Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Voľba šéfa na cayleho grafoch s lineárnym počtom správ

(diplomová práca)

Bc. Martin Námešný

9.2.1 Informatika

Vedúci: doc. RNDr. Rastislav Královič, PhD

Bratislava, 2010

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Ďakujem svojmu školiteľovi doc. RNDr. Rastislav Královič, PhD za cenné rady a postrehy, ktorými mi pomohol pri písaní tejto práce.

Abstrakt Práca sa zaoberá problémom voľby šéfa na cayleho grafoch s použitím lineárneho počtu správ. Štrukturálne vlastnosti a zmysel pre orientáciu majú veľký vplyv na počet prenesených správ. V práci sme navrhli algoritmus na voľbu šéfa na cayleho grafoch generovaných transpozičným lesom využívajúci turnajovú schému. Podarilo sa nám nájsť netriviálnu podtriedu cayleho grafov, na ktorých je možné zvoliť šéfa s použitím lineárneho počtu správ.

Kľúčové slová distribuované algoritmy, voľba šéfa, cayleho graf, lineárny počet správ, turnajová schéma, transpozičný les

Obsah

1 Úvod **9**
1.1 Členenie práce 10
1.2 Výpočtový model 10
1.2.1 Zmysel pre orientáciu 11
1.3 Voľba šéfa 11
1.3.1 Prečo volíme šéfa 12
1.3.2 Predchádzajúce výsledky 12
1.4 Cayleho graf 13
1.4.1 Cayleho graf ako komunikačná sieť 13
1.4.2 Voľba šéfa a známe cayleho grafy 14

2 Voľba šéfa a transpozičné grafy **15**
2.1 Transpozičný graf 15
2.2 Transpozičný strom 15
2.2.1 Základné vlastnosti 15
2.2.2 Dekompozičné usporiadanie 17
2.2.3 Dekompozícia 19
2.2.4 Voľba šéfa 20
2.2.5 Doručenie správy do susedného podgrafu 22
2.2.6 Doručenie správy šéfovi podgrafu 26
2.2.7 Spätné doručenie správy 26
2.2.8 Korektnosť algoritmu 27
2.2.9 Celkový počet odoslaných správ 28
2.2.10 Časová zložitosť 29
2.3 Transpozičný les 30
2.3.1 Základné vlastnosti 30
2.3.2 Dekompozičné usporiadanie 32
2.3.3 Dekompozícia 32
2.3.4 Voľba šéfa 34
2.3.5 Doručenie správy do susedného podgrafu 35
2.3.6 Doručenie správy šéfovi podgrafu 36

2.3.7 Spätné doručenie správy 36
2.3.8 Korektnosť algoritmu 36
2.3.9 Celkový počet odoslaných správ 37
2.3.10 Časová zložitosť 40

3 Záver **42**

Kapitola 1

Úvod

S rozvojom počítačových sietí a viacjadrových procesorov je úzko spätá oblasť distribuovaných výpočtov. Či už ide o peer-to-peer siete, distribuované databázy alebo ad hoc bezdrôtové siete, všetky používajú nejakú formu distribuovaného počítania. Táto práca sa venuje jednému zo základných problémov distribuovaných výpočtov - voľbe šéfa.

Ako väčšina distribuovaných algoritmov aj voľba šéfa je závislá na konkrétnej topológií komunikačnej siete. Nie je jednoduché preniesť algoritmus z jedného druhu komunikačnej siete na druhý. Existuje algoritmus [8], ktorý volí šéfa na ľubovolnom grafe, ale používa až $O(N \log N + E)$ správ, kde N je počet vrcholov a E je počet hrán. Pre všeobecné grafy platí, že to je aj optimálny výsledok.

Algoritmy na konkrétnych triedach grafov už vedia zvoliť šéfa s použitím len lineárneho počtu správ. Dobrým príkladom je hyperkocka [18] alebo star graf [17].

V tejto práci sa venujeme cayleho grafom ako komunikačnej sieti. Cayleho grafy majú z hľadiska distribuovaných výpočtov výhodné vlastnosti: sú vrcholovo aj hranovo tranzitívne, majú vysokú odolnosť voči chybám. Vhodnou voľbou grupy a množiny generátorov sa dá ovplyvniť priemer, prípadne maximálny stupeň grafu. Ďalšou výhodou cayleho grafov je, že sú často hierarchické, čo uľahčuje návrh distribuovaných algoritmov. Spomínaná hyperkocka aj star graf sú cayleho grafy.

Cieľom práce je nájsť, čo najväčšiu podmnožinu cayleho grafov, na ktorých sa dá zvoliť šéf s lineárnym počtom správ.

1.1 Členenie práce

Prvá časť diplomovej práce sa venuje definícii výpočtového modelu, formálne zavedieme definíciu cayleho grafu a algoritmus na voľbu šéfa. Ďalej popisujeme známe výsledky o voľbe šéfa na všeobecných grafoch a na známych cayleho grafoch. V tejto časti popíšeme ich niektoré základné vlastnosti.

Druhá časť diplomovej práce definuje pojem transpozičný strom a cayleho grafy generované transpozičným strom. Ukážeme základné vlastnosti ako počet vrcholov a priemer. Pomocou teórie grúp dekomponujeme takýto cayleho graf a poukážeme na jeho rekurzívnu štruktúru. V ďalšej časti navrhneme distribuovaný algoritmus na voľbu šéfa a ohraničíme vygenerovaný počet správ a časovú zložitosť algoritmu.

Tretia časť práce definuje transpozičný les a popisuje cayleho graf generovaný transpozičným lesom. Nasleduje návrh a popis algoritmu a odhad časovej zložitosti a počtu správ.

1.2 Výpočtový model

Výpočtový model v tejto diplomovej práci je daný množinou identických procesorov a komunikačnou sieťou. Každý procesor ma vlastnú nezdieľanú pamäť. Procesory si medzi sebou posielajú správy prostredníctvom liniek.

Vlastnosti procesorov:

- Identita procesora: Každý procesor má priradenú identifikačné číslo z množiny prirodzených čísel. Toto číslo je jedinečné v rámci komunikačnej siete. V texte ho budeme označovať ako PID.
- Identita susedov: Procesor vie o svojich susedoch, ale nevie ich PID. Priama adresácia nie je možná. Adresácia je možná iba pomocou lokálnych názvov liniek.
- Topologické informácie: Procesor pozná alebo vie spočítať z lokálnych informácií počet procesorov v komunikačnej sieti, priemer grafu a ďalšie.

Vlastnosti liniek:

- Spoľahlivosť: Komunikačné linky sú spoľahlivé. Každá odoslaná správa je prijatá práve raz. Správy sa po ceste nemodifikujú. Nevznikajú nové správy ani sa nestrácajú.
- Vlastnosť FIFO: Správy sú prijímané v poradí, akom boli odoslané.

- Kapacita: V jednom momente môže byť linkou prenášané neobmedzené množstvo správ.

Procesory nemajú prístup ku globálnemu časovému rámcu ani nemajú schopnosť sledovať reálny čas. Náš model je plne asynchrónny systém, ako je definovaný v [19].

Komunikačnú sieť môžme modelovať pomocou grafu $G = (V, E)$. Kde vrcholy budú procesory a hrany budú linky. V tejto práci budeme predpokladať, že ide o neorientovaný graf bez viacnásobných hrán a slučiek. V texte budeme volne zamieňať pojmy vrchol, procesor prípadne uzol a hrana a linka.

1.2.1 Zmysel pre orientáciu

Santoro v článku [16] analyzoval, aký vplyv má znalosť topológie a smerovanie komunikačných liniek na komunikačnú zložitosť. Práce [6] a [7] nadväzujú a formalizujú definíciu zmyslu pre orientáciu. My si uvedieme len neformálny popis z [6]. Zmysel pre orientáciu spĺňa nasledujúce vlastnosti:

- Každý vrchol má jednoznačne označené značkami hrany s ním incidentné. Z opisu cesty len pomocou značiek hrán, vieme určiť, či dve cesty končia v tom istom vrchole.
- Každý vrchol odkazuje na iný vrchol pomocou jeho lokálneho mena. Nech $p_x(y)$ je lokálny názov vrcholu y z pohľadu vrchola x. Potom existuje funkcia, ktorá priradí k opisu ciest medzi vrcholmi x a y lokálny názov $p_x(y)$.
- Existuje funkcia, ktorá transformuje lokálne názvy z pohľadu vrchola x na lokálne názvy z pohľadu vrchola y, ak má k dispozícií opis cesty medzi vrcholmi x a y.

Dobrý príkladom zmyslu pre orientáciu je model mriežka a hrany označené hore, dole, vpravo, vľavo. V práci budeme pracovať s hranami označenými pomocou generátorov. Takéto označenie hrán, ako neskôr uvidíme, tiež spĺňa uvedené podmienky.

1.3 Voľba šéfa

Voľba šéfa patrí medzi štandardné problémy riešené v distribuovaných systémoch. Príkladom ostatných problémov je atomický commit, konsenzus, vzajomné vylúčenie alebo replikácia. Cieľom algoritmu je, ako názov napovedá, z množiny procesorov zvoliť jeden procesor, ktorý bude "šéfom". Najprv uveďme formálnu definíciu algoritmu na voľbu šéfa:

Definícia 1. *Algoritmus voľby šéfa je algoritmus, ktorý spĺňa nasledovné podmienky*[1]*:*

1. *Na každom procesore sa vykonáva rovnaký algoritmus.*
2. *Algoritmus je decentralizovaný. Algoritmus môže byť iniciovaný s ľubovolnej neprázdnej podmnožiny procesorov.*
3. *Každý beh algoritmu skončí v terminálnej konfigurácii. A v každej terminálnej konfigurácii existuje práve jeden procesor v stave líder a ostatné procesory sú v stave porazený.*

Procesorom, ktorý ešte nespustili algoritmus, budeme hovoriť spiace. Procesory sa zvyčajne zobudia, keď sa im pošle prvá správa.

Niekedy sa vyžaduje, aby po skončení algoritmu, všetky porazené vrcholy poznali identitu šéfa alebo cestu k nemu. Ak algoritmus pri voľbe šéfa zároveň vybuduje kostru, môže ju využiť na notifikáciu porazených vrcholov. Prípadne zvoliť iný broadcast algoritmus. V tejto práci nebudeme túto podmienku vyžadovať.

1.3.1 Prečo volíme šéfa

Potreba voľby šéfa sa prirodzene vynára v distribuovaných systémoch, keď je vyžadovaná kooperácia alebo koordinácia procesov. Príkladom je už spomínaný atomický commit alebo problém vzájomného vylúčenia[15]. Často nie je možné určiť koordinátora dopredu, pretože ide o anonymnú sieť alebo sa vyžaduje, aby bol určený online v prípade porúch siete alebo zlyhania procesora. Často sa voľba šéfa spolu s najlacnejšou kostrou prezentuje ako prvý algoritmus na nových topológiach komunikačných sietí, aby sa prezentovala "očakávaná zložitosť" distribuovaných algoritmoch.

