Lab 6: PageRank
CSC 466: Knowledge Discovery in Data
Section 3

Author: Matthew Jaojoco

Overview:

For my implementation of pageRank I based my source code on the pseudocode provided for the typical iterative approach to pageRank. I originally attempted to implement this algorithm traditionally, comparing the sum of differences to an epsilon value that is passed to the algorithm, but I found that these sums turned negative after the first iterative assignment of pageRank values. So instead, my code takes as an argument the number of iterations to be run when given a small dataset. It still checks the comparison to epsilon when given one of the snap datasets.

My implementation skips vertices that are sinks in the summation when calculating pageRank values since sinks have an out-degree of zero and this would cause a division by zero error in the equation being calculated. In “the traversal”, this would cause the process to pick a new random vertex to visit, but I am unsure how to deal with sinks in this iterative approach, so I simply skip them in the calculation.

Results

Small Datasets

For the small datasets, I chose to run pageRank with 1000 iterations. I came to this number since 10 iterations and 10,000 iterations both returned very similar orderings for and pagerank values for all of the vertices in the graph. Each iteration is completed very quickly, so even though 10 iterations produces similar results, I figure there’s no harm in iterating more so long as there is still some variance in pagerank values for the different nodes.

<u>NCAA_Football</u>

Read time: 0.01393270492553711
Processing time: 0.7069649696350098
iterations: 1000
1:	Boise State with pageRank: 0.009261041711645157
2:	Utah with pageRank: 0.009126607548230385
3:	Montana with pageRank: 0.008972621447362258

4: Tulsa with pageRank: 0.008467386380859181
5: Ball State with pageRank: 0.008052580094228877
6: Rice with pageRank: 0.008043287932074669
7: Texas with pageRank: 0.008029812001547438
8: Alabama with pageRank: 0.007904587982841268
9: Florida with pageRank: 0.007879079673275061
10: TCU with pageRank: 0.007725905414115372
11: Texas Tech with pageRank: 0.00761179753950679
12: Penn State with pageRank: 0.007360908357577163
13: Cincinnati with pageRank: 0.007264718814440454
14: Oklahoma with pageRank: 0.00718587001634095
15: Richmond with pageRank: 0.006927469528167359
16: East Carolina with pageRank: 0.006688716268513091
17: James Madison with pageRank: 0.006674766521010232
18: Missouri with pageRank: 0.0066217395154335775
19: Ohio State with pageRank: 0.006587656555517754
20: Brigham Young with pageRank: 0.006499451023556769
21: Weber State with pageRank: 0.006484915333638033
22: Nebraska with pageRank: 0.006375996200346509
23: New Hampshire with pageRank: 0.006309484112733927
24: USC with pageRank: 0.006303464254990896
25: Georgia with pageRank: 0.006268874545645051
26: Michigan State with pageRank: 0.006209533487268718
27: Iowa with pageRank: 0.0062029592904825764
28: Western Michigan with pageRank: 0.006175884533654946
29: Houston with pageRank: 0.0061595911914818224
30: Northwestern with pageRank: 0.006131845737543209
31: Troy with pageRank: 0.006082930529858175
32: Pittsburgh with pageRank: 0.006063045936416921
33: Louisiana Tech with pageRank: 0.00601800800081204
34: Air Force with pageRank: 0.005969834852226772
35: Oklahoma State with pageRank: 0.005969648957722971
36: Buffalo with pageRank: 0.005922942350488311
37: West Virginia with pageRank: 0.005909393965229819
38: Virginia Tech with pageRank: 0.0059033634826189466
39: Villanova with pageRank: 0.005830625615335518
40: Mississippi with pageRank: 0.005817054060061071
41: LSU with pageRank: 0.005794476761505225
42: South Carolina State with pageRank: 0.0057914816051896165
43: Navy with pageRank: 0.005755794620734134

44: Oregon with pageRank: 0.005745139642327273
45: Boston College with pageRank: 0.005728669939166972
46: Fresno State with pageRank: 0.0056172281202073425
47: Rutgers with pageRank: 0.0055663391673920586
48: Florida Atlantic with pageRank: 0.005560032639007111
49: Florida A&M with pageRank: 0.005539139788755795
50: Connecticut with pageRank: 0.005523075284128811
51: Kansas with pageRank: 0.005518054636663345
52: Southern Miss with pageRank: 0.0054891279252995435
53: Colorado State with pageRank: 0.0054794569941693134
54: Nevada with pageRank: 0.005463804082343416
55: South Florida with pageRank: 0.005445796930211678
56: Florida State with pageRank: 0.005434730742050978
57: Appalachian State with pageRank: 0.00542229121058866
58: Maine with pageRank: 0.005411098803343747
59: San Jose State with pageRank: 0.005317548216907874
60: Georgia Tech with pageRank: 0.0052656417176762775
61: Central Michigan with pageRank: 0.005257908996404258
62: Wake Forest with pageRank: 0.005098986858221058
63: Minnesota with pageRank: 0.0050363386236504225

Generally, it seems like this algorithm worked well on this dataset, putting teams with more games generally higher in the rankings. It doesn’t work perfectly since Franklin has only 1 game in the dataset, yet appears over 4 other colleges that have more games, but it does somewhat accurately order the prestige of the colleges based on which other colleges play them. Ordering between items with similar rankings seems to be off, but generally important colleges make their way to the top and unimportant colleges make their way to the bottom.

