

RAPPORT PLP

Cyrine Chtourou, Mariem Mezghanni & Yasmine Cheikh



CentraleSupélec

2017 - 2018

PARTIE MAP REDUCE

QUESTION 2.7

DISPLAYING THE CONTENT OF A CSV FILE

INTRODUCTION DU PROBLEME

L'objectif de cet exercice est d'afficher, à partir du fichier arbres.csv, les attributs suivants pour chaque arbre (représenté par une classe Tree non instanciable) :

- Nom
- Année
- Hauteur

IMPLEMENTATION ET RESULTAT

1. Nous avons tout d'abord construit une **abstract** classe « arbre » afin de représenter chaque arbre par ses attributs Name, Year et Height. Cette classe contient des méthodes setNames, setHeight et setYear permettant d'extraire ses attributs à partir du fichier arbres.csv.
2. Les attributs sont obtenus en extrayant une sous chaîne de la ligne du document correspondant à l'arbre en question, soit en exécutant les commandes :
 - String[] split = line.split(" ; ");
 - Feature = split[i];Où i correspond à l'ordre de l'attribut qu'on souhaite extraire. Et ce pour tous les arbres du fichier.
3. Enfin, nous affichons ces attributs comme illustré ci-dessous (Il y a 97 arbres)

```
1) Oranger des Osages__Year: 1935__Height: 13.0
2) Cèdre à encens__Year: 1854__Height: 20.0
3) Pérrocarya du Caucase__Year: 1862__Height: 22.0
4) Micocoulier de Provence__Year: 1906__Height: 16.0
5) Chêne rouvre__Year: 1784__Height: 30.0
6) Platane commun__Year: 1860__Height: 45.0
7) Platane commun__Year: 1840__Height: 40.0
8) Aulne glutineux__Year: 1933__Height: 16.0
9) Marronnier d'Inde__Year: __Height: 30.0
10) Arbre aux quarante écus__Year: 1913__Height: 33.0
11) Frêne commun__Year: __Height: 30.0
12) Ailanthe__Year: __Height: 35.0
13) Cyprès chauve__Year: 1862__Height: 35.0
14) Kaki__Year: __Height: 12.0
15) Séquoia géant__Year: 1850__Height: 30.0
16) Chicot du Canada__Year: __Height: 10.0
17) Hêtre pleureur__Year: 1857__Height: 10.0
18) Pérrocarya du Caucase__Year: 1882__Height: 27.0
19) Séquoia géant__Year: 1872__Height:
20) Pin noir__Year: __Height: 30.0
21) Platane d'Orient__Year: 1843__Height: 26.0
22) Zelkova du Japon__Year: 1872__Height: 18.0
23) Noyer noir__Year: 1845__Height: 28.0
24) Cyprès chauve__Year: 1930__Height: 20.0
25) Pin Napoléon__Year: __Height: 10.0
26) Pin noir__Year: 1870__Height: 25.0
27) Robinier faux-acacia__Year: 1601__Height: 11.0
28) Arbre à gutta-percha__Year: __Height: 12.0
29) Arbre aux quarante écus__Year: 1879__Height: 22.0
30) Séquoia géant__Year: 1850__Height: 20.0
31) Platane commun__Year: __Height: 35.0
32) Marronnier d'Inde__Year: 1894__Height: 18.0
33) Hêtre pourpre__Year: __Height: 30.0
34) Séquoia sempervirent__Year: 1935__Height: 30.0
35) Sophora du Japon__Year: 1873__Height: 10.0
36) Platane d'Orient__Year: 1862__Height: 34.0
37) Erable de Montpellier__Year: 1833__Height: 12.0
38) Séquoia géant__Year: __Height: 35.0
39) Hêtre pourpre__Year: __Height: 18.0
40) Kaki__Year: 1897__Height: 14.0
```

QUESTION 2.8

DISPLAYING THE CONTENT OF A COMPACT FILE

INTRODUCTION DU PROBLEME

L'objectif de cet exercice est d'afficher, à partir du fichier isd-history.txt, les attributs suivants pour chaque station :

- Le code USAF
- Le nom
- Le pays
- L'élévation de la station

La figure ci-dessous montre un extrait du fichier isd-history.txt :

