Algoritmi
Domača naloga 4

Sara Bizjak | 27202020

Maj 2021

Problem 1 – Najmanjši skupni množitelj in modularno potenciranje

A del: *Najmanjši skupni množitelj*

Definirajmo $\texttt{lcm}(a_1, a_2, \ldots, a_n)$ kot najmanjši skupni množitelj n celih števil $a_1, a_2, \ldots, a_n$. Ta funkcija vrne najmanjše nenegativno celo število, ki je množitelj vseh števil a_i. Pokežmo, kako bi učinkovito izračunali $\texttt{lcm}(a_1, a_2, \ldots, a_n)$ z uporabo funkcije `gcd`, ki je dvoparametrna.
Definirajmo funkcijo $\texttt{lcm}_2(a, b)$ kot najmanjši skupni množitelj dveh celih števil a in b. Formula, ki povezuje funkciji za najmanjši skupni množitelj dveh števil in največji skupni delitelj dveh števil, je

$$\text{lcm}_2(a, b) = \frac{ab}{\gcd(a, b)}$$

Ker zgornja zveza velja za dva parametra, funkcijo za izračun $\texttt{lcm}(a_1, a_2, \ldots, a_n)$ definiramo rekuzrvino. Definirajmo funkcijo $\texttt{LCM}_{rek}$, ki za vhod velikosti 2 (torej za seznam dveh števil [a, b]), vrne kar $\text{lcm}_2(a, b)$, če pa je vhodnih števil več, torej za vhodni seznam $[a_1, a_2, \ldots, a_n]$, pa se kliče rekurzivno na način

$$\texttt{LCM}_{rek}([\text{lcm}_2(a_1, a_2), a_3, \ldots, a_n])$$

kar pomeni, da je zaporedje ukazov znotraj funkcije $\texttt{LCM}_{rek}$ enako

$$\text{lcm}_2(\ldots(\text{lcm}_2(\text{lcm}_2(a_1,\ a_2),\ a_3)\ldots),\ a_n)$$

oziroma, razpisano s funkcijo `gcd`, enako

$$\frac{a_1 \cdot a_2 \cdot\ \cdots\ \cdot a_n}{\gcd(\ldots(\gcd(\gcd(a_1,\ a_2),\ a_3)\ldots),\ a_n)}$$

B del: *Modularno potenciranje*

Algoritem za modularno potenciranje je predstavljen v [[1], chapter 31]. Sestavimo algoritem za modularno potenciranje, ki b bitov obhodi iz desne proti levi namesto iz leve prosti desni.

Ideja je, da ko eksponent ni potenca števila 2 (preverimo tako, da pogledamo ostanek pri deljenju z 2), moramo vmesni rezultat dodatno pomnožiti z vrednostjo trenutne osnove (po modulu), potem pa osnovo kvadriramo (po modulu) in eksponent razpolovimo. Če pa je eksponent deljiv s številom 2, naredimo samo druga dva ukaza, torej osnovo kvadriramo (po modulu) in eksponent razpolovimo.
Delujoča python koda je dostopna v priloženi datoteki `dn4.py`. Psevdo-koda opisanega algoritma:

```
def modularnoPotenciranje(a, b, n):
    """
    Vhod: a, b, n
    Izhod: a**b mod n
    """

    ce b == 0:
        vrni 1

    liha = 1            // zapisujemo liha mnozenja posebej
    osnova = a % n
    dokler b > 1:
        ce b % 2 == 1:                  // trenutni bit liho stevilo
            liha = (osnova * liha) % n
                                        // trenutni bit sodo stevilo
                                        // izvede se samo ta korak
        b = b // 2
        osnova = (osnova ** 2) % n

    vrni (osnova * liha) % n
```

Problem 2 – problem trgovskega potnika

Podano imamo naslednjo hevristiko najbljižje točke za gradnjo poti trgovskega potnika, ki upošteva trikotniško neenakost.
Začnimo z izbiro poljubnega mesta v. Predpostavimo, da je to mesto legalen cikel (sam vase), ki predstavlja pot trgovskega potnika. Poiščimo mesto u, ki je najbljižje v in ga dodajmo v cikel tako, da u vstavimo v cikel takoj za v. Sedaj imamo cikel preko dveh mest. Ponovno poiščemo mesto u, ki je najbližje kateremukoli mestu na ciklu. Recimo, da je to najbližje mesto iz cikla mesto v. Ponovno dodajmo u takoj za v in to ponavljajmo dokler še imamo mesta izven cikla. Pokažimo, da je ta hevristika dvo aproksimacijska.

To, da je hevristika dvo aproksimacijska, pomeni, da je rešitev algoritma kvečjemu dvakrat slabša od optimalne.