1.3.2 Predchádzajúce výsledky

Gallager a kolektív [8] popísali algoritmus na voľbu šéfa a minimálnu kostru na všeobecných grafoch, ktorý používa $O(N \log N + E)$ správ, kde N je počet vrcholov a E je počet hrán. Časová zložitosť tohto algoritmu môže byť v najhoršom prípade $O(N \log N)$. Algoritmus je optimálny vzhľadom na počet správ, ale časová zložitosť je suboptimálna o faktor $\log V$. Awerbuch v práci [2] publikoval algoritmus, ktorý je optimálny vzhľadom na počet správ aj potrebný čas. Príklad rýchleho algoritmu je popísaný v práci [4]. V prípade, že sú všetky procesory prebudené, dosahuje čas $\frac{n}{2} + O(1)$.

[1] Definícia je prebratá z [19].

Dolný odhad počtu správ pre orientované aj neorientované kruhy nájdeme v práci [5]. V priemernom prípade je potrebných $\Omega(N \log N)$ správ. Dolný odhad počtu správ pre kompletné grafy v prácach [10],[12].

1.4 Cayleho graf

Definícia 2. *Podmnožinu S grupy G nazveme generujúcou množinou, ak*

1. *Neobsahuje neutrálny prvok.*
2. *Je uzavretá na inverzné prvky.*

Definícia 3. *Nech G je konečná grupa a množina S je generujúca množina. Potom neorientovaný graf $\Gamma = (V, E)$ nazveme Cayleho graf, ak*

1. *Vrcholy grafu $V(\Gamma)$ sú jednoznačne priradené k prvkom grupy G.*
2. *V grafe Γ je hrana medzi vrcholmi v_1 a v_2 pravé vtedy, keď $v_1 = v_2 s$, kde s je prvok množiny S.*

Poznámka 1. *Ak cayleho graf nie je súvislý, sú jednotlivé komponenty súvislosti navzájom izomorfné. V tomto prípade budeme pracovať iba s jedným komponentom súvislosti. Preto v texte budeme zamieňať cayleho graf s jeho komponentom súvislosti.*

1.4.1 Cayleho graf ako komunikačná sieť

V práci budeme niekedy nazývať vrchol jeho kanonickým názvom. Kanonický názov vrchola je prvok grupy, ktorý reprezentuje. Napríklad, ak pôjde o grupu permutácii, vrcholy budeme nazývať pomocou permutácií. Kanonický názov budeme používať iba na popis algoritmu, ale sám algoritmus ho nebude používať. Podobne hrany budeme označovať generátormi, ktoré hranu vygenerovali. Toto označenie hrán bude využívať aj algoritmus.

Definícia 4. *Graf je vrcholovo symetrický, ak pre každú dvojicu vrcholov a a b, existuje automorfizmus grafu, ktorý zobrazí a na b.*

Inak povedané, pohľad z každého vrcholu na zvyšok grafu je rovnaký. Takýto graf umožňuje, aby bola komunikačná záťaž rovnomerne distribuovaná medzi všetky vrcholy.

Veta 1. *Každý cayleho graf je vrcholovo symetrický.*

Dôkaz. Dôkaz je v [1] □

Analogická vlastnosť pre hrany:

Definícia 5. *Graf je hranovo symetrický, ak pre každú dvojicu hrán a a b, existuje automorfizmus grafu, ktorý zobrazí hranu a na b.*

Veta 2. *Uvažujme Cayleho graf definovaný d generátormi na n symboloch. Cayleho graf je hranovo symetrický práve vtedy keď každá dvojica generátorov g_1 a g_2 existuje permutácia n symbolov, ktorý mapuje generátory na samých seba a g_1 na g_2.*

Dôkaz. Dôkaz sa dá nájsť v [1]. □

Definícia 6. *Graf, ktorý je vrcholovo aj hranovo symetrický, sa nazýva symetrický.*

1.4.2 Voľba šéfa a známe cayleho grafy

Prehľadový článok na tému cayleho grafy a voľba šéfa sa dá nájsť v [11].

- Hyperkocka Q_n je cayleho graf založený na grupe permutácii $\mathbb{S}_{2n}$. Množinu generátorov tvoria transpozície tvaru $(2i-1, 2i)$ pre i od 1 po n. Článok [18] popisuje algoritmus na voľbu šéfa, ktorý používa lineárny počet správ.

- Pancake graph P_n je cayleho graf založený na grupe permutácii $\mathbb{S}_n$. Množinu generátorov tvoria permutácie tvaru $k(k-1)\dots321(k+1)(k+2)\dots n$ pre k od 2 po n. Algoritmus v článku [13] s lineárnym počtom správ.

- Star graph S_n je tiež založený na grupe $\mathbb{S}_n$ a množina generátorov je daná transpozíciami tvaru $(1, i)$ pre i od 2 do n. V článku [17] bol publikovaný algoritmus používajúci lineárny počet správ.

- Bubblesort graph B_n Vrcholy sú z grupy $\mathbb{S}_n$. Generátory majú tvar $(i, i+1)$ pre i od 1 po $n-1$. Článok [11] uvádza, že je známy algoritmus používajúci $O(N \log N)$ správ.

V článku [3] Barrière popisuje algoritmus na voľbu šéfa v neorientovanom cayleho grafe stupňa 4.

Kapitola 2

Voľba šéfa a transpozičné grafy

2.1 Transpozičný graf

V tejto kapitole budeme skúmať cayleho grafy, ktoré sú založené na grupe permutácií S_n. Generujúcu množinu budú tvoriť iba transpozície. Takúto generujúcu množinu môžme popísať pomocou transpozičného grafu, ktorého definícia nasleduje:

Definícia 7. *Nech S je množina transpozícií z grupy $\mathbb{S}_n$. Transpozičný graf nazveme neorientovaný graf bez násobných hrán a slučiek, ktorého vrcholy označíme $1, 2, \ldots, n$ a vrcholy i a j sú spojené hranou práve vtedy, keď transpozícia (i, j) patrí do S.*

Ak je generujúca množina tvorená iba transpozíciami, budeme príslušný cayleho graf nazývať cayleho graf generovaný transpozičným grafom.

2.2 Transpozičný strom

Definícia 8. *Transpozičný graf, ktorý je strom, budeme nazývať transpozičný strom.*

Cayleho graf generovaný transpozičným stromom T_S, kde S je generujúca množina, je daný týmto stromom a číslom n, ktoré určuje grupu permutácií $\mathbb{S}_n$. Ak nebude uvedené inak budeme predpokladať že n je rovný počtu vrcholov stromu T_S.

2.2.1 Základné vlastnosti

Lema 1. *Graf generovaný transpozičným stromom s počtom vrcholov n má $n!$ vrcholov.*

Dôkaz. Dôkaz sa dá nájsť v [1]. □

Výhodou algebraického prístupu ku cayleho grafom je, že nám umožňuje určiť alebo odhadnúť niektoré vlastnosti pre celú triedu grafov. Takouto vlastnosťou je napríklad priemer grafu.

Predstavme si hru, kde každému vrcholu transpozičného stromu priradíme jednu značku z množiny $\{1, 2, \ldots, n\}$. Našou úlohou je dostať značky na svoje miesto. Vrcholy si môžu vymeniť značky ak sú spojené hranou. Keď vyriešime úlohu, usporiadame permutáciu, reprezentovanú rozmiestnením značiek na vrcholy.

Všimnime si, že cayleho graf tvorí stavový priestor spomínanej hry. Vrcholy zodpovedajú rôznym usporiadaniam značiek. Hrany reprezentujú povolené ťahy. Nájdenie cesty v cayleho grafe zodpovedá utriedeniu permutácie. Čo znamená nájdenie riešenia hry. Potom maximálny počet krokov na vyriešenie hry, zodpovedá priemeru cayleho grafu.

Veta 3. *Nech T je transpozičný strom na n vrcholoch. Označme priradenie značiek ako permutácia π vrcholov T. Permutácia π sa dá utriediť v $M(\pi)$ krokoch.*

$$M(\pi) = \eta(\pi) - n + \sum_{i=1}^{n} \delta(i, \pi(i))$$

$\eta(\pi)$ značí počet cyklov permutácie a $\delta(i, j)$ je vzdialenosť vrcholov i a j v strome T.

Dôkaz. Dôkaz je prebratý z [1]. Značkám, ktoré sú na svojom mieste budeme hovoriť, že sú v domčeku. Pre ne platí $i = \pi(i)$. V rámci dôkazu označme počet cyklov c a sumu S. Ak sú všetky značky v domčeku, tak cyklov je n a suma S je nulová. Počet krokov na usporiadanie je nulový, lebo $M(\pi) = c - n + S = n - n + 0 = 0$.

Ak m značiek nie je v domčeku, tak počet cyklov je aspoň $n - m + 1$. Zároveň však $S \geq m$, pretože m značiek je vzdialených od domčeka aspoň o jeden. Z toho vyplýva, že $M(\pi) = c - n + S \geq n - m + 1 - n + m = 1$

Teraz ukážeme, že $M(\pi)$ vieme zmenšiť na nulu po vykonaní najviac $M(\pi)$ transpozícií. Pre každú značku, ktorá nie je v domčeku, označme hranu šípkou od tohto vrchola smerom k domčeku. Vieme, že každý vrchol ma najviac jednu takúto hranu, pretože to je strom.

Všimnime si, že vieme nájsť dvojicu susedných vrcholov so značkami x a y takú, že x sa chce presunúť do vrcholu, kde je y a y je buď v domčeku alebo chce ísť na miesto x. Poznamenajme, že ak sú susedné značky x a y v jednom cykle, tak po ich výmene budú v dvoch rôznych cykloch a naopak ak boli v rôznych cykloch, potom budú v iba jednom.