Integrated Surface Database Station History, December 2017										
USAF = Air Force station ID. May contain a letter in the first position.										
WBAN = NCDC WBAN number										
CTRY = FIPS country ID										
ST = State for US stations										
ICAO = ICAO ID										
LAT = Latitude in thousandths of decimal degrees										
LON = Longitude in thousandths of decimal degrees										
ELEV = Elevation in meters										
BEGIN = Beginning Period Of Record (YYYYMMDD). There may be reporting gaps within the P.O.R.										
END = Ending Period Of Record (YYYYMMDD). There may be reporting gaps within the P.O.R.										
Notes:										
- Missing station name, etc indicate the metadata are not currently available.										
- The term "bogus" indicates that the station name, etc are not available.										
- For a small % of the station entries in this list, climatic data are not available. To determine data availability for each location, see the 'isd-inventory.txt' or 'isd-inventory.csv' file.										
USAF	WBAN	STATION NAME	CTRY	ST	CALL	LAT	LON	ELEV(M)	BEGIN	END
007026	99999	WXPOD 7026	AF			+00.000	+000.000	+7026.0	20140711	20170822
007070	99999	WXPOD 7070	AF			+00.000	+000.000	+7070.0	20140923	20150926
008403	99999	XM10							20140101	20140412
008411	99999	XM20							20140102	20160217
008414	99999	XM18							20140101	20160217
008415	99999	XM21							20140108	20160217
008416	99999	XM22							20140402	20150917
008418	99999	XM24							20140101	20160217
008421	99999	XM26							20140406	20151229
010000	99999	BOGUS NORWAY	NO	ENRS					20010927	20051231
010010	99999	JAN MAYEN(NOR-NAVY)	NO	ENJA	+70.933	-008.667	+0009.0	19310101	20171223	
010013	99999	ROST	NO						19861120	19880105
010014	99999	SORSTOKKEN	NO	ENSO	+59.792	+005.341	+0048.8	19861120	20171222	
010015	99999	BRINGELAND	NO		+61.383	+005.867	+0327.0	19870117	20111020	
010016	99999	RORVIK/RYUM	NO		+64.850	+011.233	+0014.0	19870116	19910806	
010017	99999	FRIGG	NO	ENFR	+59.980	+002.250	+0048.0	19880320	20050228	
010020	99999	VERLEGENHUKEN	NO		+80.050	+016.250	+0008.0	19861109	20171223	
010030	99999	HORNSUND	NO		+77.000	+015.500	+0012.0	19850601	20171223	
010040	99999	NY-ALESUND II	NO	ENAS	+78.917	+011.933	+0008.0	19730101	20140523	
010050	99999	ISFJORD RADIO	SV		+78.067	+013.633	+0009.0	19310103	20140523	
010060	99999	EDGEYOA	NO		+78.250	+022.817	+0014.0	19730101	20171223	
010070	99999	NY-ALESUND	SV		+78.917	+011.933	+0007.7	19730106	20171223	
010071	99999	LONGYEARBYEN	SV		+78.217	+015.583	+0037.0	20050210	20050210	
010080	99999	LONGYEAR	SV	ENSB	+78.246	+015.466	+0026.8	19750929	20171223	

Source : <https://www1.ncdc.noaa.gov/pub/data/noaa/isd-history.txt>

IMPLEMENTATION

1. Nous avons tout d'abord construit une classe "Station" pour représenter chaque station par les attributs que nous voulons afficher : **USAF, name, country, elevation** ainsi que la ligne du fichier (chaque ligne correspond à une station)

2. Chaque attribut est de type **String** et est obtenu en extrayant une sous-chaine de la chaîne constituée par la ligne considérée :
 - USAF = line.substring(1,6)
 - Name= line.substring(13,42)
 - Country= line.substring(43,45)
 - Elevation= line.substring(74,81)
3. Nous avons ajouté les Getters de chaque attribut ;
4. Dans le programme principal, nous commençons par lire le fichier ligne par ligne (à partir de la ligne 22, là où commencent les données). Pour chaque ligne, nous récupérons les attributs de chaque station à l'aide des méthodes **setUSAF(line)**, **setName(line)**, **setCountry(line)** et **setElevation(line)** ;
5. Enfin, nous affichons ces attributs à l'aide des méthodes **getUSAF()**, **getName()**, **getCountry()** et **getElevation()** ;

RESULTAT

La figure ci-dessus montre un extrait du résultat obtenu. Pour chaque ligne (station), nous pouvons lire respectivement : le code USAF, le nom, le pays, l'élévation.



```

Console
<terminated> question8 [Java Application] /usr/java/jdk1.8.0_131/bin/java
97983 DATA BUOY 41037 US +0003.0
97984 DATA BUOY 46094 US +0003.0
97985 DATA BUOY 46270 US +0003.0
97986 DATA BUOY 41043 US +0003.0
97987 DATA BUOY 42059 US +0003.0
97988 BIG BAY US +0185.0
97989 OLCOTT HARBOR US +0083.0
97990 ACE BASIN RESERVE US +0005.0
97991 HUDSON RIVER RESERVE US +0012.0
97992 APALACHICOLA RESERVE US +0005.0
97993 GREAT BAY RESERVE US +0003.0
97994 CHESAPEAKE BAY US +0003.0
97995 SOUTH SLOUGH RESERVE US +0013.0
97996 ST CLAIR SHORES US +0180.0
97997 ELKHORN SLOUGH RESERVE US +0003.0
97999 GENEVA ON THE LAKE US +0186.0
98000 GUANA TOLOMATO MATANZAS US +0004.0
98001 HURON LIGHT US +0184.0
98002 JACQUES COUSTEAU RESERVE US +0012.0
98003 JOBOS BAY RESERVE RQ +0015.0
98004 NARRAGANSETT BAY RESERVE US +0013.0
98005 NORTH INLET-WINYAH BAY US +0004.0
98006 NORTH CAROLINA RESERVE US +0005.0
98007 PADILLA BAY RESERVE US +0003.0
98008 ROOKERY BAY RESERVE US +0020.0
98009 ROCHESTER US +0083.0
98010 SAPELO ISLAND RESERVE US +0005.0
98011 SAN FRANCISCO BAY RESERVE US +0003.0
98012 ST JOSEPH US +0184.0
98013 TIJUANA RIVER RESERVE US +0004.0

```

QUESTION 5

PROBLEME 1 : TF-IDF

INTRODUCTION DU PROBLEME

TF-IDF (*Term Frequency-Inverse Document Frequency*) est une approche qui consiste à permet d'évaluer l'importance d'un terme contenu dans un document, relativement à un corpus de documents.

