Dokumenácia k 3. projektu z PRL

Priradenie poradia preorder vrcholom

Matej Marušák, Apríl 2018

1 Úvod

V tejto správe je popísaná aplikácia vytvorená v rámci projektu PRL v akademickom roku 2017/18. Jedná sa o aplikáciu na nájdenie preorder prechodu stromom v jazyku C++ s využitím knižnice MPI. V kapitole 2 je úvod do riešenej problematiky, nasledovaný popisom aplikácie v kapitole 3.

2 Paralelný preorder prechod

Paralený preorder prechod potrebuje $2n - 2$ procesorov pre n vrcholov stromu (ku každej hrane je pridaná hrana jej opačná a pre každú hrany existuje jeden procesor). Ako prvý krok sa paralelne spočíta eulerová cesta. Jej tvorba je popísaná v subsekcii 2.1. Ďalším krokom je nájdenie dopredných hrán. Každej doprednej hrane je priradená hodnota 1, spätnej hodnota 0. Posledným krokom je spočítanie sumy sufixov (priradenie každému prvku sumu hodnôt nasledovníkov až po koniec zoznamu), ktorá sa taktiež počíta paralelne. Suma sufixov priamo indexuje poradie uzlov v preorder poradí okrem koreňa.

2.1 Eulerova cesta

Eulerova cesta je cesta v orientovanom grafe, ktorá prechádza každou hranou práve raz. Ak máme binárny strom, je potrebné každú hranu nahradiť dvojicou hrán, jednu doprednú a jednu spätnú. Každý procesor priradí index následníka, čo je možné urobiť v konštantnom čase. Pri tomto kroku je možné využiť list následníkov, alebo použiť vhodné indexovanie hrán. Pre hľadanie preorder prechodu je potrebné mať určený vrchol v eulerovej ceste, preto je potrebné priradiť hrane, ktorá smeruje do vrcholu hodnotu inú, ako vrchol samotný (bežne sa tam ukladá index samého seba). Nakoľko eulerova cesta je kružnica v grafe, tak vrchol sa vytvorí rozpojením kružnice v jednom uzle.

2.2 Teoretická zložitosť

Na výpočet teoretickej zložitosti pre n vrcholov si rozložíme výpočet na časovú a priestorovú zložitosť. Pre časovú zložitosť môže algoritmus byť rozdelený na 3 časti:

1. Spočítanie Eulerovej cesty - $O(c)$
2. Nájdenie dopredných hrán - $O(c)$
3. Spočítanie sumy suffixov - $log(2n - 2) \in O(logn)$

Celková teoretická časová zložitosť je teda $O(logn)$. Priestorová zložitosť je $2n - 2 \in O(n)$. Celková teda zložitosť je $O(n * logn)$.

3 Aplikácia

Aplikácia bola napísaná v jazyku C++ s využitím knižnice OpenMPI. Preklad a beh sa vykonáva za pomoci skriptu `test.sh`. Daný skript berie jeden argument a to reťazec uzlov zapísaných po prechode binárnym stromom sprava doľava po riadkoch. Výstupom je jeden reťazec, zoznam uzlov zapísaných v preorder poradí.

3.1 Indexovanie hrán

V projekte bolo vytvorené nasledujúce indexovanie hrán:

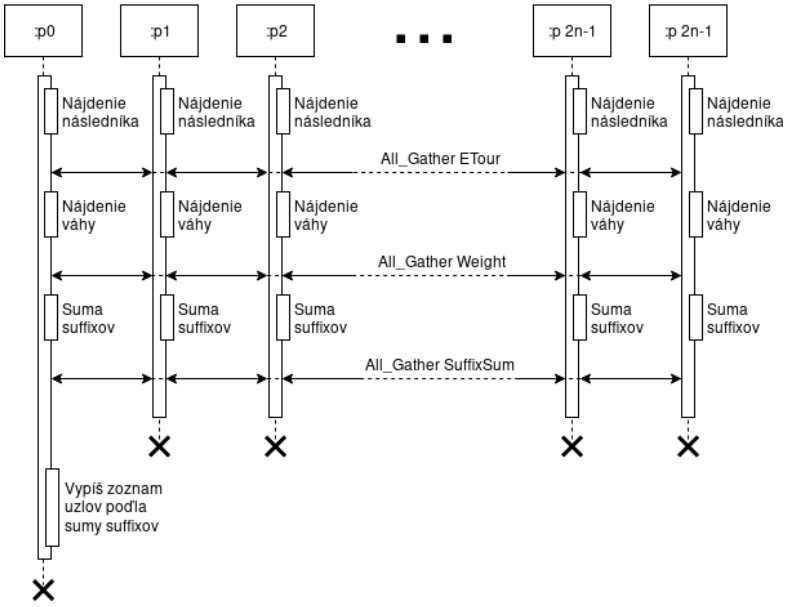
Koreňový uzol má hodnotu 0. Ľavý syn má hodnotu $2i + 1$, pravý $2i + 2$. Potom hrana má index i ak je dopredná a smeruje do uzlu s hodnotu i. Spätné hrany, ktoré smerujú z uzlu s indexom i majú index $i + n - 1$.