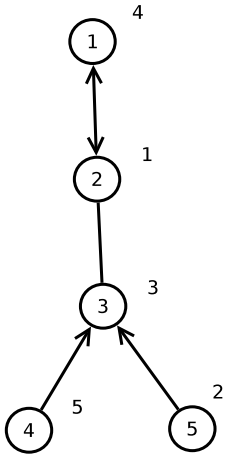
V prípade, že x a y si chcú navzájom vymeniť miesto, tak po vykonaní príslušnej transpozície sa zníži S o dva a zároveň sa zväčší počet cyklov najviac o jeden. Teda $M(\pi)$ sa zmenší aspoň o jeden. V druhom prípade je y už v domčeku. Po vykonaní výmeny zostane suma S nezmenená, pretože x sa priblíži o jedna a y sa vzdiali o jedna. Ale x a y museli byť v dvoch rôznych cykloch, pretože ak bol y v domčeku, tak y tvorilo cyklus dĺžky jedna. Po výmene sa znížil počet cyklov o jedna. □

Veta 4. *Zo všetkých stromov na n vrcholoch, cesta generuje cayleho graf s maximálny priemerom.*

Dôkaz. Dôkaz sa dá nájsť v [1]. □

Veta 5. *Priemer cayleho grafu generovaným ľubovolným stromom na n vrcholoch sa dá zhora ohraničiť pomocou výrazu $O(n^2)$.*

Dôkaz. Podľa vety 4 má bubble sort graf najväčší priemer medzi cayleho grafmi generovanými transpozičným stromom s počtom vrcholov n. Priemer bubble sort graf môžme odhadnúť pomocou $O(n^2)$ [1]. □



Obr. 2.1: Príklad inštancie hry na generujúcej množine $\{(1,2),(2,3),(3,4),(3,5)\}$

2.2.2 Dekompozičné usporiadanie

Definícia 9. *Dekompozičné usporiadanie f nazveme zobrazenie*

$$f : \{1, 2, \ldots, |S|\} \to S$$

ak

1. *Zobrazenie f je bijekcia.*

2. *Pre každé $i \in \{1, 2, \ldots, |S| - 1\}$, grupa generovaná množinou*

$$\bigcup_{l=1}^{i} \{f(l)\}$$

je vlastná podgrupa grupy generovanej množinou

$$\bigcup_{l=1}^{i+1} \{f(l)\}$$

Dekompozičné usporiadanie nám umožní postupne rozkladať cayleho graf na jeho komponenty. Najprv musíme ukázať, že také zobrazenie existuje. Majme danú generujúcu množinu S a príslušný transpozičný strom T. Vieme, že strom obsahuje vrchol so stupňom jeden. Označme ho v a vrchol, s ktorým je spojený u. Priradíme $f(1) = (v, u)$ a $T = T - \{v\}$. Postupne opakujeme až kým neodstránime všetky listy.

Algorithm 1 Konštrukcia dekompozičného usporiadania

```
T' ← T
i ← 1
while E(T') ≠ ∅ do
    v ← vrchol so stupňom jeden
    u ← susedný vrchol v
    f(i) ← (v, u)
    T' ← T' − {v}
    i ← i + 1
end while
```

Algoritmus 1 postupne odstraňuje listy zo stromu a tým postupne vytvára dekompozičné usporiadanie. Strom po odstránení listu zostane stromom. V každej iterácií s listom odstráni aj jednu hranu. Počet iterácií sa rovná počtu hrán v strome, to je $n - 1$. Žiadna transpozícia nie je priradená dvakrát a zároveň je priradená aspoň raz. Takto skonštruované zobrazenie je bijekcia.

Zostáva nám ukázať podmienku 2 z definície. Označme množinu $S_i = \{f(1), f(2), \ldots, f(i)\}$. Ľahko vidno, že grupa generovaná S_i je podgrupa grupy generovanej S_{i+1}, pretože S_i je podmnožina S_{i+1}. Z lemy 1 vyplýva, že grupa generovaná S_i má $i!$ prvkov a grupa generovaná S_{i+1} obsahuje $(i + 1)!$ prvkov. Preto sa tieto dve grupy nemôžu rovnať. Tým sme dokázali nasledujúcu vetu.

Veta 6. *Výstupom algoritmu 1 je dekompozičné usporiadanie generujúce množiny S s transpozičným stromom T.*

2.2.3 Dekompozícia

Na úvod uvedieme definíciu hierarchického cayleho grafu:

Definícia 10. *Cayleho graf nazveme hierarchický ak jeho generátory g_1, g_2, ..., g_n sa dajú usporiadať tak, že pre každé i také, že $1 \leq i \leq n$, je g_i mimo podgrupy generovanej g_1, g_2, ..., g_{i-1}.*

Definícia 11. *Cayleho graf nazveme silne hierarchický ak je hierarchický v ľubovoľnom usporiadaní generátorov.*

Graf, ktorý je hierarchický, sa dá rekurzívne dekomponovať. Príkladom silne hierarchického grafu je hyperkocka.

Ľahko vidno, že cayleho grafy generované transpozičným stromom sú hierarchické. Vyplýva to z existencie dekompozičného usporiadania. Je možné, že existuje aj iné usporiadanie generátorov ako to, ktoré získame pomocou algoritmu 1. Ale algoritmus 1 nám zaručuje, že časti na ktoré pôvodný graf dekomponujeme, sú len menšou inštanciou grafu z tej istej triedy. V našom prípade je to, že sú generované transpozičným stromom.

Ak máme daný cayleho graf pomocou transpozičného stromu T_S, tak počet generátorov v S je rovný počtu hrán v T_S. Nech n je počet vrcholov stromu T_S. Potom $|S| = n - 1$.

Označme f dekompozičné usporiadanie T_S. Teraz môžme zaviesť množinu S_i:

$$S_i = \bigcup_{j=1}^{i} f(j)$$

Teda množina S_i je podmnožinou S a obsahuje prvých i generátorov vzhľadom na dekompozičné usporiadanie f. Všimnime si, že množina S_i generuje grupu izomorfnú s grupou $\mathbb{S}_{i+1}$.

Podobne ako sme zaviedli označenie pre množinu S_i, ktorá pozostáva z prvých i generátorov, označme G_{i+1} grupu generovanú množinou S_i. Teraz môžme dekomponovať cayleho graf Γ_{n+1} pomocou rozkladu podľa podgrupy. Konkrétne máme na mysli ľavý rozklad grupy G_{n+1} podľa podgrupy G_n.

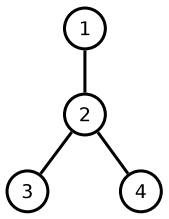
Veta 7. *Cayleho graf Γ_{n+1} obsahuje $n+1$ vrcholovo disjunktných podgrafov izomorfných s Γ_n.*

Dôkaz. Každá trieda rozkladu grupy G_{n+1} podľa G_n je izomorfná s G_n. Množina S_{n-1} generuje Γ_n. Triedy rozkladu sú navzájom disjunktné a z toho vyplýva, že aj podgrafy nimi indukované, sú vrcholovo disjunktné. Podľa vety 1 má graf Γ_n $n!$ vrcholov. Každá trieda rozkladu má rovnaký počet prvkov. Z toho vyplýva, že graf Γ_{n+1} obsahuje

$$\frac{(n+1)!}{n!} = n+1$$

vrcholov disjunktných podgrafov izomorfných s Γ_n. □

Dekompozíciu si ukážeme na príklade. Zoberme generujúcu množinu $S = \{(1,2),(2,3),(2,4)\}$ (Obrázok 2.2).Príslušný cayleho graf označme Γ_4. Dekompozičné usporiadanie je definované priamočiaro $f(1) = (1,2)$, $f(2) = (2,3)$ a $f(3) = (2,4)$. Množina S generuje grupu permutácií $\mathbb{S}_4$. Množina S_2 generuje jej podgrupu $\mathbb{S}_3$. Rozklad $\mathbb{S}_4$ podľa $\mathbb{S}_3$ má 4 triedy rozkladu - 4 kópie Γ_3. Triedy rozkladu demonštruje aj obrázok 2.3.



Obr. 2.2: Transpozičný strom

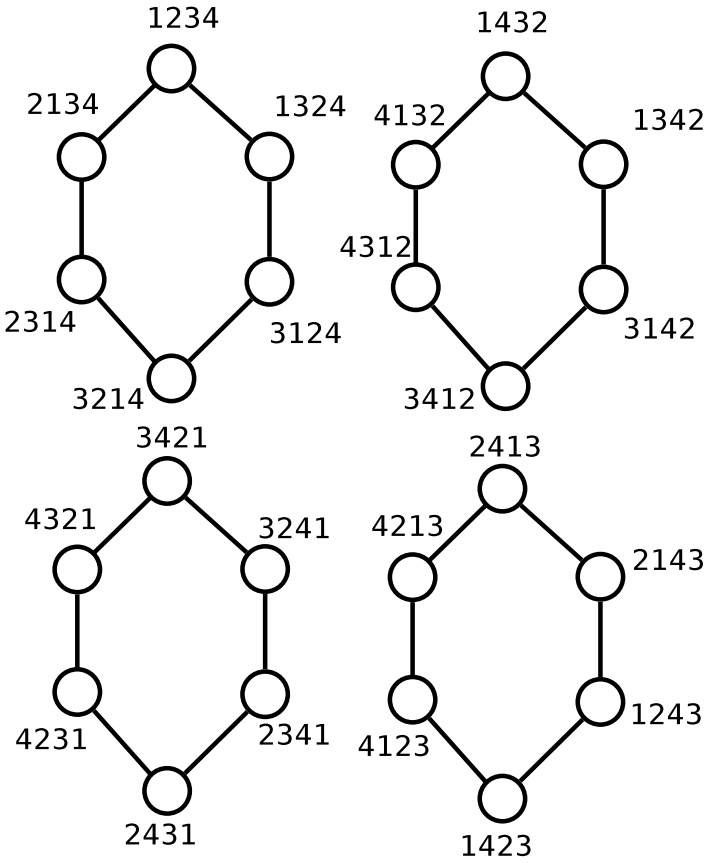
2.2.4 Voľba šéfa

V tejto časti ukážeme schému pre algoritmus na voľbu šéfa na grafoch generovaných transpozičným stromom. Využijeme jeho hierarchickú štruktúru. Majme daný cayleho graf Γ_{n+1} a generujúcu množinu $S = \{g_1, g_2, \ldots, g_n\}$ a dekompozičné usporiadanie f. Podľa vety 7 sa tento graf skladá z $n+1$ podgrafov Γ_n. Predpokladajme, že v každom podgrafe existuje práve jeden vrchol, ktorý je jedinečný. Jednou z možností, ako zvoliť šéfa celého grafu, je vybrať jeden z množiny jedinečných vrcholov. Ako ale určíme, jedinečný vrchol pre každý podgraf? Keďže každý z $n+1$ podgrafov je tiež cayleho graf generovaný transpozičným stromom a generujúcou množinou S/g_n, môžeme za jedinečný vrchol podgrafu označiť šéfa tohto podgrafu.

Náš algoritmus bude pracovať zdola nahor. Najprv sa zvolia šéfovia 1. úrovne, z nich sa zvolí šéfovia 2. úrovne až nakoniec sa zvolí šéf n. úrovne.

