Poznámky k úloze

Najděte všechny navzájem neizomorfní grafy se zadaným skóre grafu.

Ověřování izomorfizmu

Nejsnáze se izomorfizmus ověřuje, když je graf reprezentován maticí sousednosti (adjacency matrix).

Nechť A resp. B jsou matice sousednosti grafů G1 resp. G2. Grafy G1 a G2 jsou izomorfní právě tehdy, když existuje permutace p taková, že pro každé i, j platí

`A[i][j] == B[p[i]][p[j]].`

Tato podmínka se dá triviálně ověřovat v cyklu. Jak zjistíme, že taková permutace existuje? Vyzkoušíme všechny možné permutace. Odstavec Další možnosti zrychlování níže naznačuje, že je možno počínat si efektivněji, ovšem za cenu více kódování.

Postup řešení

Když dokážeme rozhodnout o izomorfizmu dvou grafů, zbývá jenom postupně generovat všechny možné kandidáty grafů, pamatovat si postupně se zvětšující množinu M již nalezených navzájem neizomorfních grafů a pro každý nově vygenerovaný graf G ověřit, zda je neizomorfní se všemi grafy v M a pokud ano, přidat G do M.

To je celá strategie řešení. Zbývá jen zvolit takovou reprezentaci grafů, která umožní jejich co nejsnazší nebo co nejpřehlednější generování. Protože jsou zadány stupně jednotlivých uzlů, nezdá se být na první pohled pro tento podnik matice sousednosti nejvýhodnější, v ní nejsou registrovány právě stupně uzlů, na jejichž hodnotě celá úloha spočívá. Dále, grafů je velké množství s nejrozmanitějšími vlastnostmi, my chceme generovat všechny grafy s danou charakteristikou a přitom si udržet v celém procesu přehled. Sáhneme proto po možnosti co nejjednodušší a tou je seznam hran. Platí, že součet stupňů všech uzlů je roven dvojnásobku počtu hran, takže známe požadovaný počet hran v grafu. Uzly v grafu očíslujeme od 0 do N−1. Graf budeme vyrábět tak, že nejprve vytvoříme diskrétní graf bez hran a potom budeme rekurzivní procedurou přidávat do seznamu hran vždy jednu hranu, to jest dvojici čísel uzlů. V jednom volání vyzkoušíme přitom všechny možné způsoby, jak hranu přidat. Celé základní schéma algoritmu řešení pak bude:

```
generuj_graf(Seznam seznam_hran_grafu) {
  if (nelze přidat hranu)
    if (seznam_hran_grafu.size < daný_počet_hran)
      return;                                  //rekurze došla do slepé uličky
    else {                                  // jinak další grafový kandidát nalezen
    Graf g = seznam_hran_grafu.vyrobMaticiSousednosti()
      if (nenílzomorfní(g, všechny_grafy_v_M))
        M.přidej(g);
    }
  while (lze přidat hranu) {
    uzel1 = zvol vhodný uzel;
    uzel2 = zvol vhodný uzel;
    seznam_hran_grafu.přidejhranu(uzel1, uzel2);
      generuj_graf(seznam_hran_grafu)
    seznam_hran_grafu.odeberhranu(uzel1, uzel2)
  } // while
} //generuj_graf
```

Volba vhodného uzlu musí být taková, aby přidáním hrany (*uzel1*, *uzel2*) do seznamu hran, to jest jejím vytvořením, nevyrostl stupeň uzlů *uzel1* a *uzel2* nad zadanou hodnotu. Označme ještě (*preduzel1*, *preduzel2*) poslední hranu v seznamu hran před pokusem o přidáním hrany (*uzel1*, *uzel2*). Promyslete, že stačí volit čísla *uzel1*, *uzel2* tak, aby:

(1) *uzel1* < *uzel2*,

(2) *preduzel1c* ≤ *uzel1*,

(3) if (*preduzel1* == *uzel1*) then *preduzel2* < *uzel2*.

Znamená to, že při daném fixním očíslování uzlů budeme probírat pouze hrany ve vzrůstajícím lexikografickém pořadí podle čísel jejich krajních uzlů. Podle toho i cyklus while uvedený podmínkou

`while (lze přidat hranu)`

skončí v okamžiku, kdy nebudou existovat čísla uzlů *uzel1*, *uzel2* s vlastnostmi (1), (2), (3).

Další možnosti zrychlování

V ověřování izomorfizmu ovšem není nutno vyzkoušet všechny permutace, stačí vyzkoušet jen takové, které zobrazují uzly určitého stupně opět jen na uzly téhož stupně. Případně lze tento výběr ještě dále zjemňovat a zobrazovat na sebe pouze uzly, které mají nejen stejný stupeň ale i stejné skóre svých sousedů. Dalších možností, jak vylučovat neperspektivní permutace uzlů, je více, čím dál tím komplikovanějších, čtenáře nabádáme k samostatnému promyšlení alespoň dvou.