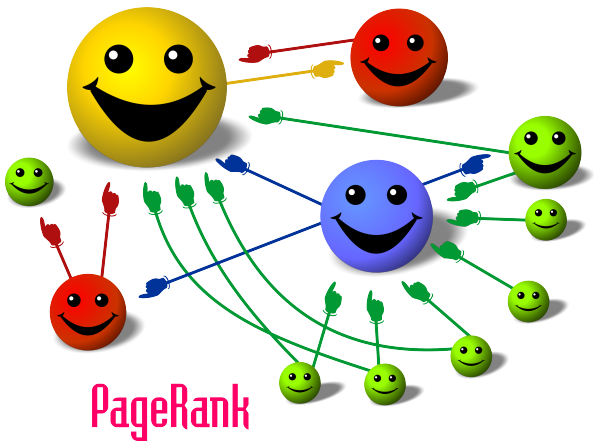
4 Pagerank

Algoritem Pagerank [22] sta leta 1998 predstavila Larry Page in Sergey Brin, ustanovitelja podjetja Google. Pagerank je uporabljal in ga v določeni meri še vedno uporablja spletni iskalnik Google pri rangiranju spletnih strani med prikazanimi zadetki za nek iskalni termin. Algoritem pri razvrščanju strani upošteva dva glavna faktorja, ki vplivata na pomembnost neke spletne strani:

- Število strani, ki vsebujejo povezavo do dane spletne strani.
- Pomembnost strani, ki vsebujejo povezave do dane spletne strani.

Drugače povedano, do neke strani lahko vodi razmeroma malo povezav, vendar se le-te nahajajo na pomembnih, oziroma zelo znanih straneh (e.g. nytimes.com), kar pomeni, da bo dana stran visoko rangirana. Lahko pa do neke strani vodi veliko število povezav iz manj pomembnih virov (razni blogi, forumi, ...), kar ji bo prav tako prineslo visok rang.



Matematično lahko zgornjo idejo zapišemo z naslednjo formulo:

$$R(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{R(p_j)}{L(p_j)}.$$

V zgornji enačbi je $R(p_i)$ rang spletne strani p_i, ki ga iščemo; $M(p_i)$ je množica strani, ki vsebujejo povezavo do strani p_i; $L(p_j)$ je število izhodnih povezav, ki jih vsebuje stran p_j in N je število vseh strani. Enačba vsebuje še teleportacijski faktor d, ki ga običajno nastavimo na 0.85. Z njim modeliramo dejstvo, da lahko posamezno stran obiščemo po naključju, brez da bi do nje vodila povezava iz neke druge strani.

PageRank lahko izračunamo iterativno. Algoritem doseže konvergenco precej hitro. V izvornem članku [22] avtorja omenjata, da je za podatkovno bazo, ki vsebuje 322 milijonov povezav potrebnih 52 iteracij.

Postopek:
1. Nastavimo t=0
2. Nastavimo začetno vrednost $R(p_i;t) = 1/N$ za vse spletne strani.
3. Ponavljamo, dokler ne dosežemo ustavitvenega pogoja $||R(\mathbf{p};t+1) - R(\mathbf{p};t)|| < \epsilon$:

$$R(p_i;t+1) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{R(p_j;t)}{L(p_j)}$$

Slika 5: Algoritem za izračun PageRank.

Iterativno enačbo v koraku 3, lahko zapišemo tudi v matrični notaciji:

$$\mathbf{R}(t+1) = d\mathbf{M}\mathbf{R}(t) + \frac{1-d}{N}\mathbf{1},$$

kjer je $\mathbf{R}(t)$ stolpični vektor rangov v koraku t, $\mathbf{1}$ stolpični vektor samih enic, $\mathbf{M}$ pa matrika definirana kot

$$\mathbf{M}_{ij} = \begin{cases} L(p_j), & p_j \to p_i \\ 0, & \text{sicer.} \end{cases} \tag{2}$$

4.1 Naloga

Kljub temu, da je sam algoritem za rangiranje spletnih strani precej enostaven, hitro naletimo na težavo, da je število spletnih strani ogromno (trenutno že preko 50 milijard), kar seveda pomeni, da s serijskim algoritmom ne bomo prišli daleč. Ideja je torej, da pripravimo vzporedno različico algoritma, ki bo prišla do rešitve hitreje. Napišite vzporedno različico algoritma s pomočjo tehnologije OpenCL in večnitenja (pThreads ali OpenMP) in ju primerjajte s sekvenčnim algoritmom. Za testni nabor podatkov lahko uporabite bazo, ki jo je objavil Google (https://snap.stanford.edu/data/web-Google.html).

Nekaj Idej:

- Primerjajte učinkovitost in pohitritev za različno število vozlišč.
- Razmislite in pripravite učinkovit način hranjenja podatkovnih struktur, ki opisujejo povezave med stranmi.
- Pri procesiranju uporabite več grafičnih kartic.
- Primerjajte vzporedno rešitev na GPU z vzporedno rešitvijo na CPU (časi, pohitritve)