Pour ce faire, il faut combiner deux notions :

1. **La fréquence d'un terme** : il est évident que le poids assigné à un mot augmente proportionnellement au nombre d'apparitions du mot dans le document.
2. **La fréquence inverse de document** : Cette fréquence est une mesure de l'importance du terme dans l'ensemble du corpus. L'objectif est d'accorder un poids plus important aux termes les moins fréquents. En effet, si un terme est très fréquent au sein du corpus (par exemple, les articles définis - le, la, les), il est en fait peu discriminant.

Cette approche assigne donc à chaque terme t un poids dans un corpus D donnée par :

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

où :

- $tf_{t,d}$ est la fréquence d'un terme t dans un document d :

$$tf_{t,d} = \frac{\text{nombre d'occurrence du terme } t \text{ dans le document}}{\text{nombre total de mots dans le document}}$$

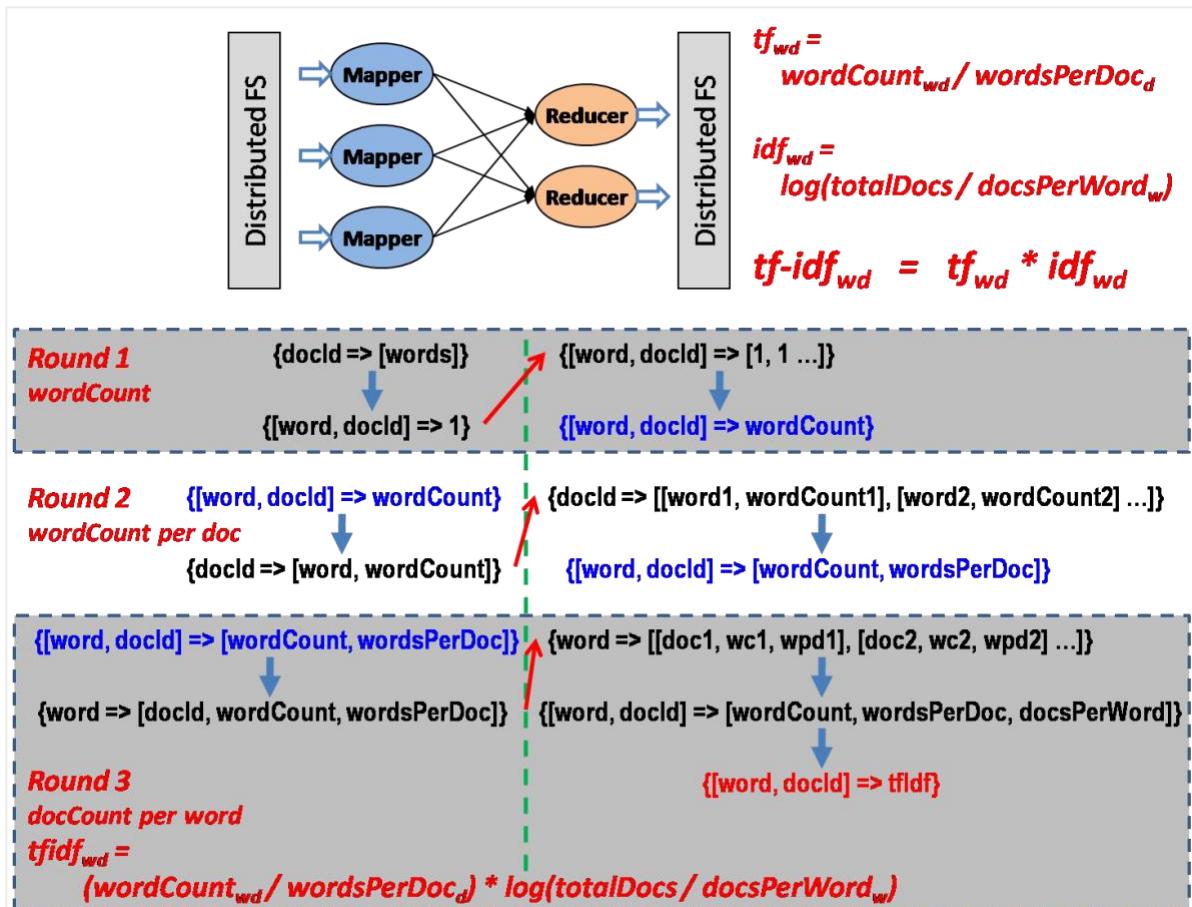
- idf_t est la fréquence inverse du terme t dans le corpus D .

$$idf_t = \log \left(\frac{\text{nombre total de documents dans le corpus}}{\text{nombre de documents contenant le terme } t} \right)$$

Le schéma propose donc d'augmenter la pertinence d'un terme en fonction de sa rareté au sein du corpus (fréquence du terme dans le corpus IDF élevée).

