

ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ

1η Προγραμματιστική Εργασία

Ομάδα:

Γραμματικόπουλος Λάμπρος 2022201800038 dit18038@uop.gr

Κολοτούρος Κωνσταντίνος 2022201800090 dit18090@uop.gr

Περιεχόμενα:

- Απάντηση ερωτήματος 1 Α Σελίδα 2
- Απάντηση ερωτήματος 1 Β Σελίδα 3
- Απάντηση ερωτήματος 2 Α Σελίδα 5
- Απάντηση ερωτήματος 2 Β Σελίδα 7
- Απάντηση ερωτήματος 3 Α Σελίδα 9
- Απάντηση ερωτήματος 3 Β Σελίδα 11
- Λεπτομέρειες υλοποίησης Σελίδα 14
- Οδηγίες για την εκτέλεση του προγράμματος Σελίδα 14

Απάντηση ερωτήματος 1 Α:

Ψευδοκώδικας:

//Mapreduce για την εύρεση των χρυσών μεταλλίων του κάθε αθλητή

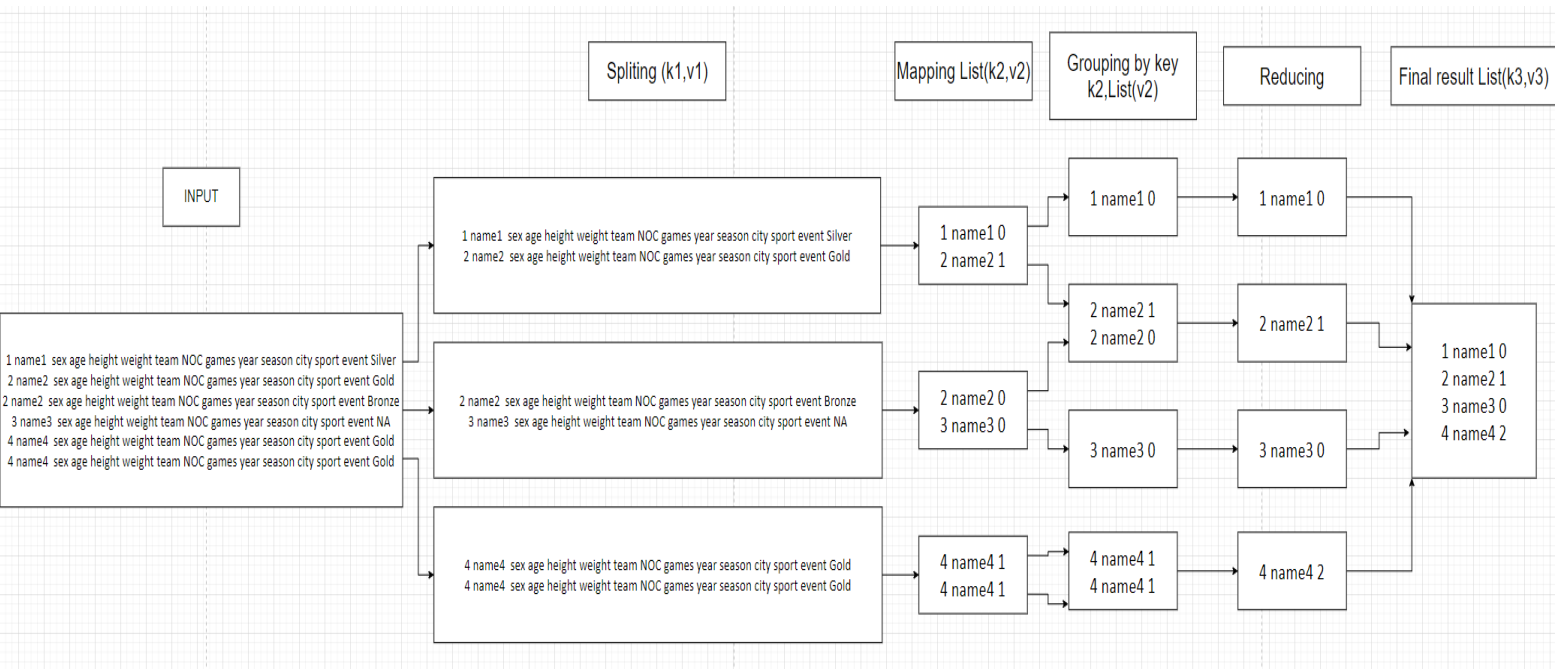
```
void Map(int key, String inputCSVline){
    if key.get() is 0 then return;           //remove first line of csv
    array[] = split inputCSVline to ",";
    foreach (Word w in inputCSVline) {
        if w is "Gold"{
            w = array[id] + array[name];
            emit(w, 1);
        }
        else{
            w = array[id] + array[name];
            emit(w, 0);
        }
    }
}

void Reduce(Word w, int[] counts){
    int sum = 0;
    foreach (int i in counts) {
        sum += i;
    }
    emit(w, sum);
}
```

Παράδειγμα:

Input CSV file	Map	Reduce	Output
1 name1 ... Silver	1 name1 0	1 name1 0	1 name1 0
2 name2 ... Gold	2 name2 1	2 name2 1	2 name2 1
2 name2 ... Bronze	2 name2 0	3 name3 0	3 name3 0
3 name3 ... NA	3 name3 0	4 name3 2	4 name3 2
4 name4 ... Gold	4 name4 1		
4 name4 ... Gold	4 name4 1		

Σχηματική εκτέλεση:



Απάντηση ερωτήματος 1 Β:

Τρόπος Εκτέλεσης:

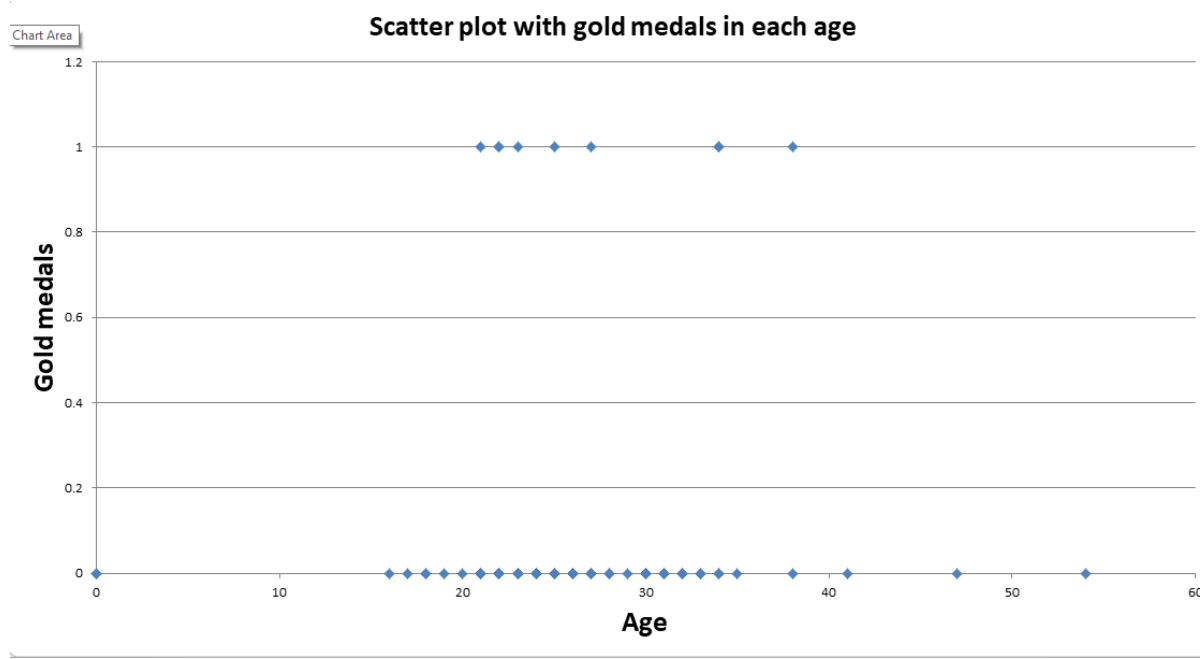
```
hadoop@lampros-VirtualBox:~/hadoop-3.2.1/sbin$ hadoop jar '/home/lampros/erwthma1.jar' Athletes /in10000 /out1 /out2
```

Ενδεικτικό αποτέλεσμα:

part-r-00000(1)				
1	1	A Dijiang,M,24,	0	
2	2	A Lamusi,M,23,	0	
3	3	Gunnar Nielsen Aaby,M,24,	0	
4	4	Edgar Lindenau Aabye,M,34,	1	
5	5	Christine Jacoba Aaftink,F,21,	0	
6	6	Per Knut Aaland,M,31,	0	
7	7	John Aalberg,M,31,	0	
8	8	"Cornelia "Cor" Aalten (-Strannood)",F,18,	0	
9	9	Antti Sami Aalto,M,26,	0	
10	10	"Einar Ferdinand "Einari" Aalto",M,26,	0	
11	11	Jorma Ilmari Aalto,M,22,	0	
12	12	Jyri Tapani Aalto,M,31,	0	
13	13	Minna Maarit Aalto,F,30,	0	
14	14	Pirjo Hannele Aalto (Mattila-),F,32,	0	
15	15	Arvo Ossian Aaltonen,M,34,	0	
16	16	Juhamatti Tapio Aaltonen,M,28,	0	
17	17	Paavo Johannes Aaltonen,M,32,	0	
18	18	Timo Antero Aaltonen,M,31,	0	
19	19	Win Valdemar Aaltonen,M,54,	0	
20	20	Kjetil Andr Aamodt,M,34,	1	

Σχολιασμός αποτελέσματος:

Τα δύο tabs δεν αφαιρέθηκαν για την ωραιοποίηση του αποτελέσματος, γιατί η χρήση ακόμα μιας `mapreduce` κρίθηκε μη σκόπιμη. Επίσης, οι αθλητές των οποίων το όνομα περιήχε «'''», αναγκαστικά λόγω του `regex` που χρησιμοποιήθηκε στην συνάρτηση `split()`, εμφανίζονται με παραπανίσια «'''».



Ανάλυση διαγράμματος:

Χρησιμοποιήθηκε scatter plot για την γραφική αναπαράσταση των χρυσών μεταλλίων σε σχέση με την ηλικία του κάθε αθλητή, για τις 100 πρώτες εγγραφές του αρχείου εξόδου. Οι αθλητές με άγνωστη ηλικία παρουσιάζονται με ηλικία 0. Παρατηρούμε, ότι για τις ηλικίες από 20-28 έχουμε τα περισσότερα χρυσά μετάλλια. Αυτό ήταν και το αναμενόμενο αποτέλεσμα αφού, γενικά, οι νεότεροι αθλητές έχουν καλύτερες επιδόσεις.