Karate

Read time: 0.003654956817626953
Processing time: 0.07393002510070801
iterations: 1000

1: 34.0 with pageRank: 0.10091918233262573
2: 1.0 with pageRank: 0.09699728538829479
3: 33.0 with pageRank: 0.07169322600575445
4: 3.0 with pageRank: 0.05707850948846202
5: 2.0 with pageRank: 0.05287692406114573
6: 32.0 with pageRank: 0.03715808706914529
7: 4.0 with pageRank: 0.03585985778641138

8: 24.0 with pageRank: 0.03152251477667803
9: 9.0 with pageRank: 0.029766056081016956
10: 14.0 with pageRank: 0.02953645615191367
11: 6.0 with pageRank: 0.02911115467837016
12: 7.0 with pageRank: 0.02911115467837016
13: 30.0 with pageRank: 0.026288537695114873
14: 28.0 with pageRank: 0.025639767482847974
15: 31.0 with pageRank: 0.024590155248580117
16: 8.0 with pageRank: 0.02449049703528238
17: 5.0 with pageRank: 0.021977952364589545
18: 11.0 with pageRank: 0.021977952364589545
19: 25.0 with pageRank: 0.021076033559222815
20: 26.0 with pageRank: 0.021006197394492998
21: 20.0 with pageRank: 0.019604636325652788
22: 29.0 with pageRank: 0.019573459463828494
23: 17.0 with pageRank: 0.016784005444189672
24: 27.0 with pageRank: 0.015044038082725551
25: 13.0 with pageRank: 0.014644892011877127
26: 18.0 with pageRank: 0.0145586772090215
27: 22.0 with pageRank: 0.0145586772090215
28: 15.0 with pageRank: 0.014535993997921246
29: 16.0 with pageRank: 0.014535993997921246
30: 19.0 with pageRank: 0.014535993997921246
31: 21.0 with pageRank: 0.014535993997921246
32: 23.0 with pageRank: 0.014535993997921246
33: 10.0 with pageRank: 0.01430939712903291
34: 12.0 with pageRank: 0.009564745492135514

The pagerank algorithm seemed to handle this dataset very well, giving an accurate representation of each fighter's prestige compared to the rest

Karate

Read time: 0.005218029022216797
Processing time: 0.12249970436096191
iterations: 1000
1: Grin with pageRank: 0.03214449277924281
2: Jet with pageRank: 0.03172814041038732
3: Trigger with pageRank: 0.0312993569046432
4: Web with pageRank: 0.03009537138486299

5: SN4 with pageRank: 0.029875338603275115
6: Topless with pageRank: 0.029514203581660632
7: Scabs with pageRank: 0.028423069732888294
8: Patchback with pageRank: 0.026458550192380536
9: Gallatin with pageRank: 0.026156875978439388
10: Beescratch with pageRank: 0.0246507168492958
11: Kringel with pageRank: 0.024640918846873926
12: SN63 with pageRank: 0.02393924439956635
13: Feather with pageRank: 0.023458479343835405
14: SN9 with pageRank: 0.021966365996160114
15: Stripes with pageRank: 0.021691125113836862
16: Upbang with pageRank: 0.021650877069331838
17: SN100 with pageRank: 0.020613389435301055
18: DN21 with pageRank: 0.0200536292995076
19: Haecksel with pageRank: 0.019883081322628417
20: Jonah with pageRank: 0.01939555066603905
21: TR99 with pageRank: 0.019231944692991634
22: SN96 with pageRank: 0.017618650342927474
23: TR77 with pageRank: 0.017339518025116643
24: Number1 with pageRank: 0.01713009110369239
25: Double with pageRank: 0.017098300843550903
26: Beak with pageRank: 0.016965391725109573
27: MN105 with pageRank: 0.016938990082180954
28: MN83 with pageRank: 0.0169057559952821
29: Hook with pageRank: 0.016626816408350314
30: SN90 with pageRank: 0.016137566914762324
31: Shmuddel with pageRank: 0.015919935882277445
32: DN63 with pageRank: 0.015643030468627007
33: PL with pageRank: 0.015302095254216337
34: Fish with pageRank: 0.015108397537453262
35: Oscar with pageRank: 0.014845738594652352
36: Zap with pageRank: 0.014767919980768464
37: DN16 with pageRank: 0.014428047632488662
38: Bumper with pageRank: 0.013338079741907372
39: Ripplefluke with pageRank: 0.013308677267526683
40: Knit with pageRank: 0.012928199949535306
41: Thumper with pageRank: 0.012830828492399735
42: TSN103 with pageRank: 0.012072590588843475
43: Mus with pageRank: 0.01150422291928506
44: Notch with pageRank: 0.01121013881870583

45: Zipfel with pageRank: 0.011039191065697548
46: MN60 with pageRank: 0.009863497611071175
47: CCL with pageRank: 0.00962906209347306
48: TR88 with pageRank: 0.008876749641654181
49: TR120 with pageRank: 0.008825895771664323
50: Wave with pageRank: 0.00832624577804376
51: TSN83 with pageRank: 0.008181047975908934
52: SN89 with pageRank: 0.007764749900931707
53: Vau with pageRank: 0.007494174336209228
54: Zig with pageRank: 0.00619014673117557
55: MN23 with pageRank: 0.005415901433024036
56: Quasi with pageRank: 0.005415901433024036
57: TR82 with pageRank: 0.005261695469502293
58: Cross with pageRank: 0.005079800175604349
59: Five with pageRank: 0.005079800175604349
60: Whitetip with pageRank: 0.004962899556163603
61: SMN5 with pageRank: 0.0049182179124345055
62: Fork with pageRank: 0.004835315766005183

The pagerank algorithm seems to have worked very well on this dataset, giving higher ranks to the more social dolphins.