Definícia 12. *Šéfa i-tej úrovne nazveme vrchol, ktorý je šéf v rámci podgrafu indukovaným triedou rozkladu podľa podgrupy G_i.*

Veta 8. *Cayleho graf Γ_n generovaný transpozičným stromom s generujúcou množinou S obsahuje $\frac{n!}{i!}$ šéfov i-tej úrovne.*



Obr. 2.3: Dekompozícia cayleho grafu Γ_4

Dôkaz. Z definície 12 vyplýva, že existuje bijekcia medzi šéfmi i-tej úrovne a triedami rozkladu podľa podgrupy G_i. Podobne ako v dôkaze vety 7, počet vrcholov v podgrafe indukovanom grupou G_i je $i!$. Cayleho graf Γ_n má $n!$ vrcholov. Z toho počet šéfov i-tej úrovne je $\frac{n!}{i!}$. □

Algoritmus bude pracovať podla nasledujúcej schémy:

1. Každý vrchol je šéfom 1. úrovne.
2. Pre i od 1 po $n-1$ zvolíme spomedzi j šéfov i-tej úrovne $\frac{j}{i+1}$ šéfov $(i+1)$. úrovne.

Šéf na i-tej úrovni musí skontaktovať $i+1$ ostatných šéfov a porovnať si s nimi PID. Túto úlohu rozdelíme na 3 podúlohy:

1. Doručenie správy do susedných podgrafov: Šéf skontaktuje po jednom vrchole v každom susednom podgrafe, ktoré spolu s ním tvoria podgraf G_i
2. Doručenie správy šéfovi podgrafu: Vrcholy skontaktované v prvom kroku skontaktujú svojich šéfov.
3. Spätné doručenie správy: Šéfovia jednotlivých podgrafov spätne pošlú správu iniciátorovi matchmakingu[1].

2.2.5 Doručenie správy do susedného podgrafu

Definícia 13. *Nech vrchol x je šéfom i-tej úrovne. Množinu vrcholov $U_i(x)$ nazveme rôznorodá ak platí:*

1. *$x \in U_i(x)$.*
2. *Ak $y_1 \in U_i(x)$ a $y_2 \in U_i(x)$ a $y_1 \neq y_2$, potom y_1 a y_2 patria do rôznych tried rozkladu grupy G_{i+1} podľa G_i.*

Na splnenie tejto podúlohy potrebujeme doručiť správu do vrcholov, ktorých množina je rôznorodá. Pozrime sa na generujúcu množinu z predchádzajúceho príkladu $S_3 = \{(1,2),(2,3),(2,4)\}$. Všimnime si, že príslušná grupa generovaná touto množinou je $\mathbb{S}_4$. Čiže permutuje všetky prvky na všetkých pozíciach. Keď ale zostrojíme rozklad tejto grupy podľa podgrupy G_3, ktorú generuje množina $S_2 = \{(1,2),(2,3)\}$, tak jednotlivé triedy rozkladu sú síce izomorfné s G_3, ale permutovať medzi sebou môžeme iba prvky na 1.,2. a 3. pozícií. Prvok na štvrtej pozícií vždy zostane na svojom mieste. Analogicky,

[1] Pozri v [14]

keby sme zobrali iné dekompozičné usporiadanie, konkrétne $f(1) = (1,2)$, $f(2) = (2,4)$ a $f(3) = (2,3)$, tak trieda rozkladu by permutovala prvky na pozíciách 1,2 a 4 a pozícia 3 by bola fixovaná.

Definícia 14. *Definujme množinu* $X_i = \{k|(k,l) \in S_{i-1} \vee (l,k) \in S_{i-1}\}$. *Množinu* X_i *budeme nazývať množinou permutovaných pozícií.*

Teraz môžme načrtnúť ako bude vyzerať množina $U_i(x)$. Šéf x patrí do grupy G_{i+1} a zároveň do nejakej triedy rozkladu podľa G_i. Použitím liniek 1 až $i-1$ sa dá správa doručiť iba v rámci podgrafu Γ_i. Až použitím linky i, ktorá zodpovedá transpozícií $f(i)$, sa dá správa doručiť v rámci podgrafu Γ_{i+1}.

Triedy rozkladu sa medzi sebou líšia prvkom na fixovanej pozícií. Bez ujmy na všeobecnosti predpokladajme, že transpozícia $f(i)$ má tvar (k,l) a $l \notin X_i$. Predpokladajme, že vrchol má na k-tej pozícií v permutácií 2. Potom po linke i môže poslať správu do triedy rozkladu, ktorá ma na fixnej pozícií dvojku. My ale potrebujeme poslať správu do všetkých susedných podgrafov. To ľahko dosiahne postupným presunutím všetkých prvkov na pozíciu k.

Veta 9. *Nech* x *je šéf* i*-tej úrovne a* $f(i)$ *má tvar* (k,l), *kde* $l \notin X_i$. *Potom množina* $U = \{x * (j,k) * (k,l) | j \in X_i\} \cup \{x\}$ *je rôznorodá.*

Dôkaz. Prvá podmienka rôznorodosti je splnená z definície množiny U.

Druhú podmienku dokážeme sporom. Vieme[2], že $aH = bH$ pravé vtedy, keď $b^{-1}a \in H$. Z toho vyplýva

$$(x(j_2,k)(k,l))^{-1}(x(j_1,k)(k,l)) \in G_i$$

pričom $j_1 \neq j_2$. Upravme ďalej tento výraz

$$\begin{aligned}(k,l)(j_2,k)x^{-1}x(j_1,k)(k,l) &= (k,l)(j_2,k)(j_1,k)(k,l)\\ &= (k,l)(j_2,k)(j_1,k)(k,l)\\ &= (l,j_2,j_1)\\ &= (l,j_1)(l,j_2)\end{aligned}$$

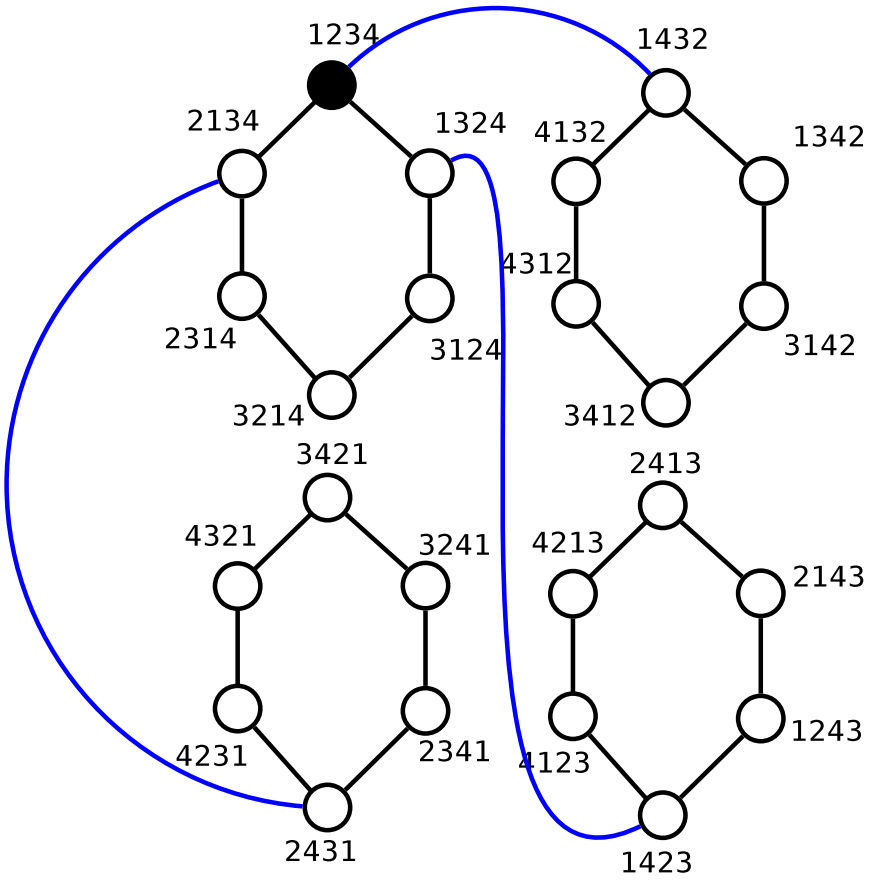
teda $(l,j_1)(l,j_2) \in G_i$, ale to je v spore s predpokladom $l \notin X_i$.

Analogicky ukážeme, že

$$x^{-1}x(j,k)(k,l) \notin G_i$$

Ak by $(j,k)(k,l) \in G_i$ potom by l muselo patriť do X_i, čo je spor. □

[2]Vetu a jej dôkaz môžme nájsť napríklad v [9]



Obr. 2.4: Kontaktovanie susedných podgrafov

Prvok $U_3(x)$	Zoznam použitých liniek	Postupnosť vrcholov
x		1234
$x(1,2)(2,4)$	$(1,2),(2,4)$	$1234, 2134, 2431$
$x(2,2)(2,4)$	$(2,4)$	$1234, 1432$
$x(3,2)(2,4)$	$(2,3),(2,4)$	$1234, 1324, 1423$

Tabuľka 2.1: Prvky množiny $U_3(x)$

Na obrázku 2.4 vidieť príklad kontaktovania susedných podgrafov. Vyplnení vrchol predstavuje šéfa 3. úrovne. Množina permutovaných pozícii vyzerá takto $X_3 = \{1,2,3\}$. A $f(3)$ je transpozícia tvaru $(2,4)$. Rôznorodá množina pre tento prípad je definovaná nasledovne:

$$U_3(x) = \{x(j,2)(2,4)|j \in X_3\} \cup \{x\}$$

Kanonický názov vrchola x je 1234. Prvky množiny $U_3(x)$ zobrazuje tabuľka 2.1. Vrchol x, použitím linky 3, priamo pošle správu do susedného podgrafu. Prostredníctvom vrcholov 2134 a 1324 a následným použitím linky 3, je šéf schopný poslať správu aj do zvyšných dvoch podgrafov.

Zostáva vyriešiť ako získať postupnosť liniek, ktorá zodpovedá transpozícií (i,j). Tu si pomôžeme transpozičným stromom. Cesta v transpozičnom strome medzi vrcholmi i, j, zodpovedá postupnosti transpozícii, ktorých súčinom je transpozícia (i,j). Algoritmus 2 pre každý vrchol z rôznorodej množiny vypočíta túto postupnosť a pošle správu po linke, ktorá zodpovedá prvej transpozícii z postupnosti. V algoritme 3 príjemca správy odhodí prvú transpozíciu z postupnosti a ak zostane neprázdna, tak pošle správu ďalej po príslušnej linke. Ak postupnosť zostala prázdna, tak príjemca je konečným príjemcom obsahu správy.

