Vysoké učení technické v Brně
Fakulta informačních technologií

Paralelní a distribuované algoritmy
2.Projekt – Přiřazení pořadí preorder vrcholům

Dominik Boboš (xbobos00)
xbobos00@stud.fit.vutbr.cz
2. mája 2022

1 Rozbor a popis algoritmu

Zadaním projektu bolo implementovať paralelný algoritmus na nájdenie pre-order prechodu stromom.

Paralelný algoritmus pre-order prechodu vyžaduje $2n - 2$ procesorov pre n vrcholov stromu – ku každej hrane v strome je pridaná aj opačná a počet hrán odpovedá počtu procesorov.

Algoritmus pozostáva z nasledujúcich krokov:

1. Výpočet Eulerovej cesty
2. Vyhľadanie dopredných hrán
3. Vypočet suffix sum

1.1 Výpočet Eulerovej cesty

Ako prvý krok v algoritme je spočítanie *Eulerovej cesty (EC)*. EC je cesta, kde sa každou hranou v orientovanom grafe prechádza práve raz. Ak je k dispozícií binárny strom, je potrebné každú z hrán hranu nahradiť dvojicou hrán: `i)` doprednú a `ii)` spätnú.

Každý z procesorov priraďuje index následníka – uskutočniteľné v konštantnom čase. Index následníka je možné získať viacerými spôsobmi, napríklad je možné využiť list následníkov (adjacency list), alebo je možné využiť vhodné indexovanie hrán, čo je aj predmetom môjho riešenia. Pre hľadanie pre-order prechodu je nutné v EC určiť vrchol – hrane, ktorá smeruje do vrcholu sa priradí iná hodnota, ako vrchol samotný. Zvykne sa ukladať index samého seba. Nakoľko EC je kružnica v grafe, tak vrchol získavame rozpojením kružnice v jednom uzle.

1.2 Výpočet suffix sum

Suffix sum je podobná prefix sum, ale je aplikovaná na grafy. Jednotlivé procesory si posielajú 2 hodnoty. Prvou je váha dotazovaného procesoru, ktorú procesor pripočítava ku svojej váhe. Druhou je pointer na nasledovníka dotazovaného procesoru. Tento pointer ukazuje na procesor, ktorého dáta budú žiadané v nasledujúcom cykle. Teda postupne každý procesor skáče o viac a viac procesorov. Posledný procesor ma sám seba za následníka, aby ostatné procesory neskákali mimo neho.

1.3 Zložitosti algoritmu

Výpočet časovej zložitosti:

- Výpočet Eulerovej cesty - $\mathcal{O}(1)$
- Vyhľadanie dopredných hrán - $\mathcal{O}(1)$
- Výpočet suffix sum - $\mathcal{O}(logn)$

Časová zložitosť: $\mathcal{O}(logn)$.
Priestorová zložitosť: $2n - 2 \rightarrow \mathcal{O}(n)$.
Celková cena: $\mathcal{O}(n * logn)$.

2 Implementácia

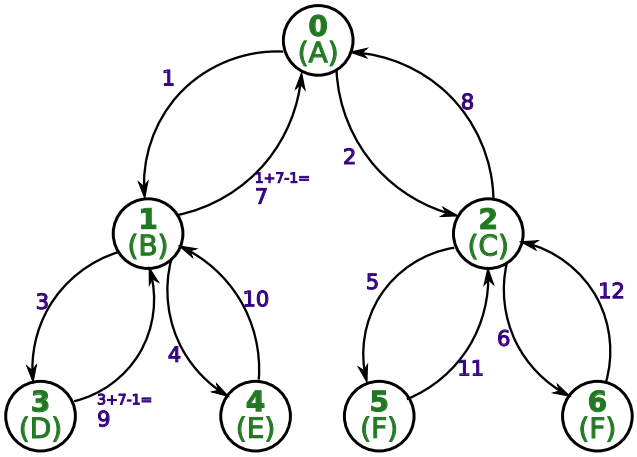
Algoritmus je implementovaný v programovacom jazyku *C++*. Pre posielanie správ medzi procesmi je použitá knižnica *Open MPI*. Implementácia sa nachádza v súbore `preorder.cpp`. Pre preklad a beh je vytvorený skript `test.sh`. Skript `test.sh` je potrebné spúšťať s jedným parametrom, ktorého reťazec značia uzly v stromovej štruktúre. Výstupom je opäť reťazec znázorňujúci uzly v pre-order poradí.

V projekte bolo vytvorené nasledujúce indexovanie hrán pre tvorbu EC.

Uzol, ktorý je koreňom má hodnotu 0. Ľavý potomok $2i + 1$, pravý $2i + 2$. Hrany indexujeme od 1. Potom hrana má index i práve vtedy ak je hrana dopredná a smeruje do uzla s hodnotou i. Naopak spätná hrana smerujúca z uzlu s hodnotou i má index $i+n-1$.

Takéto indexovanie zabezpečuje výpočet EC v $\mathcal{O}(1)$, lebo pre výpočet EC stačí poznať pre danú hranu to či je dopredná, či smeruje do listového uzlu, či má daný uzol aj pravého potomka.