Επεξήγηση υλοποίησης:

- Χρησιμοποιήθηκε μια `map` (`AMapper`) για την τμηματοποίηση του αρχείου εισόδου CSV, την εύρεση χρυσού μεταλλίου για κάθε αθλητή/αθλήτρια και την επιλογή των κατάλληλων τμημάτων. Έπειτα, χρησιμοποιήθηκε μια `reduce` (`AReducer`) για την εύρεση του αθροίσματος των χρυσών μεταλλίων και γίνεται εκτύπωση των δεδομένων στο πρώτο αρχείο εξόδου.
- Στην συνέχεια, υλοποιήθηκε μια δεύτερη `map` (`GMapper`) που λαμβάνει και τμηματοποιεί τα δεδομένα από το πρώτο αρχείο εξόδου, ενώ, η δεύτερη `reduce` (`GReducer`) εκτυπώνει όλα τα δεδομένα με βάση το `id`, αφαιρώντας έτσι όλα τα διπλότυπα.

Απάντηση ερωτήματος 2 Α:

Ψευδοκώδικας:

//Πρώτη **mapreduce** για την εύρεση των μεταλλίων του κάθε αθλητή

```
void Map(int key, String line){
    if key.get() is 0 then return;           //remove first line of csv
    array[] = split line to ",";
    for each (Word w in line) {
        w = array[id] + array[name] + array[games];
        if w is "Gold" then emit(w, "100");
        else if w is "Silver" then emit(w,"010");
        else if w is "Bronze" then emit(w,"001");
        else emit(w, "000");
    }
}

void Reduce(String w, String[] medals){
    String[] sumArray = 0,0,0;
    for each (String i in medals) {
        sumArray[] = sumArray[]+medals[];
    }
    emit(w, sumArray);
}
```

//Δεύτερη **mapreduce** για την εύρεση του συνόλου των μεταλλίων του κάθε αθλητή σε μια μόνο διοργάνωση

```
void Map(int key, String line){
    array[] = split line to ",";
    String IdNameGames = array[id]+array[name]+array[games];
    emit(IdNameGames, line);
}

void Reduce(String IdNameGames, String line){
    String[] sum = 0,0,0;
    for each (String i in line) {
        array[] = split line to ",";
        String medalFields = array[golds]+array[silvers]+array[bronzes];
        sum[] = sum[] + medalFields[];
        int sumOfMedals = sum[golds] + sum[silvers] + sum[bronzes];
        String finalString = sum[golds] + sum[silvers] + sum[bronzes] + sumOfMedals;
        emit(array[id]+array[name]+array[games] ,finalString);
    }
}
```

Παράδειγμα:

Input CSV file

```
1 name1 ...Games1 ... Silver
2 name2 ...Games1 ... Gold
2 name2 ...Games1 ... Bronze
3 name3 ...Games2 ... NA
3 name3 ...Games2 ... Silver
4 name4 ...Games1 ... Gold
4 name4 ...Games2 ... Gold
```

Map1

```
1 name1 Games1 0 1 0
2 name2 Games1 1 0 0
2 name2 Games1 0 0 1
3 name3 Games2 0 0 0
3 name3 Games2 0 1 0
4 name4 Games1 1 0 0
4 name4 Games2 1 0 0
```

Reduce1

```
1 name1 Games1 0 1 0
2 name2 Games1 1 0 1
3 name3 Games2 0 1 0
4 name4 Games1 1 0 0
4 name4 Games2 1 0 0
```

Output1

```
1 name1 Games1 0 1 0
2 name2 Games1 1 0 1
3 name3 Games2 0 1 0
4 name4 Games1 1 0 0
4 name4 Games2 1 0 0
```

Map2

```
1 name1 Games1 0 1 0
2 name2 Games1 1 0 1
3 name3 Games2 0 1 0
4 name4 Games1 1 0 0
4 name4 Games2 1 0 0
```

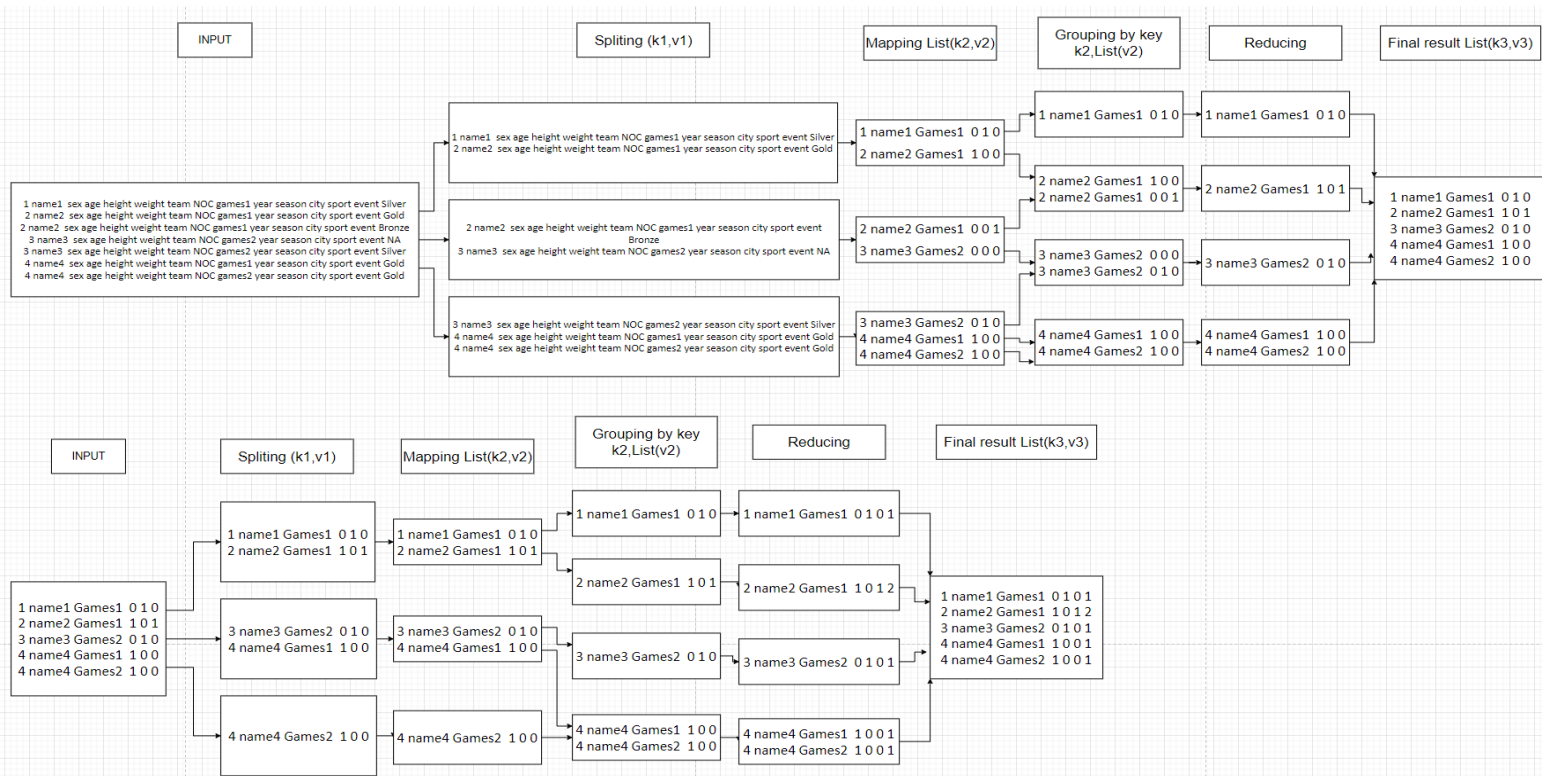
Reduce2

```
1 name1 Games1 0 1 0 1
2 name2 Games1 1 0 1 2
3 name3 Games2 0 1 0 1
4 name4 Games1 1 0 0 1
4 name4 Games2 1 0 0 1
```

