Prednáška: Depth-first search

©MišoF., jeseň 2005

- DFS: keď sa dá, idem hlbšie, keď sa nedá, vrátim sa, pre každý vrchol si pamätám, odkiaľ som doň prvýkrát prišiel.
- Terminológia: koreň, podstrom, parent/rodič/otec, children/deti/synovia.
- Implementácia s vlastným stackom je netriviálna, treba si pre každý vrchol v ňom pamätať aj spracúvanú hranu.
- Čo si vieme navyše pamätať? Čas objavu d (discovery) a čas ukončenia spracovania f (finish) pre každý vrchol – čísla od 1 do $2N$.
- Poradiu discovery a finish eventov zodpovedá dobre uzátvorkovaný výraz. Inými slovami $[d(u), f(u)]$ a $[d(v), f(v)]$ sú buď disjunktné, alebo jeden je podmnožinou druhého.
- Klasifikácia hrán: stromové, spätné, dopredné a priečne. Ak robíme DFS neorientovaného grafu, všetky hrany sú stromové alebo spätné.
- Topologický sort: kým máme vrcholy, púšťame DFS, output finished.
- Mosty: Hrana je most iff neleží na kružnici iff ak odstrihneme hranu z parenta do childa, nevieme sa dostať z podstromu von.
- Artikulácie: Koreň je artikulácia, ak má viac ako 1 dieťa. Iný vrchol je artikulácia, ak z jeho podstromu (okrem neho) nevedie žiadna spätná hrana do nejakého jeho predka. (Spätné hrany dole v podstrome môžu byť.)
- Úprava DFS na počítanie artikulácií a mostov: pre každý vrchol v si pamätáme $l(v)$ – najmenšie číslo vrcholu, kam sa vieme dostať postupnosťou $k \geq 0$ tree-hrán a jednou back-hranou.

  Pre mosty pozeráme na $l(child)$, pre artikulácie na $l(.)$ pre deti daného vrcholu.

  Až na triviálny prípad (izolovaný vrchol) platí, že koncové vrcholy mosta sú artikulácie. Opačná implikácia neplatí, existujú artikulácie, z ktorých nevedú mosty.
- Blokovo-artikulačný graf, 2-súvislé komponenty.
- Silno súvislé komponenty. Začneme transponovaným grafom G^T – otočené smery hrán. Transpozíciou sa silne súvislé komponenty nezmenia.

  Všimnime si, že *komponentový graf* – graf, v ktorom silne súvislé komponenty scucneme do vrcholov – je acyklický.
- Algoritmus pre silno súvislé komponenty:

1. $DFS(G)$, pre každý vrchol máme $f(.)$.
2. Spočítať G^T.
3. CountSort vrcholov podľa $f(.)$ zostupne.
4. V tomto poradí spracúvame vrcholy, zakaždým, keď nájdeme nespracovaný, pustíme z neho DFS v G^T, ofarbí nám jeden komponent.

- Dôkaz algoritmu: Všimnime si pre každý silne súvislý komponent, ktorý vrchol z neho $DFS(G)$ navštívi ako prvý, toho nazveme reprezentantom komponentu a jeho $d(.)$ bude čas začiatku spracúvania komponentu. Tento vrchol bude zároveň mať z neho najväčšie $f(.)$, toto bude koniec spracúvania komponentu.

  Takto definované začiatky a konce zjavne zodpovedajú korektnému DFS v komponentovom grafe. (Keď sa vraciame z vrcholu, je už ofarbené všetko, kam sme sa chceli dostať, a teda každý komponent, kam sme sa vedeli dostať, je ofarbený celý.)

  Jeho komponenty by to teda vypísalo v lexikografickom poradí (!)

  No a teraz keď spracúvame v 4. kroku jednotlivé vrcholy, tak ako prvého z komponentu nájdeme jeho reprezentanta. V tej chvíli sú už spracované všetky komponenty, ktoré boli v lexikografickom poradí pred ním, a teda všetky, ktoré boli v G nad ním, t.j. sú v G^T pod ním. Preto keď z reprezentanta začneme ofarbovať v G^T, ofarbíme práve jeho komponent.