582359 Algoritmit ongelmanratkaisussa (kevät 2013)
Viikon 4 ratkaisuja

1. DNA-ketju muodostuu merkeistä A, C, G ja T. Yksi tapa tarkastella DNA-ketjua on jakaa se samaa merkkiä toistaviin osiin. Esimerkiksi ketju AGGCTTTAAG muodostuu osista A, GG, C, TTT, AA ja G. Ketjun *pisin toisto* on pisimmän toistavan osan pituus. Esimerkiksi ketjussa AGGCTTTAAG pisin toisto on 3.

   Tee tehokkaat algoritmit seuraaviin ongelmiin:

   (a) Montako n-merkin DNA-ketjua on olemassa, joissa pisin toisto on 1?

   (b) Montako n-merkin DNA-ketjua on olemassa, joissa pisin toisto on yli 1?

   Kun $n = 10$, niin (a)- ja (b)-kohtien tulokset ovat 78732 ja 969844. Laske tulokset, kun $n = 30$.

   Ratkaisu:

   (a) Ketjun ensimmäisen merkin valintaan on 4 vaihtoehtoa, minkä jälkeen jokaisen seuraavan merkin valintaan on 3 vaihtoehtoa. Ketjuja on siis yhteensä $4 \cdot 3^{n-1}$. Tapauksessa $n = 30$ ketjuja on yhteensä $4 \cdot 3^{29} = 274521509459532$.

   (b) Tulos saadaan vähentämällä kaikkien ketjujen määrästä (a)-kohdan tulos. Kaavaksi tulee $4^n - 4 \cdot 3^{n-1}$. Tapauksessa $n = 30$ ketjuja on yhteensä $4^{30} - 4 \cdot 3^{29} = 1152646983097387444$.

2. Tee tehokas algoritmi, joka laskee, monessako n-merkin DNA-ketjussa pisin toisto on korkeintaan k.

   Esimerkiksi kun $n = 10$ ja $k = 3$, niin tulos on 959472. Laske tulos, kun $n = 30$ ja $k = 5$.

   Ratkaisu:

   Tarkastellaan ensin tapausta, jossa $n = 5$ ja $k = 3$. Haaraudutaan sen mukaan, mikä on ketjun viimeinen saman merkin osa. Jos viimeinen merkki on A, vaihtoehdot ovat ????A, ???AA, ??AAA. Vastaavasti merkeillä C, G ja T tulee kullakin 3 vaihtoehtoa lisää.

   Tämän idean yleistämällä tehtävän ratkaisuun saa seuraavan rekursiivisen funktion:

$$T(n) = \begin{cases} 0 & \text{jos } n < 0 \\ 1 & \text{jos } n = 0 \\ \sum_{i=1}^{k} 3T(n-i) + 1 & \text{jos } n \leq k \\ \sum_{i=1}^{k} 3T(n-i) & \text{jos } n > k \end{cases}$$

   Kaavassa kerroin 3 tulee siitä, että viimeisen osion merkki voi olla mikä tahansa muu kuin aiemman osion merkki. Tapauksessa $n \leq k$ lisäys $+1$ tulee siitä, että kun viimeinen osio kattaa koko merkkijonon, niin mahdollisuuksia on 4 eikä 3. Ratkaisun toteutus on tiedostossa `Ketjut.java`.

3. Taulukossa on n positiivista kokonaislukua. Tehtävänä on erottaa taulukon luvuista mahdollisimman pitkä ketju, jossa luvut ovat samassa järjestyksessä kuin taulukossa ja jokainen luku on edellistä suurempi. Esimerkiksi taulukossa [3, 2, 8, 5, 3, 4, 1, 9] pisin ketju on [2, 3, 4, 9] eli siihen kuuluu 4 lukua.

   Esitä tehokas algoritmi, joka etsii pisimmän ketjun taulukosta. Jos saat aikavaativuuden $O(n^2)$, niin se on riittävän tehokas, mutta myös aikavaativuus $O(n \log n)$ on mahdollinen.

   Ratkaisu:

   Ideana on laskea jokaiselle taulukon luvulle, kuinka pitkä on pisin siihen päättyvä ketju. Esimerkin tapauksessa tulokset ovat seuraavat:

luku	3	2	8	5	3	4	1	9
pisin ketju	1	1	2	2	2	3	1	4

   Tämän saa laskettua tehokkaasti käymällä kunkin luvun kohdalla taulukon alkuosan läpi ja etsimällä sellaisen luvun, joka on käsiteltävää lukua pienempi ja johon päättyy mahdollisimman pitkä ketju. Tätä ketjua jatkamalla syntyy pisin ketju käsiteltävään lukuun asti. Ratkaisun toteutus on tiedostossa `PisinKetju.java` ja sen aikavaativuus on $O(n^2)$.

Ratkaisua on mahdollista tehostaa toimimaan ajassa $O(n \log n)$ ottamalla käyttöön aputaulukon, joka kertoo jokaisesta ketjun pituudesta pienimmän tähän mennessä löydetyn luvun, johon päättyy kyseisen pituinen ketju. Esimerkin tapauksessa taulukosta tulee lopulta seuraava:

ketjun pituus	1	2	3	4	5	6	7	8
pienin luku	1	3	4	9	–	–	–	–

Tämän taulukon sisältö on järjestyksessä pienimmästä suurimpaan, koska pidempi ketju ei voi päättyä pienempään lukuun kuin lyhyempi ketju. Niinpä taulukosta voi etsiä ketjun edellisen luvun binäärihaulla ajassa $O(\log n)$. Tämä tuottaa aikavaativuuden $O(n \log n)$.

