Laskuharjoitus 5

15.02.2001

1. Kirjan tehtävä 4.1: Määritä valintajoukot ja kuvaile peräytymisalgoritmit seuraaville ongelmille:

   (a) Etsi kaikki tavat asettaa n kuningatarta $n \times n$-shakkilaudalle siten, etteivät kuningattaret uhkaa toisiaan.

   (b) Etsi kaikki n:n pituiset itseäänvälttävät(?) kävelyt. (Lähtee tasossa origosta; n askelta, kukin pituudeltaan 1; kunkin askelen suunta on ylös, alas, vasempaan tai oikealle, eikä missään pisteessä käydä kahdesti.)

   (c) Etsi graafin $\mathcal{G}$ solmujen k-väritykset.

2. Määritellään graafin $\mathcal{G}$ kromaattinen luku

   $$\chi\left(\mathcal{G}\right) = \min\left\{k \;:\; \mathcal{G} : \text{lla on solmujen } k - \text{varitys}\right\}$$

   ja klikkiluku

   $$\omega\left(\mathcal{G}\right) = \max\left\{k \;:\; \mathcal{G} : \text{ssa on } k : \text{n kokoinen klikki}\right\}.$$

   Kirjan lemman 4.4 mukaan $\omega\left(\mathcal{G}\right) \leq \chi\left(\mathcal{G}\right)$. Osoita, että epäyhtälö voi olla myös aito: etsi graafi, jolle $\omega\left(\mathcal{G}\right) < \chi\left(\mathcal{G}\right)$.

3. Olkoon annettuna graafi $\mathcal{G}$ ja sen solmuille jokin järjestys. Ahne graafinväritysalgoritmi värittää graafin solmu kerrallaan pienimmästä suurimpaan siten, että kulloinkin väritettävän solmun väriksi valitaan pienin (oletetaan värit järjestetyiksi) väri, joka ei esiinny väritettävän solmun jo väritetyissä naapureissa.

   (a) Esitä ääretön perhe graafeja, joiden solmut on järjestetty siten, että ahne väritysalgoritmi ei tuota optimaalista väritystä. (Väritys on optimaalinen jos värien lukumäärä on $\chi(\mathcal{G})$.)

   (b) Onko kaikille graafeille olemassa sellainen solmujen järjestys, että ahne algoritmi tuottaa optimaalisen värityksen?

4. Kirjan tehtävä 4.13: Latinalainen neliö on $n \times n$-taulukko, jonka jokaisessa rivissä ja sarakkeessa kukin luvuista $1, \dots, n$ esiintyy tasan kerran. Latinalaista neliötä kutsutaan redusoiduksi latinalaiseksi neliöksi, jos sen ensimmäisen rivin ja ensimmäisen sarakkeen alkiot esiintyvät järjestyksessä $1, 2, \dots, n$. Kirjoita ohjelma, joka laskee peräytymishaulla redusoitujen $n \times n$ latinalaisten neliöiden lukumäärän.

Laskuharjoituksen 5 vastaukset

1. Peräytymishaut voidaan rakentaa esim. seuraavasti.

   (a) Tarkastellaan kutakin laudan riviä $1, \ldots, n$ vuorollaan. Valintajoukkona on tällöin niiden tällä rivillä olevien ruutujen joukko, joita ei vielä uhkaa laudalla oleva kuningatar. Kokeillaan kutakin näistä vuorollaan ja siirrytään seuraavalle riville.

   (b) Valintajoukko voisi olla niiden tämänhetkisen sijainnin naapuripisteiden joukko, joissa ei ole vielä käyty. Kokeillaan näistä kutakin vuorollaan ja jatketaan kustakin niistä rekursiivisesti, ellei ole jo otettu n askelta.

   (c) Naiivi algoritmi: Tarkastellaan solmuja järjestyksessä. Valintajoukkona on kunkin solmun kohdalla kaikkien värien joukko $\{1, 2, \ldots, k\}$, josta kuitenkin on poistettu ne värit, joilla on jo väritetty jokin parhaillaan tarkasteltavana olevan solmun naapuri.

   Ainakin ensimmäisessä kohdassa kannattaa itse asiassa järjestää haku niin, että kulloinkin tarkasteltavaksi riviksi valitaan se, jolla sillä hetkellä on vähiten uhkaamattomia ruutuja. Tällöin haun haarautumiskerroin lähellä puun juurta on pieni, jolloin hakusolmujen määrä pienenee, vaikka kaikki vaihtoehdot käydäänkin yhtä kattavasti läpi.

2. Viiden solmun sykli C_5 (viisikulmio) on kromaattiselta luvultaan 3, mutta sen suurimman klikin koko on 2.

3. Olkoon $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ graafi, ja $\mathcal{W} \subseteq \mathcal{V}$. Solmujoukko $\mathcal{W}$ on *riippumaton* (independent), jos minkään sen kahden solmun välillä ei ole kaarta. Graafi $\mathcal{G}$ on *kaksijakoinen* (bipartite) jos sen solmujoukko $\mathcal{V}$ voidaan jakaa kahteen riippumattomaan joukkoon $\mathcal{W}$ ja $\mathcal{V} - \mathcal{W}$.

