ΠΑΝΕΠΙΣΤΉΜΙΟ ΙΩΑΝΝΊΝΩΝ

ΤΜΉΜΑ ΜΗΧΑΝΙΚΏΝ ΗΛΕΚΤΡΟΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διαχείριση Σύνθετων Δεδομένων

Εργασία 1 – Αποτίμηση ερωτημάτων

ΣΙΑΚΑΒΑΡΑ ΘΕΜΙΣΤΟΚΛΕΙΑ, 4786

ΠΑΡΑΔΟΣΗ: ΠΑΡΑΣΚΕΥΗ 29 ΜΑΡΤΙΟΥ 2024

Για την υλοποίηση της πρώτης εργασίας, χρησιμοποιήθηκε η γλώσσα προγραμματισμού python, και πιο συγκεκριμένα η έκδοση 3.10.6.

Σε όλα τα προγράμματα έχει γίνει import csv για να διαβάσουμε και να γράψουμε ένα csv αρχείο, και import sys, για να διαβάσουμε ορίσματα από τη γραμμή διαταγών.

1ο Μέρος (group-by with aggregation):

Το ζητούμενο πρόγραμμα για το πρώτο μέρος ονομάζεται **aggregation.py** και παίρνει ως ορίσματα ένα αρχείο csv (R.csv ή S.csv), ένα attribute ομαδοποίησης, ένα attribute όπου εφαρμόζεται συνάθροιση και την συνάρτηση συνάθροισης. Για παράδειγμα:

```
python aggregation.py R.csv 1 2 max
```

Στην main συνάρτηση ελέγχεται πρώτα αν έχει δοθεί ο σωστός αριθμός ορισμάτων. Αν ναι, τότε ανοίγει το αρχείο csv και μεταφέρει τα δεδομένα του σε ένα πίνακα data. Έπειτα, καλώντας την συνάρτηση groupby_with_aggregation(), ομαδοποιεί τα δεδομένα του πίνακα αυτού και βρίσκει το αποτέλεσμα της συνάρτησης συνάθροισης. Τέλος, ανοίγει το αρχείο “O1.csv”, στο οποίο γράφονται τα αποτελέσματα.

```
if __name__ == "__main__":
    if len(sys.argv)!= 5:
        print("Not enough arguments")
        sys.exit(1)

    data = []
    with open(sys.argv[1], mode = 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)

    groupAttribute = int(sys.argv[2])
    aggregationAttribute = int(sys.argv[3])
    aggregationFunction = sys.argv[4]


    finalData = groupby_with_aggregation(data, groupAttribute, aggregationAttribute, aggregationFunction)

    with open('O1.csv', 'w', newline='') as file:
        writer = csv.writer(file)
        for row in finalData:
            writer.writerow(row)
```

Η συνάρτηση groupby_with_aggregation() έχει ως ορίσματα τα ίδια με αυτά του προγράμματος, με την διαφορά ότι αντί για το αρχείο παίρνει ως πρώτο όρισμα τα δεδομένα σε μορφή πίνακα.

Στην αρχή, μέσω ενός if-statement, βρίσκουμε την συνάρτηση συνάθροισης που δόθηκε από τον χρήστη, την αποθηκεύουμε σε μια μεταβλητή ως συνάρτηση (που υπάρχει ήδη ορισμένη στην γλώσσα python), και ορίζουμε την αρχική της τιμή. Αν η συνάρτηση, δεν είναι κάποια από τις επιτρεπτές (max, min, sum), τότε το πρόγραμμα σταματάει λόγω error.

```
C:\Users\Themis Siak\Desktop\Assignments\1>python aggregation.py R.csv 1 2 avg
Traceback (most recent call last):
  File "C:\Users\Themis Siak\Desktop\Assignments\1\aggregation.py", line 72, in <module>
    finalData = groupby_with_aggregation(data, groupAttribute, aggregationAttribute, aggregationFunction)
  File "C:\Users\Themis Siak\Desktop\Assignments\1\aggregation.py", line 16, in groupby_with_aggregation
    raise ValueError("Inappropriate aggregation argument value")
ValueError: Inappropriate aggregation argument value
```

Έπειτα, ακολουθούν άλλες τρείς συναρτήσεις, οι οποίες έχουν ως στόχο την υλοποίηση του αλγορίθμου MergeSort.

