1. Mi az alábbi állításoknak a tagadása? *(Két állítás akkor tagadása egymásnak, ha a két állítás közül minden esetben pontosan az egyik igaz.)* Próbáljuk úgy megfogalmazni a tagadásokat, hogy ne szerepeljen bennük tagadószó.

   (a) Minden szerdán van algel előadás.

   (b) Minden olyan hallgató, aki jár algel gyakorlatra, teljesíti a tárgyat.

   (c) Minden olyan 17 lábú zsiráf, aki jár algel gyakorlatra, az teljesíti a tárgyat. *(Igaz ez?)*

   (d) Van olyan hallgató, aki sokat tanul, de nem teljesíti a tárgyat.

   (e) Mindenki, aki teljesíti a tárgyat, sokat tanult.

   Megoldás:

   (a) Van olyan szerda, amikor nincs algel előadás.

   (b) Van olyan hallgató, aki jár algel gyakorlatra, de nem teljesíti a tárgyat.

   (c) Van olyan 17 lábú zsiráf, aki jár algel gyakorlatra, de nem teljesíti a tárgyat. *(Az eredeti állítás igaz, a tagadás nem igaz.)*

   (d) Minden olyan hallgató, aki sokat tanul, teljesíti a tárgyat.

   (e) Van olyan hallgató, aki keveset tanult, de teljesíti a tárgyat.

2. Tudjuk, hogy minden hömpörő surjancs. Mondjuk meg minden alábbi állításra, hogy biztosan igaz, lehetséges, vagy biztosan hamis! *(Ha nehéz a feladat, akkor legyen a hömpörő=kertitörpe és surjancs=szobor.)*

   (a) Tudjuk valamiről, hogy nem hömpörő. Azt állítom, hogy ez surjancs.

   (b) Tudjuk valamiről, hogy hömpörő. Azt állítom, hogy ez nem surjancs.

   (c) Tudjuk valamiről, hogy nem surjancs. Azt állítom, hogy ez hömpörő.

   (d) Tudjuk valamiről, hogy nem surjancs. Azt állítom, hogy ez nem hömpörő.

   (e) Tudjuk valamiről, hogy surjancs. Azt állítom, hogy ez nem hömpörő.

   Megoldás:

   (a) Lehet.

   (b) Hamis.

   (c) Hamis.

   (d) Igaz.

   (e) Lehet.

3. Mi lehet a $T(n)$ függvény, ha teljesül $T(1) = 2$ és $T(n) = 3 \cdot T(n-1) + 1$ minden $n \geq 2$ esetén?

   Megoldás: **I.** Elkezdjük kibontani a rekurziót. A kibontás célja, hogy eljussunk a $T(1)$-hez, ahol konkrét értéket tudunk behelyettesíteni, majd zárt alakra hozunk.

   II. Próbaként menjünk vissza két lépést:

   $$T(n) = 3 \cdot T(n-1) + 1,$$
   $$T(n-1) = 3 \cdot T(n-2) + 1 \qquad n \text{ helyére } (n-1)\text{-et írva.}$$

   Helyettesítsük $T(n-1)$-et az eredeti egyenletbe:

   $$T(n) = 3\Big(3 \cdot T(n-2) + 1\Big) + 1 = 3^2 \cdot T(n-2) + 3 + 1.$$

   III. Folytatva az előző módszert, végül ezt kapjuk:

   $$T(n) = 3^{n-1} \cdot T(1) + \sum_{i=0}^{n-2} 3^i.$$

Mivel $T(1) = 2$, ezért:

$$T(n) = 2 \cdot 3^{n-1} + \sum_{i=0}^{n-2} 3^i.$$

IV. A zárt alak a mértani sorozat összegképletéből következik:

$$\sum_{i=0}^{n-2} 3^i = \frac{3^{n-1} - 1}{3 - 1} = \frac{3^{n-1} - 1}{2}.$$

V. Végső, kompakt forma:

$$T(n) = 2 \cdot 3^{n-1} + \frac{3^{n-1} - 1}{2} = \frac{5 \cdot 3^{n-1} - 1}{2}.$$

4. Jelölje egy algoritmus maximális lépésszámát az n hosszú bemeneteken $L(n)$. Azt tudjuk, hogy minden $n > 3$ egész számra $L(n) \leq L(n-1) + \frac{n}{2}$ teljesül, és hogy $L(3) = 3$. Milyen felső becslést adhatunk ez alapján $L(n)$-re?

