# Pràctica 5: Pagerank

### Dificultats i evolució de la implementació

Al llarg d’aquesta pràctica ens hem anat trobant amb algunes dificultats que hem anat solucionant a través de la teoria i la recerca d’informació, fet que ha fet que la nostra implementació hagi anat evolucionant.

La primera dificultat, va estar entendre el funcionament de l'algorisme i de les classes implementades per a poder desenvolupar l’algorisme de forma eficient. Després de d’algunes errades i confusions amb les classes, diccionaris i vectors vam aconseguir crear un codi que creiem que funcionava.

Segona dificultat. Al compilar donava l’error: *Key error ‘TGK’*. Vam comprovar si aquest era un aeroport de la primera ruta per veure si fallavem en la forma de llegir-los, però no era el cas, i ens vam fixar en que aquest aeroport no es troba entre els aeroports de la llista airportList. Per tant, partint d’això, hem anat comprovant si l’aeroport d’origen o destí amb el que volem treballar es troba a la llista airportList abans de voler accedir a ell.

Per últim, un cop vam poder executar el codi sense errors, vam observar que la suma resultant dels pageranks era 0.6342355498561338 en comptes d’1. Vam repassar el codi i vam adonar-nos que aquesta diferència recau en que no consideràvem els nodes sense arestes de sortida. Així, vam modificar el codi per afegir a la suma de pesos, la puntuació extra que es donaria a cada node si els nodes sense sortides ara apuntessin a tots els nodes del graf. També vam adonar-nos que aquesta puntuació extra canvia a cada iteració i s’ha de recalcular per obtenir el nou pagerank d’aquests nodes sense sortides. D’aquesta manera s’implementa aquest càlcul de manera eficient i, un cop implementat, la suma resultant ja ens donava correcta (0.99999999..).

### Elecció dels paràmetres i conclusions

*Condició de stop*

Per escollir una bona condició de stop a les iteracions del càlcul del pagerank hem obtat per fixar-nos en la similitud de dues llistes de pageranks (P) d’iteracions consecutives. Aquesta similitud es basa en veure si els valors són prou semblants, fixant en la condició un cert nombre de decimals als quals arrodonim els ranks.

Hem fet proves a partir de 2 decimals i hem trobat que les diferències disminueixen a mesura que augmentem el nombre de decimals. Per observar la diferència en la posició dels aeroports hem utilitzat la següent comanda, generant fitxers amb els aeroports ordenats de major a menor rank:

diff -U 0 file1 file2 | grep -v ^@ | wc -l

Entre 2 i 3 tenim que existeixen 0 diferències, però si provem 4 decimals la diferència amb 3 es dispara a 6646. A partir d’aquí augmentar decimals disminueix les diferències de manera exponencial: entre 4 i 5 trobem 536 i entre 5 i 6, 62. Si provem amb 10 decimals i comparem la diferència entre aquest fitxer i 5 ens dóna un resultat de 96; per tant decidim considerar a partir de 10 decimals. Finalment, provem amb 15 decimals i trobem que la diferència amb 10 decimals és molt baixa, de 6 diferències, fet que ens fa pensar que considerar **10 decimals** és un bon valor per la condició de stop.

També ens hem fixat en l’evolució del nombre d’iteracions i el temps d’execució i hem trobat els següents resultats:

| #decimals considerats | temps d’execució (segons) | #iteracions |
| --- | --- | --- |
| 5 | 1,86 | 28 |
| 10 | 4,54 | 69 |
| 15 | 8,76 | 129 |

Per tant hem concluit que 10 és un bon nombre de decimals a considerar ja que si augmentem aquest nombre el temps d’execució i les iteracions augmenten i el canvi que obtenim és gairebé irrellevant, considerant el nombre de aeroports que es tracten 5740. A més les diferències són menors ja que es tracta d’aeroports consecutius que s’intercanvien les posicions.

Cal remarcar que l’estudi de la condició del stop s’ha fet prèviament a l’elecció del damping factor, per tant els resultats han estat obtinguts amb la L inicialitzada a 0.8.

*Damping Factor*

Per a decidir amb quin damping factor quedar-nos hem provat amb diferents valors entre 0.8 i 0.9 {0.8, 0.825, 0.85, 0.875, 0.9}. Per a l’estudi ens hem fixat en les diferències d’ordre dels aeroports, les iteracions, el temps de computació i el resultat de la suma. Hem comprovat que la diferència en l’ordre dels aeroports sempre ronda per 4500 (4465, 4497, 4624…) per tant considerem que no cal tenir-la en compte.

|  | L=0.8 | L=0.825 | L=0.85 | L=0.875 | L=0.9 |
| --- | --- | --- | --- | --- | --- |
| #iteracions | 69 | 84 | 98 | 117 | 155 |
| Time of computePageRanks (segons) | 2.9246902465820312 | 3.528106212615967 | 3.997051239013672 | 5.1840009689331055 | 6.588433027267456 |
| Suma | 0.9999999999999242 | 0.9999999999999419 | 0.9999999999999435 | 1.0000000000000002 | 1.0000000000000546 |

Com es pot veure en la taula, a mesura que incrementa el damping factor també ho fan el nombre d’iteracions i per tant el temps de computació. També hem observat que millora la precisió del resultat però com que suposa un número més gran d’iteracions i de temps, hem decidit que ens quedem amb un **damping factor de 0.85** ja que és el punt mitjà.