Output2

```
1 name1 Games1 0 1 0 1
2 name2 Games1 1 0 1 2
3 name3 Games2 0 1 0 1
4 name4 Games1 1 0 0 1
4 name4 Games2 1 0 0 1
```

Σχηματική εκτέλεση:



Απάντηση ερωτήματος 2 Β:

Τρόπος Εκτέλεσης:

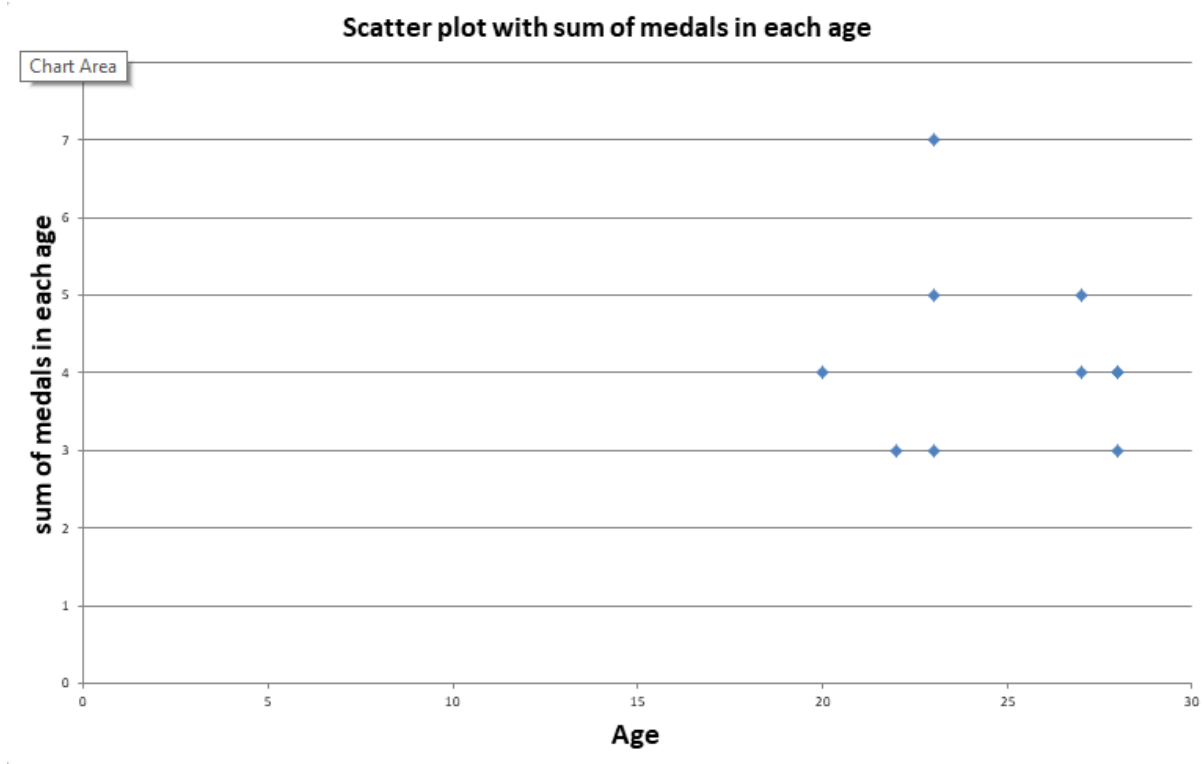
```
hadoop@lampros-VirtualBox:~/hadoop-3.2.1/sbin$ hadoop jar '/home/lampros/erwthma2.jar' Athletes /in10000 /out5 /out6 /out7 /out8
```

