ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ

ΡΟΗ Λ – ΕΞΑΜΗΝΟ 9ο
ΑΚ. ΕΤΟΣ 2021 - 2022
ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Χρήση του Apache Spark στις Βάσεις Δεδομένων

Στην παρούσα εργασία, θα γίνει χρήση του Apache Spark για τον υπολογισμό αναλυτικών ερωτημάτων πάνω σε αρχεία που περιγράφουν σύνολα δεδομένων. Το Apache Spark προσφέρει δύο βασικά APIs για την υλοποίηση αναλυτικών ερωτημάτων:

- RDD API
  - ✓ https://spark.apache.org/docs/2.4.4/rdd-programming-guide.html
- Dataframe API / Spark SQL
  - ✓ https://spark.apache.org/docs/2.4.4/sql-programming-guide.html

Στην χρήση του δευτέρου API στην εργασία συνίσταται η υλοποίηση με παραδοσιακή SQL όπως περιγράφεται στην ενότητα https://spark.apache.org/docs/2.4.4/sql-programming-guide.html#running-sql-queries-programmatically

Για τις ανάγκες της εργασίας θα χρησιμοποιηθεί ένα σύνολο δεδομένων από ταινίες, το οποίο προέρχεται από το σύνολο Full MovieLens Dataset, το οποίο μπορείτε να βρείτε εδώ. Στην εργασία θα χρησιμοποιήσουμε ένα υποσύνολο μίας εκδοχής του που διατίθεται στο Kaggle, το οποίο μπορείτε να κατεβάσετε μέσω του ακόλουθου συνδέσμου:

http://www.cslab.ntua.gr/courses/atds/movie_data.tar.gz

Στο συμπιεσμένο αρχείο που κατεβάσατε, περιέχονται τρία αρχεία κειμένου στην μορφή CSV, τα movie_genres.csv, movies.csv και ratings.csv με συνολικό μέγεθος 700MB μετά την αποσυμπίεση. Για κάθε αρχείο, δίνουμε στην συνέχεια τα πρώτα δύο rows του και μία σύντομη επεξήγηση της κάθε στήλης που υπάρχει μετά δύο διαχωριστικών («,») του CSV.

- Αρχείο movies.csv (17 MB)
  Το αρχείο movies.csv περιγράφει τις ταινίες που υπάρχουν στο σύνολο δεδομένων. Το πρώτο, το δεύτερο και το τρίτο πεδίο αποτελούν το αναγνωριστικό, τον τίτλο και την περίληψη της ταινίας. Το τέταρτο πεδίο δίνει την ημερομηνία κυκλοφορίας (ως κάποιο timestamp πχ 1995-10-30T00:00:00.000+02:00) , ενώ το πέμπτο τη διάρκεια της ταινίας σε λεπτά. Το έκτο και το έβδομο πεδίο αποτελούν το κόστος παραγωγής και τα έσοδα από την προβολή της ταινίας αντίστοιχα. Το τελευταίο και όγδοο πεδίο περιγράφει την δημοτικότητα της ταινίας. Το αρχείο αποτελείται από περίπου 45K εγγραφές.

```
862,Toy Story,Led by Woody Andys toys live happily in his room until Andys birthday brings Buzz Lightyear onto the scene Afraid of losing his place in Andys heart Woody plots against Buzz But when circumstances separate Buzz and Woody from their owner the duo eventually learns to put aside their differences,1995-10-30T00:00:00.000+02:00,81.0,30000000,373554033,21.946943

8844,Jumanji,When siblings Judy and Peter discover an enchanted board game that opens the door to a magical world they unwittingly invite Alan  an adult whos been trapped inside the game for 26 years  into their living room Alans only hope for freedom is to finish the game which proves risky as all three find themselves running from giant rhinoceroses evil monkeys and other terrifying creatures,1995-12-15T00:00:00.000+02:00,104.0,65000000,262797249,17.015539
```