IMPLEMENTATION MAPREDUCE

Comme proposé dans le sujet, nous allons implémenter un programme MapReduce qui calcule le score **TF-IDF** d'un mot dans un corpus de documents en 3 étapes illustrées dans le schéma ci-dessous :



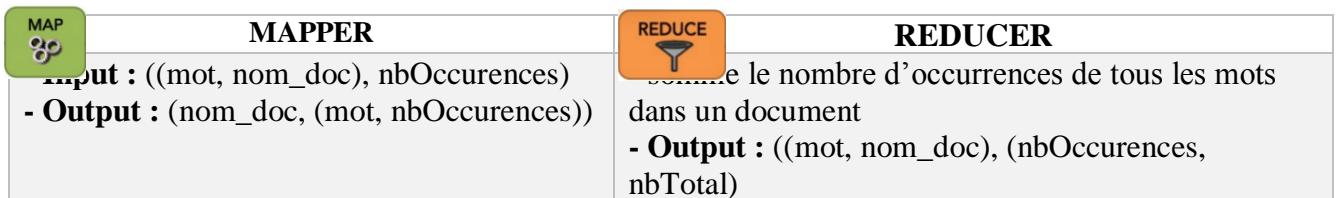
ETAPE 1 : COMPTER LA FREQUENCE D'UN MOT DANS UN DOCUMENT

MAP	MAPPER	REDUCE	REDUCER
	<ul style="list-style-type: none"> - Input : (nom_doc, contenu) - Output : ((mot, nom du document), 1) 		<ul style="list-style-type: none"> - somme pour chaque couple (mot, document) les « 1 » issus du Mapper - Output : ((mot, nom_doc), nbOccurrences)

La figure ci-dessous montre un extrait du fichier résultat de l'étape 1 :

Mot	Document	Nombre d'occurrences
above	dans THE CALL OF THE WILD.txt	11
abraham	dans ROBINSON CRUSOE.txt	2
abreast	dans THE CALL OF THE WILD.txt	1
abridgment	dans ROBINSON CRUSOE.txt	1

ETAPE 2 : COMPTER LE NOMBRE DE MOTS DANS UN DOCUMENT



La figure ci-dessous montre un extrait du fichier résultat de l'étape 2 :

Mot	Document	Nombre d'occurrences	Nombre total de mots dans le document
slope	THE CALL OF THE WILD.txt	3/31610	
advanced	THE CALL OF THE WILD.txt	7/31610	
neighbouring	THE CALL OF THE WILD.txt	1/31610	
heed	THE CALL OF THE WILD.txt	2/31610	
generation	THE CALL OF THE WILD.txt	1/31610	
timber	THE CALL OF THE WILD.txt	10/31610	
flitted	THE CALL OF THE WILD.txt	1/31610	
smashing	THE CALL OF THE WILD.txt	1/31610	
women	ROBINSON CRUSOE.txt	8/121090	
unforseen	ROBINSON CRUSOE.txt	1/121090	
shock	ROBINSON CRUSOE.txt	1/121090	
hedge	ROBINSON CRUSOE.txt	16/121090	
remembered	ROBINSON CRUSOE.txt	1/121090	
daughters	ROBINSON CRUSOE.txt	1/121090	
claimed	ROBINSON CRUSOE.txt	1/121090	
rivulet	ROBINSON CRUSOE.txt	1/121090	
busied	ROBINSON CRUSOE.txt	1/121090	
perches	THE CALL OF THE WILD.txt	1/31610	
hysteria	THE CALL OF THE WILD.txt	1/31610	
attended	THE CALL OF THE WILD.txt	1/31610	
yukon	THE CALL OF THE WILD.txt	7/31610	
holding	THE CALL OF THE WILD.txt	2/31610	
advances	THE CALL OF THE WILD.txt	4/31610	
half-friendly	THE CALL OF THE WILD.txt	1/31610	
discharged	THE CALL OF THE WILD.txt	2/31610	
answered	THE CALL OF THE WILD.txt	3/31610	
pleading	THE CALL OF THE WILD.txt	1/31610	
solleks	THE CALL OF THE WILD.txt	1/31610	

ETAPE 3 : CALCULER LA FREQUENCE D'UN MOT DANS UN CORPUS

MAPPER	REDUCE	REDUCER
<p>- Input : ((mot, nom_doc), (nbOccurrences, nbTotal))</p> <p>- Output : (mot, (nom_doc, nbOccurrences, nbTotal, 1))</p>	<p>compte le nombre d'occurrences d'un mot dans tout le corpus</p>	<p>- Output : (mot, nom_doc, (nbOccurrences, nbTotal,))</p>