Ενδεικτικό αποτέλεσμα:

part-r-00000(3)		part-r-00000	
1	1,Nikolay Yefimovich Andrianov,M,23,Soviet Union,Gymnastics,1976 Summer,4,2,1,7		
2	2,Vladimir Nikolayevich Artyomov,M,23,Soviet Union,Gymnastics,1988 Summer,4,1,0,5		
3	3,Paavo Johannes Aaltonen,M,28,Finland,Gymnastics,1948 Summer,3,0,1,4		
4	3,Viktor An,M,20,South Korea,Short Track Speed Skating,2006 Winter,3,0,1,4		
5	3,Viktor An,M,28,Russia,Short Track Speed Skating,2014 Winter,3,0,1,4		
6	4,Hjalmar Johan Andersen,M,28,Norway,Speed Skating,1952 Winter,3,0,0,3		
7	5,Nikolay Yefimovich Andrianov,M,27,Soviet Union,Gymnastics,1980 Summer,2,2,1,5		
8	6,Nathan Ghar-Jun Adrian,M,27,United States,Swimming,2016 Summer,2,0,2,4		
9	7,"Christine M. "Crissy" Ahmann-Leighton (-Perham)",F,22,United States,Swimming,1992 Summer,2,1,0,3		
10	7,Agneta Monica Andersson,F,23,Sweden,Canoeing,1984 Summer,2,1,0,3		

Σχολιασμός αποτελέσματος:

Εμφανίστηκαν οι πρώτες 10 εγγραφές με τα περισσότερα χρυσά μετάλλια ταξινομημένα με τη σωστή σειρά ως προς τα συνολικά μετάλλια και έπειτα ως προς την αλφαβητική σειρά των ομάδων. Ακόμα, εμφανίστηκε σωστά ο αριθμός κατάταξης. Επίσης, οι αθλητές των οποίων το όνομα περιήχε «""», αναγκαστικά λόγω του regex που χρησιμοποιήθηκε στην συνάρτηση split(), εμφανίζονται με παραπανίσια «""».



Ανάλυση διαγράμματος:

Χρησιμοποιήθηκε scatter plot για την γραφική αναπαράσταση των συνολικών μεταλλίων σε σχέση με την ηλικία του κάθε αθλητή, για τις 10 εγγραφές του αρχείου εξόδου. Παρατηρούμε, απο την ηλικία των 20 μέχρι 28 ετών βρίσκονται τα περισσότερα μετάλλια. Αυτό ήταν και το αναμενόμενο αποτέλεσμα αφού, οι νεότεροι αθλητές έχουν καλύτερες επιδόσεις συνεπώς υπάρχει μεγαλύτερη πιθανότητα να κερδίσουν κάποιο μετάλλιο.

Επεξήγηση υλοποίησης:

- Χρησιμοποιήθηκε μια map (AMapper) για την τμηματοποίηση του αρχείου εισόδου CSV, την εύρεση όλων των μεταλλίων για κάθε αθλητή/αθλήτρια και την επιλογή των κατάλληλων τμημάτων. Έπειτα, χρησιμοποιήθηκε μια reduce (AReducer) για την εύρεση του αθροίσματος για κάθε είδος μεταλλίου και γίνεται εκτύπωση των δεδομένων στο πρώτο αρχείο εξόδου.
- Στην συνέχεια, υλοποιήθηκε μια δεύτερη map (MMapper) που λαμβάνει και τμηματοποιεί τα δεδομένα από το πρώτο αρχείο εξόδου, ενώ, η δεύτερη reduce (MReducer) αναλαμβάνει την εύρεση του συνολικού αθροίσματος μεταλλίων και την ομαδοποίηση ως προς την διοργάνωση.
- Εν συνεχεία, υλοποιήθηκε μια τρίτη map (SortMapper) που λαμβάνει και τμηματοποιεί τα δεδομένα από το δεύτερο αρχείο εξόδου, τα οποία μετά εισάγονται σε έναν comparator (SortComparator3) όπου πραγματοποιείται όλη η ταξινόμησή τους, ενώ, η τρίτη reduce (SortReducer) τμηματοποιεί τα κατάλληλα δεδομένα που δέχεται και έτσι γίνεται η εκτύπωσή τους στο τρίτο αρχείο εξόδου.
- Κατόπιν, υλοποιήθηκε μια τέταρτη map (Top10Mapper) που λαμβάνει και τμηματοποιεί τα δεδομένα από το τρίτο αρχείο εξόδου, προσθέτει την κατάταξη των αθλητών και στέλνει στον comparator τις 10 πρώτες εγγραφές. Ο comparator (SortComparator4) πραγματοποιεί όλη την ταξινόμηση των δεδομένων. Οπότε, μετά την τέταρτη reduce (Top10Reducer) γίνεται η εκτύπωση των εγγραφών στο τέταρτο και τελικό αρχείο εξόδου.

Απάντηση ερωτήματος 3 Α:

Ψευδοκώδικας:

//Mapreduce για την εύρεση των γυναικίων συμμετοχών σε κάθε ομάδα.

```
void Map(int key, String inputCSVline){
    if key.get() is 0 then return;           //remove first line of csv
    array[] = split inputCSVline to ",";
    foreach (Word w in array){
        if w is "F"{
            w = array[games] + array[team] + array[sport];
            emit(w, 1);
        }
        else{
            w = array[games] + array[team] + array[sport];
            emit(w, 0);
        }
    }
}

void Reduce(Word w, int counts){
    int sum = 0;
    foreach (int i in counts) {
        sum += i;
    }
    emit(w, sum);
}
```

