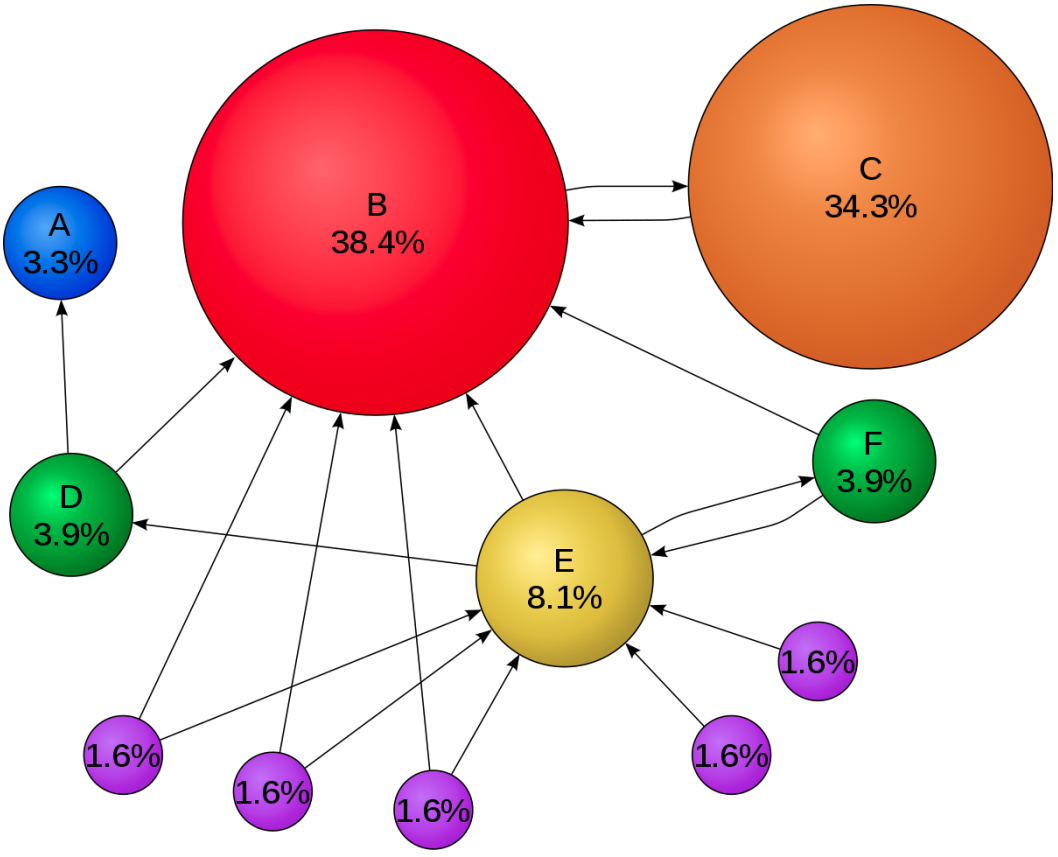
Lab 5

Page Rank



Ramon Novell Mora

Bartomeu Perelló Comas

26/11/2020

CAIM - GEI FIB

1- Estructura de codi:

Pel que fa al nostre codi, ho hem programat de manera que tenim les següents estructures de dades: AirportList, amb la llista de tots els aeroports i AirportHash amb el index de cada Aeroport dins la llista, indexats pel seu code. A cada Airport hi tenim una llista de routes, amb totes les destinacions a les quals aquests tenen rutes, també tenim routesHash, que és un diccionari que conté Edges, indexats pel seu origen. Aquests edges representen aeroports els quals tenen rutes cap a l'objecte actual. Per últim, la classe Airport també conté un outweight que representa el volum de rutes de sortida totals d'aquest aeroport. Tanmateix, Edge consta d'un origen, que representa l'aeroport d'on prové la ruta, un weight, que representa el pes d'aquesta ruta (es va incrementant a cada línia de routes.txt amb rutes repetides) i un index, que apunta al index de AirportList on es troba l'aeroport origen.

