A Palindromi

Jos merkkijono on valmiiksi palindromi, siitä voi myös poistaa yhden merkin keskeltä, minkä jälkeen se on edelleen palindromi.

Muussa tapauksessa vaihtoehtoja on kaksi: yksi merkki poistetaan merkkijonon keskikohdan vasemmalta puolelta tai oikealta puolelta. Vain yhden merkin saa poistaa, eli ensimmäinen merkki, joka rikkoo palindromiehdon, täytyy poistaa.

Yksinkertaisesta ideasta huolimatta tämä on yllättävän vaikeaa koodata toimivasti.

B Viivapeli

Uolevi voittaa, jos tarkalleen toinen luvuista n ja m on pariton.

Voittava pelaaja voi käyttää aina matkimistaktiikkaa, jossa ideana on tehdä aina vastustajan siirron kaltainen siirto symmetrisesti toiselle puolelle pelilautaa. Sitten kun jossakin 1x1-neliössä on kolme viivaa, matkija voittaa pelin täydentämällä neljännen viivan. Jos n ja m ovat molemmat parillisia tai molemmat parillisia, Maija pystyy käyttämään matkimistaktiikkaa. Jos tarkalleen toinen on pariton, Uolevi pystyy laittamaan ensimmäisen viivan ruudukon keskelle ja käyttämään sen jälkeen matkimistaktiikkaa.

C Järjestys

Ideana on laskea eri merkkien esiintymiskerrat. Tämän jälkeen yleisimmin esiintyvä merkki ratkaisee, onko järjestyksen muodostaminen mahdollista.

Oletetaan, että merkkijonon pituus on n. Jos n on parillinen, järjestäminen on mahdollista, jos yleisin merkki esiintyy $n/2$ kertaa tai vähemmän. Jos n on pariton, järjestäminen on mahdollista, jos yleisin merkki esiintyy $(n+1)/2$ kertaa tai vähemmän.

D Tukikohta

Tämä on suoraviivainen tehtävä, jossa riittää käydä läpi kaikki mahdolliset kohdat sijoittaa tukikohta. Ainoa vaikeus on, että tukikohdan reunoilla saattaa olla tyhjää. Yksi ratkaisu on poistaa tyhjät reunoilta ennen sijoittamista. Toinen ratkaisu on käydä läpi kaikki kohdat, joissa ainakin tukikohdan reunaruutu osuu metsän alueelle ja jättää huomiotta metsän ulkopuoliset tyhjät ruudut.

E Säästöpossu

Tehtävään on kaksi lähestymistapaa: kaiken koodaaminen itse tai kirjaston käyttäminen. Kaiken koodaaminen itse ei loppujen lopuksi ole vaikeaa (ks. malliratkaisu), kunhan muistaa kuukausien päivien määrät ja karkausvuoden määritelmän. Toinen vaihtoehto on käyttää esim. Javan luokkia päivämäärien käsittelyyn. Niissä voi kuitenkin tulla vastaan yllätyksiä.

F Neliö

Seuraava ratkaisu toimii ajassa $O(n^3 \log n)$: Etsitään suurinta neliön kokoa binäärihaulla. Binäärihakua voi käyttää, koska aina jos koko on suurinta kokoa pienempi, ratkaisu on myös varmasti olemassa. Kun neliön koko on kiinnitetty, käydään läpi kaikki rivit, joilla neliö voi esiintyä. Tietyn neliön koon tarkistaminen onnistuu ajassa $O(n^3)$.

Tehtävä on myös mahdollista ratkaista ajassa $O(n^3)$. Keksitkö miten?