La figure ci-dessus montre un extrait de l'output de l'étape 3 :

```
part-r-00000
abate dans ROBINSON CRUSOE.txt TF = 12 / 121090 IDF = log(1/2) TfIdf = 0.00002983]
abated dans ROBINSON CRUSOE.txt TF = 10 / 121090 IDF = log(1/2) TfIdf = 0.00002486]
abatement dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
abating dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
abed dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
abhor dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
abhorrence dans ROBINSON CRUSOE.txt TF = 6 / 121090 IDF = log(1/2) TfIdf = 0.00001492]
abide dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(2/2) TfIdf = 0.00000826]
abide dans THE CALL OF THE WILD.txt TF = 1 / 31610 IDF = log(2/2) TfIdf = 0.00003164]
abiding-place dans THE CALL OF THE WILD.txt TF = 1 / 31610 IDF = log(1/2) TfIdf = 0.00000952]
ability dans THE CALL OF THE WILD.txt TF = 2 / 31610 IDF = log(1/2) TfIdf = 0.00001905]
abjectly dans THE CALL OF THE WILD.txt TF = 1 / 31610 IDF = log(1/2) TfIdf = 0.00000952]
able dans THE CALL OF THE WILD.txt TF = 2 / 31610 IDF = log(2/2) TfIdf = 0.00006327]
able dans ROBINSON CRUSOE.txt TF = 54 / 121090 IDF = log(2/2) TfIdf = 0.00044595]
abode dans ROBINSON CRUSOE.txt TF = 3 / 121090 IDF = log(1/2) TfIdf = 0.00000746]
abominable dans ROBINSON CRUSOE.txt TF = 2 / 121090 IDF = log(1/2) TfIdf = 0.00000497]
abortive dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
about dans ROBINSON CRUSOE.txt TF = 285 / 121090 IDF = log(2/2) TfIdf = 0.00235362]
about dans THE CALL OF THE WILD.txt TF = 36 / 31610 IDF = log(2/2) TfIdf = 0.00113888]
about's dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
above dans ROBINSON CRUSOE.txt TF = 81 / 121090 IDF = log(2/2) TfIdf = 0.00066892]
above dans THE CALL OF THE WILD.txt TF = 11 / 31610 IDF = log(2/2) TfIdf = 0.00034799]
abraham dans ROBINSON CRUSOE.txt TF = 2 / 121090 IDF = log(1/2) TfIdf = 0.00000497]
abreast dans THE CALL OF THE WILD.txt TF = 2 / 31610 IDF = log(1/2) TfIdf = 0.00001905]
abridgment dans ROBINSON CRUSOE.txt TF = 1 / 121090 IDF = log(1/2) TfIdf = 0.00000249]
abroad dans ROBINSON CRUSOE.txt TF = 36 / 121090 IDF = log(2/2) TfIdf = 0.0002973]
abroad dans THE CALL OF THE WILD.txt TF = 1 / 31610 IDF = log(2/2) TfIdf = 0.00003164]
absence dans ROBINSON CRUSOE.txt TF = 5 / 121090 IDF = log(1/2) TfIdf = 0.00001243]
```

Chaque ligne contient successivement :

- le mot
- le nom du document
- le calcul de TF
- le calcul de IDF
- la valeur de TF-IDF

ETAPE 3 : RETROUVER LES 20 MOTS AVEC LE « TF-IDF » LE PLUS ELEVEE

Afin de retrouver les 20 mots avec les scores IF-IDF les plus élevés dans les deux textes, nous avons exporté le fichier texte qui est l'output de l'étape 3 dans Excel et nous avons classé tous les mots en fonction de leur TF-IDF par ordre décroissant. Voici le résultat obtenu :

	A	B	C
1	Rang	Mot	TFIDF
2		1 the dans THE CALL OF THE WILD.txt	0,07193926
3		2 the dans ROBINSON CRUSOE.txt	0,04864976
4		3 and dans THE CALL OF THE WILD.txt	0,04821259
5		4 i dans ROBINSON CRUSOE.txt	0,04232389
6		5 and dans ROBINSON CRUSOE.txt	0,03972252
7		6 to dans ROBINSON CRUSOE.txt	0,03555207
8		7 of dans ROBINSON CRUSOE.txt	0,02910232
9		8 of dans THE CALL OF THE WILD.txt	0,02752294
10		9 he dans THE CALL OF THE WILD.txt	0,02565644
11		10 was dans THE CALL OF THE WILD.txt	0,02198671
12		11 to dans THE CALL OF THE WILD.txt	0,021354
13		12 a dans THE CALL OF THE WILD.txt	0,02068966
14		13 a dans ROBINSON CRUSOE.txt	0,01864729
15		14 his dans THE CALL OF THE WILD.txt	0,01774755
16		15 my dans ROBINSON CRUSOE.txt	0,01765629
17		16 in dans THE CALL OF THE WILD.txt	0,01695666
18		17 was dans ROBINSON CRUSOE.txt	0,01664877
19		18 in dans ROBINSON CRUSOE.txt	0,01598811
20		19 that dans ROBINSON CRUSOE.txt	0,01555868
21		20 it dans ROBINSON CRUSOE.txt	0,01522008

Pour mieux comprendre la signification du nombre TF-IDF, nous avons aussi regardé les mots avec le score le plus faible :

	A	B	C
11125	11124	worthy dans ROBINSON CRUSOE.txt	0,00000249
11126	11125	woud dans ROBINSON CRUSOE.txt	0,00000249
11127	11126	wouldest dans ROBINSON CRUSOE.txt	0,00000249
11128	11127	wounding dans ROBINSON CRUSOE.txt	0,00000249
11129	11128	wranglings dans ROBINSON CRUSOE.txt	0,00000249
11130	11129	wrapping dans ROBINSON CRUSOE.txt	0,00000249
11131	11130	wrapt dans ROBINSON CRUSOE.txt	0,00000249
11132	11131	wrench dans ROBINSON CRUSOE.txt	0,00000249
11133	11132	wretchedly dans ROBINSON CRUSOE.txt	0,00000249
11134	11133	wring dans ROBINSON CRUSOE.txt	0,00000249
11135	11134	writ dans ROBINSON CRUSOE.txt	0,00000249
11136	11135	writings dans ROBINSON CRUSOE.txt	0,00000249
11137	11136	wronged dans ROBINSON CRUSOE.txt	0,00000249
11138	11137	wrights dans ROBINSON CRUSOE.txt	0,00000249
11139	11138	wrongs dans ROBINSON CRUSOE.txt	0,00000249
11140	11139	wrung dans ROBINSON CRUSOE.txt	0,00000249
11141	11140	wsy's dans ROBINSON CRUSOE.txt	0,00000249
11142	11141	xury's dans ROBINSON CRUSOE.txt	0,00000249
11143	11142	yardarm dans ROBINSON CRUSOE.txt	0,00000249
11144	11143	yearling dans ROBINSON CRUSOE.txt	0,00000249
11145	11144	yell dans ROBINSON CRUSOE.txt	0,00000249
11146	11145	yellings dans ROBINSON CRUSOE.txt	0,00000249
11147	11146	Yorkshire dans ROBINSON CRUSOE.txt	0,00000249
11148	11147	yourn dans ROBINSON CRUSOE.txt	0,00000249
11149	11148	yours dans ROBINSON CRUSOE.txt	0,00000249
11150	11149	yourself dans ROBINSON CRUSOE.txt	0,00000249
11151	11150	zee dans ROBINSON CRUSOE.txt	0,00000249