Les Miserables

Read time: 0.00500798225402832
Processing time: 0.17928504943847656
iterations: 1000
1: Valjean with pageRank: 0.0754301216327847
2: Myriel with pageRank: 0.042779281022712105
3: Gavroche with pageRank: 0.03576731819472972
4: Marius with pageRank: 0.03089493621512294
5: Javert with pageRank: 0.030302735905813585
6: Thenardier with pageRank: 0.02792652569403349
7: Fantine with pageRank: 0.027022704917205688
8: Enjolras with pageRank: 0.021882033328144535
9: Cosette with pageRank: 0.020611215084857623
10: MmeThenardier with pageRank: 0.019501134691061094
11: Bossuet with pageRank: 0.01895953011022145
12: Courfeyrac with pageRank: 0.01857844249302667
13: Eponine with pageRank: 0.01779391186324403

14: Mabeuf with pageRank: 0.017478022290751016
15: Bahorel with pageRank: 0.017199878018386
16: Joly with pageRank: 0.017199878018386
17: Gueulemer with pageRank: 0.01669183803628071
18: Babet with pageRank: 0.01669183803628071
19: Claquesous with pageRank: 0.016561020318792925
20: MlleGillenormand with pageRank: 0.016260208629667382
21: Combeferre with pageRank: 0.015892124698100082
22: Feuilly with pageRank: 0.015892124698100082
23: Tholomyes with pageRank: 0.01564742736848247
24: Bamatabois with pageRank: 0.01557626472102227
25: Montparnasse with pageRank: 0.015170928493589987
26: Gillenormand with pageRank: 0.014957475581397036
27: Grantaire with pageRank: 0.014456669473094033
28: Prouvaire with pageRank: 0.013145880776009446
29: Listolier with pageRank: 0.012618202914107791
30: Fameuil with pageRank: 0.012618202914107791
31: Blacheville with pageRank: 0.012618202914107791
32: Favourite with pageRank: 0.012618202914107791
33: Dahlia with pageRank: 0.012618202914107791
34: Zephine with pageRank: 0.012618202914107791
35: Judge with pageRank: 0.012424659363823549
36: Champmathieu with pageRank: 0.012424659363823549
37: Brevet with pageRank: 0.012424659363823549
38: Chenildieu with pageRank: 0.012424659363823549
39: Cochepaille with pageRank: 0.012424659363823549
40: Brujon with pageRank: 0.0118666609426886
41: Fauchelevent with pageRank: 0.011638047873644026
42: MmeHucheloup with pageRank: 0.010689825140848526
43: MlleBaptistine with pageRank: 0.010277134629737837
44: MmeMagloire with pageRank: 0.010277134629737837
45: Simplice with pageRank: 0.00907364696813507
46: LtGillenormand with pageRank: 0.008713597430429626
47: MmeBurgon with pageRank: 0.0078055815514132395
48: Pontmercy with pageRank: 0.007368097201041265
49: Woman2 with pageRank: 0.006836862528350659
50: Toussaint with pageRank: 0.006836862528350659
51: Anzelma with pageRank: 0.006313538586561964
52: MotherInnocent with pageRank: 0.006202126104197609
53: MmePontmercy with pageRank: 0.006010133393378012

54: Child1 with pageRank: 0.005791254017601827
55: Child2 with pageRank: 0.005791254017601827
56: Napoleon with pageRank: 0.005584290834982478
57: CountessDeLo with pageRank: 0.005584290834982478
58: Geborand with pageRank: 0.005584290834982478
59: Champtercier with pageRank: 0.005584290834982478
60: Cravatte with pageRank: 0.005584290834982478
61: Count with pageRank: 0.005584290834982478
62: OldMan with pageRank: 0.005584290834982478
63: Perpetue with pageRank: 0.00540748854075564
64: Magnon with pageRank: 0.0052712227025698164
65: Jondrette with pageRank: 0.005265424107402575
66: Marguerite with pageRank: 0.005260327543023243
67: Woman1 with pageRank: 0.0052441777263389335
68: BaronessT with pageRank: 0.005146458723386863
69: Gribier with pageRank: 0.004421137121201304
70: MlleVaubois with pageRank: 0.003922505853082988
71: Labarre with pageRank: 0.003729040931048254
72: MmeDeR with pageRank: 0.003729040931048254
73: Isabeau with pageRank: 0.003729040931048254
74: Gervais with pageRank: 0.003729040931048254
75: Scaufflaire with pageRank: 0.003729040931048254
76: Boulatruelle with pageRank: 0.0034316486255474773
77: MotherPlutarch with pageRank: 0.0032986263977917997

The pagerank algorithm seemed to work extremely well for this dataset. It correctly identifies Jean Valjean as the main character of the play, and puts other main characters such as Javert and Fantine near the top as well. It fails to recognize the importance of impactful characters that only have a brief appearance, but this is unavoidable with the implementation of pagerank that is being used.

Results

Snap Datasets

For the snap datasets that are significantly larger, I decided to only iterate until the stoppage condition mentioned in the lecture notes is met, with epsilon set to 0. For wiki-vote it was feasible to run on 1000 iterations and compare the outputs. From what I observed, the outputs appeared to be similar so I figured it wasn't worth running extra iterations for these larger datasets.

