Přiřazení pořadí preorder vrcholům

František Šumšal

April 29, 2018

1 Euler tour

Eulerova cesta (*Euler tour*) je obecný průchod binárním stromem. Speciálními případy jsou průchody preorder, inorder a postorder. Pro provedení Eulerovy cesty je nutné vstupní strom $T = (V, E)$ převést na orientovaný graf $T' = (V, E')$ tak, že každou hranu (u, v) z E nahradíme dvěma orientovanými hranami $< u, v >$ a $< v, u >$. Poté se z T' stává Eulerovský graf, který obsahuje orientovanou kružnici, procházející každým vrcholem z V pouze jednou.

Pro vytvoření Eulerovy cesty potřebujeme zkonstruovat Eulerovu kružnici, která je reprezentována funkcí $Etour$, která každé z hran $e \in E'$ přiřazuje následníka $Etour(e) \in E'$. Implementačně jsou tyto vztahy reprezentovány pomocí seznamu sousedností (*adjacency list*). Reverzní hrana k $e = (u, v)$ je značena jako $e_R = (v, u)$.

Samotná paralelní konstrukce Eulerovy cesty, kde vstupem je seznam sousedností a výstupem pole $Etour$, probíhá následovně:

1. Pokud následující prvek reverze hrany e (e_R) není roven nule, tak $Etour(e) = next(e_R)$
2. V opačném případě je následující prvek roven prvnímu prvku ze seznamu sousedností pro vrchol $v =>$ $Etour(e) = AdjList(v)$

1.1 Zavedení kořene

Výše popsaná definice Eulerovy cesty nevyužívá kořen stromu. Abychom byli schopni provádět operace nad daným stromem, je třeba takový kořen zavést. To lze provést pro kořenový vrchol r tak, že pro hranu e, vedoucí do kořene r, přiřadíme v Eulerově cestě $Etour(e) = e$. Tím Eulerovu cestu *přeřízeneme* ve vrcholu r.

1.2 Výpočet pozice hran v Eulerově cestě

Abychom mohli vypočítat sumu sufixů (viz dále), je nutné znát pořadí jednotlivých hran ve výsledné Eulerově cestě. Ve výsledku tedy kromě výpočtu pole $Etour$ a zavedení kořene vypočítáme *rank* jednotlivých hran pomocí algoritmu *ListRanking* a poté paralelně spočteme pole $posn(e)$ kde $posn(e) = 2n - 2 - Rank(e)$.

1.3 Suma sufixů

Pro preorder variantu Eulerovy cesty potřebujeme spočítat sumu sufixů pro jednotlivé hrany. Suma sufixů je součet prvků seznamu od daného prvku do konce seznamu. V případě preorder Eulerovy cesty se jedná o součet vah jednotlivých hran (viz dále).

1.4 Přiřazení pořadí preorder vrcholům

S využitím znalostí z předchozích odstavců lze sestrojit algoritmus, který pro daný binární strom vrátí seznam vrcholů v preorder pořadí. Algoritmus obsahuje následující tři kroky:

1. Nastavení vah jednotlivých hran – pokud je hrana dopředná, je její výsledná váha rovna 1, jinak 0.
2. Spočtení sumy sufixů pro každou hranu – vstupem je Eulerova cesta $Etour$ a váhy jednotlivých uzlu. Výstupem je nová váha pro daný uzel.

3. Vytvoření pole $preorder$ – pro každou dopřednou hranu e je $preorder(v) = n - weight(e) + 1$. Pro kořen platí $preorder(root) = 1$.

2 Implementace

Algoritmus je implementován v jazyce C++ s využitím knihovny OpenMPI.

Vstupem je řetězec uzlů binárního stromu, ze kterého je dle následujících pravidel zkonstruován vstupní binární strom:

- Uzly jsou indexovány od 1
- Levý podstrom uzlu x je na pozici $2x$
- Pravý podstrom uzlu x je na pozici $2x + 1$

Následuje vytvoření seznamu sousedností, rozdělení hran mezi jednotlivé procesory a výpočet následující hrany pro Eulerovu cestu. Tyto hrany jsou poté sesbírány kořenovým procesorem (ID 0) do pole $Etour$, nad kterým je proveden algoritmus *ListRanking* a výsledné ranky jsou rozeslány zpět procesorům, které je využijí ke zjištění pozice jejich hrany v Eulerově cestě.

Pro výpočet sumy sufixů je nutné znát váhy zbytku hran. Proto jsou jednotlivé váhy sesbírány kořenovým procesorem, který je uloží do pole $weights$ ve správném pořadí a rozešle zpět jednotlivým procesorům. Ty s využitím seřazeného pole $weights$ vypočítají svou sumu sufixů.

Posledním krokem je zkonstruování pole $preorder$. Každý uzel zašle kořenovému procesoru dvě hodnoty – vrchol, do kterého hrana vede, a pořadí v poli preorder ($n - weight(e) + 1$, kde $weight$ je váha získána sumou sufixů). To platí pouze v případě, že je hrana dopředná. V opačném případě procesor zašle obě hodnoty nastavené na -1, které kořenový procesor ignoruje. Posledním krokem je vypsání výsledného pole $preorder$.

3 Závěr

Výsledkem projektu je funkční implementace úlohy *Přiřazení pořadí preorder vrcholům*. Důležitou součástí bylo pochopení všech využitých algoritmů a jejich implementace bez využití kolektivních funkcí *MPI_Scan* a *MPI_Reduce*. Z tohoto pohledu je zadání splněno – aplikace generuje korektní výsledky a využívá pouze funkce *MPI_Send*, *MPI_Recv* a *MPI_Bcast*.