Algel II. gyakorlat

A dinamikus programozás jó

2009. február 17/19.

1. **Legyen $w = w_1 w_2 \cdots w_n$ egy n betűből álló szó. Hívjuk részszónak w egy tetszőleges $w_i w_{i+1} \cdots w_{i+k}$ darabját ($1 \leq i \leq n-1$, $1 \leq k \leq n-i$)! Adjunk algoritmust, ami $O(n)$ lépésben meghatározza az összes a-val kezdődő és b-re végződő részszó számát!**
Észrevesszük, hogy egy tetszőleges b karakter előtti összes a karakter meghatároz egy-egy ilyen szót. A szövegben haladva ha a-t találunk, akkor az eddigi a-k számát növeljük eggyel, ha b-t, akkor pedig a végeredményhez hozzáadjuk az a-számláló aktuális értékét. Az algoritmus egy végigolvasásból megvan, azaz $O(n)$, valamint helyes (ezt nagyon precízek indukcióval is bebizonyíthatják).

4. **Legyenek $a_1, a_2, \ldots, a_n$ tetszőleges egész számok és $m < n^2$ egész. Adjunk algoritmust, amely a bináris alakjukkal megadott $a_1, a_2, \ldots, a_n$ és m számokról eldönti polinom időben, hogy az $a_1, a_2, \ldots, a_n$ számok közül kiválasztható-e néhány úgy, hogy az összegük m-mel osztva egyet adjon maradékul!**
Vegyük fel az m-hez tartozó teljes maradékrendszert egy (mondjuk bool) A tömbbe! Kezdetben minden hely $false$. Az a_i szám kézbevételekor $\forall j : A[j] = true$ helyre állítsuk $A[(j + a_i) \pmod m]$-et $true$-ra, majd $A[a_i \pmod m]$-et is. Ha $A[1]$ $true$ lesz, akkor kiválasztható, ha a végére sem, akkor nem. Ezt be is kell bizonyítani! Azt állítom, hogy a_i kézbevétele után pontosan azok az $A[j]$-k vannak $true$-ra állítva, amik valahogy előállnak $a_1 \ldots a_i$ közül néhány összegeként $\pmod m$. Indukcióval nyilván igaz, hiszen a_1 választásával csak $a_1 \pmod m$ áll elő. Tfh k-ra igaz, ekkor nézzük $k+1$-re. $a_1 \ldots a_{k+1}$ közül néhányat választva vagy választjuk a_{k+1}-et, vagy nem. Ha nem, akkor pontosan azok állnak elő, amik $a_1 \ldots a_k$ közül, ha igen, akkor egyrészt előállhat maga a_{k+1}, másrészt az eddig előálló számok mindegyikéhez hozzávehetjük a_{k+1}-et is. Ezért a feltétel igaz maradt, hiszen az összes lehetséges esetet lefedtük. A lépésszám polinomiális, hiszen $O(nm)$ műveletet végzünk (minden számhoz a teljes táblán végigmegyünk), ami ebben a spec. esetben azért polinomiális, mert $m = O(n^2)$. (Ha m-re nem lenne korlát, akkor ez akár exponenciális is lehetne.)

