Kombinatorické Algoritmy
Luděk Kučera, LS 2010

Seminář 1x týdně (ač oficiálně 2/2), zápočet a zkouška

0 Prerekvizity

Pro pochopení výkladu je dobré znát Dijsktrův a Bellman-Fordův algoritmus hledání nejkratší cesty v grafu, stejně jako některé algoritmy hledání maximálního toku v grafu (Ford-Fulkerson, Dinitz).

1 Párování v bipartitním grafu

Pozorování: Není-li párování v G maximální, pak v G existuje cesta

V – nehrana – N – hrana – N – [nehrana – N – hrana – N]* – nehrana – V

kde V je volný vrchol (není v párování), N nevolný vrchol (je v párování) a hrana/nehrana se týká párování.

Tato cesta mi jednoduše umožňuje zvětšit párování o 1: červené párování je o 1 větší než modré.

Dk: Vezmu G s modrým a červeným párováním (červené al. o 1 větší), vyházím nebarevné a oboubarevné hrany. Pak mi zbudou cykly a cesty sudé délky (**č – m – č – m –**...), ty mají stejně červených i modrých hran, a cesty liché délky tvaru **č – [m – č]* – m – č** nebo **m – [č – m]* – č – m**. Červených hran je ale aspoň o 1 více než modrých, tedy existuje aspoň jedna cesta tvaru **č – [m – č]* – m – č**, a tedy pozorování platí.

Řeší se převedením na toky v sítích

Z → ***nové hrany*** → **1.partita** → **původní hrany** → **2.partita** → ***nové hrany*** → ***S***

Ford Fulkerson

obecně je pomalej, ale tady se to dá

Dinitz

má zde O (M * √N) – tj. nepotřebujem zde lepší alg

dk na 3 stránky

2 Párování v obecném grafu

opět hledání zlepšující cesty

Maďarská metoda

historická

ala FF, prohledávání do šířky

Edmondsův algoritmus

kontrakce lichých cyklů → bipartitní graf, v něm hledám zlepšující cestu

Algoritmy vycházející z EA

neprovádím kontrakce, místo toho označuju vrcholy jako liché a sudé (1 vrchol může být obojí), potřebuju najít hranu z S do VV

dá se dostat na O (M * √N) - pokud hledám nejkratší možné cesty - podobně jako u bipartitního grafu

3 Pravděpodobnostní a paralelní algoritmy

typicky grafové algoritmy neparalelizovatelné

nejkratší cesta – O(log N) pomocí matic

párování v O(log N) – nevíme jestli to jde

Existence perfektního párování v bipartitním grafu – teoretický algoritmus – paralelizace párování

je to pravděpodobnostní algoritmus

matice řádu N*N, A_{ij} = číslo hrany $\{v_i, v_j\}$ nebo 0 → determinant

determinant ≡ 0 → neexistuje perfektní párování

jinak mi každá nenulová permutace dává jedno perfektní párování (jako součin čísel hran)

náhodně volím čísla hran (x_i) a počítám determinant (gaussovou eliminací), opakuju třeba 100x – pstní algoritmus, umím paralelně

Pozn: zobecnění – např. tripartity matching – vede na NP úplný problém

chci zjistit to párování: odebírám hrany, když odebráním hrany párování ztratím, tak leží v max párování

chci největší párování: odebírám vrcholy, až mám graf který má perfektní párování

4 Hledání největšího toku nejmenší ceny

najdu maximální tok; transformuju ho tak, aby se snížila jeho cena (při zachování velikosti toku) – ala FF, počet iterací není zaručen

zobecněný cyklus = cyklus bez ohledu na orientaci hran

najdu nenasycený zobecněný cyklus (tj. je možné zvýšit tok cyklem), který má zápornou cenu, a ten přidám (respektive odstraním) – pokud neexistuje, má tok minimální cenu

Dk: *tok je nejlepší <=> neexistuje nenasycený zobecněný cyklus záporné ceny* (dokážeme tvar *tok není nejlepší <=> existuje nenasycený zobecněný cyklus záporné ceny*)

<= stačí ukázat, že odstraněním toho cyklu nezměním tok (divergence ve vrcholech se nezmění) a snížím cenu (z definice ceny toku)

=> existuje důkaz s cirkulacemi na 3 stránky (Kučera), který vlastně běží takhle:

Nechť mám dva maximální toky t1, t2, kde t2 je lepší než t1 [`||t1|| = ||t2||` a `w(t1) > w(t2)`].

Toky nemohou být stejné, tedy se liší v toku aspoň jednou hranou e {v0, v1}. Protože ale musí platit nulová divergence všech vrcholů (= Kirchhoff), musí se lišit i tok nějakou hranou {v1, v2}. Postupujeme indukcí, až se dostaneme do vrcholu, ve kterém už jsme byli (to se časem stane, protože je graf konečný a v každém vrcholu musí platit Kirchhoff – máme přidanou hranu t → s), tj. našli jsme zobecněný cyklus, který je nenasycený (v obou grafech, ale v každém v jiném směru). Pokud má zápornou cenu, tak jsme vyhráli, jinak ho odstraníme, dostáváme tak t2' (jakoby provedeme změnu částečně měnící t2 na t1) a opakujeme. Dokud není t1 = t2', tak

tam musí takový cyklus existovat. Pokud by všechny takové cykly měly nulovou resp. kladnou cenu, byla by cena t2 stejná resp. vyšší než cena t1, takže tam určitě najdeme aspoň jeden cyklus se zápornou cenou.

