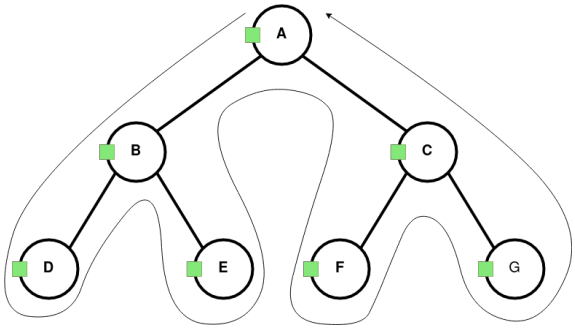
Dokumentácia k 2. projektu z predmetu PRL

René Rešetár (xreset00)

25. apríla 2022

1 Rozbor algoritmu

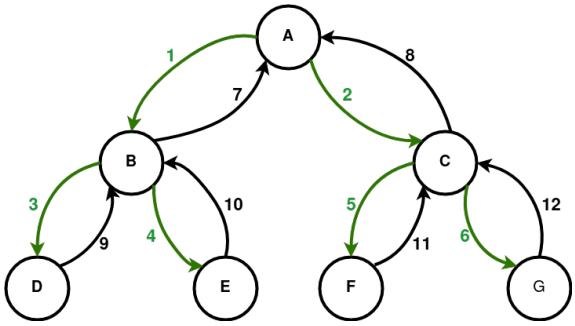
V tomto projekte sme mali za úlohu implementovať algoritmus pre priradenie poradia vrcholom grafu pri PreOrder prechode grafom binárneho stromu. Algoritmus sa skladá z troch častí a to vytvorenia Eulerovej cesty, spočítania sumy suffixouv a nakoniec konkrétneho výpočtu PreOrder poradia. Všetky príklady budeme ukazovať na strome z ukážkového vstupu v zadaní projektu **ABCDEFG**. Taktiež budeme predpokladať, že binárny strom sa bude tvoriť vždy zľava doprava, takže výsledný binárny strom by vyzeral ako na Obr. 1. Ak by sme vstup predĺžili na **ABCDEFGHI**, tak **H** sa stane ľavým a **I** pravým pod-uzlom uzla **D**.



Obr. 1: Binárny strom pre vstup ABCDEFG zobrazujúci aj spôsob získania poradia PreOrder priechodu. Dopredné hrany označené zelenou farbou.

1.1 Eulerova cesta

Tento termín označuje cestu orientovaným grafom, ktorá obsahuje každú hranu práve jeden krát. V grafe z Obr. 1 nahradíme každú neorientovanú hranu dvoma orientovanými a získame graf z Obr. 2. Týmto získame $2 * n - 2$ orientovaných hrán, kde n je počet vrcholov. Pre naše potreby bude prvá polovica hrán dopredných a druha polovica reverzných. Hrany sa podobne ako uzly očíslujú zľava doprava ako na Obr. 2.



Obr. 2: Binárny strom pre vstup ABCDEFG po nahradený neorientovaných hrán orientovanými.

S takto označenými hranami vieme vypočítať nasledujúcu hranu pre každú hranu týmito pravidlami implementovanými v *get_next_in_eltour(edge_number, number_of_vertices)*:

- Posledná hrana, ktorá vedie do koreňa má následníka samého seba.

- Reverzné hrany z ľavých pod-uzlov, ktoré majú možnosť pokračovať do pravého pod-uzlu budú mať následníka $edge\_number - number\_of\_vertices + 2$.
- Reverzné hrany z ľavých pod-uzlov, ktoré nemôžu pokračovať do pravého pod-uzla a reverzné hrany pravých pod-uzlov budú mať následníkov $(number\_of\_vertices-2)/2+(edge\_number/2)$.
- Dopredné hrany, ktoré nevedú do listov budú mať následníka $edge\_number+number\_of\_vertices-1$.
- Dopredné hrany vedúce do listov budú nasledované hranou $edge\_number * 2 + 1$.

Získavame Eulerovu cestu reprezentovanú nasledujúcim poľom (pole *euler* v kóde) na Obr. 3:

Edge:	1	2	3	4	5	6	7	8	9	10	11	12
Index:	0	1	2	3	4	5	6	7	8	9	10	11
Next edge:	3	5	9	10	11	12	2	8	4	7	6	8

Obr. 3: Cesta začína hranou 1 a pokračuje takto $1->3->9->4->10->7->2->5->11->6->12->8->8$

1.2 Suma suffixov

Je to suma hodnôt všetkých následníkov od konkrétnej hrany až po poslednú. V tomto prípade počítame s neutrálnym prvkom + a počítame sumu následných dopredných hrán. Pre jej výpočet potrebujeme pole váh (v kóde pole *weights*), v ktorom každý prvok reprezentujúci doprednú hranu bude nadobúdať hodnotu 1 a každý prvok reprezentujúci reverznú hranu hodnotu 0. Vďaka nášmu číslovaniu hrán jednoducho priradíme túto hodnotu každej hrane ako $edge < vertices\ ?\ 1 : 0$, kde *edge* je číslo hrany a *vertices* je počet vrcholov grafu. Získame:

Edge:	1	2	3	4	5	6	7	8	9	10	11	12
Index:	0	1	2	3	4	5	6	7	8	9	10	11
Weights:	1	1	1	1	1	1	0	0	0	0	0	0

Obr. 4: Pole váh.

