Algoritmuselmélet
2011. február 14.

Schlotter Ildi
ildi@cs.bme.hu

2. gyakorlat
Dinamikus programozás

1. Adott egy n és egy m hosszú sorozat: $x_1x_2\dots x_n$ és $y_1y_2\dots y_m$. A két sorozat részsorozata egy $1 \leq i_1 < \dots < i_k \leq n$, és egy $1 \leq j_1 < \dots < j_k \leq m$ indexsorozattal adható meg, ahol $x_{i_\ell} = y_{j_\ell}$ teljesül minden $1 \leq \ell \leq k$ esetén. Adjon $O(nm)$ lépésszámú algoritmust, ami a két megadott sorozatban meghatároz egy leghosszabb közös részsorozatot.

2. Egy n és egy m karakterből álló szövegben meg akarjuk találni a legnagyobb azonos darabot, azaz ha az egyik szöveg $a_1a_2\cdots a_n$ és a másik $b_1b_2\cdots b_m$, akkor olyan
$1 \leq i \leq n$ és $1 \leq j \leq m$ indexeket keresünk, hogy
$$a_{i+1} = b_{j+1}, a_{i+2} = b_{j+2}, \dots, a_{i+t} = b_{j+t}$$
teljesüljön a lehető legnagyobb t számra. Adjon erre a feladatra $O(mn)$ lépést használó algoritmust.

3. Legyenek $a_1, a_2, \dots, a_n$ tetszőleges egész számok és $m < n^2$ egész. Adjon algoritmust, amely a bináris alakjukkal megadott $a_1, a_2, \dots, a_n$ és m számokról eldönti polinom időben, hogy az $a_1, a_2, \dots, a_n$ számok közül kiválasztható-e néhány úgy, hogy az összegük m-mel osztva egyet adjon maradékul.

4. Adott egy n elemű A tömb, melynek elemei valós számok. Adjon $O(n)$ lépésszámú algoritmust, amely meghatároz olyan i és j indexeket, melyre az $A[i] + A[i+1] + \dots + A[j]$ összeg a lehető legnagyobb.

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 $t = s - (\ell_i + \ell_{i+1} + \dots + \ell_j + j - i)$ üres hely marad a sor végén. Egy ilyen sor hibája legyen t^2. A tördelés hibája a nemüres sorok hibáinak összege. Adjon $O(n^2)$ lépéses algoritmust egy legkisebb hibájú tördelés meghatározására! (A szavak sorrendje rögzített.)

6. Adjon algoritmust, ami egy n csúcsú fában lineáris időben meghatározza a fában levő leghosszabb út hosszát.

7. Adottak $M_1, M_2, \dots, M_n$ munkák $H_1, H_2, \dots, H_n$ határidőkkel és $P_1, P_2, \dots, P_n$ profitokkal. Mindegyik munka elvégzése pontosan 1 napunkba kerül (így egy napon csak egy munkát végezhetünk el). Adjunk hatékony algoritmust, amely meghatározza, hogy mely munkákat vállaljuk el a profit maximalizálása érdekében!

Gyakorlás:

1. Legyen $w = w_1w_2\cdots w_n$ egy n betűből álló szó. Hívjuk részszónak w egy tetszőleges $w_iw_{i+1}\cdots w_{i+k}$ darabját ($1 \leq i \leq n-1$, $1 \leq k \leq n-i$).

   a) Adjon algoritmust, ami $O(n)$ lépésben meghatározza az összes a-val kezdődő és b-re végződő részszó számát.

   b) Javasoljunk egy minél kevesebb összehasonlítást igénylő módszert annak az eldöntésére, hogy w tartalmazza-e részszóként az $abca$ szót. Elemezzük a módszer költségigényét!

2. Legyen F egy n csúcsú fa, tetszőleges v csúcsának a súlya pedig $s(v)$. Adjunk polinom idejű algoritmust, amely meghatározza az F fában található legnagyobb összsúlyú független ponthalmaz súlyát.

3. Legyenek $a_1, a_2, \dots, a_n$ tetszőleges pozitív egész számok és legyen b is pozitív egész szám. Adjon algoritmust, amely a bináris alakjukkal megadott $a_1, a_2, \dots, a_n$ és b számokhoz $O(nb)$ időben megadja, hogy a b szám hányféleképpen áll elő az $a_1, a_2, \dots, a_n$ számok közül néhány összegeként.

4. Adottak $t_1, t_2, \dots, t_n$ tárgyak, az i-edik tárgy súlya egy s_i egész szám ($0 < s_i < 100$). Ezeket a tárgyakat a sorrendet megőrizve ládákba kell pakolnunk, minden ládába összesen legfeljebb 100 súlyú tárgyakat rakhatunk. Minden láda után adót kell fizetnünk: ha egy ládába s összsúlyú tárgyakat rakunk, akkor $(100-s)^2$ gyémántfélkrajcárt kell fizetnünk. Adjunk algoritmust, amely meghatároz egy legolcsóbb pakolást $O(n^2)$ időben!

5. Az orosz cár sürgősen levelet szeretne küldeni a kínai császárnak. Ehhez a már jól bejáratott, Szentpétervárról Pekingbe vivő útvonalat szeretné használni, amin n város helyezkedik el sorban egymás után: $v_1, v_2, \dots, v_n$. A levelet postagalambok viszik egyik városból a másikba, a célba juttatásához persze több galambra is szükség lehet egymás után. A felesleges kitérők elkerülésének érdekében a galambok mindig olyan v_i városból viszik a levelet v_j-be, amire $i < j$ teljesül. Segítsünk megtervezni a cárnak a leggyorsabb útvonalat, ha minden galambról tudjuk, hogy honnan hova tud repülni, és mennyi idő alatt teljesíti a távot!
   Elemezzük a megadott algoritmus hatékonyságát is!