Partint d'aquesta estructura, el nostre codi llegeix els aeroports (omplint AirportList i AirportHash), llegeix les rutes amb readRoutes() que alhora completa els camps de routes i routesHash dels aeroports, aixi com incrementar els weights tant dels aeroports com de les rutes (representades en edges dins de routesHash).

Un cop fet això, ja podem executar ComputePageRanks(), en el nostre cas només hem realitzat una petita modificació a l'algorisme presentat, hem calculat quant aporta cada node que té *outweight* = 0 i ho hem dividit entre el nombre total de aeroports per a repartir el pes restant entre totes les asignacions.

Finalment el outputPageRanks() asigna a cada aeroport el seu pes i posteriorment és ordenat per ordre decreixent i se li assigna el pagerank corresponent.

2- Decisions a prendre

Pel que fa a les decisions que hem hagut de prendre en els espais de codi obert, hem optat per les següent:

-Afegir camp index a la classe Edge:

El qual apunta al index d'airportHash on es troba l'aeroport d'origen de la ruta.

-Damping factor:

La hem establert a 0.8 com ens recomana l'enunciat de la pràctica. A factors més petits, executa menys iteracions i els valors de Pagerank són més petits. A factors més grans augmenten considerablement les iteracions i convergeix a un damping factor més alt. Això es veu reflexat en la precisió dels resultats, a menys iteracions, menys temps d'execució i menys precisió i l'inversa a factors més propers a 1. Exemples d'experimentació:

Dumping factor	iteracions	temps d'execució
0.9	232	3.70s
0.8	111	1.66s
0.7	70	1.05s
0.5	37	0.55s
0.3	22	0.34s

També hem apreciat que passa una cosa semblant al modificar l'error màxim per al stopping factor, afecta lleugerament al nombre d'iteracions afectant a algunes posicions errònies en el PageRank, tal i com era esperat, tot i així no és tan rellevant com el damping factor.

-Stopping condition:

Ho hem organitzat de la següent manera, hem establert un màxim d'iteracions (en el nostre cas 1000), tot i així a cada iteració es comprova que la diferència del sumatori de Q (Pagerank) amb la iteració anterior tingui un error mínim (del 0.000000005 en el nostre cas), per tant, el bucle acabarà al cap de 1000 iteracions o quan deixi d'apreciar-se canvis en el valor del Pagerank (que hauria de convergir prop d'1).

3-Problemes obtinguts

Un dels principals problemes obtinguts a l'inici era que hem afegit un camp destí a la clase edge i treballavem amb una llista d'objectes Edge on cada

objecte era una ruta vàlida del fitxer routes.txt. Això, a part de causar-nos resultats erronis al calcular la fórmula del Pagerank, tenia un cost altíssim, i cada iteració superava els 20 segons per a computar-se. A part, tan sols ens feia 2 iteracions com a màxim fins a parar, donant-nos resultats erronis i incoherents. Hem intentat arreglar-ho augmentant el pes de les rutes repetides, de manera que aquestes nomes constaven d'una posició a la llista conjunt en comptes de totes per separat i en desordre, això millorava en molt poca mesura el cost de la funció i seguia donant resultats incoherents, a part de que la fórmula es feia molt més complexa. No ha estat fins que hem reestructurat el sistema de dades (entenent el funcionament de routes i routesHash de la classe airport) que els temps d'execució totals rondaven els pocs segons.

També relacionat amb això, hem tingut alguns problemes al decidir quins atributs afegiem i com els utilitzavem dins les clases donades, hem donat moltes voltes a quina seria la manera més fàcil i eficient d'estucturar el problema sense trencar cap de les restriccions d'eficiència de l'enunciat.

Per últim, el nostre algoritme, tot i haver seguit l'algoritme que s'indica a la pràctica al peu de la lletra, convergia a valors superiors a 1, depenent del damping factor més o menys allunyats. Tot i així després de donar-hi moltes voltes no hem sapigut veure on s'acumulava error i ens sembla que està ben implementat. De totes maneres el resultat obtingut és molt coherent i creiem que correcte i sembla que funciona com s'espera tret d'aquest petit detall. No hem sapigut entendre què podia estar malament en el plantejament de la funció.