4. *Täydellinen tekoäly* voittaa vastustajan aina, kun on olemassa varma strategia voittoon. Toteuta täydellinen tekoäly seuraaviin peleihin:

   (a) Pelin alussa pöydällä on n tikkua. Pelaajat nostavat vuorotellen 1, 2 tai 3 tikkua. Pelin häviää se, joka joutuu nostamaan viimeisen tikun.

   (b) Muutetaan peliä niin, että tikkujen poistomäärät päätetään pelin alussa. Tekoälyn tulee siis sopeutua mihin tahansa poistomääriin. Kohta (a) on tästä erikoistapaus, jossa poistomäärät ovat [1, 2, 3]. Testaa tekoälyä esimerkiksi poistomäärillä [1, 4] ja [1, 3, 7].

   Ratkaisu:

   (a) Muodostetaan taulukko, joka ilmoittaa jokaisesta tikkumäärästä, onko kyseessä voittotila (V) vai häviötila (H). Voittotilassa olevalla pelaajalla on varma strategia voittoon, kun taas häviötilassa oleva pelaaja häviää varmasti, jos toinen pelaa täydellisesti. Ensimmäinen häviötila on 1. Tila on voittotila, jos siitä pääsee häviötilaan poistamalla jokin sallittu määrä tikkuja. Vastaavasti tila on häviötila, jos mikä tahansa poistomäärä johtaa voittotilaan.

tikkuja	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	$\cdots$
tila	H	V	V	V	H	V	V	V	H	V	V	V	H	V	V	$\cdots$

   Huomataan, että tila on häviötila tarkalleen silloin, kun tikkumäärän jakojäännös 4:llä on 1. Täydellinen tekoäly valitsee aina tikkuja niin, että vastustaja joutuu häviötilaan. Jos tekoäly on itse häviötilassa, se voi tehdä minkä tahansa valinnan ja toivoa, että vastustaja tekee virheen.

   (b) Kohdan (a) päättely yleistyy mihin tahansa tilanteeseen, joskaan kaava ei ole aina yhtä yksinkertainen. Yleisen tekoälyn voi toteuttaa laskemalla kohdan (a) kaltaisen taulukon pelin alussa olevaan tikkumäärään asti. Tämän jälkeen riittää pyrkiä poistamaan tikkuja niin, että vastustaja joutuu häviötilaan. Tiedostossa `Tikkupeli.java` on yleisen tekoälyn toteutus.

5. Toteuta täydellinen tekoäly seuraavaan peliin: Taulukossa on n positiivista kokonaislukua, ja pelaajat poistavat vuorotellen yhden luvun taulukon vasemmasta tai oikeasta reunasta. Peli päättyy, kun kaikki luvut on poistettu, ja pelin voittaja on se, jonka poistamien lukujen summa on suurempi.

   Ratkaisu:

   Tavoitteena on, että tekoälyn lukujen summan ja vastustajan lukujen summan erotus olisi mahdollisimman suuri. Lasketaan jokaiselle taulukon välille, mikä on suurin mahdollinen lukujen summan erotus. Tämän saa laskettua rekursiivisesti vähentämällä poistettavasta luvusta jäljelle jäävän taulukon välin suurin erotus. Tarkastellaan esimerkiksi seuraavaa taulukkoa pelissä:

   5 2 7 1 2 8

   Jos pelaaja valitsee luvun 5 vasemmalta, vastustajan taulukko on seuraava:

   2 7 1 2 8

   Vastaavasti jos pelaaja valitsee luvun 8 oikealta, vastustajan taulukko on seuraava:

   5 2 7 1 2

   Ensimmäisessä tapauksessa erotus on $5 - A$, jossa A on taulukon loppuosan erotus. Vastaavasti toisessa tapauksessa erotus on $8 - B$, jossa B on taulukon alkuosan erotus. Jos $5 - A > 8 - B$, niin pelaajan kannattaa valita luku 5, ja muuten luku 8. Tiedostossa `Poistopeli.java` on tehokas ratkaisu, joka perustuu dynaamiseen ohjelmointiin.

6. Tehtävänä on sijoittaa luvut $1 \ldots n^2$ taulukkoon, jossa on $n \times n$ ruutua. Vaatimuksena on, että jokaisella pysty- ja vaakarivillä luvut ovat järjestyksessä pienimmästä suurimpaan. Tässä on yksi tapa muodostaa taulukko, kun $n = 5$:

1	4	7	12	13
2	5	9	15	19
3	8	11	20	22
6	14	16	21	24
10	17	18	23	25

Montako vaatimukset täyttävää taulukkoa on olemassa, kun $n = 10$?

Huom.: Vastaus on suuri luku eikä mahdu 64-bittiseen muuttujaan. Jos toteutat ratkaisun Javalla, niin `BigInteger` on hyvä valinta.

Ratkaisu:

Olennainen havainto on, että jokainen tapa sijoittaa k ensimmäistä lukua ruudukkoon muodostaa yhtenäisen alueen ruudukon vasemmasta yläkulmasta. Esimerkissä luvut 1–9 muodostavat seuraavan alueen:

1	4	7	12	13
2	5	9	15	19
3	8	11	20	22
6	14	16	21	24
10	17	18	23	25

Jokaisessa alueessa alemmalla rivillä ei ole koskaan enempää lukuja kuin ylemmällä rivillä, koska muuten tulevat luvut menisivät väärään järjestykseen. Lisäksi alueessa olevien lukujen järjestys ei vaikuta siihen, miten monta tapaa on täydentää ruudukko loppuun.

Tiedostossa `Young.java` on tehokas ratkaisu, joka perustuu yllä oleviin havaintoihin. Ideana on laskea jokaisen muotoista aluetta vastaavien ruudukoiden määrä vain kerran. Alueiden määrät tallennetaan `HashSet`-rakenteeseen, jossa avaimena on taulukosta muodostettu merkkijonoesitys.