- extractAttribute(row, groupAttribute, aggAttribute): Βοηθητική συνάρτηση, η οποία παίρνει ως ορίσματα την γραμμή του πίνακα δεδομένων, το attribute ομαδοποίησης και το attribute συνάρτησης συνάθροισης. Επιστρέφει τις δύο τιμές των δύο παραπάνω attributes από τα δεδομένα, δηλαδή αγνοεί το attribute που δεν χρησιμοποιείται.
- merge(left, right): Τα ορίσματα είναι δύο πίνακες (left, right) που δημιουργούνται στην επόμενη συνάρτηση (mergeSort). Η συνάρτηση αυτή γεμίζει και επιστρέφει τον πίνακα result, ο όποιος στο τέλος περιέχει ταξινομημένα τα δεδομένα. Στο while-loop συγκρίνει τις τιμές του groupAttribute των left και right. Αν η τιμή του left είναι μικρότερη αυτής του right, τότε στον result προστίθεται ο πίνακας left και συνεχίζει στην επόμενη του σειρά, αλλιώς γίνεται το ίδιο αλλά με τον πίνακα right. Αν όμως οι δύο τιμές των πινάκων είναι ίδιες, τότε επιτόπου υλοποιείται η συνάρτηση συνάθροισης που έχουμε αποθηκεύσει ήδη από το if- statement. Τέλος, προσθέτουμε ό,τι περίσσεψε από τους δύο πίνακες στον result.
- mergeSort(array): Αναδρομική συνάρτηση που παίρνει ως όρισμα τον πίνακα των δεδομένων και τελικά εκτελεί τον αλγόριθμο mergeSort. Αν ο πίνακας αποτελείται μόνο από ένα στοιχείο τότε επιστρέφει αυτό, αλλιώς τον χωρίζει στην μέση, δημιουργώντας δύο άλλους πίνακες (left, right). Τέλος, επιστρέφει την συγχώνευση των δύο αυτών πινάκων. Η αναδρομή γίνεται κατά τον ορισμό των left και right, διότι η ταξινόμηση ξεκινάει από τα φύλλα του δέντρου που δημιουργείται (όταν είναι μόνο ένα στοιχείο). Έτσι, υλοποιείται η

δυαδική αναζήτηση, καθώς βρίσκουμε το μισό μέγεθος του πίνακα και ορίζουμε δύο άλλους πίνακες, οι οποίοι είναι το αριστερά μισό του ορίσματος και το δεξί μισό του ορίσματος, αντίστοιχα.

Αφού έχουν οριστεί οι παραπάνω συναρτήσεις, πρώτα αφαιρούμε το attribute που δεν χρησιμοποιείται από τα δεδομένα μας και μετατρέπουμε τα υπόλοιπα από String σε int, για να μπορούν να συγκριθούν οι τιμές ως αριθμοί. Έτσι, πλέον τρέχουμε κανονικά την συνάρτηση mergeSort και επιστρέφουμε το αποτέλεσμά της.

Ο κώδικας είναι ο εξής:

```
def groupby_with_aggregation(data, groupAttribute, aggregationAttribute, aggregationFunction):
    if aggregationFunction == 'sum':
        aggFunction = sum
        initialValue = 0
    elif aggregationFunction == 'min':
        aggFunction = min
        initialValue = float('inf')
    elif aggregationFunction == 'max':
        aggFunction = max
        initialValue = float('-inf')
    else:
        raise ValueError("Inappropriate aggregation argument value")

    def extractAttribute(row, groupAttritube, aggAttribute):
        groupValue = row[groupAttribute]
        aggValue = row[aggAttribute]
        return groupValue, aggValue

    def merge(left, right):
        result = []
        i=j=0
        while i<len(left) and j<len(right):
            if left[i][0] < right[j][0]:
                result.append(left[i])
                i = i+1
            elif left[i][0] > right[j][0]:
                result.append(right[j])
                j = j+1
            else:
                aggValues = [left[i][1], right[j][1]]
                aggResult = aggFunction(aggValues)
                result.append((left[i][0], aggResult))
                i = i+1
                j = j+1
        result.extend(left[i:])
        result.extend(right[j:])
        return result
```

```
    def mergeSort(array):
        if len(array) <= 1:
            return array
        mid = len(array) // 2
        left = mergeSort(array[:mid])
        right = mergeSort(array[mid:])
        return merge(left, right)

    elimination = [extractAttribute(row, groupAttribute, aggregationAttribute) for row in data]
    elim = [(int(x), int(y)) for x, y in elimination]
    sortData = mergeSort(elim)
    return sortData
```

Στο παράδειγμα όπου τα ορίσματα είναι *R.csv 1 2 max,* στο αρχείο O1.csv έχουμε τα εξής 20 πρώτα αποτελέσματα:

A	B
1	10
2	10
3	10
4	9
5	10
6	6
7	8
8	10
9	10
10	9

Άλλο ένα παράδειγμα, αυτή την φορά με το S.csv αρχείο, το οποίο είναι ήδη ταξινομημένο με το attribute 1 (δεύτερη στήλη του):