Wiki Vote

Read time: 7.123586654663086
Processing time: 0.07787609100341797
iterations: 3
1: 2565 with pageRank: 0.008721538241739643
2: 766 with pageRank: 0.007528892219881926
3: 11 with pageRank: 0.006990255842402874
4: 457 with pageRank: 0.006595725805808572
5: 2688 with pageRank: 0.005703485998826735
6: 1166 with pageRank: 0.005294113869357573
7: 1549 with pageRank: 0.0051969051039328085
8: 1374 with pageRank: 0.004286978951785181
9: 1151 with pageRank: 0.004189780831232542
10: 5524 with pageRank: 0.0038864115846128453
11: 5802 with pageRank: 0.0036349211355400524
12: 1133 with pageRank: 0.0036080882034037892
13: 3642 with pageRank: 0.0035407457567019017
14: 4967 with pageRank: 0.003433589264337928
15: 2972 with pageRank: 0.003377285589691796
16: 1608 with pageRank: 0.0032072826689545537
17: 3449 with pageRank: 0.0031466740273047578
18: 5189 with pageRank: 0.00312968379569238
19: 2658 with pageRank: 0.003014824620682024
20: 3453 with pageRank: 0.003005798943619421
21: 1098 with pageRank: 0.0030001966955442963
22: 6 with pageRank: 0.002988725071044045
23: 2485 with pageRank: 0.002979130502264339
24: 68 with pageRank: 0.002972839890505104
25: 173 with pageRank: 0.0029196335326140094
26: 789 with pageRank: 0.002890120164957478
27: 5079 with pageRank: 0.002780293108766456
28: 24 with pageRank: 0.0027775826416825866
29: 4310 with pageRank: 0.002760475671827219
30: 722 with pageRank: 0.0027092543733103513

31: 1305 with pageRank: 0.0026971810631790066
32: 3447 with pageRank: 0.002695325074165248
33: 2871 with pageRank: 0.0026747056171032946
34: 996 with pageRank: 0.002652516889089288
35: 988 with pageRank: 0.002621588971834529
36: 3352 with pageRank: 0.0025760996823405687
37: 4045 with pageRank: 0.0025576229153783927
38: 311 with pageRank: 0.002470205531686817
39: 2967 with pageRank: 0.0024202883080426704
40: 737 with pageRank: 0.002375985709981358
41: 813 with pageRank: 0.002355296466820173
42: 5800 with pageRank: 0.0023384302191562334
43: 306 with pageRank: 0.002324453135919089
44: 3787 with pageRank: 0.0023063811013607372
45: 2651 with pageRank: 0.002304872794597472
46: 5531 with pageRank: 0.002294031875518876
47: 4632 with pageRank: 0.002288284023645477
48: 312 with pageRank: 0.002263096302992538
49: 5179 with pageRank: 0.002259515745465002
50: 1542 with pageRank: 0.002250949590422378
51: 3614 with pageRank: 0.002249700384625857
52: 3456 with pageRank: 0.002240028445716048
53: 5697 with pageRank: 0.002213304398413412
54: 1210 with pageRank: 0.002201186137107734
55: 2326 with pageRank: 0.002195741932351278
56: 4948 with pageRank: 0.002178875091170554
57: 826 with pageRank: 0.0021546229607223816
58: 3032 with pageRank: 0.002150136012014973
59: 2256 with pageRank: 0.0021402059431476273
60: 310 with pageRank: 0.002139626484871775
61: 2237 with pageRank: 0.0021282090902516645
62: 5188 with pageRank: 0.0021067824584113147
63: 600 with pageRank: 0.002099804263841493
64: 993 with pageRank: 0.002059792665872878
65: 8 with pageRank: 0.002052429421628067
66: 122 with pageRank: 0.001968157105335653
67: 6907 with pageRank: 0.0019525244190718078
68: 1395 with pageRank: 0.0019341072109179218
69: 36 with pageRank: 0.001930719242663684
70: 6305 with pageRank: 0.0019134691454771719

When observing a handful of the nodes at the very bottom and very top of the listing, it appears that the algorithm works well for this dataset. It appears that users higher on the list are recommended more frequently, or by users that are frequently recommended. The algorithm also takes into account when a user votes for many users, and properly adjusts its calculations in response. This process seems to be extremely effective in this scenario where qualification to be an administrator needs to be quantified and ranked.

Slashdot Zoo

Read time: 284.3285582065582
Processing time: 0.6906208992004395
iterations: 3
1: 642 with pageRank: 0.0009190260416232486
2: 190 with pageRank: 0.0009041283130145637
3: 3124 with pageRank: 0.0008984640648436576
4: 8794 with pageRank: 0.0008975065107456393
5: 10274 with pageRank: 0.000897036045824318
6: 6487 with pageRank: 0.0008906883480283991
7: 7262 with pageRank: 0.0008880176938199646
8: 1491 with pageRank: 0.0008840269962635668
9: 1287 with pageRank: 0.0008659314993264025
10: 1990 with pageRank: 0.0008604457525204207
11: 9130 with pageRank: 0.0008472573865482842
12: 5071 with pageRank: 0.000845779723204766
13: 2784 with pageRank: 0.0008349052959805843
14: 11065 with pageRank: 0.000833898071568622
15: 825 with pageRank: 0.0008212681455087507
16: 13001 with pageRank: 0.0008191599270326799
17: 184 with pageRank: 0.0008155697754028222
18: 4175 with pageRank: 0.0008138003648799463
19: 3041 with pageRank: 0.0008095713919351931
20: 173 with pageRank: 0.0008059044617318716
21: 3588 with pageRank: 0.0008036426531935215
22: 2328 with pageRank: 0.0007965200541156605
23: 4808 with pageRank: 0.0007916279078793095
24: 2666 with pageRank: 0.0007888573518351636
25: 4261 with pageRank: 0.0007773836688983247
26: 5027 with pageRank: 0.0007756927007529474