Problem prevedemo na iskanje Hamiltonovega cikla na grafu G z minimalno ceno, tj. zgradimo najmanjše vpeto drevo T in pre-order obhod tega drevesa s pomočjo Primovega algoritma.
Denimo, da imamo podan poln neusmerjen graf $G(V, E)$, povezava med dvema vozliščema grafa u in v pa ima dodeljeno ceno $c(u, v) > 0$, ki je porojena z evklidsko dolžino posamezne povezave – krajša kot je povezava, nižja je njena cena.
Na $G(V, E)$ nato poženemo Primov algoritem in dobimo najmanjše vpeto drevo T, na katerem algoritem nadaljujemo.
Označimo z $v \in V(G)$ poljubno vozlišče, v katerem začnemo obhod p. Označimo sosede vozlišča v v najmanjšem vpetem drevesu T z $N_T(v) = \{u_1, \ldots, u_n\}$. Algoritem najprej obišče tisto vozlišče iz $N_T(v)$, ki je od vozlišča v najbolj oddaljeno, ker je bilo nazadnje dodano. Označimo to vozlišče z u_{max}. Naj bo $N_T(u_{max}) = \{w_1, \ldots, w_n\}$ množica sosedov vozlišla u_{max}. Potem algoritem obhod nadaljuje na sosednjem vozlišču w_{max}, ki je od u_{max} najbolj oddaljeno in še ni bilo obiskano. Če takega vozlišča ni, tj. če so bila vsa vozlišča iz množice $N(u_{max})$ že obiskana, se vrne po isti povezavi, po kateri je v u_{max} prišel, sicer nadaljuje rekurzivno.
Obhod p gradi torej tako, da vozlišča obiskuje v vrtsnem redu po padajočih razdaljah (sosednja vozlišča), v obhod pa doda še neobiskano vozlišče prvič, ko do njega pride. Ko algoritem konča, dobimo vrstni red vozlišč drevesa oz. vrstni red obiska mest kot jih po vrsti obišče trgovski potnik.

Opisan algoritem očitno deluje pravilno in nam poda pravo rešitev na zastavljen problem. Pokažimo sedaj, da je opisana hevristika dvo aproksimacijska.
Označimo z $l(p_{opt})$ dolžino optimalne poti za zgoraj opisan problem. Za $l(p_{opt})$ velja, da je gotovo večja ali enaka ceni najmanjšega vpetega drevesa T, ki jo označimo s $c(T)$, torej

$$l(p_{opt}) \geq c(T) \tag{1}$$

Označimo s p obhod, ki ga vrne naš algoritem, s $p_{pre-order}$ pa zaporedje vseh obiskanih vozlišč med pre-order obhodom. Pri obhodu $p_{pre-order}$ drevesa T gremo preko vsake povezave v T natanko dvakrat – enkrat, ko jo prehodnimo prvič, in enkrat, ko se po njen vračamo po obhodu poddrevesa. Velja torej

$$l(p_{pre-order}) = 2 \cdot c(T) \tag{2}$$

Po trikotniški neenakosti velja

$$l(p) \leq l(p_{pre-order}) \tag{3}$$

saj obhod p iz obhoda $p_{pre-order}$ dobimo le z morebitnim potrebnim odstranjevanjem vozlišč. Neenakost (3) sicer sledi iz trikotniške neenakosti, ki velja za vsako pot med tremi vozlišči.

Imejmo torej pot uwv. Po trikotniški enakosti velja, da je

$$l(uv) \leq l(uw) + l(wv) \quad (= l(uwv)) \tag{4}$$

kar potrdi, da neenakost (3) velja.
Če združimo (2) in (3) dobimo

$$l(p) \leq 2 \cdot c(T)$$

Ker vemo, da velja, da je optimalni obhod po dolžini kvečjemu enak dolžini obhoda, dobljenega z našim algoritmom, velja

$$l(p_{opt}) \leq l(p) \tag{5}$$

Če združimo zgornje zveze, dobimo iskano konstanto

$$\frac{l(p)}{l(p_{opt})} \leq \frac{2 \cdot c(T)}{c(T)} = 2$$

$$\Rightarrow l(p) \leq 2 \cdot l(p_{opt})$$

Dobljena pot je torej največ za 2 daljša od optimalne, kar pomeni, da je opisana hevristika res dvo aproksimacijska.
Algoritem z dokazom prirejen po [[1], chapter 35] in po [3].

Literatura

[1] Thomas H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[2] *Binary exponentiation*, [ogled 5. 5. 2021], dostopno na `https://primes.utm.edu/glossary/page.php?sort=BinaryExponentiation`.

[3] *The travelling salesman problem*, [ogled 10. 5. 2021], dostopno na `www.cs.tufts.edu/~cowen/advanced/2002/adv-lect3.pdf?fbclid=IwAR2A0DJpcNyD5LCgyH4FC9aZz8fcu2ioTGfxgG8nKUTYYWtRH52XnDRULjo`.