- Αρχείο movie_genres.csv (1.3 MB)
  Το αρχείο movie_genres.csv περιέχει σε κάθε γραμμή ένας ζεύγος αναγνωριστικού ταινίας και κατηγορίας ταινιών στην πρώτη και δεύτερη θέση αντίστοιχα. Μία ταινία μπορεί να ανήκει σε περισσότερες από μία κατηγορίες και έτσι για κάθε ταινία ενδέχεται να υπάρχουν πάνω από μία γραμμές. Το αρχείο αποτελείται από περίπου 91K εγγραφές.

```
862,Animation
862,Comedy
862,Family
8844,Adventure
8844,Fantasy
8844,Family
```

- Αρχείο ratings.csv (677 MB)
  Το αρχείο ratings.csv περιγράφει τις αξιολογήσεις των χρηστών για τις ταινίες, δηλαδή ότι ο χρήστης x (πεδίο 1ο) αξιολόγησε την ταινία y (πεδίο 2ο) με βαθμολογία z (πεδίο 3ο) σε κάποια χρονική στιγμή (πεδίο 4ο). Το αρχείο αποτελείται από περίπου 26M εγγραφές.

```
1,110,1.0,1425941529
1,147,4.5,1425942435
1,858,5.0,1425941523
1,1221,5.0,1425941546
```

Μέρος 1ο: Υπολογισμός Αναλυτικών Ερωτημάτων με τα APIs του Apache Spark

Στο πρώτο μέρος της εργασίας θα υπολογιστούν τα αποτελέσματα για 5 ερωτήματα που παρουσιάζουν ενδιαφέρον από το διαθέσιμο σύνολο δεδομένων. Τα ερωτήματα παρουσιάζονται στον Πίνακας **1**.

Πίνακας 1: Τα ερωτήματα που ζητείται να απαντηθούν στο 1ο Μέρος της άσκησης.

# Ερωτήματος	Λεκτική Περιγραφή
Q1	Από το 2000 και μετά, να βρεθεί για κάθε χρονιά η ταινία με το μεγαλύτερο κέρδος, Αγνοείστε εγγραφές που δεν έχουν τιμή στην ημερομηνία κυκλοφορίας ή μηδενική τιμή στα έσοδα ή στον προϋπολογισμό.
Q2	Να βρεθεί το ποσοστό των χρηστών (%) που έχουν δώσει σε ταινίες μέση βαθμολογία μεγαλύτερη από 3.
Q3	Για κάθε είδος ταινίας, να βρεθεί η μέση βαθμολογία του είδους και το πλήθος των ταινιών που υπάγονται σε αυτό το είδος. Αν μία ταινία αντιστοιχεί σε περισσότερα του ενός είδη, θεωρούμε ότι μετριέται σε κάθε είδος.
Q4	Για τις ταινίες του είδους "Drama", να βρεθεί το μέσο μήκος περίληψης ταινίας σε λέξεις ανά 5ετία από το 2000 και μετά (1η 5ετία: 2000-2004, 2η 2005-2009, 3η 2010 – 2014, 4η 2015 - 2019). Αγνοείστε ταινίες στις οποίες απουσιάζει η περίληψη.
Q5	Για κάθε είδος ταινίας, να βρεθεί ο χρήστης με τις περισσότερες κριτικές, μαζί με την περισσότερο και την λιγότερο αγαπημένη του ταινία σύμφωνα με τις αξιολογήσεις του. Για την περίπτωση που ο χρήστης έχει την ίδια ψηλότερη / χαμηλότερη βαθμολογία σε περισσότερες από 1 ταινίες επιλέξτε την πιο δημοφιλή ταινία από αυτές της κατηγορίας που συμπίπτει η βέλτιστη / χείριστη βαθμολογία του χρήστη. Τα αποτελέσματα να είναι σε αλφαβητική σειρά ως προς την κατηγορία ταινίας και να παρουσιαστούν σε ένα πίνακα με τις ακόλουθες στήλες. • είδος • χρήστης με περισσότερες κριτικές • πλήθος κριτικών, • περισσότερο αγαπημένη ταινία • βαθμολογία περισσότερο αγαπημένης ταινίας • λιγότερο αγαπημένη ταινία • βαθμολογία λιγότερο αγαπημένης ταινίας

Επιπλέον διευκρινίσεις και βοήθεια σχετικά με τα ερωτήματα δίνεται στις *Υποδείξεις* της παρούσας ενότητας. Τα ζητούμενα του πρώτου μέρους είναι τα ακόλουθα:

Ζητούμενο 1 (0.5 Μονάδες): Φορτώστε τα 3 CSV αρχεία που σας δόθηκαν στο hdfs σε ένα φάκελο **files**.

