582359 Algoritmit ongelmanratkaisussa (kevät 2013)
Viikon 10 tehtävät (12.4.)

1. Toteuta $O(n^2)$-aikainen brute-force-algoritmi, joka etsii annetusta pistejoukosta pienimmän etäisyyden kahden pisteen välillä. Testaa algoritmin tehokkuutta tiedostolla `pisteet.txt`, joka sisältää 100000 pistettä. Tiedoston jokaisella rivillä on yhden pisteen x- ja y-koordinaatit.

   Ratkaisu:

   Tiedostossa `HidasEtaisyys.java` on brute-force-algoritmin toteutus. Algoritmi käy läpi kaikki pisteparit ja valitsee niistä pienimmän etäisyyden. Sen suoritusaika testikoneella on reilut 5 minuuttia.

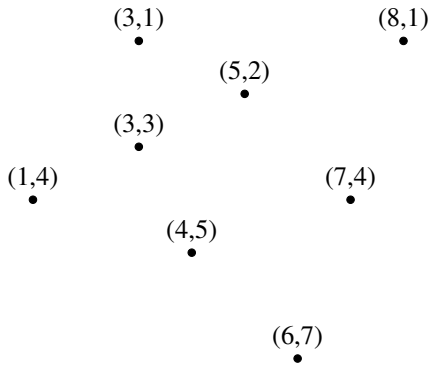
   Tiedostossa `HidasEtaisyys2.java` on hieman tehostettu brute-force-algoritmi. Se järjestää ensin pisteet x-koordinaatin mukaan, minkä ansiosta toisen silmukan voi katkaista, jos x-koordinaattien erotus on liian suuri. Tämän muutoksen ansiosta suoritusaika putoaa alle sekuntiin testikoneella.

2. Toteuta samaan tehtävään tehokas $O(n \log n)$-aikainen algoritmi. Kuinka nopeasti se käsittelee tiedoston `pisteet.txt`?

   Ratkaisu:

   Tiedostossa `NopeaEtaisyys.java` on pyyhkäisyviivaan perustuva algoritmi, joka vastaa kurssisivun linkin kuvausta. Algoritmin suoritusaika on alle sekunti. Kiinnostava havainto on, että tällä testiaineistolla tehostettu $O(n^2)$-aikainen brute-force-algoritmi on yhtä nopea kuin $O(n \log n)$-algoritmi.

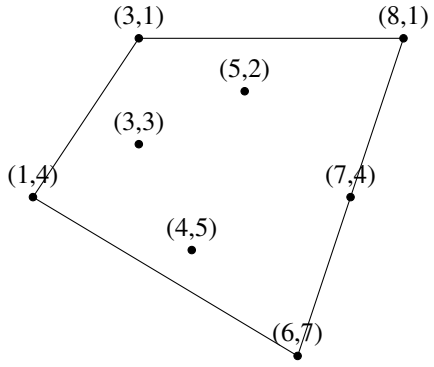
3. Tutustu haluamaasi algoritmiin, joka muodostaa konveksin peitteen annetulle pistejoukolle. Miten algoritmi käsittelee seuraavan pistejoukon?



   Ratkaisu:

   Yksinkertainen tapa muodostaa konveksi peite on lahjanpaketointialgoritmi. Se lähtee liikkeelle jostain peitteeseen kuuluvasta pisteestä ja valitsee aina seuraavaksi pisteeksi sellaisen, että kaikki muut pisteet ovat syntyvän suoran oikealla puolella. Sopiva aloituspiste on piste, jonka x-koordinaatti on pienin. Jos tällaisia pisteitä on useita, valitaan niistä piste, jonka y-koordinaatti on pienin. Jos uuden pisteen valinnassa vaihtoehtoja on useita, valitaan niistä läheisin piste.

   Esimerkkitilanteessa algoritmi tuottaa seuraavan konveksin peitteen:



4. Syötteenä on joukko suorakulmioita, joiden sivut ovat vaaka- ja pystysuuntaisia, ja tehtävänä on laskea suorakulmioiden peittämän alueen kokonaispinta-ala. Toteuta tähän tehtävään algoritmi ja testaa sitä tiedostolla `pinta-ala.txt`, joka sisältää 1000 suorakulmiota. Tiedoston jokaisella rivillä on yhden suorakulmion vasemman yläkulman ja oikean alakulman koordinaatit.

   Ratkaisu:

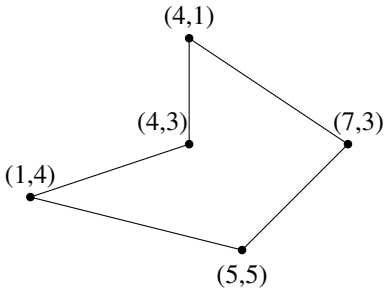
   Tiedostossa `PintaAla.java` on pyyhkäisyviivaan perustuva algoritmi. Ideana on käydä läpi kaikki x-koordinaatit, joissa jokin suorakulmio alkaa tai päättyy. Algoritmi laskee kokonaispinta-alan summana x-koordinaattien välisistä osista. Niissä suorakulmioiden y-suunnassa kattama osuus säilyy vakiona. Algoritmi käsittelee testiaineiston alle sekunnissa.

