Paralelné a distribuované algoritmy

Projekt č. 2 - Priradenie poradia preorder vrcholom

Adam Švenk (xsvenk00)

05.05.2022

Implementácia a inicializácia

Program je implementovaný v programovacom jazyku `C++`, avšak bez objektovo-orientovaných vlastností. Na implementáciu rozhrania prenosu správ umožňujúceho komunikáciu medzi viacerými procesmi je použitá knižnica `Open MPI`.

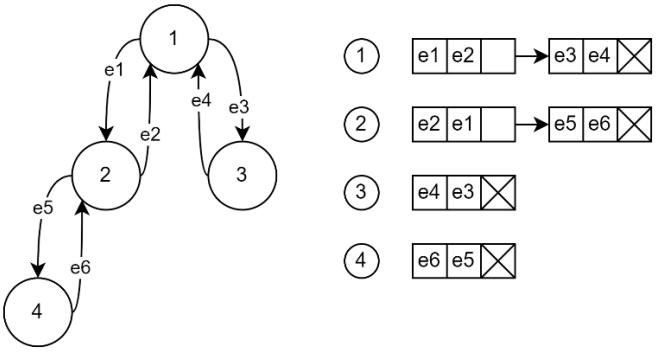
V prvej časti behu programu dochádza k inicializácii komunikačného prostredia poskytovaného knižnicou `Open MPI`. Nasleduje načítanie vstupného argumentu, ktorý obsahuje vstupný reťazec reprezentujúci vstupný binárny strom. V ďalších krokoch prebieha inicializácia jednotlivých dátových štruktúr a inicializácia hodnôt ich elementov, ktoré obsahujú dáta o binárnom strome – jeho uzly, hrany a zoznam susednosti.

Samotný algoritmus priradenia poradia preorder vrcholom binárneho stromu je zložený zo štyroch nasledovných podkrokov:

1. Vytvorenie Eulerovej cesty
2. Vytvorenie poľa hodnôt
3. Spočítanie sumy suffixov nad vytvoreným poľom hodnôt
4. Prevedenie finálnej korekcie

Vytvorenie Eulerovej cesty

Vytvorenie Eulerovej cesty predstavuje počiatočný krok algoritmu, bez ktorého sa nezaobídu nasledujúce časti. Eulerova cesta predstavuje všeobecný priechod stromovou štruktúrou. Špeciálnou variantou sú priechody preorder, postorder a inorder, ktoré špecifikujú poradie výberu v prípade uzla, ktorý obsahuje viacero synovských uzlov. Vstupnou štruktúru implementovaného paralelného algoritmu vytvorenia Eulerovej cesty z prednášky je zoznam susednosti jednotlivých uzlov vstupného binárneho stromu (adjacency list). Tento zoznam pre ukážkový vzorový binárny strom vyzerá nasledovne:



Obrázok 1 Binárny strom spoločne so zoznamom susednosti

Algoritmus funguje nasledovne: Každý proces paralelne reprezentuje jednu hranu, pričom v prípade, že reverzná hrana reprezentovanej hrany má v zozname susedov následníka, ktorý nie je rovný `NULL`, vloží na jej pozíciu v zozname tvoriacom Eulerovu cestu následníka spomínanej reverznej hrany. Ak by náhodou reverzná hrana nemala následníka (jej následník by bol rovný `NULL`), vloží na svoju pozíciu v zozname tvoriacom Eulerovu cestu prvú hranu zo zoznamu susednosti pre uzol, ktorý je cieľovým uzlom hrany, ktorú reprezentuje v paralelnom algoritme. Celý algoritmus má vďaka tomu, že každá hrana je reprezentovaná jedným z paralelne bežiacich procesov, časovú náročnosť `t(n) = n`. Každá z hrán následne odošle získanú nasledujúcu hranu, konkrétne jej identifikátor, otcovskému procesu, ktorý zo získaných hrán vytvorí samotnú Eulerovu cestu. Táto Eulerova cesta prechádza ešte drobnou korekciou, nakoľko v získaných dátach z jednotlivých procesov absentuje prvá hrana tvoriaca Eulerovu cestu, ktorá vychádza z koreňového uzla binárneho stromu do jeho ľavého podstromu.

Vytvorenie poľa hodnôt

Vytvorenie poľa hodnôt predstavuje ďalší krok. Každý z procesov reprezentuje jednu z hrán tvoriacich binárny strom, pričom algoritmus vytvorenia poľa hodnôt prebieha paralelne na všetkých procesoch. Algoritmus je vskutku jednoduchý – pokiaľ je hrana dopredná, nastaví jej hodnotu (váhu) na hodnotu `1`, v prípade, že nie je, nastaví na hodnotu `0`. V prípade zachovania spôsobu indexovania jednotlivých znakov vstupného reťazca ako uzlov, uvedených v zadaní projektu (`(2 * i) + 1`) je možné jednoduchou podmienkou overiť, či je hrana dopredná. Toto overovanie prebieha paralelne na všetkých procesoch. Po overení odošle každý z procesov získanú hodnotu (doprednosť) otcovskému procesu, ktorý zo získaných hodnôt vytvorí pole.

Spočítanie sumy suffixov nad vytvoreným poľom hodnôt

Vytvorené pole hodnôt z predchádzajúceho kroku predstavuje vstup algoritmu spočítania sumy suffixov. Suma suffixov pričítava `+ 1` za každú jednu doprednú hranu, po ktorej vo svojom cykle prejde, a danú hodnotu uloží do poľa. Výsledkom sumy suffixov je pole, ktoré obsahuje súčty jednotlivých hrán od konca, ktoré odpovedajú dopredným hranám. Výpočet sumy suffixov prebieha len na otcovskom procese.

Prevedenie finálnej korekcie

Finálna korekcia spočíva v overení na každom z procesov reprezentujúcich hranu, či sa jedná o doprednú hranu alebo nie. V prípade, že áno, je potrebné v zozname reprezentujúci samotný preorder prechod, nahradiť index cieľového uzla reprezentovanej hrany výsledkom výpočtu `n - weight(e) + 1`. V prípade, že hrana nie je dopredná, nastaví hodnotu na `-1`. Takto upravené hodnoty sú následne odoslané otcovskému procesu. Otcovský proces počas prijímania hodnôt vytvára finálne pole preorder, pričom vie, že v prípade hodnoty `-1` neupravuje hodnotu na indexe danej hrany. Po obdržaní všetkých hodnôt, úprave a vytvorení poľa dochádza k vypísaniu poradia preorder otcovským procesom na štandardný výstup stdout.

Analýza zložitosti algoritmu

Nakoľko je algoritmus zložený z viacerých podalgoritmov, je v prvom kroku potrebné vykonať ich analýzu.

1. Vytvorenie Eulerovej cesty
    a. Vzhľadom na implementáciu je zložitosť algoritmu $O(c)$.
2. Vytvorenie poľa hodnôt
    a. V prípade uvedenej implementácie predstavuje zložitosť $O(c)$.
3. Spočítanie sumy suffixov nad vytvoreným poľom hodnôt
    a. Suma suffixov má zložitosť $O(\log n)$.
4. Prevedenie finálnej korekcie
    a. Korekcia má zložitosť $O(c)$.

Celková časová zložitosť $t(n)$ algoritmu je $O(\log n)$. Vzhľadom na spôsob výpočtu sumy suffixov je celkový počet procesov potrebných na výpočet $c(n) = 2 * n - 2$.