582359 Algoritmit ongelmanratkaisussa (kevät 2013)
Viikon 2 tehtävät (25.1.)

1. Tehtävänä on laskea, monellako tavalla kaksi ratsua voidaan sijoittaa $n \times n$ -shakkilaudalle niin, että ne eivät uhkaa toisiaan. Esimerkiksi tapauksissa 3×3 ja 10×10 vastaukset ovat 28 ja 4662.

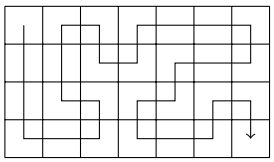
   (a) Ohjelmoi tehtävään brute-force-algoritmi. Mikä on sen aikavaativuus? Kuinka suuria tapauksia pystyt laskemaan algoritmilla?

   (b) Ohjelmoi tehtävään $O(1)$-aikainen algoritmi eli käytännössä mieti tilannetta matemaattisesti ja kehitä kaava, jolla vastauksen voi laskea suoraan. *Vihje*: laske ensin, moneenko suuntaan ratsu voi liikkua kussakin ruudukon ruudussa.

2. Muutetaan edellistä tehtävää niin, että ratsuja onkin kolme. Nyt mikään ratsu ei saa uhata toista asetelmassa. Ohjelmoi tehtävään brute-force-algoritmi, jolla voit laskea pienten tapausten vastauksia. Muodosta suoraan niiden perusteella polynomikaava esimerkiksi materiaalin menetelmällä.

   Voit halutessasi myös perustella matemaattisesti, miksi kyseinen polynomikaava antaa oikean vastauksen. Tämä on kuitenkin selvästi vaikeampaa kuin edellisessä tehtävässä.

3. Tehtävänä on laskea, montako erilaista reittiä on ruudukon vasemmasta ylänurkasta oikeaan alanurkkaan. Reitin aikana täytyy käydä tasan kerran jokaisessa ruudussa, ja ruudukossa saa liikkua vasemmalle, oikealle, ylöspäin ja alaspäin. Tässä on yksi mahdollinen reitti 4×7 -ruudukossa:

   Esimerkiksi tapauksessa 4×7 reittejä on yhteensä 111.

   Ohjelmoi tehtävään brute-force-algoritmi. Kuinka suuria tapauksia pystyt laskemaan algoritmilla?

4. (a) Osoita, että jos sekä ruudukon korkeus että leveys on parillinen, niin edellisen tehtävän reittejä ei ole olemassa, ja muuten niitä on aina olemassa.

   (b) Tehosta algoritmiasi niin, että se pystyy sopivissa tilanteissa huomaamaan kesken reitin muodostuksen, ettei reittiä ole mahdollista muodostaa loppuun. Tällöin nykyisen hakuhaaran voi katkaista. Kuinka suuria tapauksia pystyt nyt laskemaan? *Tavoite*: tapauksen 7×7 tulisi ratketa hetkessä.

5. *Heapin algoritmi* on erikoinen tapa käydä läpi joukon permutaatiot. Sen Java-toteutus on kurssisivulla tiedostossa `Heap.java`. Perustele, miksi Heapin algoritmi toimii.

6. Lähtökohtana on lukujen $1 \ldots n$ permutaatio ja tehtävänä on järjestää luvut pienimmästä suurimpaan kääntöoperaatioiden avulla. Jokainen kääntöoperaatio kääntää valitun välin permutaatiosta takaperin.

   Esimerkiksi jos permutaatio on $(4, 2, 3, 1)$, lyhin mahdollinen kääntösarja on seuraava:

   $$(\underline{4, 2, 3, 1}) \rightarrow (1, \underline{3, 2}, 4) \rightarrow (1, 2, 3, 4)$$

   (a) Osoita, että minkä tahansa permutaation saa järjestykseen kääntöoperaatioilla.

   (b) Etsi lyhin kääntösarja seuraavalle permutaatiolle:

   $$(7, 13, 9, 4, 1, 14, 5, 3, 12, 10, 15, 2, 8, 6, 11)$$

   Vihje: IDA*-algoritmi on yksi hyvä lähestymistapa.