Algorithm 2 $sendByTransposition(i,j,M)$

```
1: list ← findPathInTrasposition Tree(i, j)
2: if list = ∅ then
3:     send(0, M, ∅)
4:     return
5: else
6:     link ← head(list)
7: end if
8: send(link, ForwardMessage(list, M), ∅)
```

Algorithm 3 $OnReceiveForwardMessage(list, M, traveledLinks)$

```
1: list ← tail(list)
2: if list = ∅ then
3:     send(0, M, traveledLinks)
4: else
5:     send(link, ForwardMessage(list, M), traveledLinks)
6: end if
```

2.2.6 Doručenie správy šéfovi podgrafu

Už vieme poslať správu do množiny vrcholov $U_i(x)$. Jednotlivé vrcholy teraz musia doručiť správu do svojho šéfa i-tej úrovne. Problémom je, že daný vrchol nemusí vedieť, kto je tým šéfom. Preto budeme požadovať, aby si každý vrchol pamätal najkratšiu cestu k svojmu šéfovi úrovne o jedna väčšej. Ak vrchol u z množiny $U_i(x)$ je šéfom na úrovni j, vieme sa dostať k šéfovi na úrovni $j+1$, odtiaľ ku $j+2$ až k šéfovi na úrovni i. V najhoršom prípade vrchol u prehral hneď na začiatku a bude šéfom nultej úrovne.

Algorithm 4 $OnReceiveForwardToLeaderMessage(targetLevel, M, traveledLinks)$

```
1: if targetLevel = currentLevel then
2:     send(0, M, traveledLinks)
3:     return
4: else
5:     path ← getPathToCurrentLeader()
6:     link ← head(path)
7:     leaderMessage ← ForwardToLeaderMessage(targetLevel, M)
8:     message ← ForwardMessage(path, leaderMessage)
9:     send(head(path), message, traveledLinks)
10: end if
```

Algoritmus 4 popisuje spôsob doručenia správy z vrcholu u do šéfa i-tej úrovne. Na preposielanie správy medzi šéfom j-tej úrovne do šéfa $(j+1)$-tej úrovne sa používa správa typu $ForwardMessage$ podobne ako pri doručovaní správy do susedného podgrafu.

2.2.7 Spätné doručenie správy

Keď šéf i-tej úrovne dostane správu od šéfa rovnakej úrovne zo susedného podgrafu, správa obsahuje zoznam liniek, ktoré boli použité pri preposielaní správy. Potom pomocou usporiadanie permutácie vie nájsť cestu k pôvodnému odosielateľovi.

Zoznam liniek po ktorých putovala správa predstavuje permutáciu π. Podobne ako v dôkaze vety 3 priradíme vrcholom transpozičného stromu značky. Vrchol v bude mať značku $\pi(v)$. Naším cieľom je premiestniť značky na svoje miesto, to je tak aby platilo $v = \pi(v)$. Značky môžeme premiestňovať iba výmenou medzi dvoma susednými vrcholmi.

Algorithm 5 Utriedenie permutácie π pomocou množiny transpozícii

```
1: T ← transpozičný strom
2: π ← permutácia
3: resultList ← nil
4: while existuje značka, ktorá nie je na svojom mieste do
5:    while existuje list, ktorý má správnu značku do
6:       v ← list, ktorý má značku v domčeku, t.j v = π(v)
7:       T ← T − {v}
8:    end while
9:    u ← list, ktorý nemá svoju značku
10:   v ← vrchol pre ktorý platí π(v) = u
11:   list ← findPathInTrasposition Tree(v, u)
12:   π ← π ∗ list
13:   resultList ← resultList + list
14: end while
```

Výstupom algoritmu 5 je postupnosť transpozícii z množiny generátorov, ktoré utriedia permutáciu π. Algoritmus v každom kole vonkajšieho cyklu umiestni aspoň jednu značku na svoje miesto. Zároveň túto značku aj s príslušným vrcholom odstráni zo stromu. To znamená, že táto značka zostane na svojom mieste až do konca algoritmu. V i-tom opakovaní cyklu máme aspoň i značiek v domčeku. Po n-tom opakovaní musia byť všetky značky na svojom mieste.

Pomocou výstupu algoritmu 5 vieme doručiť správu *TournamentReply* vyzývateľovi. Tá bude obsahovať PID vrchola. Vyzývateľ, šéf i-tej úrovne počká na $i+1$ správ *TournamentReply* a podľa nich vyhodnotí, či sa stane šéfom $i+1$ úrovne. Ak prehrá, zapamätá si cestu k vrcholu s najväčším PID. Ak vyhrá pokračuje vo vyzývaní ostatných šéfov úrovne $i+1$ pomocou správy *Tournament*.

2.2.8 Korektnosť algoritmu

Veta 10. *Algoritmus zvolí šéfa grafu* Γ_n.

Dôkaz. Musíme ukázať, že vrchol, ktorý bol zvolený algoritmom je jediný vrchol v stave líder. Budeme postupovať indukciou vzhľadom na úroveň šéfa.

$i = 1$ Každý vrchol je šéfom 1. úrovne v rámci podgrafu indukovanom triedou rozkladu podľa grupy G_1, pretože tento podgraf obsahuje iba jeden vrchol.

IP Máme zvolených $i + 1$ šéfov i-tej úrovne. Príslušné podgrafy spolu tvoria podgraf indukovaný triedou rozkladu podľa podgrupy G_{i+1}. V predchádzajúcom texte sme ukázali, že si títo šéfovia medzi sebou vymenia svoj PID. Každý z nich má k dispozícií rovnakú konečnú podmnožinu totálne usporiadanej množiny. Táto podmnožina má jediný najväčší prvok a preto výber šéfa $i + 1$. úrovne z pomedzi $i + 1$ šéfov i-tej úrovne je jednoznačný.

Všimnime si, že algoritmus neskonči skôr ako sa zvolí šéf celého podgrafu. V každom kole sa vyberie aspoň jeden nový šéf a ten spustí ďalšie kolo výpočtu. □

2.2.9 Celkový počet odoslaných správ

Veta 11. *Nech T je transpozičný strom z množiny generátorov S_i. Výstupom algoritmu 5 je postupnosť dĺžky najviac $O(i^2)$.*

Dôkaz. Algoritmus v každom opakovaní vonkajšieho cyklu umiestni značku na svoje miesto. Pri tom vygeneruje postupnosť transpozícii, ktoré použil. Dĺžka tejto postupnosti nie je väčšia ako priemer transpozičného stromu. Zároveň v každej iterácií odstráni najmenej jeden list. Preto ak transpozičný strom má na začiatku algoritmu $i+1$ vrcholov potom postupnosť transpozícii bude mať dĺžku l, kde

$$l \leq \sum_{j=1}^{i+1}(j-1) = \sum_{j=0}^{i} j = O(i^2)$$

□

Veta 12. *Šéf i-tej úrovne vie skontaktovať šéfa rovnakej úrovne zo susedného podgrafu použitím $O(i^3)$ správ.*

Dôkaz. Dôkaz rozdelíme na 3 časti, rovnako ako opis algoritmu.

- Doručenie správy do vrchola z množiny $U_i(x)$. Procedúra *sendByTransposition* naplánuje cestu podľa cesty v transpozičnom strome. Každá linka v postupnosti predstavuje jednu správu. Transpozičný strom je určený množinou S_{i-1}. Tento strom má i vrcholov a jeho priemer je i. Na konci použijem ešte linku $f(i)$. To je najviac i správ.

- Doručenie správy svojmu šéfovi i-tej úrovne. Predpokladajme, že vrchol u je šéf k-tej úrovne. Vrchol u pozná cestu k šéfovi úrovne $k+1$. Podľa vety 11 má dĺžku najviac $O(i^2)$. Celkový počet správ bude

$$\sum_{j=k}^{i-1} O(j^2) \leq \sum_{j=0}^{i} O(j^2) = O(i^3)$$

- Spätné doručenie správy tiež používa algoritmus 5 na naplánovanie cesty. Počet odoslaných správ je $O(i^2)$.

Celkový počet správ pre šéfa i-tej úrovne je $O(i^3)$. □

Veta 13. *Algoritmus pri voľbe šéfa na grafe Γ_n generovaný transpozičným stromom T použije $O(n!)$ správ.*

Dôkaz. Šéf na úrovni i kontaktuje $i+1$ šéfov rovnakej úrovne. Spolu šéf i-tej úrovne priamo či nepriamo vygeneruje $O(i^4)$ správ. Šéfov na úrovni i v grafe Γ_n je $\frac{n!}{i!}$. Celkový počet správ bude

$$\sum_{i=2}^{n} \frac{n!}{i!} O(i^4) = n! O(\sum_{i=2}^{n} \frac{i^4}{i!}) = n! O(1) = O(n!)$$

□

2.2.10 Časová zložitosť

Podobne ako v prácach [18] a [17], budeme predpokladať, že všetky uzly(vrcholy) spustia algoritmus nezávisle a posledný uzol začne v čase t_0.

Veta 14. *Algoritmus zvolí šéfa v grafe Γ_n generovanom transpozičným stromom v čase $O(n^4)$.*

Dôkaz. Predpokladajme, že šéfovia i-tej úrovne sú zvolený v čase t_i. Kontaktovanie susedných šéfov prebieha paralelne, stačí nám spočítať koľko času trvá matchmaking s jedným susedom.

- Doručenie správy do susedného podgrafu trvá $O(i)$ krokov.
- Doručenie správy svojmu šéfovi trvá $O(i^3)$ krokov.
- Spätné doručenie správy trvá $O(i^2)$ krokov.

Celkový čas bude:

$$t_n = t_0 + \sum_{i=1}^{n-1} O(i^3) = t_0 + O(n^4)$$

□

2.3 Transpozičný les

Definícia 15. *Transpozičný graf, ktorý je les, nazveme transpozičný les.*

Les pozostáva z k stromov. Označme jednotlivé stromy $T_1, T_2, \ldots, T_k$. Počet vrcholov jednotlivých stromov označme n_i pre $i = 1, 2, \ldots, k$. Ak $k > 1$, tak príslušný cayleho graf je nesúvislý. Podľa poznámky 1 sú jednotlivé komponenty izomorfné.

2.3.1 Základné vlastnosti

Veta 15. *Komponent súvislosti cayleho grafu, ktorý je generovaný transpozičným lesom s k komponentami, má*

$$N = \prod_{i=1}^{k} (n_i)!$$

vrcholov.

Dôkaz. Budeme postupovať indukciou vzhľadom na k:

$k = 0$ Generujúca množina je prázdna, teda cayleho graf neobsahuje žiadne hrany. Z toho vyplýva, že každá komponenta súvislosti má práve jeden vrchol.

$k = 1$ Dôsledok vety 1.