27: 1389 with pageRank: 0.0007727842381054478
28: 1339 with pageRank: 0.00076363757671257
29: 522 with pageRank: 0.0007633409107166967
30: 2300 with pageRank: 0.0007534565116786476
31: 1200 with pageRank: 0.0007481631880034753
32: 798 with pageRank: 0.0007426981841536485
33: 8 with pageRank: 0.0007408158804453353
34: 1694 with pageRank: 0.000732495868857324
35: 791 with pageRank: 0.000732080011489887
36: 3414 with pageRank: 0.0007266552272799037
37: 936 with pageRank: 0.0007234564790398564
38: 7084 with pageRank: 0.000720002896303837
39: 62 with pageRank: 0.0007178145103025602
40: 12821 with pageRank: 0.000706709010133388
41: 8091 with pageRank: 0.0007027703952593168
42: 6453 with pageRank: 0.0007001821345367942
43: 3891 with pageRank: 0.0006998609294413021
44: 4949 with pageRank: 0.0006944443312092934
45: 1381 with pageRank: 0.0006912878972763399
46: 653 with pageRank: 0.0006896410288978241
47: 4707 with pageRank: 0.000679334821944057
48: 82 with pageRank: 0.0006724322631847627
49: 25103 with pageRank: 0.0006684705388999846
50: 1325 with pageRank: 0.0006653949728445193
51: 913 with pageRank: 0.0006600603477249789
52: 142 with pageRank: 0.0006535253399755296
53: 7526 with pageRank: 0.0006266213287855316
54: 1053 with pageRank: 0.0006257845674630135
55: 7462 with pageRank: 0.0006229782324655343
56: 898 with pageRank: 0.0006220865242965425
57: 8687 with pageRank: 0.0006201565185999672
58: 822 with pageRank: 0.0006139835455661441
59: 8611 with pageRank: 0.0006138425674190573
60: 6780 with pageRank: 0.0006103220546695803

This dataset contains data about friendships between users on a social media site, very similar to scenarios that we went over in class using pageRank. pageRank is a very effective algorithm for this scenario, and identifies important users, considering the importance of each user's friends. Users with many friends are placed higher in the output list, and friends of popular users also appear higher.

Still waiting on this to finish running so for now I will provide Gnutella data

Amazon May

Gnutella Peer 2 Peer

Read time: 1.7818889617919922
Processing time: 0.011790037155151367
iterations: 2

1:	3451 with pageRank: 0.0016144660894660892
2:	6319 with pageRank: 0.0014642135642135637
3:	3002 with pageRank: 0.0012021315640880856
4:	4003 with pageRank: 0.0011115800865800864
5:	4990 with pageRank: 0.001068037518037518
6:	5308 with pageRank: 0.0010324675324675323
7:	7525 with pageRank: 0.0009932178932178933
8:	5103 with pageRank: 0.0009883116883116882
9:	8041 with pageRank: 0.000971139971139971
10:	8277 with pageRank: 0.0009662337662337663
11:	7644 with pageRank: 0.0008607503607503608
12:	4745 with pageRank: 0.0008582972582972584
13:	7012 with pageRank: 0.0008558441558441559
14:	8308 with pageRank: 0.0008362193362193363
15:	2262 with pageRank: 0.0008356936714079572
16:	926 with pageRank: 0.0008251803751803752
17:	209 with pageRank: 0.0008215007215007214
18:	4684 with pageRank: 0.0008165945165945166
19:	7605 with pageRank: 0.0008116883116883117
20:	4551 with pageRank: 0.0007961519961519961
21:	2172 with pageRank: 0.000792063492063492
22:	7056 with pageRank: 0.000792063492063492
23:	4785 with pageRank: 0.0007856588403386433
24:	8261 with pageRank: 0.0007845154845154845
25:	7108 with pageRank: 0.000773051948051948
26:	2588 with pageRank: 0.00076998556998557
27:	7239 with pageRank: 0.00076998556998557
28:	1113 with pageRank: 0.0007675324675324675
29:	7044 with pageRank: 0.0007675324675324675
30:	6019 with pageRank: 0.0007675324675324674
31:	8815 with pageRank: 0.0007479076479076478