//Mapreduce για την αφαίρεση των δεδομένων με τα πεδία "ομάδα" ίδια, τα οποία δεν έχουν τις μέγιστες γυναικείες συμμετοχές, για την εμφάνιση του αθλήματος πρώτης επιλογής.

```
void Map(int key, String inputline){
    array[]=inputline.split();
    String gameteam = array[game]+ array[team];
    emit(gameteam, inputline);
}

void Reduce(Word gameteam, String FullLine){
    emit(gameteam, FullLine);
}

}
```

//Mapreduce για την εμφάνιση των τριών πρώτων ομάδων σε κάθε διαφορετική διοργάνωση.

```
void Map(int gameteam, String FullLine){
```

```

    array[]=inputline.split();
    remove gamesteam from array;
    foreach (FullLine) find max1;
    foreach (FullLine) find max2;
    foreach (FullLine) find max3;
    foreach (FullLine){
        if(array[femaleParticipations]=(max1|max2|max3)){
            emit(FullLine,Null);
        }
    }
}

void Reduce(Word FullLine, Null){
    emit(FullLine, Null);
}
}

```

Παράδειγμα:

Input CSV file

```

... F ... team1 ... games1 ... sport1 ...
... F ... team2 ... games1 ... sport2 ...
... F ... team3 ... games2 ... sport1 ...
... F ... team2 ... games1 ... sport2 ...
... F ... team1 ... games1 ... sport2 ...
... F ... team4 ... games2 ... sport2 ...
... F ... team3 ... games2 ... sport1 ...
... F ... team4 ... games2 ... sport2 ...
... F ... team4 ... games2 ... sport2 ...
... F ... team4 ... games2 ... sport2 ...
... F ... team4 ... games2 ... sport1 ...
... F ... team5 ... games2 ... sport3 ...
... F ... team5 ... games2 ... sport3 ...
... F ... team5 ... games2 ... sport3 ...
... F ... team6 ... games2 ... sport3 ...
... M ... team6 ... games1 ... sport2 ...
... M ... team2 ... games2 ... sport2 ...

```

Map1

```

games1 team1 sport1
games1 team2 sport2
games2 team3 sport1
games1 team2 sport2
games1 team1 sport2
games2 team4 sport2
games2 team3 sport1
games2 team4 sport2
games2 team4 sport2
games2 team4 sport2
games2 team4 sport1
games2 team5 sport3
games2 team5 sport3
games2 team5 sport3
games2 team6 sport3

```

Reduce1

```

games1 team2 2 sport2
games1 team1 1 sport2
games1 team1 1 sport1
games2 team3 2 sport1
games2 team4 4 sport2
games2 team4 1 sport1
games2 team5 3 sport3
games2 team6 1 sport3

```

Output1

```

games1 team2 2 sport2
games1 team1 1 sport2
games1 team1 1 sport1
games2 team3 2 sport1
games2 team4 4 sport2
games2 team4 1 sport1
games2 team5 3 sport3
games2 team6 1 sport3

```

Map2

```

games1 team2 2 sport2
games1 team1 1 sport2
games1 team1 1 sport1
games2 team3 2 sport1
games2 team4 4 sport2
games2 team4 1 sport1
games2 team5 3 sport3
games2 team6 1 sport3

```

Reduce2

```

games1 team2 2 sport2
games1 team1 1 sport2
games2 team3 2 sport1
games2 team4 4 sport2
games2 team5 3 sport3
games2 team6 1 sport3

```

Output2

games1 team2 2 sport2
games1 team1 1 sport2
games2 team3 2 sport1
games2 team4 4 sport2
games2 team5 3 sport3
games2 team6 1 sport3

Map3

games1 team2 2 sport2
games1 team1 1 sport2
games2 team3 2 sport1
games2 team4 4 sport2
games2 team5 3 sport3
games2 team6 1 sport3