IP Každá hrana v transpozičnom lese predstavuje transpozíciu v generujúcej množine a každá transpozícia generuje množinu hrán v príslušnom cayleho grafu. Označme všetky hrany v cayleho grafe, ktoré boli vygenerované hranou zo stromu T_k, červenou farbou a ostatné hrany modrou.

Jednoduché pozorovanie je, že keď vynecháme červené hrany, dostaneme cayleho graf generovaný stromami $T_1, T_2, \ldots, T_{k-1}$. Naopak ak vynecháme modré hrany dostane cayleho graf generovaný stromom T_k.

Modrým komponentom budeme nazývať komponent súvislosti grafu, ktorý má odstránené červené hrany. Analogicky definujeme červený komponent. Modrá cesta je cesta pozostávajúca iba z modrých hrán. Červená cesta je cesta z červených hrán.

Ak medzi vrcholmi existuje modrá cesta tak medzi nimi neexistuje červená. Ak by existovali obe, znamenalo by to, že existujú dve rôzne permutácie π_1 a π_2 také, že $x\pi_1 = x\pi_2$. Ale v grupe môžme krátiť zľava, potom $\pi_1 = \pi_2$, čo je spor.

Zoberme si ľubovolný vrchol, ten súčasne patrí do nejakého modrého komponentu aj do nejakého červeného. Červený komponent obsahuje podľa indukčného predpokladu $n_k!$ vrcholov. Modrý komponent má $\prod_{i=1}^{k-1}(n_i)!$ vrcholov, pričom každý patrí do nejakej červeného komponentu. Tieto komponenty sú ale navzájom rôzne. Z toho vyplýva, že

$$N \geq \prod_{i=1}^{k}(n_i)!$$

Musíme sa ešte presvedčiť, či to nie je viac. S každým vrcholom v z modrého komponentu M_1 zarátame celý červený komponent C_1 do ktorého v patrí. Vrcholy z červeného komponentu sú spojené modrou hranou aj s inými vrcholmi. Ukážeme, že tieto vrcholy patria do červeného komponentu C_2, ktorý je spojený s vrcholom u z modrého komponentu M_1 a teda už ho máme zarátaný.

Vrchol v je z modrého komponentu M_1 a zároveň z červeného komponentu C_1. Nech vrchol w je z červeného komponentu C_1. Existuje červená cesta medzi vrcholmi v a w, označme ju permutáciou π_1. Zoberme ľubovolný vrchol u, ktorý je spojený modrou cestou, označme ju π_2, s vrcholom w. Potom existuje cesta medzi vrcholmi v a u a má tvar $\pi_1\pi_2$. Keďže permutácie π_1 a π_2 sú disjunktné, existuje aj druhá cesta tvaru $\pi_2\pi_1$. Vrchol $v\pi_2$ je z modrého komponentu M_1 a s ním je zarátaný aj vrchol u. Z toho

$$N = \prod_{i=1}^{k}(n_i)!$$

□

Poznámka 2. *Aby sme sa vyhli opakovanému zdôrazňovaniu, že nehovoríme o celom cayleho grafe, ale iba o jednej jeho komponente súvislosti, nazveme ju β_S.*

Nasledujúca veta hovorí o priemere cayleho grafu generovaného transpozičným lesom:

Veta 16. *Priemer cayleho grafu generovaného transpozičným lesom je $O(n^2)$, kde*

$$n = \sum_{i=1}^{k} n_i$$

Dôkaz. Horné ohraničenie vyplýva z algoritmu 7 a vety 21. □

2.3.2 Dekompozičné usporiadanie

Dekompozičné usporiadanie sme definovali v definícií 9. Označme volanie algoritmu 1 na strome T ako volanie funkcie $decomposeTree(T)$.

Algorithm 6 Konštrukcia dekompozičného usporiadania pre transpozičný les

```
1: base ← 0
2: for i = 1 to k do
3:    h ← decomposeTree(T_i)
4:    for j = 1 to n_i − 1 do
5:       f(base + j) = h(j)
6:    end for
7:    base = base + n_i − 1
8: end for
```

Algoritmus 6 postupne usporiada generátory z množiny S. Prvých n_1 generátorov tvorí strom T_1, nasledujúcich n_2 generátorov tvorí strom T_2 a tak ďalej. V rámci stromu T_i je usporiadanie určené algoritmom 1.

Kvôli jednoduchosti vyjadrovania zaveďme ešte dve pomocné funkcie. Funkcie t aj s sú $\mathbb{N} \to \mathbb{N}$. Zobrazenie $t(i)$ vráti číslo stromu, do ktorého patrí transpozícia $f(i)$. Zobrazenie $s(i)$ vráti poradové číslo transpozície $f(i)$ v rámci stromu.

Musíme preveriť ešte vlastnosť dekompozičného usporiadania, že grupa generovaná množinou S_i je vlastná podgrupa grupy generovanej množinou S_{i+1}. Pripomeňme, že množina S_i je definovaná takto:

$$S_i = \bigcup_{l=1}^{i} \{f(l)\}$$

S_i je podmnožina S_{i+1}. Rozdiel množín S_{i+1} a S_i je transpozícia (m, n). Platí, že transpozícia (m, n) nepatrí do grupy generovanej množinou S_i, pretože by v transpozičnom lese vznikol cyklus. Ale transpozícia (m, n) patrí do grupy generovanej množinou S_{i+1}.

2.3.3 Dekompozícia

Cayleho grafy generované transpozičným lesom sú hierarchické. Poradie generátorov, vyžadované v definícií 10 je dané dekompozičným usporiadaním

f. Označme symbolom G_i[3] grupu generovanú množinou S_i. Podobne ako v prípade transpozičných stromov, existujú aj iné usporiadania generátorov. Dekompozičné usporiadanie f má nasledujúce vlastnosti:

1. Transpozičné grafy zadané množinou generátorov S_i sú lesy.
2. S rastúcim i neklesá počet komponentov súvislosti v transpozičnom lese. Inak povedané pridaním transpozície $f(i+1)$ do množiny S_i nespojíme dva stromy do jedného.

Symbol β_S bude označovať cayleho graf generovaný transpozičným lesom z množiny transpozícii S. Nech $S_n = S$ potom grupu permutácii generovanú množinou S_n označíme G_n.

Veta 17. *Nech β_{S_j} je cayleho graf generovaný transpozičným lesom. Transpozičný les pozostáva so stromov $T_1, T_2, \ldots, T_k$ a strom T_k obsahuje n_k vrcholov. Potom graf β_S sa skladá z n_k vrcholovo disjunktných podgrafov izomorfných s grafom $\beta_{S_{j-1}}$.*

Dôkaz. Veta je obdobou vety 7. Trieda rozkladu grupy G_j podľa podgrupy G_{j-1} je izomorfná s grupou G_{j-1}. Množina S_{j-1} generujúca grupu G_{j-1} je generujúca množina grafu $\beta_{S_{j-1}}$. Keďže triedy rozkladu sú navzájom disjunktné, tak aj podgrafy nimi indukované, sú vrcholovo disjunktné.

Môžeme predpokladať, že strom T_k obsahuje aspoň dva vrcholy. Transpozičný les nemôže obsahovať strom, ktorý je tvorený izolovaným vrcholom, pretože každá transpozícia v množine S_j pridáva vrchol a hranou ho pripája do stromu, alebo pridáva dva vrcholy spojené hranou.

Podľa vety 15 graf β_{S_j} obsahuje

$$(\prod_{i=1}^{k-1} n_i!) * n_k!$$

vrcholov. Analogicky graf $\beta_{S_{j-1}}$ má

$$(\prod_{i=1}^{k-1} n_i!) * (n_k - 1)!$$

vrcholov. Triedy rozkladu grupy G_j podľa G_{j-1} majú rovnaký počet prvkov ako grupa G_{j-1}. Potom rozklad obsahuje

$$\frac{\prod_{i=1}^{k-1} n_i!}{\prod_{i=1}^{k-1} n_i!} * \frac{n_k!}{(n_k-1)!} = n_k$$

[3]Narozdiel od grupy G_i používanej pri voľbe na cayleho grafe generovanom transpozičným stromom, dolný index neurčuje počet permutovaných symbolov, ale mohutnosť množiny generátorov.

tried. □

2.3.4 Voľba šéfa

Aj v tomto prípade budeme postupne voliť šéfa zdola nahor. Zavedieme pojem šéfa i-tej úrovne pre cayleho graf generovanom transpozičným lesom.

Definícia 16. *Šéfom i-tej úrovne nazveme vrchol, ktorý je šéf v rámci podgrafu indukovanom triedou rozkladu G_i.*

Veta 18. *Graf β_{S_n} obsahuje*

$$\frac{\prod_{j=t(i)}^{k} n_j!}{(s(i)+1)!}$$

šéfov i-tej úrovne.

Dôkaz. Podľa definície 16 šéfov i-tej úrovne môžme jednoznačne priradiť k triedam rozkladu podľa podgrupy G_i. Podobne ako v dôkaze vety 17 určíme počet prvkov rozkladu. Grupa G_i je generovaná množinou S_i. Množina S_i obsahuje prvých i generátorov. Ak funkcia $t(i)$ vracia číslo stromu do ktorého patrí transpozícia $f(i)$, tak potom do S_i patria všetky transpozície, ktoré tvoria stromy $T_1, T_2, \ldots, T_{t(i)-1}$ a počet vrcholov jednotlivých stromov je $n_1, n_2, \ldots, n_{t(i)-1}$. Funkcia $s(i)$ určuje počet transpozícii, ktoré patria do S_i a zároveň tvoria strom $T_{t(i)}$. Takýto podstrom s $s(i)$ hranami obsahuje $s(i)+1$ vrcholov. Použitím vety 15 dostaneme počet vrcholov patriacich do podgrafu indukovanom triedou rozkladu:

$$(\prod_{j=1}^{t(i)-1} n_j!) * (s(i)+1)!$$

Počet šéfov i-tej úrovne dostaneme ako pomer celkového počtu vrcholov grafu β_S a počet vrcholov podgrafu:

$$\frac{\prod_{j=1}^{k} n_j!}{\prod_{j=1}^{t(i)-1} n_j!} * \frac{1}{(s(i)+1)!} = \frac{\prod_{j=t(i)}^{k} n_j!}{(s(i)+1)!}$$

□

Algoritmus bude pracovať podľa schémy:

1. Každý vrchol je šéfom 0-tej úrovne.

2. Z množiny šéfov i-tej úrovne zvolíme menšiu množinu šéfov $(i+1)$. úrovne.

Šéf i-tej úrovne musí skontaktovať $s(i)+2$ šéfov rovnakej úrovne. Podobne ako v podkapitole 2.2.4 rozdelíme úlohu na tri podúlohy:

1. Doručenie správy do susedného podgrafu
2. Doručenie správy šéfovi podgrafu
3. Spätné doručenie správy

2.3.5 Doručenie správy do susedného podgrafu

Šéf i-tej úrovne vie pomocou liniek $1, 2, \ldots, f(i)$ komunikovať v rámci podgrafu β_{S_i}. Linka $f(i+1)$ umožňuje odosielanie správ do susedných podgrafov β_{S_i}. Na doručenie správ do rôznych susedných podgrafov použijeme definíciu rôznorodej množiny zo strany 22. Veľkosť množiny $U_i(x)$ je daná počtom tried rozkladu grupy G_{i+1} podľa podgrupy G_i.