Vďaka tomuto indexovaniu je možné nájdenie eulerovej cesty v konštantnom čase. K nájdeniu eulerovej cesty stačí poznať tri predikáty pre každú hranu a to, či smeruje do listu ($i >= n/2 \&\& i < n$), či je hrana dopredná ($i < n$), alebo či má uzol, do ktorého smeruje spätná hrana aj pravého potomka ($((i - n) mod 2 == 0) \&\& (i < 2 * n - 2)$).

3.2 Beh programu

Každý procesor začne tým, že zistí svoj index (počíta sa od 1, takže procesor 0 vie, že indexuje hranu 1) a počet uzlov. Následne nájde svojho následníka, pre tvorbu eulerovej cesty. Procesor s indexom $n + 1$ považuje za svojho následníka svoj index (jedná sa o hranu smerujúcu do koreňa). Následne všetky procesory zapíšu svojho následníka do poľa reprezentujúceho eulerovu cestu za využitia kolektívnych funkcií. Druhým krokom je výpočet váh. Každý procesor, ktorý reprezentuje doprednú hranu zapíše do poľa váh číslo 1, spätnú číslo 0 opäť za využitia kolektívnych funkcií. Posledný krok, ktorý robia všetky procesory je výpočet sumy sufixov. Každý procesor sčíta hodnoty z poľa váh po prechode od svojej hrany až po hranu, ktorá smeruje do koreňa stromu. Potom už len prvý procesor na základe výsledku sumy prefixov a vstupného reťazca vypíše uzly v správnom poradí.

3.3 Komunikačný protokol

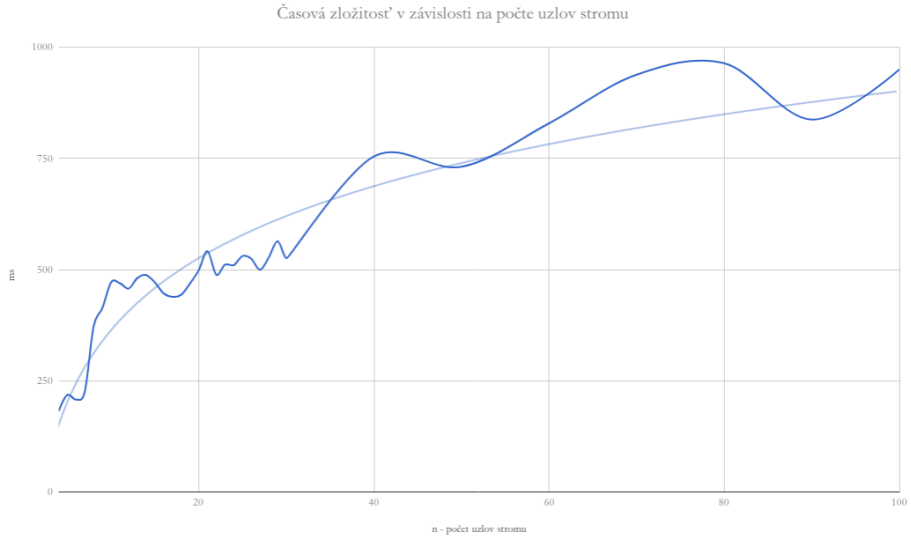


Obr. 1: Komunikačný protokol procesorov

3.4 Experimentálne výsledky

S výslednou aplikáciou bolo experimentované aj v sledovaní času, ktorý je potrebný na získanie preorder poradia. Obrázok 2 ukazuje ako sa správa aplikácia pri stúpajúcom počte uzlov. Čas sa začal merať každému procesoru po jeho inicializácii a príprave dátových štruktúr a meranie

sa ukončilo po poslednom kroku procesoru (výpis procesorom 0 sa neuvažoval). Následne čas každého procesora bol spriemerovaný aritmetickým priemerom. Pre každý počet uzlov bolo vykonaných 50 meraní. Všetky merania boli taktiež spriemerované a výsledná hodnota je vyobrazená v grafe. Výhoda tohto prístupu oproti meraniu celkového behu programu je v tom, že najväčšiu časť behu programu trvá inicializácia procesorov a výpis výsledku. Je potrebné ale uvažovať, že meranie bolo vykonávané na počítači so 4 jadrami a teda desiatky procesorov sú len simulované. Grafom bola preložená trendová čiara zobrazujúca teoretickú zložitosť (logaritmickú). Veľkosť kroku n je jedna pre hodnoty v rozsahu $< 4; 30 >$ a 10 pre hodnoty $< 40; 100 >$.



Obr. 2: Časové trvanie behu pre stúpajúci počet uzlov

4 Záver

V tejto správe bola popísaná aplikácia na paralelný prechod preorder stromom. Aplikácia funguje podľa očakávaní, jej funkcionalita bola overená na veľkej sade testov a boli zmerané časové výhody danej aplikácie, ktoré sú prezentované v grafe v kapitole 3.4.