32: 5634 with pageRank: 0.0007479076479076479
33: 535 with pageRank: 0.0007454545454545455
34: 5029 with pageRank: 0.0007454545454545455
35: 7247 with pageRank: 0.0007454545454545455
36: 8807 with pageRank: 0.0007454545454545455
37: 1186 with pageRank: 0.0007288961038961039
38: 8712 with pageRank: 0.0007258297258297257
39: 3228 with pageRank: 0.0007233766233766234
40: 5837 with pageRank: 0.0007233766233766234
41: 7349 with pageRank: 0.0007233766233766234
42: 6048 with pageRank: 0.000711111111111111
43: 5701 with pageRank: 0.0007037518037518038
44: 4631 with pageRank: 0.0006822548796233007
45: 6870 with pageRank: 0.0006816738816738817
46: 5510 with pageRank: 0.0006620490620490621
47: 818 with pageRank: 0.0006620490620490619
48: 7262 with pageRank: 0.0006595959595959596
49: 7640 with pageRank: 0.0006595959595959596
50: 3953 with pageRank: 0.0006571428571428571
51: 4802 with pageRank: 0.0006571428571428571
52: 6478 with pageRank: 0.0006571428571428571
53: 6058 with pageRank: 0.0006524994846423418
54: 2143 with pageRank: 0.000650834879406308
55: 341 with pageRank: 0.0006497835497835498
56: 8830 with pageRank: 0.0006485569985569985
57: 5055 with pageRank: 0.0006473304473304473
58: 8423 with pageRank: 0.0006448773448773449
59: 5265 with pageRank: 0.0006424242424242424
60: 4233 with pageRank: 0.0006405844155844155
61: 1068 with pageRank: 0.0006399711399711401
62: 7818 with pageRank: 0.0006399711399711401
63: 1781 with pageRank: 0.00063997113997114
64: 2341 with pageRank: 0.00063997113997114
65: 2410 with pageRank: 0.00063997113997114
66: 2952 with pageRank: 0.00063997113997114
67: 3663 with pageRank: 0.00063997113997114
68: 3976 with pageRank: 0.00063997113997114
69: 1662 with pageRank: 0.0006375180375180376
70: 2722 with pageRank: 0.0006375180375180376

This dataset appears to present a scenario in which pageRank can be useful. The algorithm orders the users so that users with more prestige (and thus likely more trustworthy in this p2p network) are presented higher on the list, according to which users they interact with in the network.

Overall Summary

Generally, this implementation of the pageRank algorithm performed extremely well when looking at the general relevance of a given node in the graph.

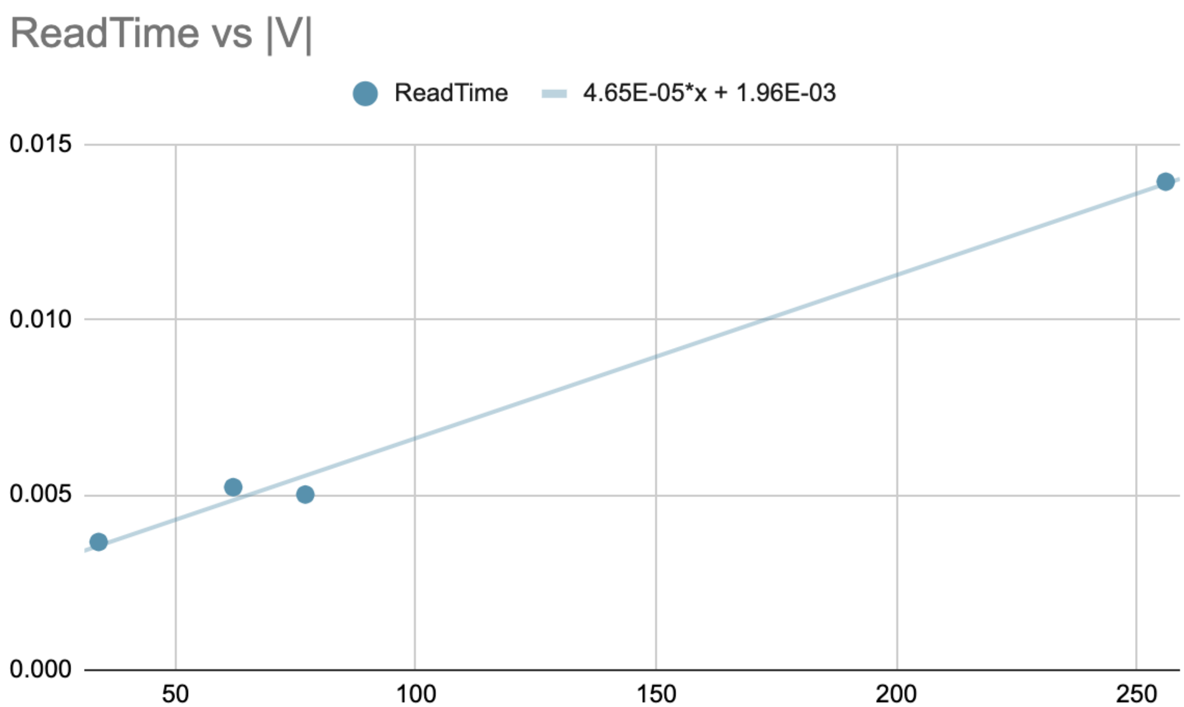
Looking back on performance of the code, most of the runtime is in reading in the values and creating the graph, so I could have afforded to predetermine the number of iterations and forced the algorithm to iterate further, but I assume that like Wiki-vote, the other 2 snap datasets will have marginal changes in the output when forced to iterate past the stopping condition with epsilon set to 0.