PROBLEME 2 : PAGE RANK

INTRODUCTION DU PROBLEME

PageRank est une méthode qui sert à mesurer l'importance des pages du web. Distribution de probabilité sur les pages web qui représente la chance qu'un utilisateur naviguant au hasard arrive à une page web particulière.

REMARQUES :

- Le web est considéré comme un graphe orienté où une page est un nœud et les hyperliens sont des arcs.
- Le programme Map Reduce calcule itérativement la probabilité de toutes les pages jusqu'à convergence.

Le Page Rank d'une page u peut être exprimé comme suit :

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

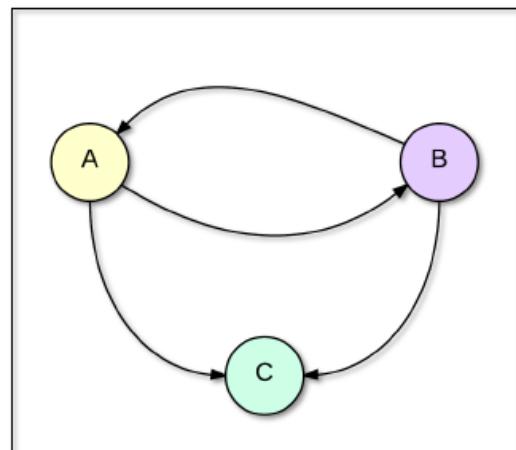
avec :

- B_u l'ensemble des pages ayant un lien vers la page u ;
- $L(v)$ le nombre de liens sortant de la page v ;

EXEMPLE :

Prenons un exemple simplifié où le web est constitué de 3 pages A, B et C. PageRank est initialisé à la même valeur pour les 3 pages : $PR(A) = PR(B) = PR(C) = 0.33$.

$$\left\{ \begin{array}{l} PR(A) = \frac{PR(B)}{2} \\ PR(B) = \frac{PR(A)}{2} \\ PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{2} \end{array} \right.$$



IMPLEMENTATION MAPREDUCE

Nous allons calculer le PageRank des utilisateurs du réseau social *Epinions who-trust-whom* en 3 étapes que nous allons décrire :

1. Bâtir et initialiser le graphe
2. Recalculer PageRank pour chaque page web jusqu'à convergence
3. Classer les PageRank et retourner les 10 premières valeurs

ETAPE 1 : BATIR LE GRAPHE



MAPPER



REDUCER

- **Input** : un nœud (un utilisateur A)
- **Output** : pour chaque lien de l'utilisateur A vers l'utilisateur B, on émet :

(utilisateur A, utilisateur B)

- **Input** : (utilisateur A, [utilisateur B₁...])
 - **Output** : (utilisateur A, « PageRank ; [utilisateurs B_i] »)
- Le PageRank est également initialisé à
- $$\frac{\text{damping factor}}{\text{nombre total de noeuds}}$$

Voici un extrait de l'output de l'étape 1 où on souligne les résultats obtenus pour l'utilisateur 10000

	part-r-00000	part-r-00000
0	1.1202045361694277E-5	
39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,		
1	1.1202045361694277E-5	
24,21,20,17,16,11,10,5,4,3,19889,11779,10731,8842,8498,6401,5412,5269,4950,4939,4903,41		
10	1.1202045361694277E-5	
48,50,51,61,65,66,69,72,73,74,78,83,88,90,92,98,100,102,103,104,110,117,124,141,165,23		
100	1.1202045361694277E-5	
447,546,550,551,553,555,582,588,597,611,449,2,4,5,6,10,12,19,21,22,26,27,28,30,38,43,48		
1000	1.1202045361694277E-5	
682,639,634,295,195,22,90,135,663,16675,16674,16673,12453,10700,10498,9750,7714,6878,56		
10000	1.1202045361694277E-5	128,64,9646,7590,5514,4040,3984,1719,1619,1304,1179,73
10001	1.1202045361694277E-5	7445,24954,395,639,972,2042,3637
10002	1.1202045361694277E-5	
13960,47619,663,143,135,18,735,737,849,852,854,860,909,918,1177,1268,1274,1394,1401,14		
10003	1.1202045361694277E-5	3723,3717,2025,1914,1741,395
10004	1.1202045361694277E-5	395
10005	1.1202045361694277E-5	401,395