Megoldás: **1. lépés:** Az előző megoldásban ismertetett startégiát követjük. Bontsuk ki a rekurziót két lépésig:

$$L(n) \leq L(n-1) + \frac{n}{2},$$
$$L(n-1) \leq L(n-2) + \frac{n-1}{2}.$$

Helyettesítsük be a második egyenletet az elsőbe:

$$\begin{aligned} L(n) &\leq \left(L(n-2) + \frac{n-1}{2} \right) + \frac{n}{2} \\ &= L(n-2) + \frac{n-1}{2} + \frac{n}{2} \end{aligned}$$

Ezt folytatva eljutunk:

$$L(n) \leq L(3) + \sum_{k=4}^{n} \frac{k}{2}.$$

Mivel $L(3) = 3$, tovább írhatjuk:

$$L(n) \leq 3 + \frac{1}{2} \sum_{k=4}^{n} k.$$

2.1. lehetőség a befejezésre: (Pontosabb számolás) A számtani sorozat összegképletét szerint:

$$\sum_{k=4}^{n} k = \frac{(4+n)(n-3)}{2}.$$

Így:

$$L(n) \leq 3 + \frac{(4+n)(n-3)}{4}.$$

Mivel

$$(n+4)(n-3) = n^2 + n - 12,$$

így

$$L(n) \leq 3 + \frac{n^2 + n - 12}{4} = \frac{n^2 + n}{4}.$$

Mivel $\frac{n^2+n}{4} \leq \frac{1}{2}n^2$ minden $n \geq 1$ esetén, végül:

$$L(n) \leq \frac{1}{2}n^2.$$

2.2. lehetőség a befejezésre: Minden $\frac{k}{2}$ tagot felülről $\frac{n}{2}$-vel becsülve

Mivel minden k értéke a sorozatban legfeljebb n (hiszen $4 \leq k \leq n$), ezért

$$\frac{k}{2} \leq \frac{n}{2} \quad \text{minden } k \text{ esetén.}$$

Az alábbi becslést kapjuk:

$$L(n) \leq 3 + (n-3) \cdot \frac{n}{2}.$$

Erről peidg nyilvánvalóan látszik, hogy legfeljebb n^2, így

$$L(n) \leq n^2.$$

Megjegyzés. A későbbiekben látni fogjuk, hogy a multiplikatív konstansokkal nem törődünk, így nekünk a második becslés ugyanolyan jó lesz, mint az első.

5. Tegyük fel, hogy van egy számítógépes programunk, ami egy k méretű feladaton a jelenlegi gépünkön 1 nap alatt fut le. Beszereztünk egy százszor gyorsabb számítógépet. Ugyanazon programmal mekkora feladatot lehet az új gépen egy nap alatt megoldani, ha a program lépésszáma n méretű feladat esetén
(a) n-nel arányos, (b) n^3-bel arányos, (c) 2^n-nel arányos?

Megoldás: Legyen a gépünk A (jelenlegi) és B (új, 100-szor gyorsabb) indexe, illetve $T_A(n)$ és $T_B(n)$ az algoritmus futási ideje az adott gépen n méretű feladat esetén. Mivel a gép B 100-szor gyorsabb, a B gépen az idő:

$$T_B(n) = \frac{T_A(n)}{100}.$$

Tegyük fel, hogy a program az A gépen egy k méretű feladatot 1 nap alatt old meg:

$$T_A(k) = 1\text{nap}.$$

Ez alapján a B gépen 1 nap alatt végrehajtott műveletek száma megfelel annak, amit az A gép 100 nap alatt végez el. Megnézzük, ehhez mekkora inputok tartoznak a különböző esetekben.

(a) Ha a futási idő n-nel arányos, azaz $T_A(n) = c \cdot n$.

Tudjuk, hogy $c \cdot k = 1$, mennyi k_{uj}, ha $c \cdot k_{uj} = 100$? Ebből

$$k_{uj} = 100\, k.$$

(b) Ha a futási idő n^3-mal arányos, azaz $T_A(n) = c \cdot n^3$.

Tudjuk, hogy $c \cdot k^3 = 1$, mennyi k_{uj}, ha $c \cdot k_{uj}^3 = 100$? Ebből

$$k_{uj}^3 = 100\, k^3 \quad \Longrightarrow \quad k_{uj} = k \cdot 100^{1/3} \approx 4{,}64k.$$

(c) Ha a futási idő 2^n-nel arányos, azaz $T_A(n) = c \cdot 2^n$.