Ζητούμενο 2 (0.5 Μονάδες): Όπως αναφέρθηκε τα δεδομένα σας δίνονται σε μορφή απλού κειμένου (csv). Παρόλα αυτά, είναι γνωστό ότι ο υπολογισμός ερωτημάτων αναλυτικής επεξεργασίας απευθείας πάνω σε αρχεία csv δεν είναι αποδοτικός. Για να βελτιστοποιηθεί η πρόσβαση των δεδομένων, παραδοσιακά οι βάσεις δεδομένων φορτώνουν τα δεδομένα σε ειδικά σχεδιασμένα binary formats. Παρ'ότι το Spark δεν είναι μια τυπική βάση δεδομένων, αλλά ένα σύστημα κατανεμημένης επεξεργασίας, για λόγους απόδοσης, υποστηρίζει κι αυτό μια παρόμοια λογική. Αντί να τρέξουμε τα ερωτήματά μας απευθείας πάνω στα csv αρχεία, μπορούμε να μετατρέψουμε πρώτα το dataset σε μια ειδική μορφή που:

- Έχει μικρότερο αποτύπωμα στη μνήμη και στον δίσκο και άρα βελτιστοποιεί το I/O (input/output) μειώνοντας τον χρόνο εκτέλεσης.
- Διατηρεί επιπλέον πληροφορία, όπως στατιστικά πάνω στο dataset, τα οποία βοηθούν στην πιο αποτελεσματική επεξεργασία του. Για παράδειγμα, αν ψάχνω σε ένα σύνολο δεδομένων τις τιμές που είναι μεγαλύτερες από 100 και σε κάθε block του dataset έχω πληροφορία για το ποια είναι η min και ποια η max τιμή, τότε μπορώ να παρακάμψω την επεξεργασία των blocks με max τιμή < 100 γλιτώνοντας έτσι χρόνο επεξεργασίας.

Το ειδικό format που χρησιμοποιούμε για να επιτύχουμε τα παραπάνω είναι το Apache Parquet. Όταν φορτώνουμε έναν πίνακα σε Parquet, αυτός μετατρέπεται κι αποθηκεύεται σε ένα columnar format που βελτιστοποιεί το I/O και τη χρήση της μνήμης κι έχει τα χαρακτηριστικά που αναφέραμε. Περισσότερες πληροφορίες σχετικά με το Parquet μπορείτε να βρείτε εδώ. Από άποψη κώδικα, η μετατροπή ενός dataset σε Parquet είναι ιδιαίτερα απλή. Παραδείγματα και πληροφορίες για το πώς διαβάζω και γράφω Parquet αρχεία μπορείτε να βρείτε εδώ.
Στο συγκεκριμένο ερώτημα, ζητείται να μετατρέψετε κάθε ένα csv που υπάρχει στο hdfs σε Parquet μορφή, διαβάζοντας κάθε CSV σε dataframe και αποθηκεύοντας το στη συνέχεια σε parquet μορφή πίσω στο hdfs (συμβουλευτείτε και τις παραπάνω οδηγίες). Τελικά θα πρέπει να υπάρχουν 6 αρχεία στο hdfs, 3 CSV και 3 parquet.

Ζητούμενο 3 (5.5 Μονάδες): Για κάθε ερώτημα του Πίνακα 1 υλοποιείστε μία λύση με το RDD API και μία με Spark SQL, η οποία θα μπορεί να διαβάζει είτε αρχεία CSV χρησιμοποιώντας το option inferSchema είτε αρχεία Parquet. Οι μονάδες κατανέμονται ως εξής: Q1 - 0.75 μονάδες, Q2 - 0.75 μονάδες, Q3 - 1 μονάδα, Q4 - 1 μονάδα, Q5 - 2 μονάδες, μοιρασμένες για κάθε υλοποίηση.

