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 csütörtökön van algel előadás.

   (b) Minden olyan hallgató, aki jár algel gyakorlatra, átmegy a vizsgán.

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

   (d) Van olyan hallgató, aki sokat tanul, de nem megy át a vizsgán.

   (e) Mindenki, aki átmegy a vizsgán, sokat tanult.

   Megoldás:

   (a) Van olyan csütörtök amikor nincs algel előadás.

   (b) Van olyan hallgató, aki jár algel gyakorlatra, de elégtelenre vizsgázik.

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

   (d) Minden olyan hallgató, aki sokat tanul, átmegy a vizsgán.

   (e) Van olyan hallgató, aki keveset tanult, de átmegy a vizsgán.

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 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: $T(1) = 2;\ T(2) = 3 \cdot 2 + 1;\ T(3) = 3^2 \cdot 2 + 3 + 1;\ T(4) = 3^3 \cdot 2 + 3^2 + 3 + 1;\ \ldots$
   $T(n) = 3^{n-1} + (3^{n-1} + 3^{n-2} + \ldots + 1) = 3^{n-1} + \frac{3^n-1}{3-1} = \frac{2 \cdot 3^{n-1} + 3 \cdot 3^{n-1} - 1}{2} = \frac{5}{2}3^{n-1} - \frac{1}{2}$.

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: $L(n) \leq 3 + \frac{1}{2}(4 + 5 + \cdots + n) = 3 + \frac{1}{2}\frac{(n+4)(n-3)}{2} = \frac{n^2+n}{4} \leq \frac{n^2}{2} \leq n^2$.

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:

   (a) $100k$

   (b) $k\sqrt[3]{100} = 4.64k$

(c) $k + \log_2 100 = k + 6.64$

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 és mindig felfelé lépünk. 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: x_i legyen 1 ha jó a fok és 0 ha rossz. $y_i = x_i$ ha $i \leq 3$, $y_i = \max\{y_{i-1}, y_{i-2}, y_{i-3}\}$ ha $x_i = 1$, különben 0. Fel tudunk jutni, ha $y_f = 1$.

Hányféleképp: Hasonlóan, minden y_i legyen 0 ha $x_i = 0$. Különben $y_1 = x_1$, $y_2 = 1 + y_1, y_3 = 1 + y_1 + y_2$ és $y_i = y_{i-1} + y_{i-2} + y_{i-3}$ ha $i \geq 3$.

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:

1. chip	2. chip	1. kimenet	2. kimenet
jó	jó	jó	jó
jó	rossz	rossz	jó/rossz
rossz	jó	jó/rossz	rossz
rossz	rossz	jó/rossz	jó/rossz

Készítsünk elő 3 dobozt: T=tesztelt, N=nem tesztelt, S=szemét. Kezdetben minden chip az N dobozban van. Vegyünk ki kettőt belőle. Ha egy pár jó-jót mond, akkor egyformák, vagy jó-jó, vagy rossz-rossz. Ha a pár jó-jót mond, akkor tegyük mindkettőt a T dobozba. Ha nem jó-jót mondanak, akkor legalább az egyik rossz, ilyenkor dobjuk ki mindkettőt az S dobozba. Így még mindig igaz, hogy az N és T dobozban lévők összességének több mint a fele jó. Ha az N doboz még nem üres, akkor mindig a T dobozból egyet párosítunk egy N dobozban levővel. Ha jó-jó, akkor a T dobozba tesszük mindkettőt, különben az S dobozba. Ha kiürül a T doboz, akkor két N dobozbelit veszünk megint.

Tehát minden lépésben kiveszünk az N dobozból egyet vagy kettőt, ezért az N doboz legfeljebb $n - 1$ összehasonlítás után kiürül. Ekkor véget ér az algoritmus. Az nem lehetséges, hogy ekkor a T doboz is üres, hiszen akkor az utolsó összehasonlítás előtt már nem teljesült volna, hogy az N és T dobozban lévők összességének több mint a fele jó. Tehát a végén T nem üres. Mivel tudjuk, hogy csupa egyformák vannak benne és N üres, de megmaradtak több mint fele jó, ezért a T dobozban minden chip jó. Már láttuk, hogy ez legfeljebb $n - 1$ összehasonlítás.

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: Világos, hogy egyik gombot sem érdemes egynél többször megnyomni. Tegyük fel, hogy megnyomtunk néhány gombot és minden lámpa elaludt. Mi történik ha most az összes gombot még egyszer megnyomjuk? Minden lámpának a sorához és oszlopához tartozó gombot is megnyomtuk, ezért a lámpa le lesz kapcsolva. Viszont ezt ez eredményt úgy is elérhetjük, hogy pont azokat nyomjuk meg, amiket eredetileg nem. Ha először k gombot nyomtunk, akkor most $2n - k$ darabot fogunk. Viszont $\min(k; 2n - k) \leq n$, válasszuk tehát azt a gombnyomást, amikor kevesebbet kell nyomni. Tehát legfeljebb n gombnyomás kell.

Másrészt, ha minden lámpa ég, akkor egy gombnyomással nem lehet megváltoztatni az átlóban két különböző lámpa állapotát. Vagyis az átló minden lámpájához kell legalább egy gombnyomás, azaz n gombnyomásra szükség lehet legrosszabb esetben.

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) Először legfeljebb 1 lépésben elérhető, hogy az üres hely alul legyen.

   Ha ezután minden kék az alsó sorban van, akkor minden piros a felső sorban van, viszont alul az üres hely bárhol lehet. Könnyen látszik, hogy ebből a helyzetből legfeljebb $n-1$ lépésből elérhető a megfelelő elrendezés.

   Ha van a felső sorban kék, pl. az i-edik oszlopban, akkor mindenképp van az alsó sorban piros. Az alsó sor négyzeteinek mozgatásával legfeljebb $n-1$ lépésben elérhető, hogy az üres hely az i-edik oszlop alsó sorában legyen. 1 lépésben toljuk le a kéket, ezzel csökken a fent levő kékek száma. Lent még mindig kell lenni pirosnak pl. a j-edik oszlopban, ahol most $j \neq i$, hiszen ott egy kék van. Megint csak legfeljebb $n-1$ lépésben az üres helyet mozgassuk a j-edik oszlopba és toljuk fel a pirosat. Tehát legfeljebb $2n$ lépésben eggyel kevesebb kék lesz fent (és eggyel kevesebb piros lent). Ha még mindig van kék fent, akkor ismételjük meg ugyanezt.

   Mivel legfeljebb $n-1$ kék lehet fent kezdetben, így $1+2n(n-1)$ lépésben elérhető, hogy minden kék lent legyen. Mivel akkor az üres hely lent lesz, ezért biztos, hogy minden piros fent van már. Már csak az üres helyet kell a bal alsó sarokba mozgatni legfeljebb $n-1$ lépésben. Ez az algoritmus tehát legfeljebb $1 + 2n^2 - 2n + n - 1 = 2n^2 - n \leq 2n^2$ lépést használ.

   (b) Ha a pirosak mind a bal oldalon vannak, az első $n/2$ oszlopban, akkor a felső sor utolsó $n/4$ helyére pirosakat kell mozgatni valahogy. De ezek a pirosak legalább $n/4$ távolságra vannak. Ez már legalább $n^2/16$ lépés.