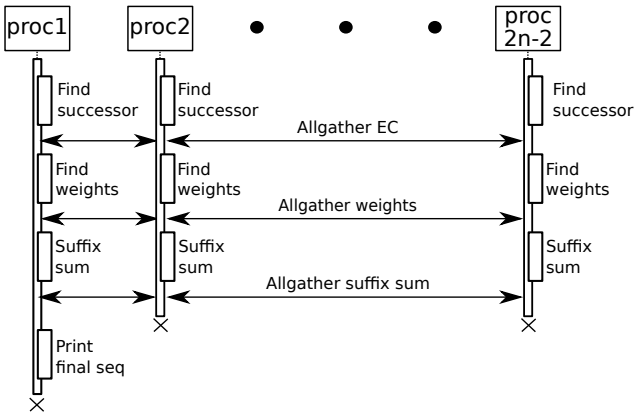
Hrana je dopredná ak `i < n`. Hrana smeruje do listového uzlu ak `(i >= n/2) && Dopredná`. Uzol má pravého potomka ak `(i - n) % 2 == 0 && i < (2*n - 2)`. Ukážka indexovania pre sekvenciu „ABCDEF“ je zobrazená na obrázku 1.



Obr. 1: Ukážka indexovania hrán pre sekvenciu „ABCDEF“

Samotný beh programu prebieha tak, že každý procesor začne tým, že zistí svoj index (pričom sa indexuje od 1, takže procesor s priradeným číslom 0 vie, že indexuje hranu 1). Zistí sa aj celkový počet uzlov z reťazca n. Potom procesor nájde svojho následníka, pre tvorbu EC. Prípad kedy procesor má index $n + 1$ tak za svojho následníka považuje svoj index – lebo ide o hranu smerujúcu do koreňa. Následne všetky procesory zapíšu svojho následníka do poľa reprezentujúceho EC za využitia `MPI_Allgather`. Ďalej nasleduje výpočet váh. Pre každý procesor, ktorého index reprezentuje doprednú hranu, zapíše do poľa váh číslo 1, pre spätnú hranu číslo 0 – opäť s využitím `MPI_Allgather`. Nakoniec sa uskutoční suffix sum. Z poľa váh každý procesor sčíta hodnoty po prechode od svojej hrany až po hranu, ktorá smeruje do koreňového uzlu. Po dokončení hlavný procesor (ten s indexom 0, resp.1) na základe výsledku a vstupného reťazca vypíše na STDOUT uzly v pre-order poradí.

Komunikačný protokol je zobrazený na obrázku 2.



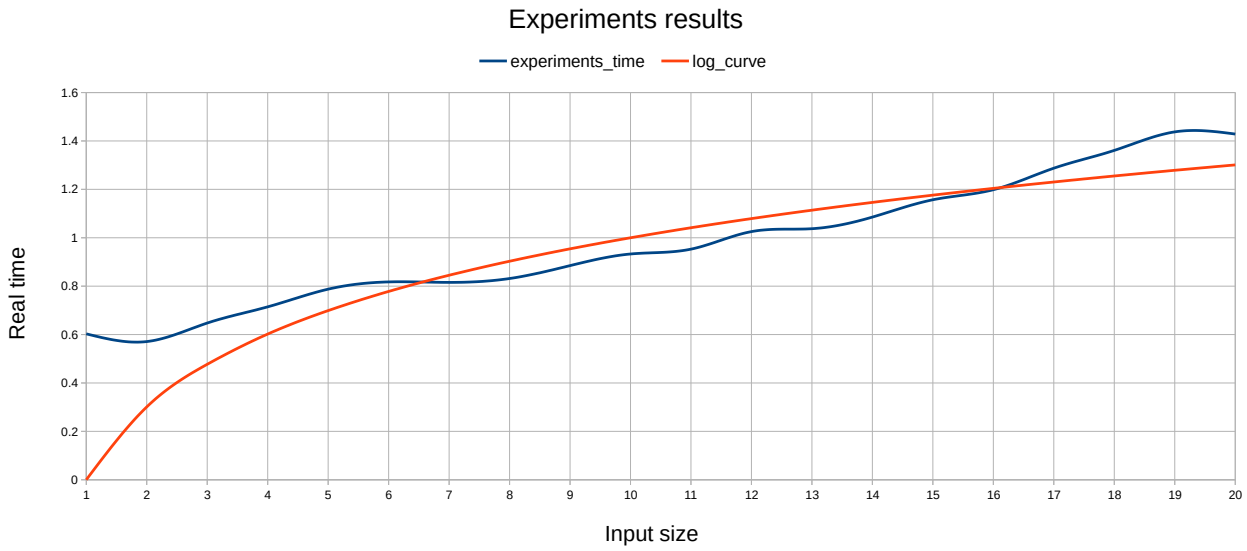
Obr. 2: Komunikačný protokol procesorov.

3 Experimentálne výsledky

Pre porovnanie časovej zložitosti implementácie s teoretickou boli vykonané testy. Postupne pre sekvencie od dlžky 1 až po dlžku reťazcov 20 bol spúšťaný program. Pre každú sekvenciu bolo vykonaných 50 behov a z nich sa uskutočnil priemer. Z týchto hodnôt bola následne zostavená tabuľka a potom vykreslený graf, ktorý je možný pozorovať na obrázku 3.

4 Záver

Vrámci tohto projektu bola úspešne vytvorená implementácia algoritmu pre priradenie pre-order prechodu vrcholom. Bol uvedený rozbor algoritmu a jeho popis. Bola predstavená a vysvetlená implementácia. Program funguje podľa očakávaní, funkčnosť bola overená pomocou experimentov.



Obr. 3: Experimentálne namerané výsledky preložené s teoretickou časovou zložitosťou, ktorá odpovedá logaritmickej krivke

.