PRL - Priradenie poradia preorder vrcholom

Natália Holková
xholko02@stud.fit.vutbr.cz
21. apríla 2022

1 Teória

1.1 Poradie preorder

Poradie **preorder** v binárnom strome znamená, že najskôr sa navštívi otcovský uzol, následne ľavý podstrom a nakoniec pravý podstrom. Pre paralelný výpočet poradia preorder pre binárny strom je potrebné najskôr vypočítať *Eulerovu cestu*, potom *pole váh* a napokon *sumu sufixov*.

1.2 Eulerova cesta

Eulerova cesta je definovaná ako taký spôsob prechádzania stromom, že každý vrchol je pridaný do cesty keď ho navštívime. Buď sa pridá pri prechode z otcovského uzla do synovského, alebo opačne pri spätnom prechode zo synovského uzla do otcovského. Začneneme v určitom vrchole a vrátime sa do tohto vrcholu až po prejdení všetkých vrcholov.

Algoritmus Eulerovej cesty je možné vykonávať paralelne. Najskôr je potrebné zostaviť binárny strom s n uzlami, ktorý má preto celkovo $n-1$ hrán. Strom sa následne prevedie na orientovaný graf tak, že každé hrana (u, v), kde u je hodnota uzla odkiaľ hrana vychádza a v je hodnota uzla do ktorého hrana smeruje, sa nahradí *dvoma* hranami (u, v) a (v, u), teda ku každej hrane sa pridá reverzná hrana. Vytvorený orientovaný graf bude mať preto $2n-2$ hrán. Paralelný algoritmus Eulerovej cesty bude využívať **2n - 2 procesorov** a každý procesor sa bude starať o jednu hranu e a jej nasledovníka v Eulerovej ceste $Etour(e)$.

Využíva sa tu zoznam susedov uzlov. Každý uzol stromu má príslušný zoznam jeho susedov (0, 1 alebo 2 susedia), kde každý sused má tvar $(e, e_R, next)$, kde e je hrana vychádzajúca z riešeného uzla, e_R je reverzná hrana na e a $next$ je ukazovať na nasledujúceho suseda. Každý procesor má informácie o celom zozname susedov a na základe toho vie vypočítať $Etour(e)$ pre svoju danú hranu e.

Každý procesor prechádza postupne zoznam susedov a hľadá prvú položku, suseda, ktorý obsahuje na prvom mieste jeho danú hranu e. Zisťuje reverznú hranu a potom rovnakým spôsobom ju vyhľadáva v zozname susedov. Keď ju nájde, pozrie sa, či sused v ktorom sa nachádzala má $next$ odkazujúci na ďalšieho suseda. Pokiaľ má, tak výsledok $Etour(e)$ je $next(e_R)$. Inak je výsledkom hrana prvého suseda v zozname susedov toho uzlu, v ktorom bol aj sused obsahujúci e_R.

1.2.1 Korekcia Eulerovej cesty

Samotná Eulerova cesta neberie v úvahu koreň stromu, toto treba dodatočne upraviť. Upravuje sa to tak, že hrana e, ktorej následník posledný vedie do koreňového uzla, bude mať zmenený $Etour(e) = e$. V podstate sa tým z cyklického zoznamu spraví obyčajný zoznam, ktorého posledný prvok ukazuje sám na seba.

1.3 Pole váh

Výpočet poľa váh je opäť možné paralelizovať na **2n - 2 procesorov**, kde každý procesorov rieši váhu jednej hrany. Váha v tomto prípade znamená, či je hrana *doprednú* (1) alebo *spätná* (0). Na zistenie váhy sa používa zoznam susedov, ktorý sa postupne prehľadáva. Ak sa hľadaná hrana v ňom najskôr objaví ako hlavná hrana e smerujúca z uzla, jedná sa o hranu doprednú, inak ak sa ukáže skôr ako reverzná hrana e_R smerujúca *do* uzla, bude *spätná*.

1.4 Suma suffixov

Suma suffixov sa vykonáva nad poľom váh a Eulerovou cestou a vykonáva súčet nad poľom váh. Ako sa postupuje udáva Eulerova cesta. Suffix predstavuje podzoznam medzi riešeným prvkom a koncom zoznamu.

Pri paralelnom riešení ak zisťujeme suffix sum pre hranu e, tak začíname počítať súčet od prvku z poľa váh pre hranu e a nasledujúci index poľa budeme brať z $Etour(e)$. Ukončovacou podmienkou tohto cyklu bude, keď narazíme na koniec Eulerovej cesty, ktorý sa prejavuje ako $Etour(e) = e$.

1.5 Určenie poradia preorder

Pre určenie poradia preorder zo sumy suffixov je potrebné ignorovať spätné hrany. Zostupný zoznam hrán zoradený podľa ich hodnuty sumy suffixov nám určuje poradie preorder. Pre výpis uzlov v poradí preorder stačí iba vypísať koreň a následne postupovať po hranách v tomto poradí a vypisovať hodnotu uzla do ktorého smerujú.