Zlepšující cyklus hledám pomocí Bellman-Forda (nevaděj mu záporný ceny cest, pracuje O(M*N) – najde mi zápornej cyklus)

5 Isomorfismus 3-souvislých rovinných grafů

Pro G 3-souvislý rovinný a v něm kružnici C platí:

C je obvodem nějaké stěny <=> G\C je souvislý

Dk: vrcholy A a B, mezi nima dvě cesty přes kružnici C, když vytrhnu tu kružnici, pořád mi tam zbyde ještě třetí cesta mezi nima; a uvnitř kužnice mi nic nezustalo pač tam byla stěna

Důsledek: 3-souvislé grafy mají jednoznačné nakreslení (každá hrana má dvě stěny, tj. můžu iterativně hledat ty stěny a kreslit je)

Isomorfismus typicky využije jednoznačnosti nakreslení

Kvadratický algoritmus

v G1 si zvolím vrchol v0 a z něj hranu e0, zkusím jim přiřadit každý vrchol v G2 a každou jeho hranu (zbytek pak díky jednoznačnosti dopočítám tranzitivně)

Složitost: N*6N*2 (všechny vrcholy, všechny hrany, normálně a zrcadlově) = $O(N^2)$

Kódovací algoritmus

hledám signaturu grafu přes signaturu hrany – čtveřice [stupeň jednoho vrcholu, stupeň druhého vrcholu, délka jedné stěny, délka druhé stěny]

získané kódy lexikograficky uspořádám, iteruju dokud se to zjemňuje, pak porovnám signatury grafů

Redukovací algoritmus

mám spoustu pravidel, podle kterejch redukuju graf (zachovávaj isomorfismus), na konci mám dvě platónská tělesa, ta porovnám v konečném čase

6 Toky v rovinných sítích

Upper-most path algoritmus

Tok proženu nejhořejší cestou → aspoň jedna hrana se nasytí (~ vypadne) → získávám novou nejhořejší cestu → iteruju

Zdroj i spotřebič musí ležet na téže (v nakreslení vnější) stěně

Je v podstatě duální k Dijkstrovi: prohodím vrcholy a stěny, odtsraním vrcholy stupně 2, hrany si odpovídají (nové jsou „kolmo“ k těm starým), tj. z kapacit udělám délky hran a hledám nejkratší cestu shora dolů (z vrcholu odpovídajícímu horní polorovině do vrcholu odpovídajícího dolní polorovině) = maximální řez v G; tok hranou = rozdíl čísel ve vrcholu na začátku a na konci hrany (tj. rozdíl vzdáleností od horní stěny)

Implementuje se právě pomocí Dijkstry

O(n log n) – respektive O(m log m), ale v rovinném grafu $m \leq 3n$

7 Hledání separátoru v rovinném grafu

Každý rovinný graf G lze rozdělit na dvě části (A, B) a separátor (C), kde všechna spojení A a B vedou přes C a kde $|A| \leq \frac{2}{3} n$, $|B| \leq \frac{2}{3} n$ a $|C| \leq 2\sqrt{2}\sqrt{n}$

Využití pro divide-and-conquer.

Teorie

BÚNO graf je triangulace (dotriangulování mu neuškodí)

poloměr grafu: zvolím vrchol jako střed grafu, poloměr je maximum z nejkratších cest ze středu do vrcholu (pro každý vrchol) – střed volím tak aby poloměr byl co nejmenší

lemma 1: pro G o poloměru ≤ r lze najít cyklus – separátor o délce ≤ 2r+1 že |A|, |B| ≤ ⅔ n

střed → cesty, mezi nima příčky (každá příčka určuje 1 cyklus: střed → jedna cesta → příčka → druhá cesta), ten je určitě ≤ 2r+1

separátorový cyklus mi dává příčka s nejnižším ohodnocením, kde ohonocení je max {# vrcholů uvnitř cyklu, # vrcholů vně cyklu | oboje bez vrcholů ležících přímo na cyklu}; preferuju cyklus s nižším počtem stěn uvnitř cyklu

dk sporem – vezmu stěnu přiléhající na příčku zevnitř, tj. mám třetí vrchol a rozeberu jeho možné polohy

lemma 2: vrstvy a jejich separátor, používá lemma 1

lemma 3: používá lemma 2, je technické (počítací)

Algoritmus

V čase O(n) najdeme separátor.

8 Hledání maximální kliky

Jednoduchý pravděpodobnostní algoritmus

Vezmu množinu všech vrcholů V', přidám do kliky v1, z V' odstraním vrcholy nespojené s v1, přidám do kliky v2 z V', odstraním z V' vrcholy nespojené s v2... Na konci je V' klika.

Najde kliku o velikosti asi poloviny maximální kliky (což je slušný)

Dk tohoto tvrzení – pomocí nástrojů teorie pravděpodobnosti (náhodná veličina, Čebyševova nerovnost apod.)

Jiný pokus

Vezmu si n vrcholů (prázdný graf), náhodně vyberu k vrcholů, na nich postavím kliku, náhodně přidám další hrany.

Jednoduchý algoritmus tuto kliku skoro jistě nenajde.

Dá se využít toho, že vrcholy v klice mívají vyšší stupeň – při vybírání vrcholu preferuju ten s vyšším stupněm. (ale: vrchol s vysokým stupněm nemuí nutně ležet v klice)