1. Adjon meg a $10, 3, 5, 2, 7, 1, 18, 4, 12, 17, 6$ sorozatban egy leghosszabb növekvő részsorozatot az órán tanult dinamikus programozást használó algoritmussal!

2. Egy f fokú létrán bizonyos fokok annyira rozogák, hogy ha rálépünk, leszakadnak. Szerencsére tudjuk, hogy melyik fokok ilyenek, hova nem szabad lépnünk. Egy lépéssel legfeljebb 3 fokot tudunk lépni. Adjon 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! *(Feltehető, hogy a legfelső fokra rá szabad lépni.)* Mennyi az algoritmusok lépésszáma?

---

3. 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 legfeljebb egyszer használható fel az összegben.) Adjunk dinamikus programozást használó $O(n \cdot b)$ lépésszámú algoritmust erre a feladatra (melynek neve Részhalmaz-összeg probléma, rövidítése RH) az alábbi részfeladatok felhasználásával:
$L(i,c)$ jelentése ($0 \leq i \leq n$ és $0 \leq c \leq b$ esetén) legyen az, hogy elő lehet-e állítani az $s_1, \ldots, s_i$ számokból a c számot.

4. Egy $m \times m$ 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. Adjon egy dinamikus programozást használó eljárást, ami meghatározza, hogy hányféleképpen érhetünk célba!

5. Legyen $s_1 s_2 \ldots s_n$ és $t_1 t_2 \ldots t_m$ egy n és egy m hosszú karaktersorozat. Azt szeretnénk, hogy az $n \times m$ méretű A mátrix $A[i,j]$ eleme tartalmazza azt a legnagyobb k számot, melyre az $s_1 s_2 \ldots s_i$ és a $t_1 t_2 \ldots t_j$ sorozatok utolsó k karaktere megegyezik. Adjon eljárást, ami az A tömböt $O(nm)$ lépésben kitölti.

6. Egy $n \times n$ méretű táblázat minden eleme egy pozitív egész szám. A táblázat bal alsó sarkából akarunk eljutni a jobb felső sarkába úgy, hogy egy lépésben a táblázatban vagy felfelé vagy jobbra egyet lépünk. Azt szeretnénk, hogy a lépegetés során látott elemek növekvő sorrendben kövessék egymást. Adjon $O(n^2)$ lépésszámú algoritmust, ami meghatározza, hogy (a) hány a szabályoknak megfelelő út van! (b) mekkora a legnagyobb értékű, a szabályoknak megfelelő út, ha egy út értéke a benne szereplő számok szorzata!

---

7. Van n fájlunk, az i-edik fájl hosszát jelölje a h_i. Tegyük fel, hogy a h_i számok pozitív egészek. Mentéshez két egyformán L méretű lemez áll rendelkezésünkre (L pozitív egész szám). A cél, hogy minél nagyobb k számra az első k darab fájl mindegyikét mentsük ki a lemezekre (a fájlok sorrendje rögzített). Fájlokat szétvágni nem szabad, minden fájl teljes egészében kerül az egyik vagy a másik lemezre. Adjon algoritmust, ami adott L és h_i számokhoz meghatározza, hogy melyik fájlt melyik lemezre tegyük ahhoz, hogy k a lehető legnagyobb legyen. Az algoritmus lépésszáma legyen $O(L^2)$.

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

9. Adott egy n és egy m hosszú 0-1 sorozat, $a_1, a_2, \ldots, a_n$, illetve $b_1, b_2, \ldots, b_m$. Ezek alapján egy T tömböt töltöttünk ki a következő módon:

   Ha $0 \leq i \leq n$, akkor $T[i,0] = 0$. Ha $0 \leq j \leq m$, akkor $T[0,j] = 0$.

   Ha $1 \leq i \leq n$ és $1 \leq j \leq m$, akkor $T[i,j] = \begin{cases} T[i-1,j-1]+1 & \text{ha } a_i = b_j \\ \max\{T[i,j-1], T[i-1,j]\} & \text{ha } a_i \neq b_j \end{cases}$

   Mi a jelentése a $T[i,j]$ értéknek! A két sorozatnak milyen tulajdonságát adja meg a $T[n,m]$ érték?

10. Tekintsük azt a rekurzív algoritmust az n. Fibonacci-szám, F_n kiszámolására, ami $n = 0$ és $n = 1$ esetben 1-et ad vissza, $n \geq 2$ esetben 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 $\Omega(2^{\frac{n}{2}})$.