Utilisateur A

PageRank initial (le même pour tous les utilisateurs)

Utilisateurs Bi

ETAPE 2 : CALCULER PAGERANK PAR ITERATION

MAP ☒	MAPPER	REDUCE ✖	REDUCER
<ul style="list-style-type: none"> - Input : (utilisateur A, « PageRank ; [utilisateurs B_i] ») - Output : on émet 2 sorties <ol style="list-style-type: none"> La liste des liens de chaque utilisateur : (utilisateur A, [utilisateur B₁...]) pour chaque utilisateur B_i, on émet le PageRank de l'utilisateur source et le nombre total : (utilisateur B_i ; « PageRank ; nb total de liens sortant de l'utilisateur A») 	<ul style="list-style-type: none"> - Input : (utilisateur A, « PageRank ; [utilisateurs B_i] ») - Output : on calcule le PageRank par itération et on retourne : (utilisateur A , « PageRank ; [utilisateur B₁...] ») 		

Voici un extrait de l'output de l'étape 2 avec les résultats pour le même utilisateur que précédemment (utilisateur 10000) :

*part-r-00000		part-r-00000
0	5.28547452423924	
39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70		
1	6.51139360397876	
24,21,20,17,16,11,10,5,4,3,19889,11779,10731,8842,8498,6401,5412,5269,4950,4939,4903,4706,4703,		
10	1.9485397599046488	
48,50,51,61,65,66,69,72,73,74,78,83,88,90,92,98,100,102,103,104,110,117,124,141,165,233,301,302		
100	1.7839886090313541	
447,546,550,551,553,555,582,588,597,611,449,2,4,5,6,10,12,19,21,22,26,27,28,30,38,43,48,50,51,5		
1000	0.3464834397865254	
682,639,634,295,195,22,90,135,663,16675,16674,16673,12453,10700,10498,9750,7714,6878,5659,5417,		
10000	0.33990728081018196	128,64,9646,7590,5514,4040,3984,1719,1619,1304,1179,737,394,141
10001	0.1805557400485695	7445,24954,395,639,972,2042,3637
10002	0.6022130915108739	
13960,47619,663,143,135,18,735,737,849,852,854,860,909,918,1177,1268,1274,1394,1401,1430,1433,1		
10003	0.26829763570499315	3723,3717,2025,1914,1741,395
10004	0.15000000000000002	395
10005	0.15000000000000002	401,395

Utilisateur A

PageRank final

Utilisateurs Bi

ETAPE 3 : CLASSER LES PAGERANK PAR ORDRE DECROISSANT

Afin de retrouver les 10 utilisateurs avec les PageRank les plus élevés dans le réseau social, nous avons exporté le fichier texte qui est l'output de l'étape 2 dans Excel et nous avons classé tous les utilisateurs en fonction de leur PageRank par ordre décroissant. Voici les 10 meilleurs utilisateurs :

	A	B
1	Classement	Utilisateur
2	1	41330
3	2	24919
4	3	41329
5	4	41328
6	5	41327
7	6	10006
8	7	29285
9	8	9992
10	9	29286
11	10	10007

PROBLEME 3 : THE TREES OF PARIS

INTRODUCTION DU PROBLEME

Il s'agit d'extraire et de calculer des informations sur les arbres de Paris en utilisant MapReduce et ayant pour input le fichier arbres.csv :

- Calculer le nombre d'arbres par type.
- Pour chaque type d'arbres, calculer la hauteur maximale des arbres.

IMPLEMENTATION MAPREDUCE

Pour calculer le nombre d'arbres par type, nous avons :

1. Crée une classe Driver contenant la méthode main qui configure le Job 'TreeTypes' utilisé par la suite et définit les entrées et les sorties utilisées.

2. Crée les deux classes Mapper et Reducer qui consistent en :

MAP	MAPPER	REDUCE	REDUCER
	<ul style="list-style-type: none"> - Input : (arbres.csv, contenu) - Output : (Type ,1) 		<ul style="list-style-type: none"> - somme pour chaque Type les « 1 » issus du Mapper - Output : (Type, nombre d'arbres)

Pour calculer la hauteur maximale des arbres par type, nous avons :

- Créé une classe Driver contenant la méthode main qui configure le Job 'TreeMaxHeightPerType' utilisé par la suite et définit les entrées et les sorties utilisées.
- Créé les deux classes Mapper et Reducer qui consistent en :

MAP	MAPPER	REDUCE	REDUCER
	<ul style="list-style-type: none"> - Input : (arbres.csv, contenu) - Output : for each Tree : (Type ,Height) 		<ul style="list-style-type: none"> - trouve pour chaque Type la hauteur maximale à partir des hauteurs issues du Mapper - Output : (Type, hauteur maximale)

RESULTAT

Les figures ci-dessus montrent un extrait du résultat obtenu :

