Osnove programiranja v diskretni matematiki
Zapiski predavanj

2023/24

Povzetek

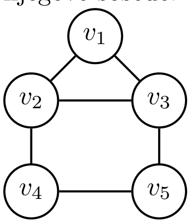
Dokument vsebuje zapiske predavanj predmeta Osnove programiranja v diskretni matematiki profesorja Vesela v okviru študija prvega letnika magistrskega študija matematike na FNM.

Kazalo

1 Predstavitve grafov in onsnovni algoritmi nad njimi

Definicija 1: Graf G je sestavljen iz množice ogljišč $V(G)$ in množice povezav $E(G)$. Pišemo: $G = (V, E)$. Pri tem pogosto označimo $|V|= n$ in $|E|= m$. Za $v \in V(G)$ definiramo $d(v) =$ *število sosed vozlišča v v grafu G*

Zgled 1: Poglejmo si naslednji graf in v tabeli določimo vsakemu vozlišču njegove sosede. G :



Vozlišče	v_1	v_2	v_3	v_4	v_5
Sosedi	v_2, v_3	v_1, v_3, v_4	v_1, v_2, v_5	v_2, v_5	v_3, v_4

Za predstavitev grafa imamo več možnosti. Prva možnost je matrika sosednosti, druga možnost (ki nas bo zanimala), pa so seznami sosedov. Seznami sosedov se izkažejo za bolj prostorsko varčne, saj matrika sosednosti tipično zavzame $O(n^2)$ mest, seznami sosedov pa manj. Koliko manj, je odvisno od načina implementacije.

1.1 Enostavni seznami sosedov

Sestavimo seznam S vozlišč, tipično urejenih in oštevilčenih (npr 1 predstavlja vozlišče v_1, 2 vozlišče v_2 itd.). Nato sestavimo seznam sosedov P tako, da po vrsti naštejemo sosede vozlišča 1, nato vozlišča 2, itd. Pri tem vsako vozlišče iz S opremimo s kazalcem, ki kaže v P na mesto, kjer se začnejo njegovi sosedi.

Zgled 2: Predstavimo graf G iz zgleda 1 s pomočjo enostavnega seznama sosedov.

S:

1	2	3	4	5

P:

2	3	1	3	4	1	2	5	2	5	3	4

Z S_i označimo seznam sosedov vozlišča v_i. Seznam S ima ravno n mest, seznam P pa $\sum_{i=1}^{n} d(v_i) = 2m$ mest. Prostorska zahtevnost enostavnih seznamov sosedov je torej $O(n + m)$, kar je po navadi res manj kot $O(n^2)$.

1.2 Razširjeni seznami sosedov

Najprej za vsako vozlišče v_i sestavimo seznam sosedov S_i, v katerem vsakega od sosedov v_i predstavimo s poljem p, ki je sestavljeno iz petih komponent: $p = (v_i, v_j, \&r, \&n, \&u)$. Pri tem je v_i vozlišče, katerega sosede obravnavamo, v_j ime soseda, $\&r$ kazalec, ki kaže na predhodnika p v S_i, $\&n$ kazalec, ki kaže na naslednika p v S_i in $\&u$ kazalec, ki kaže na polje v seznamu S_j, ki predstavlja povezavo v_iv_j. Če je p prvi v seznamu S_i je $\&r = 0$, če je pa zadnji, je pa $\&n = 0$.

Zgled 3:

Seznam	Ime polja	Polje
S_1	a	$(v_1, v_2, 0, \&c, \&b)$
	c	$(v_1, v_3, \&a, 0, \&d)$
S_2	b	$(v_2, v_1, 0, \&e, \&a)$
	e	$(v_2, v_3, \&b, \&g, \&f)$
	g	$(v_2, v_4, \&e, 0, \&h)$
S_3	d	$(v_3, v_1, 0, \&f, \&c)$
	f	$(v_3, v_2, \&d, \&i, \&e)$
	i	$(v_3, v_5, \&f, 0, \&j)$
S_4	h	$(v_4, v_2, 0, \&k, \&g)$
	k	$(v_4, v_5, \&h, 0, \&l)$
S_5	j	$(v_5, v_3, 0, \&l, \&i)$
	l	$(v_5, v_4, \&j, 0, \&k)$

Omenimo tukaj, da kadar grafu dodamo povezavo v_iv_j, to naredimo tako, da dodamo novo polje na začetek seznamov S_i in S_j.

1.3 Časovna zahtevnost nekaterih operacij na grafih

Sedaj, ko smo uvedli različne sezname sosedov, lahko primerjamo časovno zahtevnost nekaterih klasičnih operacij na grafih.

Operacija	Enostavni sez.	Razširjeni sez.
Preveri, če G vsebuje povezavo v_iv_j	$O(d(v_i))$	$O(d(v_i))$
Označi vse sosede vozlišča v_i	$O(d(v_i))$	$O(d(v_i))$
Označi vse povezave	$O(m)$	$O(m)$
Dodaj povezavo v_iv_j	$O(m)$	$O(1)$
Odstrani povezavo v_iv_j	$O(m)$	$O(d(v_i))$
Odstrani vse povezave s krajiščem v v_i	$O(m)$	$O(d(v_i))$

Izkaže se torej, da četudi imamo malenkost več dela z reprezentacijo grafov v razširjenih seznamih sosedov, s tem pridobimo na časovni zahtevnosti.

1.4 Urejanje seznamov sosedov

Za urejanje lahko, v splošnem, uporabimo algoritem z zahtevnostjo $O(k\log(k)$, kjer je k število elementov seznama. Če to naredimo za vsak seznam dobimo:

$$\begin{aligned}\sum_{i=1}^{n} O(d(v_i)\log(d(v_i))) \leq \sum_{i=1}^{n} O(d(v_i)\log(n)) &= O(\log(n)\sum_{i=1}^{n} d(v_i)) \\ &= O(m\log(n))\end{aligned}$$

To hitrost lahko torej pričakujemo pri enostavnih seznamih sosedov. Obstaja pa še hitrejši način, če si pomagamo z razširjenimi seznami sosedov. To naredimo tako, da upoštevamo, da je v vsakem polju seznama S_i na v komponenti vozlišče v_i. Nove sezname gradimo tako, da se sprehajamo od seznama S_n, do S_1 in na vsakem koraku odstranimo polje $(v_i, v_j, \ldots)$ iz S_i. Nato odstranjenemu polju zamenjamo prvo in drugo komponento, da dobimo polje oblike $(v_j, v_i, \ldots)$ in to novo polje vstavimo na začetek novega seznama, ki ustreza vozlišču v_j. Algoritem še zapišemo bolj pregledno:

Data: Razširjeni seznami sosedov S_1,. . . , S_n
Result: Urejeni razširjeni seznami sosedov B_1,. . . , B_n
Ustvari prazne razširjene sezname sosedov B_1,. . . , B_n;
for i = n downto 1 **do**
    while $S_i \neq \emptyset$ **do**
        Odstrani prvo polje $p = (v_i, v_j, \ldots)$ iz S_i;
        Dodaj $\acute{p} = (v_j, v_i, \ldots)$ na začetek B_j;
    end
end

Ta algoritem ima časovno zahtevnost $O(m)$, torej je hitrejši od algoritma na enostavnih seznamih sosedov, katerih urejanje je imelo časovno zahtevnost $O(m\log(n))$.