   Kaksijakoinen graafi voidaan aina värittää kahdella värillä, koska molempiin riippumattomiin joukkoihin riittää yksi väri.

   (a) *Täydellinen* kaksijakoinen (complete bipartite) graafi sisältää kaikki mahdolliset kaaret sen kahden riippumattoman joukon välillä, ts. kaikilla $u \in \mathcal{W}$ ja kaikilla $v \in \mathcal{V} - \mathcal{W}$ pätee $\{u, v\} \in \mathcal{E}$. Konstruoidaan graafiperhe seuraavasti: Otetaan täydellinen kaksijakoinen graafi, jonka molemmissa riippumattomissa joukoissa on n solmua, ja järjestetään solmut siten, että ensimmäinen riippumaton joukko koostuu parittomista solmuista $(1, 3, \ldots, 2n - 1)$, ja toinen parillisista $(2, 4, \ldots, 2n)$. Poistetaan graafista kaaret $\{1, 2\}, \{3, 4\}, \{5, 6\}, \ldots, \{2n - 1, 2n\}$.
   Ahne algoritmi värittää graafin seuraavasti: Solmulle 1 annetaan väri 1. Solmulle 2 annetaan väri 1, koska graafissa ei ole kaarta $\{1, 2\}$. Solmulle 3 annetaan väri 2, koska graafissa on kaari $\{2, 3\}$, ja väriä 1 ei näin ollen voi käyttää. Solmulle 4 annetaan väri 2, koska kaaren $\{1, 4\}$ takia väriä 1 ei voi käyttää. Solmulle 5 annetaan väri 3, koska kaarien $\{2, 5\}$ ja $\{4, 5\}$ takia värejä 1 ja 2 ei voi käyttää. Solmulle 6 annetaan väri 3, koska kaarien $\{1, 6\}$ ja $\{3, 6\}$ takia värejä 1 ja 2 ei voi käyttää jne.
   Helposti nähdään, että ahneeseen väritykseen värejä tarvitaan lopulta n kappaletta, vaikka graafi voidaan värittää kahdella värillä.

(b) On selvää, että värityksestä riippumatta samalla värillä väritetyt graafin solmut muodostavat riippumattoman joukon. Oletetaan, että käytettävissä on optimaalinen väritys. Jos graafin solmut nyt listataan väriluokka kerrallaan (esim. ensin kaikki punaiset solmut, sitten siniset, jne.), saadaan järjestys, jolla ahne algoritmi tuottaa optimaalisen värityksen. (Ahne väritys ei välttämättä ole identtinen alkuperäisen optimaalisen värityksen kanssa, mutta värien lukumäärä on sama.)

4. Oheinen ohjelma toteuttaa tehtävän yksinkertaisella tavalla. Koska redusoidussa latinalaisessa neliössä ensimmäinen rivi ja sarake ovat määrätyt, $n \times n$-kokoisessa latinalaisessa neliössä riittää sijoittaa $(n-1)^2$ lukua, mikä siis on haun maksimisyvyys. Ohjelma ei pidä yllä konstruoitavaa latinalaista neliötä, vaan ainoastaan boolean-muuttujia `n_in_row_i` ja `n_in_col_j`, joista nähdään, joko luku n on laitettu riville i tai sarakkeeseen j.

```c
/* Etsi kertalukua N olevat redusoidut latinalaiset neliöt */
#define N 4

int n_in_row_i[N+1][N+1]; /* onko numero n jo rivissä i      */
int n_in_col_j[N+1][N+1]; /* onko numero n jo sarakkeessa j */

int place(int depth) {
  int i,j,n,sum;

  if (depth==(N-1)*(N-1)) /* yksi neliö löytyi, koska on */
    return 1;             /* sijoitettu (N-1)**2 numeroa */

  i=2+depth%(N-1); /* lasketaan depthistä, mihin kohtaan */
  j=2+depth/(N-1); /* valitaan numero seuraavaksi        */

  sum=0;
  for (n=1; n<=N; n++) {
    if (!n_in_row_i[n][i] && !n_in_col_j[n][j]) {
      n_in_row_i[n][i]=1;  /* kullekin n:lle vuorollaan:         */
      n_in_col_j[n][j]=1;  /* jos n:ä ei ole laitettu riviin i   */
      sum+=place(depth+1); /* eikä sarakkeeseen j, laitetaan se  */
      n_in_row_i[n][i]=0;  /* näiden risteyskohtaan ja jatketaan */
      n_in_col_j[n][j]=0;  /* hakua; lopuksi otetaan luku pois   */
    }
  }
  return sum;
}

int main(int argc, char **argv) {
  int i,j;

  for(i=1;i<=N;i++) {        /* nollataan taulukot */
    for(j=1;j<=N;j++) {
      n_in_row_i[i][j]=n_in_col_j[i][j]=0;
    }
  }

  for(i=1;i<=N;i++) {     /* asetetaan luvut 1...n      */
    n_in_row_i[i][i]=1; /* ensimmäiseen sarakkeeseen */
    n_in_col_j[i][i]=1; /* ja riviin                 */
  }
  printf("%d\n", place(0));
  return 0;
}
```