Ζητούμενο 4 (0.5 Μονάδες): Να εκτελεστούν οι υλοποιήσεις του ζητούμενου 3 για κάθε query. Συγκεκριμένα, θέλουμε τα αποτελέσματα και τους χρόνους εκτέλεσης από τις 3 ακόλουθες περιπτώσεις:

1. Map Reduce Queries – RDD API
2. Spark SQL με είσοδο το csv αρχείο (συμπεριλάβετε infer schema)
3. Spark SQL με είσοδο το parquet αρχείο

Δώστε τους χρόνους εκτέλεσης σε ένα ραβδόγραμμα, ομαδοποιημένους ανά Ερώτημα Σχολιάστε τα αποτελέσματα σε κάθε query. Για να λάβετε σωστά τους χρόνους εκτέλεσης, φροντίστε να κάνετε write το αποτέλεσμα του κάθε query σε csv στο hdfs (**σε ένα φάκελο outputs**), καθώς το spark έχει lazy evaluation, υπολογίζει ότι χρειάζεται και επομένως για αντικειμενικές μετρήσεις χρειάζεται να ζητηθεί ολόκληρο το αποτέλεσμα. Τι παρατηρείται με τη χρήση του parquet ? Γιατί δεν χρησιμοποιείται το infer schema? (0.5 μονάδες)

Υποδείξεις:

1. Για το διάβασμα του αρχείου movies, χρειάζεται προσοχή στο RDD API καθώς υπάρχει ο διαχωριστικός χαρακτήρας «,» μέσα σε κάποιους τίτλους ταινιών. Καθαρά για αυτήν την περίπτωση δίνεται στην συνέχεια μία συνάρτηση διάσπασης μίας γραμμής του αρχείου αυτού σε γλώσσα Python.

```
from io import StringIO
import csv
def split_complex(x):
      return list(csv.reader(StringIO(x), delimiter=','))[0]
```

2. Ως κέρδος στο Q1 θεωρείτε το
$$\frac{Έσοδα - Κόστος}{Κόστος} \cdot 100$$
3. Ως μέση βαθμολογία του είδους στο Q3, θεωρούμε τη μέση τιμή της μέσης βαθμολογίας της κάθε ταίνιας, σύμφωνα με τις αξιολογήσεις των χρηστών, δηλαδή το:

$$\frac{\Sigma_{ταινίες}\left(\frac{1}{\#αξιολόγησεων\ ταινίας}\cdot\Sigma_{αξιολογήσεις}\,αξιολόγηση\right)}{\#ταινιών\ στην\ κατηγορία}$$

Ενδεικτικά, για επαλήθευση της λύσης σας δίνεται η γραμμή της απαντήσης του ερωτήματος για την κατηγορία "Action"

(Action, 3.1585329294510713, 1175)

4. Για την εύρεση της 5ετίας στην οποία ανήκει μια ταινία και για την εύρεση του μήκους της περίληψης μίας ταινίας μπορείτε να χρησιμοποιήσετε udfs για την περίπτωση των sql ερωτημάτων για το Q4.
5. Ενδεικτικά, για επαλήθευση της λύσης σας στο Q5 δίνεται η γραμμή της απαντήσης του ερωτήματος για την κατηγορία "Action"

('Action', 8659, 588, Hulk, 5.0, The Day After Tomorrow, 2.0)

Μέρος 2ο: Υλοποίηση και μελέτη συνένωσης σε ερωτήματα και Μελέτη του βελτιστοποιητή του Spark

Στο δεύτερο θέμα καλείστε να μελετήσετε και να αξιολογήσετε τις διαφορετικές υλοποιήσεις που υπάρχουν στο περιβάλλον Map-Reduce του Spark για τη συνένωση (join) δεδομένων και συγκεκριμένα το repartition join (aka Reduce-Side join) (Παράγραφος 3.1 και ψευδοκώδικας Α.1 της δημοσίευσης) και το broadcast join (aka Map-Side join) (Παράγραφος 3.2 και ψευδοκώδικας Α.4) όπως έχουν περιγραφεί στην δημοσίευση "A Comparison of Join Algorithms for Log Processing in MapReduce", Blanas et al , in Sigmod 2010". Το broadcast join θεωρείται πιο 2 αποδοτικό σε περίπτωση join ενός μεγάλου fact table και ενός σχετικά μικρότερου dimension table .