Number of Trees per Type:		Maximum Height per Type:	
Acer	3	Acer	16
Aesculus	3	Aesculus	30
Ailanthus	1	Ailanthus	35
Alnus	1	Alnus	16
Araucaria	1	Araucaria	9
Broussonetia	1	Broussonetia	12
Calocedrus	1	Calocedrus	20
Catalpa	1	Catalpa	15
Cedrus	4	Cedrus	30
Celtis	1	Celtis	16
Corylus	3	Corylus	20
Davidaia	1	Davidaia	12
Diospyros	4	Diospyros	14
Eucommia	1	Eucommia	12
Fagus	8	Fagus	30
Fraxinus	1	Fraxinus	30
GENRE	1	GENRE	0
Ginkgo	5	Ginkgo	33
Gymnocladus	1	Gymnocladus	10
Juglans	1	Juglans	28
Liriodendron	2	Liriodendron	35
Maclura	1	Maclura	13
Magnolia	1	Magnolia	12
Paulownia	1	Paulownia	20
Pinus	5	Pinus	30
Platanus	19	Platanus	45
Pterocarya	3		
Quercus	4		

ANNEXE: SCREEN-SHOTS IMAGE OF EMR JOB FLOWS CONSOLE

EXERCICE 5.1:

```
Console >
<terminated> TfIdfDriver [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Jan 8, 2018, 3:00:46 AM)
18/01/08 03:01:04 INFO mapreduce.Job: map 100% reduce 100%
18/01/08 03:01:04 INFO mapreduce.Job: Job job_local303296152_0001 completed successfully
18/01/08 03:01:04 INFO mapreduce.Job: Counters: 33
  File System Counters
    FILE: Number of bytes read=1895532
    FILE: Number of bytes written=2854656
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  Map-Reduce Framework
    Map input records=11150
    Map output records=11150
    Map output bytes=432839
    Map output materialized bytes=455145
    Input split bytes=127
    Combine input records=0
    Combine output records=0
    Reduce input groups=8962
    Reduce shuffle bytes=455145
    Reduce input records=11150
    Reduce output records=11150
    Spilled Records=22300
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=91
    CPU time spent (ms)=0
    Physical memory (bytes) snapshot=0
    Virtual memory (bytes) snapshot=0
    Total committed heap usage (bytes)=331227136
Shuffle Errors
```

EXERCICE 5.2

```
Console >
<terminated> PageRankDriver [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Jan 8, 2018, 2:56:13 AM)
18/01/08 02:59:20 INFO mapreduce.Job: map 100% reduce 100%
18/01/08 02:59:20 INFO mapreduce.Job: Job job_local1004827118_0003 completed successfully
18/01/08 02:59:20 INFO mapreduce.Job: Counters: 33
  File System Counters
    FILE: Number of bytes read=168822444
    FILE: Number of bytes written=182392788
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
  Map-Reduce Framework
    Map input records=75879
    Map output records=600254
    Map output bytes=17581719
    Map output materialized bytes=18788846
    Input split bytes=131
    Combine input records=0
    Combine output records=0
    Reduce input groups=75880
    Reduce shuffle bytes=18788846
    Reduce input records=600254
    Reduce output records=75880
    Spilled Records=1200508
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=221
    CPU time spent (ms)=0
    Physical memory (bytes) snapshot=0
    Virtual memory (bytes) snapshot=0
    Total committed heap usage (bytes)=396271616
Shuffle Errors
```

EXERCICE 5.3

The screenshot shows two separate Eclipse IDE sessions running on a Cloudera system. Both sessions have the title bar "Applications Places System" and the date/time "Mon Jan 8, 05:50" and "Mon Jan 8, 05:52".

Session 1 (Top): The package explorer shows a project structure with packages like maxHeightByType, NoaaHistoryDisplay, and nTreesByType. The nTreesByType package contains a src folder with files TreeTypesDriver.java, TreeTypesMapper.java, TreeTypesReducer.java, and log4j.properties. The TreeTypesDriver.java file is selected. The console tab displays the output of a completed Hadoop job named "TreeTypesDriver". The log includes various counters such as Map input records=98, Map output records=98, Map output bytes=1233, and Map output materialized bytes=1435.

```
<terminated> TreeTypesDriver [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Jan 8, 2018, 3:44:47 AM)
File System Counters
FILE: Number of bytes read=35445
FILE: Number of bytes written=379042
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=98
Map output records=98
Map output bytes=1233
Map output materialized bytes=1435
Input split bytes=87
Combine input records=0
Combine output records=0
Reduce input groups=37
Reduce shuffle bytes=0
Reduce input records=98
Reduce output records=37
Spilled Records=196
Shuffled Maps =0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=58
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=331227136
File Input Format Counters
Bytes Read=16862
File Output Format Counters
Bytes Written=410
```

Session 2 (Bottom): The package explorer shows a similar project structure. The TreeMaxHeightPerTypeDriver.java file is selected in the TreeMaxHeightPerTypeDriver package. The console tab displays the output of a completed Hadoop job named "TreeMaxHeightPerTypeDriver". The log includes various counters such as Map input records=98, Map output records=98, Map output bytes=1233, and Map output materialized bytes=1435.

```
<terminated> TreeMaxHeightPerTypeDriver [Java Application] /usr/java/jdk1.7.0_67-cloudera/bin/java (Jan 8, 2018, 5:51:45 AM)
File System Counters
FILE: Number of bytes read=35445
FILE: Number of bytes written=379224
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=98
Map output records=98
Map output bytes=1233
Map output materialized bytes=1435
Input split bytes=87
Combine input records=0
Combine output records=0
Reduce input groups=37
Reduce shuffle bytes=0
Reduce input records=98
Reduce output records=37
Spilled Records=196
Shuffled Maps =0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=55
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=331227136
File Input Format Counters
Bytes Read=16862
File Output Format Counters
Bytes Written=444
```