Potom sumu suffixov (v kóde pole *suffix_sum*) získame prechodom Eulerovej cesty (kde každá hrana začína na svojej pozícií) a pripočítaním hodnoty z poľa váh do tejto sumy. Získavame:

Edge:	1	2	3	4	5	6	7	8	9	10	11	12
Index:	0	1	2	3	4	5	6	7	8	9	10	11
Suffix sum:	6	3	5	4	2	1	3	0	4	3	1	0

Obr. 5: Suma suffixov.

1.3 Priradenie Preorder poradia vrcholom

Poradie PreOrder nám označuje poradie, v ktorom prvok (vrchol) pridáme do zoradenej postupnosti pri prvom kontakte s ním pri prechode grafom ako je zobrazené na Obr. 1. Teda vždy keď šípka prechádzajúca graf narazí na zelený štvorček vrchol sa pridá do postupnosti. Pre **ABCDEFG** získavame **ABDECFG**. K tomuto využijeme pole sumy suffixov vypočítane v predchádzajúcom kroku a v ktorom nás budú zaujímať iba sumy dopredných hrán, teda prvej polovice poľa. Pomocou týchto hodnôt získame indexy pre všetky vrcholy v PreOrder poradí okrem prvého. Prvý prvok postupnosti predstavuje koreň a teda pre **ABCDEFG** bude **A** prvý (v kóde nultý index v poli *fin*). Zvyšok vypočítame následne:

Vertices = počet vrcholov = 7							
Vstupná postupnosť:	**A**	**B**	**C**	**D**	**E**	**F**	**G**
Suma suffixov:	0	6	3	5	4	2	1
Vertices - Suma suffixov:	0	1	4	2	3	5	6

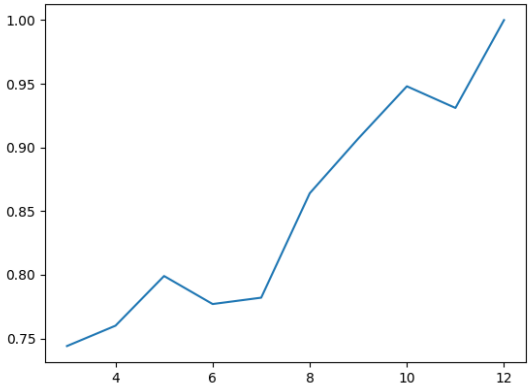
Obr. 6: Výpočet PreOrder poradia. Ak vrcholy zoradíme podľa vypočítaných výsledkov získame PreOrder poradie **ABDECFG**.

2 Priradenie práce procesom a časová zložitosť

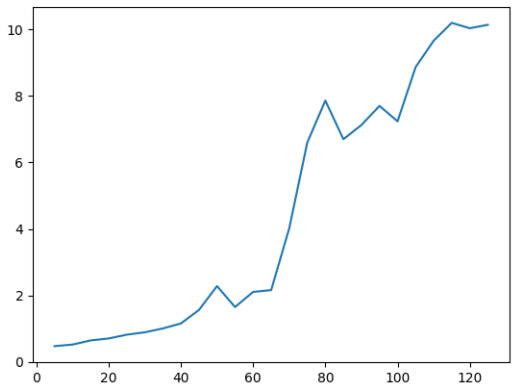
Všetky výpočty až na vytvorenie PreOrder poradia prebiehajú paralelne pomocou $2*pocet\_vrcholov-2$ procesov. Teda každý proces akoby predstavoval jednu hranu. Každý proces si sám vypočítal číslo svojej hrany, nasledujúcej hrany, váhy a sumy suffixov. Tieto hodnoty potom pošlú (MPI_Send) root procesu, ktorý ich uložil do príslušného poľa (MPI_Recv) a to obdobne rozpošle každému procesu. Výpočet Eulerovej cesty a váh teda prebiehajú v konštantnej časovej zložitosti $O(c)$. Výpočet sumy suffixov prebieha v logaritmickej časovej zložitosti $O(log(n))$. Po pridaní priestorovej zložitosti $O(n)$ teda získame celkovú časovú zložitosť $O(n * log(n))$. (*Pozn.* Pre grafy do 3 vrcholov sa len vypíše vstupný reťazec.)

3 Meranie

Osa x predstavuje počet vrcholov. Osa y predstavuje čas behu skriptu v sekundách. Meranie bolo prevedené pomocou reálnej časovej zložky z *time* v bash. Meral sa čas priebehu skriptu *test.sh* následne: *$ time test.sh <sequence>*.



Obr. 7: Meranie na Merlinovi. Maximálny možný počet vrcholov bol 13, keďže pre 24 procesov bolo maximum po ktoré ma Merlin pustil. Teda osa x = <3,13> s krokom 1.



Obr. 8: Meranie na mojom PC kde ma to pustilo po 130 vrcholov. Osa x = <5,130> s krokom 5.