Ζητούμενο 1 (1 Μονάδα): Υλοποιείστε το broadcast join στο RDD API (Map Reduce). Θεωρείστε στην υλοποίηση σας ότι ο ένας πίνακας είναι πάντα αρκετά μικρός ώστε να μπορεί να γίνει broadcast ολόκληρος, (αγνοείστε δηλαδή το μέρος του αλγορίθμου που σπάει τον πίνακα σε μικρότερα κομμάτια κάνοντας τα διαδοχικά broadcast).

Ζητούμενο 2 (1 Μονάδα): Υλοποιείστε το repartition join στο RDD API (Map Reduce)

Ζητούμενο 3 (0,5 Μονάδες): Απομονώστε 100 γραμμές του πίνακα movie genres σε ένα άλλο CSV. Συγκρίνεται τους χρόνους εκτέλεσης των δύο υλοποιήσεων σας για την συνένωση των 100 γραμμών με τον πίνακα ratings και συγκρίνετε τα αποτελεσμάτων. Τι παρατηρείται? Γιατί?

Ζητούμενο 4 (0,5 Μονάδες): Το SparkSQL έχει υλοποιημένα και τα δύο είδη ερωτημάτων συνένωσης στο DataFrame API. Συγκεκριμένα, με βάση τη δομή των δεδομένων και των υπολογισμών που θέλουμε καθώς και τις ρυθμίσεις του χρήστη, πραγματοποιεί από μόνο του κάποιες βελτιστοποιήσεις στην εκτέλεση του ερωτήματος χρησιμοποιώντας έναν βελτιστοποιητή ερωτημάτων (query optimizer), κάτι που όλες οι βάσεις δεδομένων έχουν. Μια τέτοια βελτιστοποίηση είναι ότι επιλέγει αυτόματα την υλοποίηση που θα χρησιμοποιήσει για ένα ερώτημα join λαμβάνοντας υπόψη το μέγεθος των δεδομένων και πολλές φορές αλλάζει και την σειρά ορισμένων τελεστών προσπαθώντας να μειώσει τον συνολικό χρόνο εκτέλεσης του ερωτήματος. Αν ο ένας πίνακας είναι αρκετά μικρός (με βάση ένα όριο που ρυθμίζει ο χρήστης) θα χρησιμοποιήσει το broadcast join, αλλιώς θα κάνει ένα repartition join. Περισσότερες πληροφορίες για τις ρυθμίσεις βελτιστοποίησης του SparkSQL υπάρχουν εδώ.

Χρησιμοποιώντας το script της επόμενης σελίδας, συμπληρώστε τις <> ώστε να μπορείτε να απενεργοποιήσετε την επιλογή του join από το βελτιστοποιητή. Εκτελέστε το query με και χωρίς βελτιστοποιητή και παρουσιάστε τα αποτελέσματα με την μορφή ενός ραβδογράμματος και το πλάνο εκτέλεσης που παράγει ο βελτιστοποιητής στην κάθε περίπτωση. Τι παρατηρείτε? Εξηγείστε.

```
from pyspark.sql import SparkSession
import sys, time

disabled = sys.argv[1]

spark = SparkSession.builder.appName('query1-sql').getOrCreate()

if disabled == "Y":
        spark.conf.set(<ονομα ιδιότητας>, <τιμή ιδιότητας>)
elif disabled == 'N':
        pass
else:
        raise Exception ("This setting is not available.")

df = spark.read.format("parquet")

df1 = df.load(<Εδώ path για τον πίνακα ratings στο hdfs>)
df2 = df.load(<Εδώ path για τον πίνακα movie_genres στο hdfs>)

df1.registerTempTable("ratings")
df2.registerTempTable("movie_genres")

sqlString = \
"SELECT * " + \
"FROM " + \
"       (SELECT * FROM movie_genres LIMIT 100) as g, " + \
"       ratings as r " + \
"WHERE " + \
"       r._c1 = g._c0"

t1 = time.time()
spark.sql(sqlString).collect()
t2 = time.time()

spark.sql(sqlString).explain()

print("Time with choosing join type %s is %.4f sec."%("enabled" if
disabled == 'N' else "disabled", t2-t1))

'''
Εκτέλεση script ως
-> spark-submit <name>.py Y για απενεργοποίηση του βελτιστοποιητή
-> spark-submit <name>.py N για ενεργοποίηση του βελτιστοποιητή
'''
```