Tudjuk, hogy $c \cdot 2^k = 1$, mennyi k_{uj}, ha $c \cdot 2^{k_{uj}} = 100$? Ebből

$$2^{k_{uj}} = 100 \cdot 2^k \quad \Longrightarrow \quad 2^{k_{uj}-k} = 100 \quad \Longrightarrow \quad k_{uj} = k + \log_2 100 \approx k + 6{,}64.$$

6. 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 algoritmust ami meghatározza, hogy a létra aljától fel tudunk-e jutni a létra legfelső fokára! *(Feltehető, hogy a legfelső fokra rá szabad lépni.)* Az algoritmus lépésszáma legyen $c \cdot f$, ahol c valami fix konstans.
Hogyan kell módosítani az algoritmust, hogy azt is kiszámolja, hogy hányféleképpen lehet feljutni a legfelső fokra?

Megoldás: **Az elérhetőség vizsgálata**

Legyen x_i egy bináris változó, amely megmutatja, hogy a i-edik fok használható-e:

$$x_i = \begin{cases} 1, & \text{ha a fok ép és rá lehet lépni,} \\ 0, & \text{ha a fok sérült és nem használható.} \end{cases}$$

Az algoritmus célja, hogy meghatározza, elérhető-e az f-edik fok a létra aljától, ha minden lépésben legfeljebb 3 fokot léphetünk előre. Definiáljunk egy új változót, y_i-t, amely azt jelöli, hogy az i-edik fok elérhető-e:

$$y_i = \begin{cases} x_i, & \text{ha } i \leq 3, \\ \max(y_{i-1}, y_{i-2}, y_{i-3}), & \text{ha } x_i = 1,\ i > 3 \\ 0, & \text{ha } x_i = 0. \end{cases}$$

`A képlet értelmezése`:

- Az első három fok esetén a fok elérhető, ha az adott fok ép ($x_i = 1$).
- Ha $x_i = 1$, akkor az i-edik fok elérhető, ha az előző három fok valamelyike is elérhető (y_{i-1}, y_{i-2} vagy y_{i-3}).
- Ha az adott fok törött ($x_i = 0$), akkor oda nem tudunk lépni ($y_i = 0$).

A megoldás végén ellenőrizzük y_f értékét:

- Ha $y_f = 1$, akkor el lehet jutni az f-edik fokra.

- Ha $y_f = 0$, akkor nem lehet feljutni a létra tetejére.

Lépésszám: minden i-re y_i kiszámolása konstans sok lépés valamilyen fix konstansra. Így összesen $c \cdot f$ a lépésszám

Hányféleképpen lehet eljutni a legfelső fokra?

Ha nemcsak az elérhetőséget akarjuk meghatározni, hanem a lehetséges utak számát is, akkor a következő módosított formulát használjuk y_i-kre, amelynek a jelentése most, hogy hányféleképp lehet feljutni az i-edik lépcsőre (használjuk az előző részben bevezetett x_i-ket):

$$y_i = \begin{cases} 1, & \text{ha } i = 1 \text{ és } x_1 = 1 \\ y_1 + 1, & \text{ha } i = 2 \text{ és } x_2 = 1 \\ y_2 + y_1 + 1, & \text{ha } i = 3 \text{ és } x_3 = 1 \\ y_{i-1} + y_{i-2} + y_{i-3}, & \text{ha } x_i = 1 \text{ és } i > 3 \\ 0, & \text{ha } x_i = 0. \end{cases}$$

`A képlet értelmezése`:

- Ha az i-edik fok elérhető, akkor az oda vezető utak száma az előző három fokhoz vezető utak számának összege ($i > 3$).
- Ha a fok törött, akkor oda egyetlen úton sem lehet eljutni, tehát $y_i = 0$.
- Az első 3 fokot ki kell számolni.

A megoldás végén y_f értéke megadja, hány különböző módon lehet elérni az f-edik fokot.

Lépésszám: minden i-re y_i kiszámolása konstans sok lépés valamilyen fix konstansra. Így összesen $c \cdot f$ a lépésszám.

7. Adott n chip, melyek képesek egymás tesztelésére a következő módon: ha összekapcsolunk két chipet, mindkét chip nyilatkozik a másikról, hogy hibásnak találta-e. Egy hibátlan chip korrektül felismeri, hogy a másik hibás-e, míg egy hibás chip akármilyen választ adhat. Tegyük fel, hogy a chipek több, mint a fele korrekt. Adjunk algoritmust, mely n-nél kevesebb fenti tesztet használva kikeres egy jó chipet.