Na rozdiel od prípadu generovania transpozičným stromom, v tomto prípade sa nepermutujú všetky pozície, ktoré sa vyskytnú v transpozíciách z množiny S_i. Predpokladajme, že transpozícia $f(i+1)$ má tvar (k,l) pričom l je číslo vrcholu, ktorý je list na strome $T_{t(i+1)}$ v transpozičnom lese generovanom množinou S_{i+1}. Potom dva vrcholy, ktoré chcú doručiť správu do susedného podgrafu pomocou linky (k,l), musia mať na k-tom mieste v kanonickom názve rôzne prvky. Ak šéf i-tej úrovne chce doručiť správu do dvoch vrcholov, ktoré majú rôzne prvky na k-tom mieste, stačí mu použiť transpozície, ktoré tvoria strom $T_{t(i+1)}$, pretože transpozície, ktoré tvoria ostatné stromy lesa, generujú grupu, ktorá nepermutuje prvky na pozícií k.

Označme $S^{(i)}$ množinu transpozícii, ktoré tvoria strom T_i. Potom platí

$$\bigcup_{i=1}^{k} S^{(i)} = S$$

Definícia 17. *Množinu*

$$X_i = \begin{cases} \{k|(k,l) \in S_i \cap S^{(t(i))} \vee (l,k) \in S_i \cap S^{(t(i))}\} & ak\ t(i) = t(i+1) \\ \{k|(k,l) = f(i+1)\} & inak \end{cases}$$

budeme nazývať množinou permutovaných pozícií.

Ľahko vidno, že aj takto definovaná množina permutovaných pozícií zachová platnosť vety 9. Šéf x i-tej úrovne použitím vety 9 zostrojí postupnosť liniek na ceste od neho do vrcholov množiny $U_i(x)$. Potom pomocou procedúry *sendByTransposition* uvedenej na strane 25 vykoná doručenie správy.

2.3.6 Doručenie správy šéfovi podgrafu

Vrchol v z množiny $U_i(x)$ musí doručiť správu svojmu šéfovi i-tej úrovne. Každý vrchol si bude pamätať cestu k šéfovi, ktorý ho porazil. Ak vrchol v bol šéf úrovne l, potom pozná cestu k šéfovi úrovne $l+1$, ten pozná cestu k šéfovi úrovne $l+2$ a tak ďalej. Táto podúloha má identické riešenie ako je uvedené v podkapitole 2.2.6.

2.3.7 Spätné doručenie správy

Rovnako ako v predchádzajúcom algoritme na voľbu šéfa, keď správa od vrcholu x dorazí do susedného šéfa i-tej úrovne, nesie v sebe aj zoznam liniek po ktorých prešla. Úlohou príjemcu je usporiadať permutáciu, ktorá je daná tým to zoznamom a poslať po takto skrátenej ceste vyzývateľovi správu $TournamentReply$.

Veta 19. *Skladanie disjunktných transpozícii je komutatívne.*

Dôsledkom vety 19 je, že postupnosť transpozícii z množiny S, môžeme preusporiadať tak, aby prvých m_1 transpozícii bolo z množiny $S^{(1)}$, ďalších m_2 transpozícii z množiny $S^{(2)}$ a tak ďalej. Permutácia π je súčin transpozícii z množiny S. Zaveďme značku pre každý vrchol z transpozičného grafu. Vrchol u bude mať značku $\pi(u)$. Úloha je dostať značky na svoje miesto. Využijeme algoritmus 5. Spustíme ho na každom strome zvlášť a spojíme postupnosti za sebou do jednej postupnosti. Algoritmus 5 na každom strome umiestni všetky značky na svoje miesto, pretože každá značka ma svoje miesto v tom istom strome ako bola umiestnená na začiatku algoritmu. Ak by to tak nebolo, znamenalo by to, že v ceste bola transpozícia, ktorá nepatrí do S. Označme algoritmus 5 ako funkcia $sortPermTree(T, \pi)$, ktorá vráti postupnosť transpozícii, pomocou ktorých je možne premiestniť značky na svoje miesto na strome T.

Algorithm 7 Usporiadanie permutácie π

```
1: list ← nil
2: for i = 1 to k do
3:     list ← list + sortPermTree(T_i, π)
4: end for
```

2.3.8 Korektnosť algoritmu

Veta 20. *Algoritmus zvolí šéfa grafu β_S.*

Dôkaz. Musíme ukázať, že vrchol, ktorý bol zvolený algoritmom je jediný vrchol v stave líder. Ukážeme, že po každom kole algoritmu sa udržuje invariant, že šéf i-tej úrovne je jediný vrchol v stave líder v rámci svojho podgrafu indukovanom triedou rozkladu podľa podgrupy G_i.

$i = 1$ Každý vrchol je šéfom 0. úrovne v rámci podgrafu indukovanom triedou rozkladu podľa grupy G_0, pretože tento podgraf obsahuje iba jeden vrchol.

IP Máme zvolených $s(i) + 2$ šéfov i-tej úrovne. Príslušné podgrafy spolu tvoria podgraf indukovaný triedou rozkladu podľa podgrupy G_{i+1}. V predchádzajúcom texte sme ukázali, že si títo šéfovia medzi sebou vymenia svoj PID. Každý z nich má k dispozícií rovnakú konečnú podmnožinu totálne usporiadanej množiny. Táto podmnožina má jediný najväčší prvok a preto výber šéfa $i + 1$. úrovne z pomedzi $i + 1$ šéfov i-tej úrovne je jednoznačný.

Všimnime si, že algoritmus neskonči skôr ako sa zvolí šéf celého podgrafu. V každom kole sa vyberie aspoň jeden nový šéf a ten spustí ďalšie kolo výpočtu, pokiaľ nie je šéfom celého grafu, teda kým $i < |S|$. □

2.3.9 Celkový počet odoslaných správ

Veta 21. *Nech L je transpozičný les z množiny generátorov S_i. Nech počty vrcholov jednotlivých stromov lesu L sú $n_1, n_2, \ldots, n_k$. Potom výstupom algoritmu 7 je postupnosť dĺžky $O(n_1^2 + n_2^2, \ldots, n_k^2)$.*

Dôkaz. Algoritmus 7 len spája výstupy funkcie *sortPermTree*. Podľa vety 11 funkcia *sortPermTree* na strome T_i so $n_i - 1$ hranami vráti postupnosť dĺžky najviac $O((n_i - 1)^2)$. To môžme upraviť na $O(n_i^2)$. Potom celková dĺžka zloženej postupnosti je:

$$\sum_{i=1}^{k} O(n_i^2) = O(\sum_{i=1}^{k} n_i^2)$$

□

Veta 22. *Nech $n_1, n_2, \ldots, n_k$ sú počty vrcholov stromov v lese L. Potom platí*

$$n_1 + n_2 + \ldots + n_{t(i)-1} + s(i) + 3 \geq n_1 + n_2 + \ldots + n_{t(i+1)-1} + s(i+1) + 1$$

Dôkaz. Budeme postupovať rozborom prípadov:

- Ak $t(i) = t(i+1)$ potom platí $s(i)+1 \geq s(i+1)$, z toho dostaneme

  $$n_1 + n_2 + \ldots + n_{t(i)-1} + s(i) + 2 \geq n_1 + n_2 + \ldots + n_{t(i+1)-1} + s(i+1) + 1$$

- Ak $t(i)+1 = t(i+1)$, potom transpozícia $f(i)$ je posledná transpozícia v strome $T_{t(i)}$ a transpozícia $f(i+1)$ je prvá v strome $T_{t(i+1)}$. Z toho vyplýva, že $s(i+1) = 1$ a $s(i) = n_{t(i)} - 1$. Zároveň platí, že $n_{t(i+1)-1} = n_{t(i)}$

  $$\begin{aligned} & n_1 + n_2 + \ldots + n_{t(i+1)-1} + s(i+1) + 1 \\ = \; & n_1 + n_2 + \ldots + n_{t(i)} + 2 \\ = \; & n_1 + n_2 + \ldots + n_{t(i)-1} + s(i) + 3 \end{aligned}$$

□

Veta 23. *Šéf i-tej úrovne skontaktuje susedného šéfa i-tej úrovne s použitím $(O((n_1 + n_2 + \ldots + n_{t(i)-1} + s(i) + 1)^3))$ správ.*

Dôkaz. Dôkaz rozdelíme na tri časti:

- Doručenie správy do vrcholu v z množiny $U_i(x)$. Šéf i-tej úrovne pomocou metódy *sendByTransposition* odošle správu do vrcholu v. Pritom používa transpozície zo stromu $T_{t(i+1)}$, ktorý má priemer $s(i)+1$. Potom preposielanie pomocou metódy *sendByTransposition* spotrebuj najviac $s(i)+1$ správ.

- Doručenie správy svojmu šéfovi i-tej úrovne. Predpokladajme, že vrchol u je šéf l-tej úrovne. Potom cesta k $l+1$ šéfovi má dĺžku najviac $O(n_1^2 + n_2^2 + \ldots + n_{t(l+1)-1}^2 + (s(i+1)+1)^2)$. Potom počet správ pri preposielaní šéfovi i-tej úrovne bude

  $$\begin{aligned} & \textstyle\sum_{j=l}^{i-1} O(n_1^2 + n_2^2 + \ldots + n_{t(j+1)-1}^2 + (s(j+1)+1)^2) \\ & \leq \textstyle\sum_{j=0}^{i} O((n_1 + n_2 + \ldots + n_{t(j+1)-1} + (s(j+1)+1))^2) \\ & = O((n_1 + n_2 + \ldots + n_{t(j+1)-1} + (s(j+1)+1))^3) \end{aligned}$$

- Spätné doručenie správy. Správa sa doručuje po skrátenej linke, ktorej dĺžka podľa vety 21 je $O(n_1^2 + n_2^2 + \ldots + n_{t(i+1)-1}^2 + (s(i+1)+1)^2)$.

