PRL - Projekt 2. - Přiřazení pořadí preorder vrcholům

Michal Rein

22. dubna 2022

1 Potřebné konstrukty pro výpočty ve stromu

Přiřazení pořadí preorder vrcholům binárního stromu lze v paralelním prostředí realizovat pomocí následujícího obecného postupu:

1. Vytvoření **Eulerovy cesty**
2. Vytvoření pole hodnot
3. Spočtení **sumy suffixů** nad polem hodnot
4. Korekce spočtené hodnoty

Tento postup lze rovněž aplikovat na mnoho dalších úloh, například pro výpočet úrovně vrcholu, počtu následníků nebo postorder, případně inorder pořadí vrcholů.

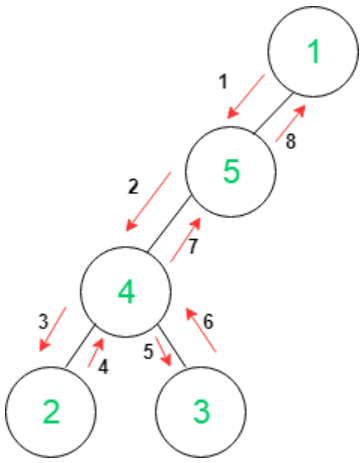
1.1 Eulerova cesta

Pojmem Eulerova cesta rozumíme takový průchod grafem, ve kterém jsou všechny hrany grafu navštíveny pouze jednou. Takový průchod klasickým binárním stromem $T = (V, E)$, kde V označujeme množinu všech vrcholů a E množinu všech hran, by nebyl možný, neboť každý z vrcholů takového stromu je s ostatními vrcholy spojen pouze jednou neorientovanou hranou.

Ze stromu T tedy vytvoříme orientovaný graf $T' = (V, E')$ takový, že každou hranu (u, v) nahradíme dvěma orientovanými hranami $< u, v >$ a $< v, u >$. Graf T' pak označíme za **Eulerovský graf**, neboť obsahuje **orientovanou kružnici**[1], která prochází každou hranou právě jednou. Příklad takového grafu lze vidět na obrázku 1, kde červené šipky představují nově vytvořené orientované hrany.

V paralelním prostředí lze Eulerovu kružnici reprezentovat tzv. funkcí následníka $Etour(e) \mid e \in E'$, která každé hraně e přiřazuje hranu e_{next}, která ji následuje. Výpočet Eulerovy cesty za předpokladu,

[1] Také nazývána jako Eulerova kružnice



Obrázek 1: Ukázka Eulerovského grafu vytvořeného z binárního stromu.

že každá hrana je reprezentována jedním procesorem, je velmi rychlý a lze jej provést v konstantním čase $t(n) = \mathcal{O}(c)$.

1.2 Suma suffixů

Sumou suffixů rozumíme součet všech podseznamů mezi prvkem a koncem seznamu. Mějme pole o n prvcích $A = (e_1, e_2, e_3 \ldots, e_n)$. Pak výsledkem sumy suffixů je n-tice $S = (S_1, S_2, S_3, \ldots, S_n)$, kde $S_1 = e_1 + e_2 + e_3 + \cdots + e_n$, $S_2 = e_2 + e_3 + \cdots + e_n \ldots S_{n-1} = e_{n-1} + e_n$ a $S_n = e_n$.
Za operátor sumy lze dosadit libovolný binární asociativní operátor $\bigoplus$.

V paralelním prostředí lze provést sumu suffixů s časovou složitostí $t(n) = \mathcal{O}(log.n)$ tak, že v $log(n)$ krocích každý procesor provádí (zjednodušeně) následující činnost:

1. Spočti: Hodnota = Hodnota $\bigoplus$ Hodnota následníka
2. Nastav: Ukazatel na následníka = Ukazatel následníka na svého následníka

2 Implementace

Úloha je implementována v jazyce C++ s využitím knihovny *Open MPI* jako prostředek pro komunikaci mezi procesy. Zdrojový kód s implementací se nachází v souboru `pro.cpp`. Program vykonává přiřazení pořadí preorder vrcholům binárního stromu, kde samotný strom je reprezentován řetězcem, jehož jednotlivé znaky označují vrcholy (uzly stromu). Počet potřebných procesorů je dán rovnicí $N = (2.n) - 2$, kde n je počet vrcholů, neboli délka vstupního řetězce. Vstupní řetězec je programu předán jako vstupní argument.