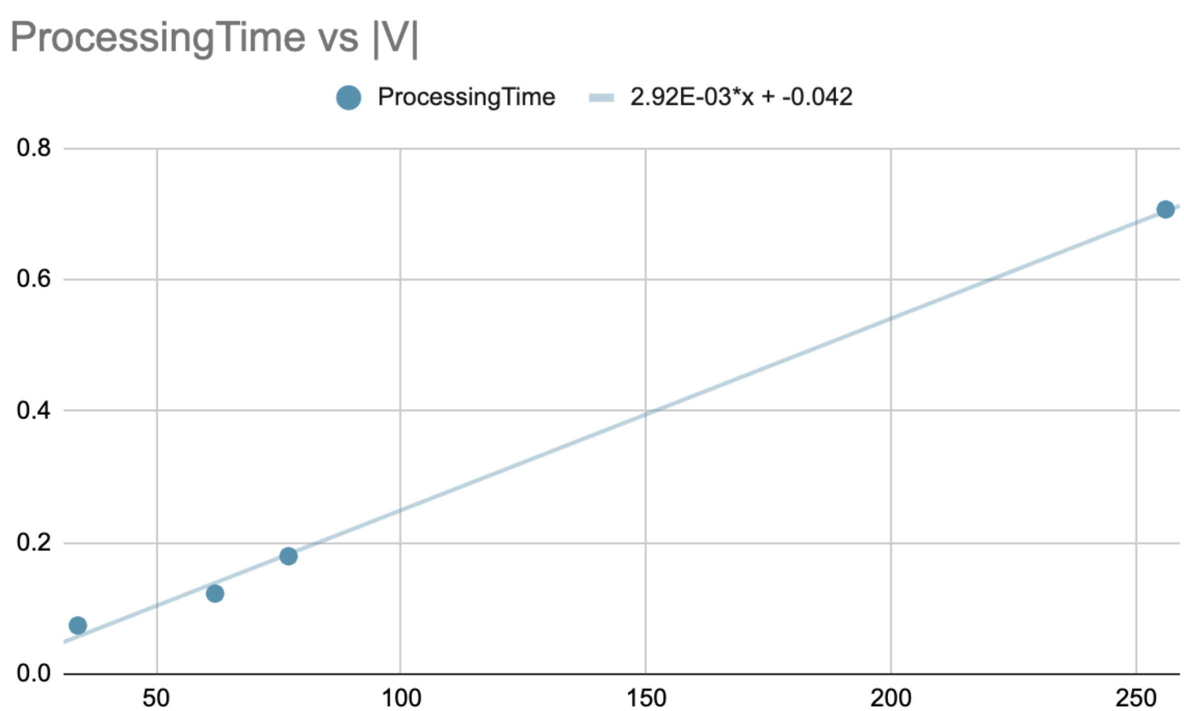
In datasets such as Les Miserables, there are some qualities in each vertice that cannot be represented by the number of interactions with other edges or importance of the vertices it has an interaction with. In Les Miserables this would be qualities such as their purpose within the plot and whether or not they impact the flow of events in the story. A character could also have a single dialogue that is more impactful than 5 conversations from a different character. The importance of interactions is not considered, only the importance of the characters as determined by number of appearances. From this, I conclude that the pageRank algorithm works less optimally in scenarios where not all edges should be considered to have equal value.

The algorithm still worked well in these cases (when not all edges should be considered equal), there are just some inaccuracies. Some characters with an early death or late introduction are listed below unnamed characters just because they don’t appear as often as others. This could be solved if not all edges were valued the same. It worked even better in scenarios where all interactions between two vertices can be treated the same with little to no repercussion, such as the Slashdot dataset and the dolphins dataset (assuming you are trying to find the user or dolphin with the most social prestige).

