket $f_n(x_1,\ldots,x_n) = f(x_1,\ldots,x_i, x_{i+1},\ldots,x_n)$, azaz f_n egy n - változós függvény, a lerögzített változóknak megfelelő tag értéke a Hf összegből. Továbbá minden i-re definiáljuk a f_i i változós függvényt, amely az a részösszeg, ahol az első i darab változó van lerögzítve, ekkor ezen függvényekre teljesül, hogy $f_i(x_1,\ldots,x_i) = f_{i+1}(x_1,\ldots,x_i, 0) + f_i(x_1,\ldots,x_i,1)$. A fenti rekurziók alapján definiált f_0 konstans (0-változós függvény) pontosan a vizsgált Hf(x) értéket adja vissza. Tehát az eldöntendő kérdés az, hogy f_0=S igaz -e.
A következő interaktív sémát használjuk:

Elsőként a bizonyítót megkérjük, hogy adja meg nekünk az f_1 egyváltozós függvényt (ennek a fokszáma legfeljebb 3n). Ellenőrizzük, hogy a kapott g függvényre (nem tudhatjuk biztosan, hogy valóban f_1-et kaptuk meg) g(0)+ g(1) = S teljesül -e. Ha nem, akkor a választ nem fogadjuk el, egyébként ellenőrizzük, hogy valóban az f_1 függvényt kaptuk e meg. Ehhez legyen p egy prím, ami nagyobb, mint 3n, és válasszunk egy r értéket a p feletti prímtestből véletlenszerűen. Vizsgáljuk meg, hogy $f_1(r) = g(r)$ teljesül -e, ha a válasz igen, akkor elfogadjuk a bizonyítást egyébként nem. A korábbiakhoz (előző óra) hasonlóan kapjuk a hiba valószínűségére, hogy Pr(hiba)=Pr(nem ugyanazt a függvényt adja vissza, de ugyanazt a kiértékelést kapjuk) $\leq 3m / p$.

Az egyetlen megmaradt probléma az, hogy mi nem tudjuk $f_1(r)$ pontos értékét. Ennek meghatározásához vegyük észre, hogy $f_1(r) = \sum^1_{x2=0} \ldots \sum^1_{xn=0} f(r,x_2,\ldots,x_n)$, tehát annak eldöntése, hogy $f_1(r) = g(r)$ teljesül -e, pontosan ugyanolyan feladat mint az eredeti feladatunk, csak egy változóval kevesebb van, így a sémát rekurzívan újra felhasználva ezt az egyenlőséget is ellenőrizhetjük.
Mivel összesen n db kérdés-felelet van, a teljes hiba valószínűségére azt kapjuk, hogy
$P(\text{hiba}) = P(g(r) = f_1(r) \mid g(z) \neq f_1(z)) \leq n * 3m / p \leq ½$, ha p értékét elegendően nagynak választjuk.

VISSZAFELÉ ELEMZÉS

Vegyük a következő rendező algoritmust. Vegyük a számok egy véletlen sorrendjét (ha determinisztikus algoritmusként tekintjük, akkor az átlagos futási időre kapunk ugyanilyen korlátot). Utána használjuk a beszúró rendezés egy olyan változatát, ahol minden még be nem szúrt elemre egy kétirányú mutató számon tartja, hogy a már rendezett számok közé hova kell beszúrni.

Például, ha a kezdeti sorrend 1 7 5 4 8 3 9 6, akkor az algoritmus a következő iterációkkal rendez:
1
7 5 4 8 3 9 6
1 7
3 5 4 6 8 9
1 5 7
3 4 6 8 9 stb..