5. Monikulmio on *yksinkertainen*, jos mitkään sen sivut eivät leikkaa toisiaan. Yksinkertaisen monikulmion pinta-alan voi laskea seuraavalla kaavalla:

$$A = \frac{\sum_{i=1}^{i=n}(x_i y_{i+1} - x_{i+1} y_i)}{2}$$

   Kaavassa n on monikulmion kärkipisteiden määrä ja kärkipiste k on kohdassa (x_k, y_k). Kaava olettaa, että kärkipisteet kierretään myötäpäivään monikulmion reunaa pitkin ja $(x_{n+1}, y_{n+1}) = (x_1, y_1)$.

   (a) Laske kaavan avulla seuraavan monikulmion pinta-ala:



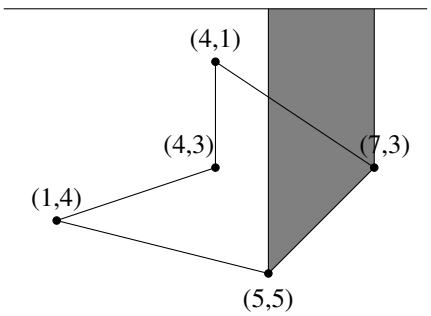
   (b) Hahmottele todistus sille, miksi kaava toimii.

   Ratkaisu:

   (a) Lasketaan pinta-ala ylimmästä pisteestä aloittaen ja myötäpäivään kulkien:

$$A = \frac{(4 \cdot 3 - 7 \cdot 1) + (7 \cdot 5 - 5 \cdot 3) + (5 \cdot 4 - 1 \cdot 5) + (1 \cdot 3 - 4 \cdot 4) + (4 \cdot 1 - 4 \cdot 3)}{2} = 19/2$$

   (b) Lisätään kuvaan vaakasuuntainen suora, jonka y-koordinaatti on 0. Nyt voidaan ajatella, että monikulmion pinta-ala saadaan laskemalla yhteen ja vähentämällä puolisuunnikkaiden pinta-aloja. Jokaisen puolisuunnikkaan yksi sivu on monikulmion sivu ja vastakkainen sivu on vaakasuuntaisella suoralla. Esimerkiksi monikulmion sivua pisteestä $(7, 3)$ pisteeseen $(5, 5)$ vastaa seuraava puolisuunnikas:



Kun liikutaan pisteestä k pisteeseen $k+1$, puolisuunnikkaan pinta-ala on seuraava:

$$\frac{(x_k - x_{k+1})(y_k + y_{k+1})}{2} = \frac{x_k y_k + x_k y_{k+1} - x_{k+1} y_k - x_{k+1} y_{k+1}}{2}$$

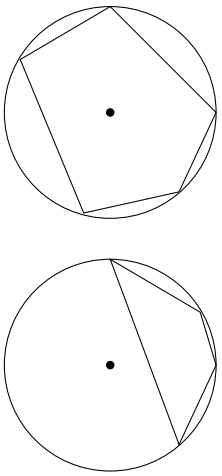
Liikuttaessa vasemmalta oikealle pinta-ala on negatiivinen ja liikuttaessa oikealta vasemmalle pinta-ala on positiivinen. Kun kaikki pinta-alat lasketaan yhteen, saadaan tehtävänannossa oleva kaava, koska termit muotoa $x_k y_k$ ja $x_{k+1} y_{k+1}$ kumoavat toisensa. Tämä on monikulmion pinta-ala, koska positiiviset puolisuunnikkaat lisäävät pala kerrallaan monikulmion osat ja negatiiviset puolisuunnikkaat poistavat monikulmion ulkopuolelle jäävät osat.

6. Esitä algoritmi, joka muodostaa yksinkertaisen monikulmion, kun sille annetaan halutut sivujen pituudet. Algoritmin tulee ilmoittaa monikulmion jokaisen kärkipisteen koordinaatit. Algoritmin täytyy myös huomata, jos monikulmiota ei voi muodostaa.

   Ratkaisu:

   Jos jokin sivu on pidempi kuin kaikki muut sivut yhteensä, monikulmiota ei voi muodostaa. Muuten monikulmion voi muodostaa. Seuraavassa ideassa monikulmio muodostetaan ympyrän sisään niin, että sivut sijoitetaan suuruusjärjestyksessä ympyrän kehälle.

   Tilanteesta riippuen ympyrän keskipiste tulee monikulmion sisään tai ulkopuolelle. Seuraavassa on esimerkit molemmista tilanteista:



   Oletetaan ensin, että ympyrän säde on tiedossa. Tällöin monikulmion kärkipisteet pystyy laskemaan trigonometrian avulla. Mistä sitten tietää, mikä on oikea valinta ympyrän säteeksi? Tämän voi selvittää binäärihaun avulla. Ylemmässä tilanteessa mitä suurempi ympyrä on, sitä suurempi on monikulmion piiri. Alemmassa tilanteessa mitä suurempi ympyrä on, sitä pienempi on monikulmion piiri.