S.csv 1 2 min

A	B
45	1
47	5
50	1
51	2
59	1
62	1
78	1
87	2
88	1
98	2

S.csv 1 2 sum

A	B
45	63
47	44
50	37
51	57
59	55
62	70
78	60
87	50
88	37
98	39

Μπορούμε εύκολα να συμπεράνουμε ότι τα αποτελέσματα είναι σωστά. Για παράδειγμα min(45) = min(3,9,7,6,4,2,1,4,7,10,10) και sum(45) = 3+9+7+6+4+2+1+4+7+10+10
Οι αρχικές τιμές του αρχείου S.csv:

	A	B	C
1	1	45	3
2	2	45	9
3	3	45	7
4	4	45	6
5	5	45	4
6	6	45	2
7	7	45	1
8	8	45	4
9	9	45	7
10	10	45	10
11	11	45	10
12	12	47	5
13	13	47	6
14	14	47	9
15	15	47	5
16	16	47	9
17	17	47	10
18	18	50	2
19	19	50	1
20	20	50	6
21	21	50	5
22	22	50	5
23	23	50	8

	A	B	C
24	24	50	4
25	25	50	1
26	26	50	5
27	27	51	8
28	28	51	4
29	29	51	10
30	30	51	3
31	31	51	4
32	32	51	8
33	33	51	3
34	34	51	2
35	35	51	9
36	36	51	6
37	37	59	10
38	38	59	5
39	39	59	4
40	40	59	10
41	41	59	4
42	42	59	1
43	43	59	4
44	44	59	3
45	45	59	8
46	46	59	6

	A	B	C
47	47	62	4
48	48	62	6
49	49	62	7
50	50	62	10
51	51	62	9
52	52	62	3
53	53	62	1
54	54	62	10
55	55	62	10
56	56	62	10
57	57	78	4
58	58	78	7
59	59	78	1
60	60	78	10
61	61	78	8
62	62	78	6
63	63	78	10
64	64	78	7
65	65	78	7
66	66	87	9
67	67	87	5
68	68	87	3
69	69	87	2

2ο Μέρος (merge join):

Στο δεύτερο μέρος της πρώτης εργασίας ζητείται η υλοποίηση της φυσικής συνένωσης (natural join) δύο αρχείων, ως προς ένα πεδίο. Το πρόγραμμα ονομάζεται **mergejoin.py**

Το πρόγραμμα δεν παίρνει κάποιο όρισμα. Κατευθείαν στην main μέθοδο ανοίγει ταυτόχρονα τα δύο αρχεία (R.csv και S.csv) για να τα διαβάσει, καθώς και ένα τρίτο αρχείο που δημιουργεί (O2.csv), για να γράψει το αποτέλεσμα. Αφού ανοίξει και το τρίτο αρχείο, καλείται η συνάρτηση mergeJoin() που υλοποιεί το ζητούμενο.

Ο κώδικας είναι ο εξής:

```
if __name__ == "__main__":
    with open('R.csv', mode='r') as file1:
        with open('S.csv', mode = 'r') as file2:
                reader1 = csv.reader(file1)
                reader2 = csv.reader(file2)

                with open('O2.csv', mode='w', newline='') as output:
                    mergeJoin(reader1, reader2, output)
```

Πέρα από την main, το αρχείο αποτελείται και από τη συνάρτηση mergeJoin(), η οποία παίρνει ως ορίσματα τους δύο readers των αρχείων και το τελικό αρχείο των αποτελεσμάτων. Αρχικά, δημιουργεί τον writer για το “O2.csv”, και διαβάζει την πρώτη γραμμή των R.csv και S.csv. Το while-loop τρέχει μέχρι ένα από τα δύο αρχεία που διαβάζουμε να μην έχει άλλα δεδομένα. Αν κάποια γραμμή είναι κενή, την παραλείπουμε, αλλιώς ελέγχουμε αν το κοινό attribute των αρχείων (R.A και S.A) έχουν ίση, μικρότερη ή μεγαλύτερη τιμή. Σημαντικό είναι να μετατρέπουμε κάθε φορά τον τύπο από String σε int, για να ελέγχονται οι τιμές. Αν είναι ίσες τότε γράφουμε στο αρχείο την σειρά αυτή με όλα τα attributes και των δύο αρχείων και διαβάζουμε την επόμενη γραμμή του δεύτερου αρχείου. Αν η τιμή του R.csv είναι μικρότερη του S.csv διαβάζουμε την επόμενη γραμμή του πρώτου, ενώ στην αντίθετη περίπτωση διαβάζουμε την επόμενη γραμμή του δεύτερου. Η ανάγνωση των αρχείων, η φυσική συνένωση, καθώς και η εγγραφή των αποτελεσμάτων γίνεται παράλληλα, γραμμή προς γραμμή.

