Tretja domača naloga pri algoritmih

Lucija Fekonja

2. junij 2024

Naloga 1: Najmanjši skupni množitelj

Najmanjši skupni množitelj števil a in b lahko izračunamo s pomočjo največjega skupnega delitelja po formuli

$$\operatorname{lcm}(a, b) = \frac{|a \cdot b|}{\gcd(a, b)}. \tag{1}$$

Naša naloga je, da s pomočjo operatorja gcd najdemo najmanjši skupni večkratnih števil $a_1 \ldots a_n \in \mathbb{Z}$. To lahko naredimo iterativno v eni zanki. Najprej shranimo prvi dve števili a_1 in a_2 v novi spremenljivki x in y. Nato izračunamo najmanjši skupni večkratnik x in y po formuli (1) in ga shranimo v novo spremenljivko z. Prej definiran x nastavimo na z, spremenljivko y pa na naslednje število a_i. Ponavljamo dokler ne pridemo do a_n. Pseudokoda je naslednja (v njej začnemo indeksirati z 1):

Algorithm 1 Algoritem za iskanje najmanjšega skupnega večkratnika števil v seznamu $A = [a_1 \ldots a_n]$.

procedure LCM(A)
  $x \leftarrow A[1]$
  $x \leftarrow A[2]$
  for $i = 3 \ldots n$ **do**
    $z \leftarrow \frac{|x \cdot y|}{\gcd(x,y)}$
    $x \leftarrow z$
    $y \leftarrow A[i]$
  end for
  return x
end procedure

Algoritem v vsaki iteraciji izračuna najmanjši skupni večkratnik števila a_i in vseh števil $a_1 \ldots a_{i-1}$, ki jih je že obhodil. Zato je izhod ravno njihov skupni najmanjši večkratnik.

Naloga 2: Modularno potenciranje

Naša naloga je napisati algoritem, ki izračuna $a^b \mod n$, tako da obhodi bite števila b iz desne proti levi. Pri tem imamo b shranjen v dvojiškem zapisu. To lahko storimo rekurzivno s pomočjo formule, ki velja tako v $\mathbb{Z}_n$ kot tudi v $\mathbb{Z}$:

$$a^b = \begin{cases} 1 \mod n & b = 0 \\ a^{b/2} \mod n & b > 0 \text{ in } b \text{ je sod} \\ a^{b-1} \mod n & b > 0 \text{ in } b \text{ je lih} \end{cases}.$$

Oglejmo si kaj nam pove formula.

1. Ustavitveni pogoj $b = 0$. Tedaj vrnemo 1.

2. Če je b sodo število, torej če je njegov najbolj desni bit enak 0, moramo izračunati $a^{b/2} \mod n$. V dvojiškem zapisu $b/2$ izračunamo tako, da preprosto izbrišemo najbolj desno ničlo. Vidimo, da lahko začetni a^b dobimo po $a^b = a^{b/2} \cdot a^{b/2} \mod n$.

3. Če pa je b liho število, torej če je njegov najbolj desni bit enak 1, pa izračunamo $a^{b-1} \mod n$ in začetni a^b dobimo po enačbi $a^b = a \cdot a^{b-1} \mod n$. Pri tem lahko $b - 1$ v dvojiškem zapisu izračunamo tako, da najbolj desni bit iz 1 spremenimo v 2.

Ker moramo v vsakem klicu preveriti zgolj najbolj desni bit števila b, res obhodimo b od desne proti levi. Pseudokoda algoritma je zapisana spodaj.

Algorithm 2 Modularno množenje $a^b \mod n$.

```
procedure MODULARNO MNOŽENJE(a, b, n)
    Zapišimo b v bitih kot b = b_n ... b_0
    if b = 0
        return 1
    else if b_0 = 0
        b ← b_n ... b_1
        d = MODULARNO MNOŽENJE (a, b, n)
        return (d · d) mod n
    else (* b_0 = 1 *)
        b ← b_n ... b_1 0
        d = MODULARNO MNOŽENJE (a, b, n)
        return (a · d) mod n
end procedure
```

Naloga 3: Problem trgovskega potnika

Naj H_n označuje pot pridobljeno z hevristiko najbližje točke, ko vstavimo n točk, in naj H_n^* označuje optimalno pot. Naj bo $c(v,u)$ cena oz. razdalja med točkama u in v. Podobno naj bo $c(H_i)$ vsota cen vseh povezav v poti H_i.

V vsakem koraku opisanega algoritma izbrižemo eno povezavo in dodamo dve novi. Recimo, da je v H_{n-1} obstajala povezava $u \to w$ in da še v ni v ciklu. Naj bo v točka, ki je najbližje ciklu in jo v naslednjem koraku vstavimo takoj za u. Tako dobimo povezave $u \to v \to w$. Posledično se skupna cena poviša za

$$c(u,v) + c(v,w) - c(u,w). \tag{2}$$

Po trikotniški neenakosti vemo, da je $c(u,v) \geq c(u,w) - c(v,w)$, zato lahko omejimo enačbo (2) z

$$c(u,v) + c(v,w) - c(u,w) \leq 2c(u,v).$$

Tako lahko ceno n-tega koraka omejimo s pomočjo cene $n-1$-vega koraka na naslednji način:

$$c(H_n) \leq c(H_{n-1}) + 2c(u,v).$$

Torej če dodajamo povezave $(u_1, v_1) \dots (u_n, v_n)$, lahko ceno poti H_n omejimo z

$$\begin{aligned} c(H_n) \leq c(H_{n-1}) + 2c(u_n, v_i) \\ &\leq c(H_{n-1}) + 2c(u_{n-1}, v_{n-1}) + 2c(u_n, v_n) \\ &\leq \cdots \\ &\leq 2\sum_{i=1}^{n} c(u_i, v_i) \end{aligned}$$

Sedaj si oglejmo kako deluje algoritem. Začne se s poljubno izbiro mesta v in tvori cikel sam vase. Nato na vsakem koraku poišče masto u, ki še ni v ciklu, ampak je najbližje kateremu koli mestu v ciklu, recimo v, in u postavi takoj za v v cikel.

Oglejmo si kako deluje Primov algoritem za iskanje minimalnega vpetega drevesa (ang. minimal spanning tree oz. MST). Na začetku izbere naključno vozlišče v grafu G. Nato izbere povezavo, katere eno vozližče je že bilo obiskano, drugo pa ne in ima najmanjšo možno ceno. Ko najde to povezavo, jo doda vljučno z njenimi vozlišči v MST. Ponavlja, dokler niso vsa vozlišča iz G v MST.

V principu sta oba algoritma enaka, saj na vsakem koraku iteracije dodamo vozlišče, ki je njmanj oddaljeno od katerega koli že dodanega vozlišča.

V Primovem algoritmu je cena vseh dodanih povezav enaka $\sum_{i=1}^{n} c(u_i, v_i)$. Torej

$$c(MST) = \sum_{i=1}^{n} c(u_i, v_i).$$

Po definiciji je minimalno vpeto drevo tisto, katerega vsota teži vozlišč je najmanjša možna, kar implicira $c(MST) \leq c(H_n^*)$.

Tako pridemo do željene neenakosti:

$$c(H_n) \leq 2\sum_{j=1}^{n} c(u_i, v_i) = 2c(MST) \leq c(H_n^*).$$

To pa ravno dokazuje dvo aproksimacijo. Namreč slednje pove, da cena heuristike ni večja od dvakratnika cene optimalne poti. Tako smo zaključili naš dokaz.