5. **Egy n szóból álló szöveget kell sorokra tördelni. A szöveg i-edik szava ℓ_i karakterből áll, egy sor s karakter hosszú. Ha egy sor a szöveg i-edik szavával kezdődik és a j-edik szóval végződik, akkor az elválasztó szóközöket is figyelembe véve $t = s - (\ell_i + \ell_{i+1} + \cdots + \ell_j + j - i)$ üres hely marad a sor végén. Egy ilyen sor hibája legyen t^2. A tördelés hibája a nem üres sorok hibáinak összege. Adjunk $O(n^2)$ lépéses algoritmust egy legkisebb hibájú tördelés meghatározására! (A szavak sorrendje rögzített.)**
Elég csúnyán hangzik, de nem az. Szokásosan valami olyasmire kell gondolni, hogy egy sor vizsgálatánál már rendelkezésünkre álljon az összes megelőző érdekes adat. Vegyünk fel egy D tömböt, ahol $D[i]$ értéke legyen az $1 \ldots i$ szavak legkisebb hibájú tördelésének értéke! $D[0] = 0$ jelentse azt, hogy 0 szót 0 sorba hiba nélkül be tudunk rakni. Amikor épp az i-edik szót vizsgáljuk, akkor a következő lehetőségeink vannak: i-t külön sorba vesszük, az $i-1$-ig bezárólag pedig a lehető legjobb tördelést adjuk (jé, ez pont $D[i-1]$!); $i-1$-et és i-t vesszük egy sorba, az előttük levőket pedig a legjobban tördeljük (csak nem $D[i-2]$?);...; 1-től i-ig csinálunk egy sort. Meg is van a képlet: $D[i] = \min_{j=1\ldots i}(t_{j,i}^2 + D[j-1])$, a helyességet a fenti gondolatmenet adja. $D[n]$ fogja az optimális tördelés értékét megadni. Ha meg is akarjuk kapni a tényleges tördelést, akkor egy T tömbben nyilvántarthatjuk, hogy adott i-hez $D[i]$-nél hogyan kaptuk a minimumot, azaz hol van a sortörés. $T[n]$-től visszafele lépkedve megkapjuk a sortörések helyét. Mivel az i-edik szónál 1-től i-ig végignézzük az eddigi tömböt (azaz $1, 2, \ldots, n$ műveletet végzünk), a lépésszám $O(n^2)$ lesz.

8. **Adott egy fa, melynek csúcsaihoz súlyok vannak rendelve. Adjunk lineáris algoritmust, ami meghatározza a fában található maximális súlyú független ponthalmaz súlyát!**
Csak nem dinamikus programozással oldjuk meg ezt is? Dehogynem! A feladat szerint nincs megadva, hogy mi a fa gyökere, de bármikor megtehetjük, hogy hasraütéssel kijelölünk egyet. Így már van értelme szintekről, szülőkről és egyebekről beszélni. Észrevehetjük, hogy ha veszünk egy csúcsot, és egy belőle kiinduló részfát, akkor két lehetőségünk van. Vagy nem vesszük be függetlennek az aktuális csúcsot, és ekkor a gyerekeiből kiinduló részfákban vehetjük a maximális súlyú független ponthalmazt (hiszen az aktuális csúcs nem zavar be semmibe, mert úgysincs kiválasztva); vagy pedig bevesszük őt is. Ha bevesszük, akkor viszont a gyerekeit semmiképp nem használhatjuk, az unokáitól kezdve viszont nincs annak hatása, hogy vele mit csinálunk. Innen már adódik az algoritmus: minden v csúcshoz tároljuk a v-ből induló részfában a maximális súlyú független ponthalmaz súlyát ($W(v)$). Egy levélben ez triviálisan a levél súlya (negatív súlyok léte esetén $\max(0, s(v))$). A levelektől a gyökér fele haladva egy belső v csúcsnál a súly: $\max(s(v) + \sum_{u \in unoka} W(u), \sum_{u \in gyerek} W(u))$. A gyökérnél levő érték fogja a keresett súlyt adni. (Nagyon precízek ezt is leírhatják szépen indukcióval, de a fentebbi gondolatmenet is elég.) A lépésszámról se feledkezzünk meg: mivel a fokszám nincs korlátozva, ilyesmi érveléssel könnyen rossz eredményre juthatnánk, vagy sokat kéne számolni. Azt érdemes megnézni, hogy egy adott v csúcsot hányszor veszünk a kezünkbe. Egyszer foglalkozunk vele, amikor őt, egyszer amikor a szülőjét, egyszer pedig amikor a nagyszülőjét vizsgáljuk. Így a legpesszimistább esetben is minden csúcsot legfeljebb háromszor piszkálunk, tehát n csúcs esetén $O(3n) = O(n)$, azaz lineáris lépésszámunk van.