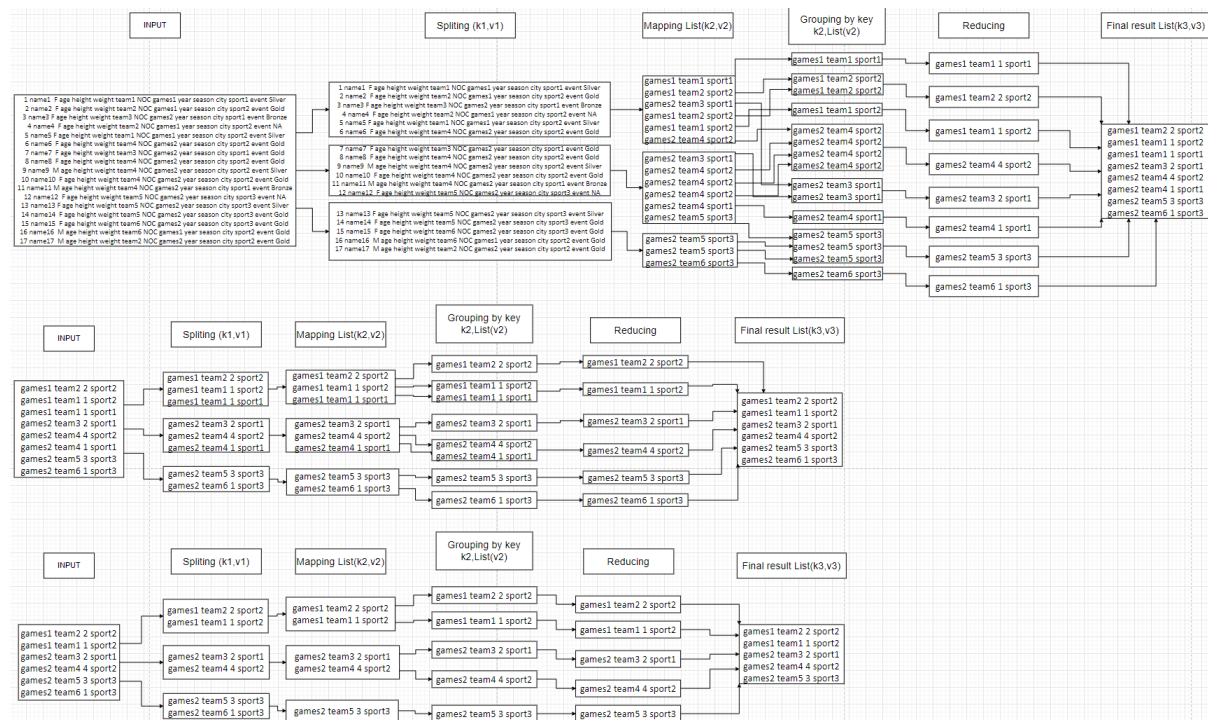
Reduce3

games1 team2 2 sport2
games1 team1 1 sport2
games2 team3 2 sport1
games2 team4 4 sport2
games2 team5 3 sport3

Output3

games1 team2 2 sport2
games1 team1 1 sport2
games2 team3 2 sport1
games2 team4 4 sport2
games2 team5 3 sport3

Σχηματική εκτέλεση:





Απάντηση ερωτήματος 3 Β:


Τρόπος Εκτέλεσης:

```
hadoop@lanpros-VirtualBox:~/hadoop-3.2.1/sbin$ hadoop jar '/home/lanpros/erwthma3.jar' Athletes /in10000 /out1 /out2 /out3 /out4 /out5
```

Ενδεικτικό αποτέλεσμα:

Activities  Text Editor ▾

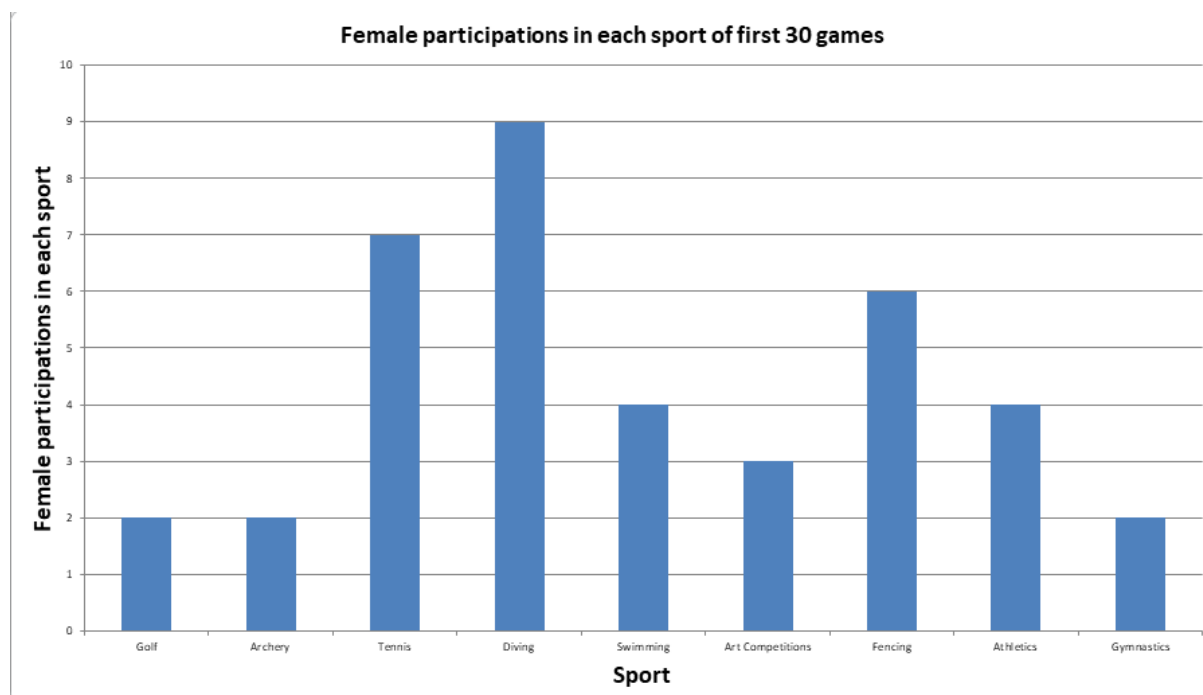
Open ▾ 

part-r-00000(4) 

1	1900	Summer, United States, USA, 2, Golf
2	1908	Summer, Great Britain, GBR, 2, Archery
3	1908	Summer, Sweden, SWE, 1, Tennis
4	1912	Summer, Sweden, SWE, 3, Diving
5	1912	Summer, Austria, AUT, 2, Swimming
6	1912	Summer, Great Britain, GBR, 1, Tennis
7	1912	Summer, Great Britain-1, GBR, 1, Tennis
8	1912	Summer, Sweden-1, SWE, 1, Tennis
9	1912	Summer, Sweden-3, SWE, 1, Tennis
10	1920	Summer, Belgium, BEL, 2, Tennis
11	1920	Summer, Sweden, SWE, 2, Diving
12	1920	Summer, Great Britain, GBR, 1, Diving
13	1920	Summer, United States, USA, 1, Diving
14	1924	Summer, Spain, ESP, 2, Tennis
15	1924	Summer, Austria, AUT, 1, Diving
16	1924	Summer, France, FRA, 1, Art Competitions
17	1924	Summer, Great Britain, GBR, 1, Diving
18	1924	Summer, Netherlands, NED, 1, Fencing
19	1924	Summer, Spain-1, ESP, 1, Tennis
20	1928	Summer, Belgium, BEL, 2, Fencing

Σχολιασμός αποτελέσματος:

Εμφανίστηκαν σωστά οι εγγραφές ταξινομημένες πρώτα ως προς την διοργάνωση και έπειτα ως προς τις γυναικείες συμμετοχές ή την αλφαβητική σειρά των ομάδων.