Megoldás: **A chip-tesztelés viselkedése**

Két chip egymás tesztelésekor a következő eredményeket kaphatjuk:

1. chip	**2. chip**	**1. válasz**	**2. válasz**
hibátlan	hibátlan	hibátlan	hibátlan
hibátlan	hibás	hibás	*bármi*
hibás	hibátlan	*bármi*	hibás
hibás	hibás	*bármi*	*bármi*

Az algoritmus működése Készítsünk elő három halmazt:

- N (nem tesztelt chipek halmaza),
- T (tesztelt chipek halmaza),
- S (kidobott chipek halmaza).

Kezdetben minden chip az N halmazban van. Az algoritmus a következő lépéseket hajtja végre:

(a) Vegyünk ki két chipet az N halmazból, és teszteljük őket egymással.

(b) Ha mindkét chip azt mondja a másikról, hogy hibátlan, akkor vagy mindkettő hibátlan, vagy mindkettő hibás. Ekkor mindkettőt áthelyezzük a T halmazba.

(c) Ha valamelyik chip a másikat hibásnak mondja, akkor legalább az egyik hibás, ezért mindkettőt kidobjuk (S halmazba kerülnek).

(d) Ha az N és T halmaz még nem üres, akkor mindig egy már tesztelt chipet (T-beli) teszteljünk egy új chip-pel (N-beli).

(e) Ha a két chip egymásról hibátlan visszajelzést ad, akkor a T halmazba kerülnek, különben az S halmazba.

(f) Ha a T halmaz kiürül, és még maradt az N-ben legalább két chip, akkor két új chippel kezdjük elölről a folyamatot.

Az algoritmus végállapota

Az algoritmus végén két eset lehetséges:

(a) Ha az N halmazban pontosan egy chip maradt és a T halmaz üres, akkor az N-ben lévő chip biztosan hibátlan.

(b) Ha az N halmaz üres és a T halmaz nem üres, akkor a T halmazban minden chip ugyanolyan típusú. Mivel tudjuk, hogy a hibátlan chipek száma több mint a hibás chipeké, és az algoritmus során a T halmazban mindig azonos típusú chipek vannak, ezért a T halmaz csupa hibátlan chipből áll.

Az algoritmus helyessége

A tesztelés során végig érvényesülnek az alábbi invariánsok:

(1) Az $N \cup T$ halmazban mindig több hibátlan chip van, mint hibás.

(2) A T halmazban mindig azonos típusú chipek vannak (vagy mind hibátlan, vagy mind hibás, vagy üres).

A fenti tulajdonságok garantálják, hogy a végén a $T \cup N$ halmazban maradt chipek közül biztosan találunk egy hibátlant.

Az algoritmus lépésszáma

Minden lépésben vagy az N-ből veszünk ki két chipet, vagy az egyik chippet áthelyezzük T-be, vagy kidobjuk. Ezért legfeljebb $n - 1$ tesztet végzünk.

8. Egy tanteremben fel van szerelve egy $n \times n$-es tábla, melyen n^2 villanykörte helyezkedik el. A tábla minden egyes sorához illetve oszlopához tartozik egy-egy nyomógomb, mellyel a megfelelő sorban (oszlopban) található n darab villanykörte állapotát egyszerre lehet átváltoztatni az ellenkezőjére. *(Egy gombnyomásra az adott sorban illetve oszlopban égő körték elalszanak, az alvók pedig kigyulladnak.)* A szünet kezdetekor az összes körte leoltott állapotban van. Szünetben a nebulók össze-vissza nyomogatják a gombokat. Hány kapcsolással tudja a tanár visszaállítani az eredeti állapotot? *(A gombok egyállapotúak, azaz nem látszik rajtuk, hogy megnyomták-e őket vagy sem.)*

   Megoldás: Mivel egy gombnyomás kétszer alkalmazva semmit nem jelent, így a végső állapotot egy adott gombcsalád (a páratlanszor megnyomott gombok halmaza) határozza meg. Ugyanakkor vegyük észre, hogy ha a $2n$ gomb közül az eredetileg (páratlanszor) megnyomott k gomb helyett a maradék $2n-k$ gombot nyomnánk meg, az az aktuális állapotot "invertálná" – azaz visszaállítaná az eredeti állapotot, azaz az eredmény az lenne, hogy mindegyik lámpa 0-ra vált. Így a visszaállításhoz választhatjuk a kisebbik halmazt, azaz:

$$\min\{k, 2n-k\} \leq n.$$

   Ez azt jelenti, hogy mindig találunk olyan gombkombinációt, amellyel legfeljebb n lépésben visszaállítható az eredeti állapot.