Celkový počet správ môžeme ohraničiť výrazom $O((n_1+n_2+\ldots+n_{t(i+1)-1}+(s(i+1)+1))^3)$. Pomocou vety 22 a schopnosti O pohlcovať konštanty upravíme na konečný tvar $O((n_1 + n_2 + \ldots + n_{t(i)-1} + s(i))^3)$. □

Veta 24. *Algoritmus voľby šéfa na grafe β_S použije $O(N)$ správ, kde N je počet vrcholov grafu β_S.*

Dôkaz. Šéf na úrovni i kontaktuje $s(i)+2$ susedných šéfov. Keďže $s(i) \leq s(i)+1+\sum_{j=1}^{t(i)-1} n_j$, šéf i-tej úrovne vygeneruje priamo či nepriamo najviac $O((n_1+n_2+\ldots+n_{t(i)-1}+s(i))^4)$ správ. Graf β_S obsahuje

$$\frac{\prod_{j=1}^{k} n_j!}{\prod_{j=1}^{t(i)-1} n_j!} * \frac{1}{(s(i)+1)!}$$

šéfov i-tej úrovne. Algoritmus postupne volí šéfov vyššej a vyššej úrovne až sa zvolí šéf celého grafu β_S. Celkový počet správ bude

$$\sum_{i=0}^{|S|-1} \frac{\prod_{j=1}^{k} n_j!}{\prod_{j=1}^{t(i)-1} n_j!} * \frac{O(n_1+n_2+\ldots+n_{t(i)-1}+s(i))^4)}{(s(i)+1)!}$$

Výraz $\prod_{j=1}^{k} n_j!$ nezávisí od premennej i a môžme ho vyňať pred sumu. Ďalej rozdeľme sumu na súčet súm podľa hodnoty $t(i)$.

$$\leq \prod_{j=1}^{k} n_j!(\sum_{i=1}^{n_1} \frac{O(i^4)}{i!} + \sum_{i=1}^{n_2} \frac{O((n_1+i)^4)}{n_1!i!} + \ldots \sum_{i=1}^{n_k} \frac{O((n_1+n_2+\ldots+n_{k-1}+i)^4)}{n_1!n_2!\ldots n_{k-1}!i!})$$

Vieme, že $n_i \geq 2$, pretože izolované vrcholy v transpozičnom lese nemôžu existovať. Dosadením do výrazu dostaneme

$$\leq \prod_{j=1}^{k} n_j!(\sum_{j=1}^{k} \frac{O((2j-1)^4)}{2^{j-1}} + \frac{O((2j)^4)}{2^j})$$

vypíšme pár sčítancou:

$$\frac{O(1^4)}{2^0} + \frac{O(2^4)}{2^1} + \frac{O(3^4)}{2^1} + \frac{O(4^4)}{2^2} + \frac{O(5^4)}{2^2} + \ldots$$

vidíme, že túto postupnosť vieme zhora ohraničiť radom:

$$\sum_{i=1}^{k} \frac{O((2i-1)^4)}{2^{\frac{2i-1}{2}-1}} + \frac{O((2i)^4)}{2^{\frac{2i}{2}-1}}$$

upravíme ho na tvar:

$$\sum_{i=1}^{n} \frac{O(i^4)}{2^{\frac{i}{2}-1}}$$

Celkový počet správ odhadneme výrazom

$$\begin{aligned} &\leq \prod_{j=1}^{k} n_j!(\sum_{i=1}^{n} \frac{O(i^4)}{2^{\frac{i}{2}-1}}) \\ &\leq N * \sum_{i=1}^{\infty} \frac{O(i^4)}{2^{\frac{i}{2}-1}} \\ &= N * O(1) \\ &= O(N) \end{aligned}$$

□

2.3.10 Časová zložitosť

Podobne ako v kapitole 2.2.10, budeme predpokladať, že všetky uzly(vrcholy) spustia algoritmus nezávisle a posledný uzol začne v čase t_0.

Veta 25.

$$n_1 + n_2 + \ldots + n_{t(i)-1} + s(i) \leq 2i$$

Dôkaz. Porovnajme hodnoty pre i a $i+1$

$$n_1 + n_2 + \ldots + n_{t(i+1)-1} + s(i+1) - (n_1 + n_2 + \ldots + n_{t(i)-1} + s(i))$$

- Ak $t(i) = t(i+1)$, potom sa rozdiel z redukuje na $s(i+1) - s(i)$, keďže transpozície sú obe zo stromu $T_{t(i)}$ tak $s(i+1) - s(i) = 1$.
- Ak $t(i) + 1 = t(i+1)$, potom sa rozdiel zredukuje na $n_{t(i+1)-1} + s(i+1) - s(i)$. Keďže transpozícia $f(i)$ je posledná zo stromu $T_{t(i)}$, tak $s(i) = n_{t(i)} - 1$. Transpozícia $f(i+1)$ je prvá na strome $T_{t(i+1)}$ tak $s(i+1) = 1$. Po dosadení dostaneme $n_{t(i)} + 1 - (n_{t(i)} - 1) = 2$.

□

Veta 26. *Algoritmus zvolí šéfa v grafe β_S generovanom transpozičným lesom v čase $O(n^4)$, kde*

$$n = \sum_{i=1}^{k} n_i$$

Dôkaz. Predpokladajme, že šéfovia i-tej úrovne sú zvolený v čase t_i. Kontaktovanie susedných šéfov prebieha paralelne, stačí nám spočítať koľko času trvá matchmaking s jedným susedom.

- Doručenie správy do susedného podgrafu trvá $O(s(i) + 1)$ krokov. Použitím nerovnosti $s(i) \leq i$, odhadneme počet krokov na najviac $O(i)$.

- Doručenie správy svojmu šéfovi trvá $O((n_1 + n_2 + \ldots + n_{t(i)-1} + s(i))^3)$ krokov. Pomocou vety 25 zjednodušíme odhad na $O(i^3)$.
- Spätné doručenie správy trvá $O(i^2)$ krokov.

Celkový čas bude:

$$t_n = t_0 + \sum_{i=1}^{|S|} O(i^3) \leq t_0 + \sum_{i=1}^{n} O(i^3) = t_0 + O(n^4)$$

□

Kapitola 3

Záver

V tejto práci sme skúmali problém voľby šéfa na cayleho grafoch. Na začiatku sme presne zadefinovali problém. Zaoberali sme prípadom, keď cayleho graf je generovaný množinou transpozícii. Skúmali sme štruktúru cayleho grafov generovaných transpozičným stromom a lesom. Podarilo sa využiť ich algebraickú štruktúru na ich dekompozíciu. Ich rekurzívna štruktúra sa stala základom algoritmov na voľbu šéfa.

Navrhli sme algoritmy na voľbu šéfa pomocou turnajovej schémy. Obidva algoritmy používajú najviac lineárny počet správ. Podarilo sa nám zjednotiť algoritmy na voľbu šéfa okrem iných pre hyperkocky, star grafy a bubble sort grafy. Našli sme netriviálnu podtriedu cayleho grafov, na ktorých sa dá zvoliť šéf s lineárnym počtom správ.

Zaujímavé rozšírenie by bolo nasadiť navrhnutý algoritmus na cayleho grafy generované transpozičnými grafmi prípadne involúciami. Iným rozšírením by bolo zrušenie identifikátorov a pracovať s anonymnou sieťou. Dokázali by sme navrhnúť efektívny matchmaking? Ďalšia možnosť by bola, snažiť sa zlepšiť časovú zložitosť algoritmov.

Oblasť distribuovaných výpočtov je dnes veľmi živa. S rastúcou dostupnosťou hardvéru, schopného zabezpečiť distribuované výpočty, bude počet zaujímavých problémov rásť.

Zoznam tabuliek

2.1 Prvky množiny $U_3(x)$ 25

Zoznam obrázkov

2.1 Príklad inštancie hry na generujúcej množine $\{(1,2),(2,3),(3,4),(3,5)\}$ 17
2.2 Transpozičný strom 20
2.3 Dekompozícia cayleho grafu Γ_4 21
2.4 Kontaktovanie susedných podgrafov 24

Literatúra

[1] Akers, S. B., and Krishnamurthy, B. A group-theoretic model for symmetric interconnection networks. *IEEE Trans. Comput. 38*, 4 (1989), 555–566.

[2] Awerbuch, B. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing* (New York, NY, USA, 1987), ACM, pp. 230–240.

[3] Barrière, L. Leader election in abelian cayley graphs. In *In 8th Colloquium on Structural Information and Communication Complexity (SIROCCO '01), Carleton Scienti* (2001), pp. 5–20.

[4] Blin, L., and Butelle, F. A very fast (linear time) distributed algorithm, on general graphs, for the minimum-weight spanning tree.

[5] Bodlaender, H. L. New lower bound techniques for distributed leader finding and other problems on rings of processors. *Theor. Comput. Sci. 81*, 2 (1991), 237–256.

[6] Flocchini, P., Mans, B., and Santoro, N. Sense of direction: Definitions, properties, and classes. *Networks 32*, 3 (1998), 165–180.

[7] Flocchini, P., Mans, B., and Santoro, N. Sense of direction in distributed computing. *Theor. Comput. Sci. 291*, 1 (2003), 29–53.

[8] Gallager, R. G., Humblet, P. A., and Spira, P. M. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst. 5*, 1 (1983), 66–77.

[9] Katriňák, T., Gavalec, M., Gedeonová, E., and Smítal, J. *Algebra a teoretická aritmetika I*, 2 ed. Univerzita Komenského, Bratislava, 1995.

[10] Korach, E., Moran, S., and Zaks, S. Tight lower and upper bounds for some distributed algorithms for a complete network of processors. In *PODC '84: Proceedings of the third annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 1984), ACM, pp. 199–207.

[11] Lavault, C. Interconnection networks: Graph- and group-theoretic modelling. *Proc. CSCS12 2* (1999), 207–214.

[12] Loui, M. C., Matsushita, T. A., and West, D. B. Election in a complete network with a sense of direction. *Inf. Process. Lett. 22*, 4 (1986), 185–187.

[13] Mohamed, A., and Ramakrishna, R. S. Linear election in pancake graphs. *Inf. Process. Lett. 106*, 3 (2008), 127–131.

[14] Mullender, S. J., and Vitányi, P. M. B. Distributed match-making for processes in computer networks (preliminary version). In *PODC '85: Proceedings of the fourth annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 1985), ACM, pp. 261–271.

[15] Ozsu, M. T., and Valduriez, P. *Principles of distributed database systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[16] Santoro, N. Sense of direction, topological awareness and communication complexity. *SIGACT News 16*, 2 (1984), 50–56.

[17] Shi, W., Bouabdallah, A., and Srimani, P. K. Leader election in oriented star graphs. *Netw. 45*, 3 (2005), 169–179.

[18] Tel, G. Linear election for oriented hypercubes. Tech. rep., Parallel Processing Letters 5, 1993.

[19] Tel, G. *Introduction to Distributed Algorithms*. Cambridge University Press, New York, NY, USA, 2001.