CAIM - Lab5
PageRank

Pau Núñez Amorós

18/11/2019

1 Dificultats i decisions preses

Després d'entendre com funcionava l'algorisme de *PageRank*, lleugerament diferent de la versió matricial que es va explicar a classe (tenim arestes amb **pesos** i s'imposa que el graf sigui tot ell un *SCC* excepte els aeroports aïllats), vaig haver de rumiar una forma prou eficient per calcular el resultat.

El document ja avisava que no era bona idea el fet de construir una matriu de mida $n * n$, per tant la meva solució, agafant com a base del codi el fitxer `.py` proporcionat amb l'enunciat, fa servir `lists` i `dicts`. Aquestes estructures, especialment els diccionaris permeten un accés molt ràpid a la informació però obliguen a crear un sistema d'índexs per relacionar el contingut dels diccionaris amb la corresponent posició a la llista.

`list[Airport]`		`dict{IATA: Airport}`
`list[Edge]`	⟵ `index` points to ⟶	`dict{IATA: index}`

Una altra dificultat amb la qual he topat és la verificació que a cada iteració de `computePageRanks()` es complís que `sum(P)` ≈ 1. Calia tenir en compte, com també diu l'enunciat, els aeroports aïllats (no són destí ni origen de cap vol), que són un total de 2455. A cada iteració s'ha d'assignar un pes a cada aeroport aïllat (el mateix valor a tots, ja que tots estan desconnectats del graf) per tal que la seva contribució al *PageRank* faci que `sum(P)` ≈ 1.

Com podem comprovar amb el resultat, tots els aeroports aïllats tenen el mateix valor PR i tots queden emptats en última posició (tots són els menys populars, com és lògic intuïtivament)

2 Experimentació

2.1 Damping Factor

Aquest valor queda fixat a l'inici de l'algorisme i compleix la següent propietat: $0 \leq \texttt{L} \leq 1$, però a l'enunciat se'ns proposa usar un rang valors populars: $0.8 \leq \texttt{L} \leq 0.9$. Tot i així he experimentat amb valors de `L` fora d'aquests valors típics proposats:

Valors petits i mitjans ($0 \leq \texttt{L} \leq 0.7$)

`L`	#iteracions	segons	Valor PageRank de		Diferència (max-min)
			Aeroport més popular	Aeroport menys popular	
0.1	12	0.34	0.000762	0.000164	0.000598
0.3	22	0.61	0.001933	0.000139	0.001794
0.5	38	1.08	0.003132	0.000110	0.003022
0.7	74	2.07	0.004431	0.000074	0.004357

Com podem comprovar, amb una `L` molt petita l'algorisme executa molt poques iteracions i acaba molt ràpid. Els valors *PageRank* són molt similars entre ells, tal i com indica la columna "Diferència", degut a que l'algorisme no té temps per calcular més precisament els valors segons la popularitat de l'aeroport.

Per cada 0.2 que creix `L` veiem que molt aproximadament, el nombre d'iteracions es duplica i el temps que tarda en acabar l'algorisme també. Els valors PR també van diferint més i més però de forma més progressiva.

Valors recomanats ($0.8 \leq$ `L` ≤ 0.9)

`L`	#iteracions	segons	Valor PageRank de		Diferència (max-min)
			Aeroport més popular	Aeroport menys popular	
0.8	118	3.42	0.005155	0.000053	0.005102
0.85	162	4.67	0.005575	0.000041	0.005534
0.9	249	7.21	0.006200	0.000028	0.006172

Aquest és el rang de valors recomanats, el nombre d'iteracions ja són diversos centenars, però el temps que tarda segueix sent molt raonable. Podem comprovar també que la diferència entre valors PR segueix creixent poc a poc. Mentre que amb `L=0.1` el valor PR més petit era un 21.5% del valor PR més gran, amb la `L` recomanada aquesta relació és $\sim 0.7\%$. Podem concloure doncs, els valors PR dels aeroports són força diferents i el cost computacional és perfectament assumible.

Valors alts ($0.9 <$ `L` ≤ 1)

`L`	#iteracions	segons	Valor PageRank de		Diferència (max-min)
			Aeroport més popular	Aeroport menys popular	
0.95	512	14.73	0.006889	0.000015	0.006871
0.97	861	25.11	0.007182	0.000009	0.007173
1	-	∞	-	-	-

En aquesta forquilla de valors tan alts pel Damping Factor l'execució dura una quantitat de temps significativa, tant el nombre d'iteracions com els segons mesurats comencen a escalar ràpidament. En contrast però, la diferència de valors PR ja no millora gaire:

Entre els valors 0.97 i 0.9 per L veiem que la diferència de valors PR millora un
$(0.007173/0.006172 - 1) * 100 \approx$ **16%**

Però en comprovar quant augmenta el temps de computació requerit entre els mateixos valors per `L` que abans veiem que no surt gens a compte escollir aquests valors tan alts: $(25.11/7.21 - 1) * 100 \approx$ **248%**

2.2 Condició de parada

La condició de parada determina quan `computePageRanks()` pot aturar-se perquè ja estem satisfets amb els valors *PageRank* obtinguts. A la meva implementació he seguit la idea proposada a l'enunciat: aturar l'algorisme quan la convergència és prou alta, és a dir, quan dos vectors `P` a iteracions consecutives són prou similars.

Vegem la següent taula comparativa dels valors possibles per a `threshold` per treure algunes conclusions:

`threshold`	#iteracions	segons	Valor PageRank de	
			Aeroport més popular	Aeroport menys popular
$1 * 10^{-5}$	20	0.6	0.005575	0.000041
$1 * 10^{-10}$	91	2.65	0.005575	0.000041
$1 * 10^{-15}$	162	4.75	0.005575	0.000041
$1 * 10^{-18}$	205	5.94	0.005575	0.000041
$1 * 10^{-20}$	-	∞	-	-

Com podem comprovar, el valor del `threshold` haurà de ser tan baix com tanta precisió en els decimals del valor *PageRank* volguem. (per això els valors PR a la taula no varien). Un bon valor és $1 * 10^{-5}$ (execució per sota 1s). Per uns quants segons podem obtenir precisió de 10 o 15 decimals i a partir de $1 * 10^{-19}$ i més petits l'execució no acaba mai perquè no es pot obtenir una diferència tan baixa.