   Viszont a legrosszabb esetben – például ha minden lámpa ég – valóban n gombnyomás szükséges, hiszen egy gombnyomással legfeljebb n lámpa állapotát tudjuk megváltoztatni. Így

n gombnyomás szükséges és elegendő a visszaállításhoz.

9. Egy $2 \times n$-es sakktábla mezőin n piros és $n-1$ kék négyzetet helyezünk el. Ezeket olyan módon akarjuk átrendezni, hogy a felső sorban piros, az alsóban kék négyzetek legyenek, s a bal alsó sarok maradjon üres. Ehhez egy-egy lépés során az üres mezőre tolhatjuk valamelyik szomszédját. Bizonyítsuk be, hogy

   (a) van olyan algoritmus, ami ezt megoldja $c \cdot n^2$ lépéssel, ahol c valamilyen fix konstans

   (b) létezik olyan d konstans, hogy minden algoritmus, ami ezt megoldja, szerencsétlen inputon használ legalább $d \cdot n^2$ lépést.

   Megoldás: **A rendezés végrehajtása adott lépésszámmal**

   A cél, hogy minden kék az alsó sorba, minden piros a felső sorba kerüljön, az üres hely pedig a bal alsó sarokba.

   (a) **Első fázis: Az üres helyet le kell juttatni az alsó sorba.**
      i. Ha az üres hely már az alsó sorban van, nincs teendő.
      ii. Ha az üres hely a felső sorban van, akkor egy lefelé tolással (legfeljebb 1 lépés) az alsó sorba mozgathatjuk.

   (b) **Második fázis: Kékek elhelyezése az alsó sorba.** Ha már minden kék alsó sorban van (az első fázis után), akkor minden piros is automatikusan fent lesz. De kezdetben lehetnek kékek a felső sorban.
      i. Ha egy i-edik oszlopban kék van fent, akkor biztosan van alul egy piros négyzet.
      ii. Mozgassuk az üres helyet az alsó sorban az i-edik oszlopba. Ehhez legfeljebb $n-1$ lépés kell.
      iii. Toljuk le a felső sorban lévő kéket az üres helyre (1 lépés).
      iv. Az alsó sorban biztosan marad egy piros négyzet, amit az üres helyre fel lehet tolni. Ehhez előbb az üres helyet a megfelelő oszlopba kell mozgatni ($n-1$ lépés), majd fel kell tolni a pirosat (1 lépés).

      Összesen tehát $2n$ lépés alatt egy kéket levittünk az alsó sorba.

   (c) **Harmadik fázis: Ismételjük a második fázist, amíg az összes kék lent nincs.** Mivel legfeljebb $n-1$ kék lehet kezdetben fent, a harmadik fázis legfeljebb $n-1$-szer ismétlődik.

   (d) **Negyedik fázis: Az üres helyet balra mozgatjuk az alsó sorban.** A rendezés végén az üres hely biztosan az alsó sorban marad, de bárhol lehet. Ahhoz, hogy a bal alsó sarokba kerüljön, legfeljebb $n-1$ lépést kell megtennie.

   (e) **Végső lépésszám:** Az algoritmus tehát legfeljebb:

$$1 + 2n(n-1) + (n-1) = 2n^2 - n + 1$$

   lépést használ. Mivel egy konstans szorzót keresünk, így elmondható, hogy 2 jó lesz, hiszen:

$$2n^2 - n + 1 \leq 2 \cdot n^2.$$

Alsó korlát bizonyítása

Most azt kell megmutatni, hogy léteznek olyan kezdőállapotok, amelyeket bármilyen algoritmus csak $d \cdot n^2$ lépéssel tud rendezni valamilyen fix d-re.

Tekintsünk egy olyan kezdeti konfigurációt, ahol (az egyszerűség kedvéért tekintsünk el az egészrészektől):

(a) Az összes piros négyzet az első $n/2$ oszlopban van.

(b) A felső sor utolsó $n/4$ cellájába kell pirosakat mozgatni.

(c) Ezek a négyszetek legalább $n/4$ távolságra vannak bármely pirostól.

A piros négyzetek egyenkénti mozgatása során minden egyes piros négyzet mozgatása legalább $n/4$ lépésbe kerül. Ha $n/4$ piros négyzetet kell áthelyezni, akkor az összes mozgatási lépés száma legalább:

$$(n/4) \times (n/4) = \frac{n^2}{16}.$$

Ez azt jelenti, hogy d választható $\frac{1}{16}$-nak.