Σχόλια αναφορικά με την εργασία

1. Η εργασία να εκπονηθεί σε ομάδες το πολύ των **3 ατόμων.**
2. Ως ημερομηνία παράδοσης της εργασίας ορίζεται το χρονικό περιθώριο των 3 εβδομάδων μετά το πέρας της κανονικής εξεταστικής (όχι επί πτυχίο). Η εργασία θα υποβληθεί στο helios στην σελίδα του μαθήματος σε link που θα ανοίξει αργότερα.
3. Η εργασία αποτελεί το **25%** του συνολικού βαθμού του μαθήματος. Για να υπολογιστεί ο βαθμός της εργασίας, η κάθε ομάδα θα πρέπει να περάσει επιτυχώς την **υποχρεωτική** προφορική εξέταση στο αντικείμενο της εργασίας. Η εξέταση θα γίνει μετά την παράδοση της εργασίας και θα αναρτηθεί σχετικό πρόγραμμα αφού ολοκληρωθεί η υποβολή των εργασιών. Σημειώνεται επίσης ότι η παράδοση και η εξέταση της εργασίας είναι υποχρεωτικά ώστε να προκύψει προβιβάσιμος βαθμός στο μάθημα συνολικά.
4. Απορίες για την εργασία θα γίνονται στο forum που δημιουργήθηκε στην σελίδα του μαθήματος στο helios. Για να μπορούν να βλέπουν και οι συμφοιτητές σας τυχόν απορίες με τις απαντήσεις τους, μη στέλνετε τις απορίες σας στα email των διδασκόντων αλλά να τις υποβάλλετε όπως αναφέρεται.
5. Ως παραδοτέο να υποβληθεί ένα **zip** αρχείο με όνομα τους ΑΜ των μελών της ομάδας χωρισμένα με κάτω παύλα (ή το ΑΜ του φοιτητή σε περίπτωση μονομελούς ομάδας), π.χ. 03100000_03100001_03100002.zip, 03100000_03100001.zip ή 03100000.zip (ανάλογα με το πλήθος των ατόμων της ομάδας). Το συμπιεσμένο αρχείο θα περιέχει τα ακόλουθα
   I. η public_ip του okeanos σε ένα αρχείο με όνομα **ip** (στόχος είναι να ελεγχθεί ότι υπάρχουν τα αρχεία της εργασίας στο hdfs). Στο hdfs υπενθυμίζεται ότι θα πρέπει να υπάρχουν δύο φακέλοι, ένας με όνομα **files** και ένας με όνομα **outputs** που ο πρώτος θα περιέχει τα αρχεία που σας δόθηκαν και ο δεύτερος τα αποτελέσματα των ερωτημάτων που υλοποιήσατε.
   II. μία αναφορά (**αυστηρά με όσα ζητούνται στην εκφώνηση**) σε **pdf** η οποία θα περιέχει αποκλειστικά της απαντήσεις στις ερωτήσεις που παρατίθενται, τα σχετικά διαγράμματα και ψευδοκώδικα σε Map Reduce που να περιγράφει τις υλοποιήσεις σας στο Part 1 για το κάθε ερώτημα με το RDD API.
   III. ένα φάκελο output με τα αποτελέσματα των ερωτημάτων τόσο μέσω του Map Reduce όσο και μέσω του SparkSQL.
   IV. ένα φάκελο code που θα περιέχει όλους τους κώδικες που έχετε υλοποιήσει, όπως και τον δοσμένο κώδικα για το μέρος 2 συμπληρωμένο με τις κατάλληλες τιμές.