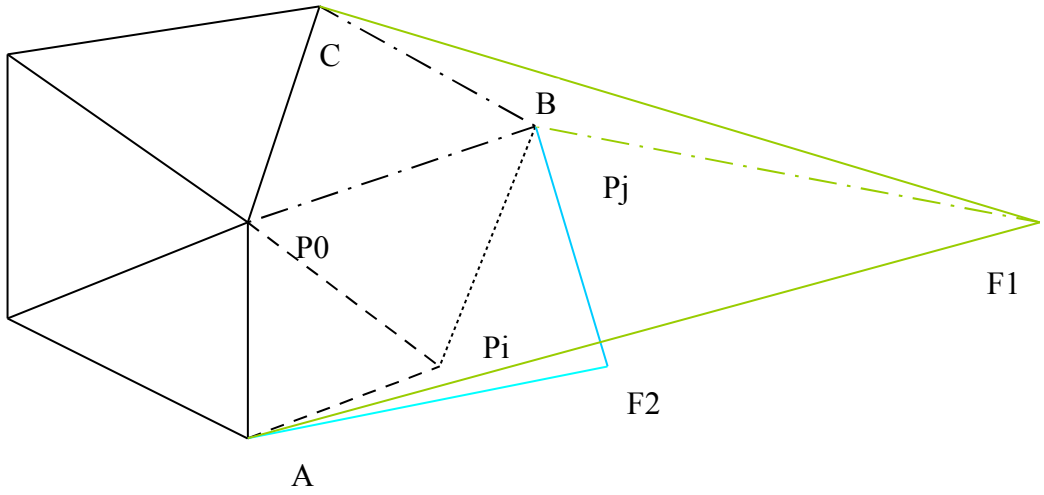
Várható futási időt nyilvánvalóan a mutatók átállításának a költsége határozza meg, maga a beszúrás a mutatók alapján konstans időben megtehető, így a beszúrások összköltsége lineáris. A visszafele elemzés alapötlete, hogy azt vizsgáljuk mennyi a költség, ha nem beszúrunk hanem töröljük az elemeket (a két költség megegyezik, csak ezt az utóbbit könnyebb kiszámolni). Tehát adottak a sorozatban rendezett x_1, x_2 … , x_i számok és egyenletes eloszlás alapján törlünk egy elemet, azaz minden elemet 1 / i valószínűséggel. Eddig az x_{i+1}, …, x_n elemeket már töröltük, tehát róluk már mutat mutató a jelenlegi elemek közötti intervallumokba.
Az egyenletes eloszlás miatt a mutató átállítások várható száma: (n-i) / i. Minden $j \in x_{i+1}$, …, x_n ,-re annak a valószínűsége, hogy j-t átállítjuk 1/i, tehát az adott elem átállításának várható értéke 1/i. Viszont az összes átállítások várható értéke, ezen várható értékek összege, tehát E(átállítás száma) = (n-i) / i.
Következésképpen az algoritmus futása során az összes átállítás várható költsége, így az algoritmus futási ideje legfeljebb $\sum^n_{i=1} (n-i) /i \leq \sum^n_{i=1} n/i = n \sum^n_{i=1} 1/i = O(n \log n)$.

KONVEX BUROK

Def: Egy ponthalmaz konvex, ha bármely két pontjára az őket összekötő szakasz bármely pontja is a ponthalmazban van.
Def: Egy ponthalmaz konvex burka a legszűkebb olyan konvex halmaz, amely a pontok mindegyikét tartalmazza.
A következő véletlenített algoritmust használhatjuk adott ponthalmaz konvex burkának meghatározására. (A két dimenziós problémát tekintjük, de az eljárás kiterjeszthető több dimenzióra is)

1. Véletlen sorrendbe állítjuk a pontokat
2. Veszünk 3 pontot, és vesszük egy belső pontját az általuk meghatározott háromszögnek pl a súlyközéppontját (P0)
3. Minden pontra meghatározzuk, hogy melyik tartományba esik a P0 és a csúcsok által meghatározott egyenesekkel meghatározott tartományok közül, ezt számon tartjuk egy kétirányú mutatóval.
4. Vesszük sorban a pontokat. Az adott pontot beszúrjuk a konvex burokba felhasználván, hogy tudjuk melyik tartományban van.
4.a Ha az új pont az eddigi burkon belül esik, akkor nincs semmi teendőnk. (Ezt konstans időben meghatározhatjuk, mivel tudjuk melyik tartományban van).
4.b Ha az új pont kívülre esett, akkor meg kell vizsgálni hogy keletkezett e konkáv boruk rész (Pj), vagy belső burok rész (Pi), ha igen akkor azt a pontot ki kell szedni a burokalkotók sorából. Majd át kell állítani a mutatókat az új tartományoknak megfelelően.

A következő ábrán a kék színnel jelölt két él hatására a belső Pi pontot és a hozzá tartozó éleket kell törölni. Ha viszont nem a kék hanem a zöldes élek kerülnek felvételre, akkor Pi-t és Pj -t is törölni kell, és a hozzájuk tartozó éleket (itt először az A F1 és a B F1 él kerül felvételre, majd a B pontnál látjuk hogy konkáv burok rész van, így B csúcsot és éleit is törölni kell, ekkor jön létre C F1 él)



Az algoritmus futási ideje két részből adódik össze:

Az élek törlésének száma. Ez lineáris, hiszen csak ha már létre lett hozva egy él akkor törölhetjük, viszont lépésenként legfeljebb 2 élet hozunk létre és n lépés van.

Az átállított mutatók száma. Ennek összeszámolásához ismét a visszafele elemzést használjuk. Tegyük fel, hogy töröljük a pontokat a konvex burokból nem pedig beszúrjuk őket. Amikor n-i pontot már kitöröltünk, akkor annak a valószínűsége, hogy a kitörölt pontok közül egy adott P pontra a (P0, P) mutatót át kell állítani megegyezik azzal a valószínűséggel, hogy annak az élnek valamely végpontját töröljük, amely él P tartományát meghatározza. Ennek pedig 2/i a valószínűsége. Tehát a várható értéke az átállított mutatók számának legfeljebb 2n / i az i-dik lépésben, így összesen O(n log n), azaz az algoritmus futási ideje is : O(nlog n).
Megjegyzés A feladat két dimenzióban megoldható determinisztikus algoritmussal is O(nlog n) időben.

Készítette: Sánta Róbert