Dinamikus programozás

Algoritmuselmélet

2. gyakorlat

2023.

1. Jelölje egy algoritmus maximális lépésszámát az n-méretű bemeneteken $L(n)$. Tegyük fel, hogy az algoritmus szerkezetéből következik, hogy tetszőleges $n \geq 2$ esetén $L(n) \leq 2L\big(\lceil n/2 \rceil\big) + n$, valamint $L(1) = 1$.

   (a) Lássuk be, hogy ekkor $L(n) \in O(n \log n)$. Igaz-e, hogy $L(n) \in O(n^2)$?

   (b) Igaz-e, hogy ha tetszőleges $n \geq 2$ esetén $L(n) \leq 2L\big(\lceil n/2 \rceil\big) + dn$ valamely d konstansra, valamint $L(1) = 1$, akkor is $L(n) \in O(n \log n)$ teljesül?

   A feladatot elég abban az esetben meggondolni, amikor n egy 2-hatvány.

2. Egy $n \times n$ méretű táblázat mezőin lépkedünk a bal alsó sarokból a jobb felső sarokba úgy, hogy egy lépésben a táblázatban vagy felfelé vagy jobbra egyet lépünk, de van néhány „tiltott” mező, ahova nem léphetünk. Adjunk egy dinamikus programozást használó eljárást, ami meghatározza, hogy hányféleképpen érhetünk célba. Mi az algoritmus lépésszáma?

3. Adjunk dinamikus programozást használó algoritmust, ami egy n egész számhoz, mint bemenethez $O(n)$ lépésben kiszámolja F_n-et, azaz az n-edik Fibonacci-számot. A Fibonacci számok sorozatát az alábbi rekurzióval definiáljuk: $F_0 = F_1 = 1$ és tetszőleges $n \geq 2$ esetén $F_n = F_{n-1} + F_{n-2}$.

4. Egy f-fokú létrán bizonyos fokok annyira rozogák, hogy ha rálépünk, leszakadnak. Szerencsére tudjuk, hogy melyik fokok ilyenek, hova nem szabad lépnünk. Tegyük fel, hogy a legfelső fokra rá szabad lépni. Egy lépéssel legfeljebb 3 fokot tudunk lépni. Adjunk dinamikus programozást használó algoritmust, ami meghatározza, hogy

   (a) a létra aljától fel tudunk-e jutni a létra legfelső fokára;

   (b) a létra aljától hányféleképpen tudunk feljutni a létra legfelső fokára.

   Mennyi az algoritmusok lépésszáma?

5. (a) Adott $n+1$ darab ($n \geq 1$) pozitív egész szám: $s_1, s_2, \ldots, s_n, b$ és azt szeretnénk eldönteni, hogy előáll-e a b szám néhány s_i szám összegeként. (Mindegyik s_i szám legfeljebb egyszer használható fel az összegben.) Adjunk dinamikus programozást használó $O(n \cdot b)$ lépésszámú algoritmust erre a feladatra (melynek neve Részhalmaz-összeg probléma, röviden RH) az alábbi részfeladatok felhasználásával: tetszőleges $0 \leq i \leq n$ és $0 \leq c \leq b$ esetén $T(i, c)$ jelentése legyen az, hogy elő lehet-e állítani az $s_1, \ldots, s_i$ számokból a c számot.

   (b) Adjunk algoritmust arra az esetre is, ha azt is meg akarjuk határozni, hogy hányféleképpen áll elő a b szám az elvárt alakban.

6. Egy n és egy m karakterből álló szövegben meg akarjuk találni a legnagyobb azonos darabot, azaz ha az egyik szöveg $a_1 a_2 \ldots a_n$ és a másik $b_1 b_2 \ldots b_m$, akkor olyan $1 \leq i \leq n$ és $1 \leq j \leq m$ indexeket keresünk, hogy $a_{i+1} = b_{j+1}, a_{i+2} = b_{j+2}, \ldots, a_{i+t} = b_{j+t}$ teljesüljön a lehető legnagyobb t számra. Adjunk erre a feladatra $O(mn)$ lépést használó algoritmust.

7. A $b_1, b_2, \ldots, b_n$ sorozat különböző nemnegatív egészekből áll. Szeretnénk meghatározni a legnagyobb összeget a monoton növő részsorozatok körében. Például ha a sorozat a 2, 1, 10, 6, 3, 8, 4, 9, akkor a legnagyobb összeg 25, amit a 2, 6, 8, 9 részsorozatnál érünk el. Adjunk $O(n^2)$ futásidejű algoritmust, ami megtalálja a legnagyobb elérhető összeg értékét.

8. Egy w méter széles folyón szeretnénk átkelni a folyó medrébe lerakott n darab cölöp segítségével. A cölöpök a folyó két partja között, a folyó partjára merőleges egyenes vonalban helyezkednek el, $0 < x_1 < x_2 < \ldots < x_n < w$ méterre a kiindulási parttól (w és az összes x_i távolság egész szám). Az első lépésben egy métert ugorhatunk, utána pedig az igaz, hogy minden ugrás vagy pontosan egy méterrel nagyobb, vagy pontosan egy méterrel kisebb, vagy ugyanakkora, mint az előző. Adjunk algoritmust, ami az x_i számok ismeretében $O(nw)$ lépésben eldönti, hogy át tudunk-e jutni a túlsó partra anélkül, hogy a vízbe esnénk.

9. Zárthelyit szervezünk, amin összesen H hallgató fog részt venni, és ehhez T darab terem áll rendelkezésünkre. Tudjuk, hogy az i-edik terembe h_i hallgató fér be. de lehet benne kevesebb is; $\sum h_i \geq H$. A terem geometriája, oszlopok, stb. miatt ha az i-edik terembe kerül zh-t író hallgató, akkor itt f_i fő zh-felügyelőre van szükségünk. (Tegyük fel, hogy f_i független a terembe ténylegesen kerülő hallgatók számától, feltéve, hogy ez legalább egy fő.) Adjunk dinamikus programozást használó algoritmust, amivel $O(T \cdot H)$ lépésben meghatározható, mely termeket használjuk a T közül ahhoz, hogy elég legyen a hely a hallgatóknak, de az összesen szükséges teremfelügyelők száma minimális legyen.

10. Tekintsük azt a rekurzív algoritmust F_n, azaz az n-edik Fibonacci-szám kiszámolására, ami az $n = 0$ és $n = 1$ esetben 1-et ad vissza, tetszőleges $n \geq 2$ esetén pedig meghívja saját magát $n - 1$ és $n - 2$ inputtal, majd visszaadja az így kapott két érték összegét. Lássuk be, hogy ennek az eljárásnak a lépésszáma $\Omega(2^{n/2})$.