V první fázi programu dochází k rozdělení komunikátoru světa *MPI_COMM_WORLD* na *FORWARD_COMM*, ve kterém jsou procesy rozděleny podle orientace hrany, kterou má každý z nich na starosti. Orientace hrany každého procesu lze deterministicky určit, neboť rank procesů dopředných hran (od otce k synům), v kontextu komunikátoru světa, je vždy sudý. Hrany reverzní pak mají na starosti procesy, jejichž rank je lichý. Toto rozdělení hran vyplývá z vlastností Eulerovy cesty, kterou je nutné vytvořit a je popsána v kapitole 1.1.

Jak již bylo zmíněno, každý proces má na starosti jednu z hran, kterou si uchovává ve struktuře *edge*, jejíž obsahem jsou indexy vrcholů, které daná hrana spojuje. Samotný strom je uložený v poli, kde každý vrchol s indexem i má vždy svého levého potomka (pakliže existuje) na indexu $2*i$ a pravého na indexu $(2*i)+1$. Jelikož je takto vytvořený list vždy vyvážený a seřazený, lze jednoduše vypočítat výchozí index vrcholu dané hrany vztahem $floor(rank/4)$, kde rank je označení procesu v kontextu světa a $floor()$ funkce zaokrouhlení směrem dolů, neboť hrany k oběma synům z libovolného vrcholu číslovaného od 0 tvoří vždy 4 bezprostředně sousedící hrany (procesy). Obdobně lze spočítat index cílového vrcholu hrany vztahem $floor(rank/2) + 1$.

Jakmile každý proces spočítá parametry hrany, která mu náleží, dochází k vytvoření části Eulerovy cesty (lokální kružnice) do struktury, která se nazývá *Adjacency list*. Každá hrana si vytváří tento list vzhledem k vrcholu, do kterého směřuje. Pro výsledek ve formě pořadí preorder vrcholů je nutné jednotlivé dvojice hran vkládat do listu v pořadí:

1. Hrany k/od rodiče
2. Hrany k/od levého potomka
3. Hrany k/od pravého potomka

Po vytvoření této struktury je možné pomocí metody *adjacency_next()* získat index svého následníka, tak jak bylo definováno na přednášce. Po získání této informace se procesy představí svým následníkům zasláním zpráv pomocí metod *MPI_Send()* a *MPI_Recv()*. Po této fázi už všechny procesy znají svoje následníky a předchůdce.

V poslední fázi programu dochází ke spočtení výsledného pořadí vrcholů pomocí sumy suffixů. Jako reprezentant pole hodnot je vždy inicializována hodnota do proměnné *weight*, která je nastavena na 1 v případě dopředných hran, 0 pro zpětné hrany. Binárním asociativním operátorem je v tomto případě operace sčítání (+). Následuje proces distribuce hodnot a převázání ukazatelů tak jak bylo popsáno v kapitole 1.2, přičemž režie komunikace funguje pomocí zasílání zpráv a pracuje v těchto fázích:

1. Informuj následníka o ranku předchůdce (zpráva Send)
2. Přijmi rank předchůdce svého předchůdce (zpráva Recv)
3. Informuj předchůdce o své hodnotě a ranku následníka (zpráva Send)
4. Přijmi hodnotu a rank následníka od svého následníka (zpráva Recv)
5. Připočti ke své hodnotě novou hodnotu
6. Nahraď ukazatel na následníka nově přijatým ukazatelem

Proces *MASTER* slouží jako zarážka a je opatřen tagem, který se postupně šíří od poslední hrany a brání ostatním procesům, aby mu posílali data.

Po vypočtení sumy suffixů všechny dopředné hrany provedou korekci své hodnoty a odešlou finální zprávu s hodnotami své váhy (reprezentující index pořadí v preorder průchodu) a indexu vrcholu, do kterého směřují. Pomocí metody *MPI_Gather()* dochází k agregování všech hodnot do procesu *MASTER* od procesů v komunikátoru *FORWARD_COMM*, který následně převede indexy vrcholů na skutečné znaky a provede jejich indexaci do výsledného pole.

3 Závěr

Výsledná implementace skutečně provádí přiřazení pořadí preorder vrcholům a funguje dle očekávání.