Ανάλυση διαγράμματος:

Χρησιμοποιήθηκε bar graph για την γραφική αναπαράσταση των γυναικείων συμμετοχών σε κάθε άθλημα, για τις 30 πρώτες εγγραφές του αρχείου εξόδου. Παρατηρούμε, ότι τα αθλήματα της κολύμβησης, της ξιφασκίας και του tennis έχουν τις περισσότερες γυναικείες συμμετοχές.

Επεξήγηση υλοποίησης:

- Χρησιμοποιήθηκε μια map (FemalesMapper) για την τμηματοποίηση του αρχείου εισόδου CSV και την εύρεση γυναικείας συμμετοχής. Έπειτα, χρησιμοποιήθηκε μια reduce (FemalesReducer) για την εύρεση του αθροίσματος των γυναικείων συμμετοχών για κάθε διαφορετική ομάδα και γίνεται εκτύπωση των δεδομένων στο πρώτο αρχείο εξόδου.
- Στην συνέχεια, υλοποιήθηκε μια δεύτερη map (TabsMapper) που λαμβάνει, τμηματοποιεί τα δεδομένα από το πρώτο αρχείο εξόδου και αναδιατάσει τα πεδία στην σωστή σειρά. Μετά, χρησιμοποιήθηκε ένας comparator ο οποίος αναλαμβάνει την ταξινόμηση (SortComparator) αρχικά ως προς την διοργάνωση και τέλος ως προς το πλήθος των γυναικείων συμμετοχών ή ως προς την αλφαβητική σειρά των ομάδων (όπου αυτό χρειάζεται). Έπειτα, μετά την δεύτερη reduce (TabsReducer) γίνεται εκτύπωση των νέων δεδομένων στο δεύτερο αρχείο εξόδου.
- Εν συνεχεία, υλοποιήθηκε μια τρίτη map (DuplicatesMapper) που λαμβάνει, τμηματοποιεί τα δεδομένα από το δεύτερο αρχείο εξόδου και τα εκτυπώνει με τα 3 πρώτα πεδία (GTN) ως key και όλα τα δεδομένα ως value. Η τρίτη reduce (DuplicatesReducer) τμηματοποιεί τα δεδομένα που δέχεται από την map, αφαιρεί τις εγγραφές που έχουν τα πεδία “ομάδες” και “NOC” όμοια που δεν έχουν τις μέγιστες γυναικείες συμμετοχές σε μία διοργάνωση. Έπειτα, γίνεται η εκτύπωσή τους στο τρίτο αρχείο εξόδου.
- Κατόπιν, υλοποιήθηκε μια τέταρτη map (GTNMapper) που λαμβάνει και τμηματοποιεί τα δεδομένα από το τρίτο αρχείο εξόδου, διαγράφει τα πεδία Game Team Noc που είχαν προστεθεί από την DuplicatesMapper και έπειτα ταξινομούνται πάλι τα δεδομένα με τον ίδιο comparator (SortComparator). Οπότε, η τέταρτη reduce (GTNReducer) λαμβάνει στην σωστή μορφή τις εγγραφές και γίνεται η εκτύπωσή τους στο τέταρτο αρχείο εξόδου.
- Στην συνέχεια, υλοποιήθηκε μια πέμπτη map (ThreeGamesMapper) που λαμβάνει, τμηματοποιεί τα δεδομένα από το τέταρτο αρχείο εξόδου και κρατάει μόνο τις πρώτες 3 γραμμές για κάθε διαφορετική διοργάνωση (μπορεί να είναι περισσότερες ή λιγότερες). Μετά την map, ο comparator (SortComparator) αναλαμβάνει την ταξινόμηση. Τέλος, μετά την πέμπτη reduce γίνεται η εκτύπωσή των δεδομένων στο πέμπτο και τελικό αρχείο εξόδου.

Λεπτομέριες υλοποίησης:

- Όλα τα πεδία των εγγραφών διαχωρίζονται με κόμματα αντί για κενά.
- Το regex που χρησιμοποιήθηκε στην συνάρτηση split() λαμβάνει υπόψη την περίπτωση όπου ένα πεδίο στο αρχείο περιέχει κόμμα (αυτό είναι απαραίτητο διότι έχουν προστεθεί κόμματα ανάμεσα στα πεδία).
- Τα ερωτήματα έχουν ελεγχθεί με τις πρώτες 10.000 εγγραφές του αρχείου athlete_events.csv (το συγκεκριμένο αρχείο παρέχεται στα έγγραφα της άσκησης).
- Το κάθε αρχείο που παράγει η mapreduce χρησιμοποιείται σαν όρισμα για την επόμενη.

Αναλυτικές οδηγίες για την εκτέλεση του προγράμματος:

Αφού έχουμε εγκαταστήσει σωστά την Java και το Hadoop εκτελούμε τις ακόλουθες εντολές με την σειρά για την εκτέλεση του προγράμματος:

```
su hadoop
```

```
cd /hadoop/hadoop-3.2.1/sbin
```

```
/start-dfs.sh
```

```
/start-yarn.sh
```

```
jps
```

```
hadoop fs -mkdir -p /input
```

```
hadoop fs -put /input_folder_from_local_files /input
```

```
hadoop jar /jar_from_local_files ClassName /input /output
```

```
hadoop fs -cat /output/part-r-00000
```