1.6 Teoretická zložitosť

Tvorba poľa váh má konštatnú časovú zložitosť $\mathcal{O}(c)$. Výpočet sumy suffixov by mal mať časovú zložitosť $\mathcal{O}(logn)$.

2 Implementácia

Implementácia je riešená v jazyku C++ s využitím knižnice *Open MPI* pre paralelizáciu. Pri zadaní vstupného reťazca uzlov o dĺžke n je spustený program s celkovo **2n - 2 procesormi**.

2.1 Načítanie hodnôt, vytvorenie zoznamu susedov

Načítanie vstupného reťazca uzlov z argumentov príkazového riadku rieši procesor 0, označovaný ako `ROOT`. Reťazec s hodnotami uzlov v prvom rade konvertuje na objekt triedy `Tree` funkciou `createTree()`, binárny strom, ktorý sa skladá so reťazca hodnôt uzlov a vektoru hrán typu `Edge`. Každá hrana `Edge` má vlastné unikátne `id` podľa ktorého sú prideľované procesorom. Binárny strom konvertujeme na orientovaný graf ako bolo popísané v 1.2 funkciou `convertTreeToOrientedGraph()`.

Z orientovaného grafu sme už schopný vytvoriť zoznam susedov. Trieda `Neighbour` predstavuje suseda obsahujúceho hlavnú hranu a k nej reverznú. Ukazovateľ *next* nahradzujeme tým, že objekty `Neighbour` budú udržované vždy vo vektore, čím teda simulujeme štruktúru zoznamu. Celkový zoznam susedov vytvárame sekvenčne pomocou funkcie `createNeighboursVector()`.

2.2 Eulerova cesta

Výpočet Eulerovej cesty sa už vykonáva paralelne. Pred začatím výpočtu musí `ROOT` procesor rozoslať všetkým procesorom kópu celého zoznamu susedov. Keďže zoznam susedov je reprezentovaný ako vektor vektorov, musí sa pred zaslaním konvertovať na obyčajné pole. Každému procesoru sú v cykle zasielané zoznamy susedov pre jednotlivé uzle vo funkcii `sendNeighboursToProcessor()`. Pri zasielaní sa používa iba obyčajné `MPI_Send()`. Jednotlivé procesory to príjmajú postupným `receiveNeighbours`, ktorý využíva `MPI_Recv()`. Po zostaveni zoznamu susedov všetkými procesormi je možné vykonať výpočet Eulerovej cesty.

Každý procesor sa stará o hranu, ktorej `id` sa rovná hodnote jeho ranku. Výpočet elementu Eulerovej cesty zaisťuje funkcia `getEtourElement()`. Implementácia je podľa toho, ako je to definované v 1.2.

2.3 Pole váh

Výpočet poľa váh prebieha paralelne po jednotlivých prvkov vo funkcii `determineEdgeWeight()` na základe `id` spracovávanej hrany a zoznamu susedov. Implementácia je presne ako je definové v 1.3.

2.4 Suma suffixov

Po tom, čo jednotlivé procesory vypočítajú svoje hodnoty poľa váh, navzájom si všetky procesory preposielajú hodnoty pomocou funkcie `MPI_AllGather()`, vďaka čomu každý procesory skončí s kompletnou kópiou poľa váh. Hodnoty jednotlivých elementov Eulerovej cesty musia byť avšak najskôr zaslané `ROOT` elementu, ktorý vykoná korekciou cesty vo `correctEtour()` ako je spomenuté v 1.2.1, a následne opravené pole rozošle celé všetkým procesorom funkciou `MPI_Bcast()`.

Keď majú procesory svoje kópie celého poľa váh a poľa Eulerovej cesty, je už možné vykonať paralelne sumu suffixov, ktorá je nadefinovaná vo funkcii `suffixSum()`. Nakoľko každý procesor má celú kópiu oboch polí, nie je potrebné žiadna ďalšia komunikácia medzi nimi počas sumy suffixov. Implementácie postupuje podľa 1.4.

2.5 Poradie preorder a výpis uzlov

Po získaní sumy suffixov procesory zasielajú hromadne svoje výsledky `ROOT` procesoru pomocou `MPI_Gather`. Je využitá mapa `preorderMap` na namapovanie sumy suffixov na `id` hrany kvôli jednoduchosti neskoršieho výpisu týmto spôsobom. Pri vkladaní do `preorderMap` sa ignorujú spätné hrany a vkladajú sa položky iba pre dopredné hrany.

Pri finálnom výpise poradia preorder vrcholov sa najskôr samostatne vypísať hodnota koreňového uzla a následne zostupne zoradená `preorderMap` vypisuje hodnotu uzla, do ktorého smeruje hrana.

3 Záver

V projekte bol úspešne implementovaný paralelný algoritmus pre priradenie poradia **preorder** vrcholom binárneho stromu.