Runtime Analysis

Graphs for Small Datasets

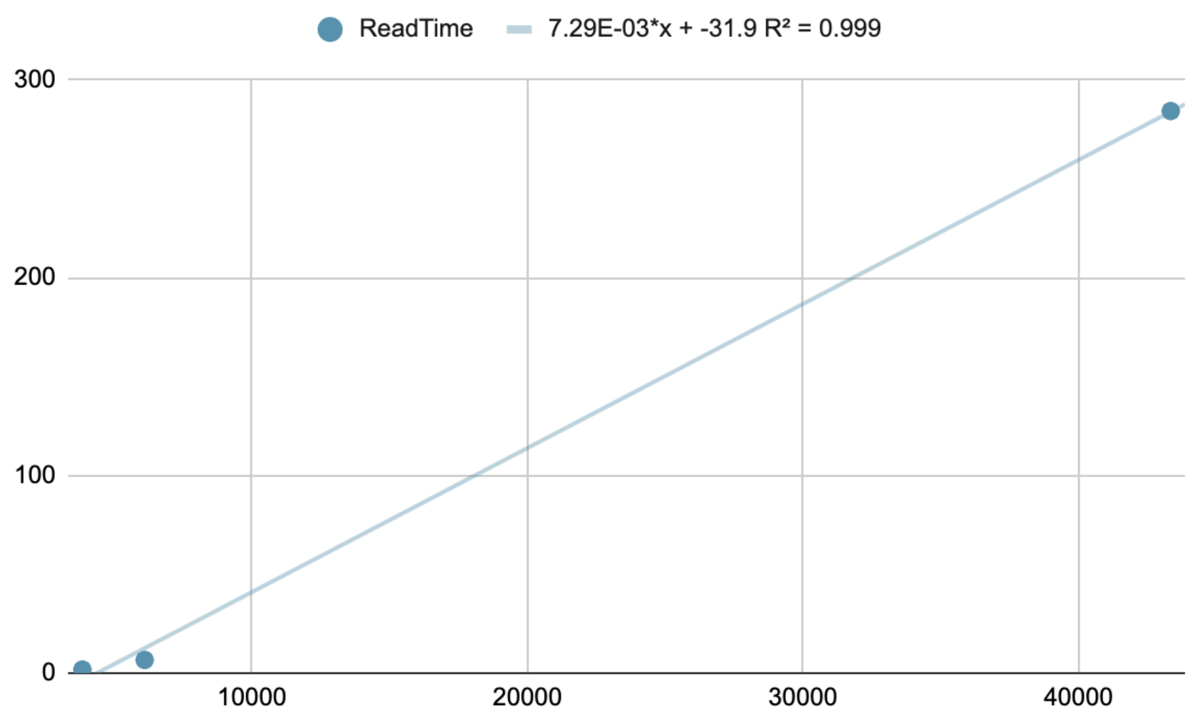

ReadTime vs |V|
ReadTime
4.65E-05*x + 1.96E-03
0.015
0.010
0.005
0.000
50
100
150
200
250

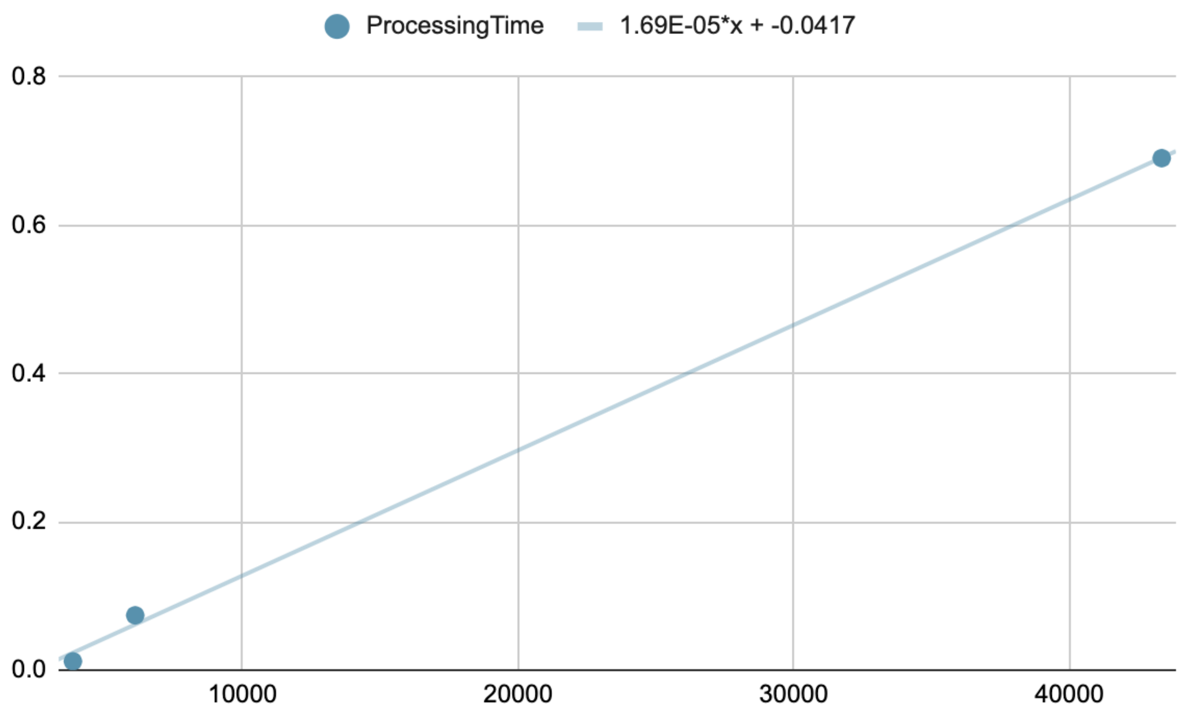

ProcessingTime vs |V|
ProcessingTime
2.92E-03*x + -0.042
0.8
0.6
0.4
0.2
0.0
50
100
150
200
250

For the simple datasets, the runtimes for both reading and processing the data fit well to a linear model, where each node in the graph adds 0.0000465 sec to read time and 0.00292 sec to processing time.

All of my datasets converge on the third iteration (second if you don't count the 0th iteration), but I have the pageRank algorithm continue for 1000 iterations regardless. Since it always executes 1000 iterations, that means a single iteration takes 0.00000292 sec on average. So a single iteration takes significantly less time than reading the data and building the graph.

Graphs for Snap Datasets





For the snap datasets, we can observe that the trends for the runtimes still follow for bigger datasets. Since the algorithm only iterates 2-3 times in each dataset, we can see that it still takes significantly less time to iterate once over the nodes than to set up the graph, so it is probably worth iterating more times than less, since it won't cost you much time and the behavior of the pageRank values converging will prevent further iterations from ruining the accuracy of the results.

Appendix: README

```
Author: Matthew Jaojoco
pageRank.py stores the implementation for my pageRank algorithm. It takes in either
epsilon or iterations depending on the format of the file that stores the graph.
It returns the pageRank of each node in the graph in descending order,
as well as the number of iterations and the time it took to
    1. construct the graph and 2. calculate pageRank

usage: python3 pageRank.py <filename> <iterations/epsilon> <small/snap>
    <small/snap> is a string that represents the format of the data.
    <iterations/epsilon> is an integer/float representing the corresponding variable
        iterations is used for small formatted data
        epsilon is used for snap formatted data

note: the algorithm always attempts the snap format if anything other than
    "small" is provided for the final argument
```