Ο κώδικας της συνάρτησης είναι ο εξής:

```
def mergeJoin(reader1, reader2, output):
    writer = csv.writer(output)
    R = next(reader1, None)
    S = next(reader2, None)

    while R is not None and S is not None:
        if not R or not S:
            continue

        if int(R[0]) == int(S[1]):
            R.append(S[0])
            R.append(S[2])
            writer.writerow(R)

            R.pop()
            R.pop()
            S = next(reader2, None)

        elif int(R[0]) < int(S[1]):
            R = next(reader1, None)
        else:
            S = next(reader2, None)
```

Παρατηρούμε εύκολα ότι τα αποτελέσματα είναι σωστά:

Αρχείο R.csv

	A	B	C
45	45	41	7
46	46	90	6
47	47	43	7
48	48	90	5
49	49	88	6
50	50	37	5
51	51	92	8
52	52	29	3
53	53	63	10
54	54	35	7
55	55	8	10
56	56	100	9
57	57	18	4
58	58	47	10
59	59	95	6
60	60	33	9
61	61	75	10
62	62	11	10
63	63	73	5
64	64	92	6
65	65	67	1
66	66	46	9
67	67	78	8

Αρχείο S.csv

	A	B	C
1	1	45	3
2	2	45	9
3	3	45	7
4	4	45	6
5	5	45	4
6	6	45	2
7	7	45	1
8	8	45	4
9	9	45	7
10	10	45	10
11	11	45	10
12	12	47	5
13	13	47	6
14	14	47	9
15	15	47	5
16	16	47	9
17	17	47	10
18	18	50	2
19	19	50	1
20	20	50	6
21	21	50	5
22	22	50	5
23	23	50	8

Για το αρχείο R.csv κοιτάμε για R.A=45 και κάτω, καθώς πιο πριν δεν υπάρχει κοινή τιμή με το S.A.

Τα 20 πρώτα αποτελέσματα:

	A	B	C	D	E
1	45	41	7	1	3
2	45	41	7	2	9
3	45	41	7	3	7
4	45	41	7	4	6
5	45	41	7	5	4
6	45	41	7	6	2
7	45	41	7	7	1
8	45	41	7	8	4
9	45	41	7	9	7
10	45	41	7	10	10
11	45	41	7	11	10
12	47	43	7	12	5
13	47	43	7	13	6
14	47	43	7	14	9
15	47	43	7	15	5
16	47	43	7	16	9
17	47	43	7	17	10
18	50	37	5	18	2
19	50	37	5	19	1
20	50	37	5	20	6

3ο Μέρος (composite query):

Στο τρίτο και τελευταίο μέρος της εργασίας, ζητούμενο είναι μια η αποτίμηση μίας πιο σύνθετης ερώτησης SQL: $_{sum(E)}\gamma_A$ ($\sigma_{C=7}$(R) ⋈ S) ή αλλιώς:

```
SELECT S.A, SUM(S.E)
FROM R, S
WHERE R.A = S.A AND R.C = 7
GROUP BY S.A
```

Επεξηγηματικά, όπως και στο 2ο μέρος, πρέπει το κοινό κλειδί να έχει την ιδία τιμή, και συμπληρωματικά το attribute C του R.csv αρχείου να έχει την τιμή 7. Έτσι, βρίσκουμε το αποτέλεσμα της πρόσθεσης των S.E που συμφωνούν με την παραπάνω συνθήκη, και το γράφουμε μαζί με το S.A. Το πρόγραμμα που δημιουργήθηκε ονομάζεται **compositequery.py** και δεν παίρνει κάποιο όρισμα.

Η main του συνάρτηση είναι ίδια με αυτή του δεύτερου μέρους, μόνο που τώρα καλείται άλλη συνάρτηση για την παραγωγή αποτελεσμάτων. Αυτή είναι η compositeQuery().

Ο κώδικας της main:

```
if __name__ == "__main__":
    with open('R.csv', mode='r') as file1:
        with open('S.csv', mode = 'r') as file2:
                reader1 = csv.reader(file1)
                reader2 = csv.reader(file2)

                with open('O3.csv', mode='w', newline='') as output:
                    compositeQuery(reader1, reader2, output)
```

Η άλλη συνάρτηση του αρχείου (compositeQuery) παίρνει ως ορίσματα τους δύο readers των αρχείων και το τελικό αρχείο των αποτελεσμάτων. Αρχικά, δημιουργεί τον writer για το “O3.csv”, και διαβάζει την πρώτη γραμμή των R.csv και S.csv. Γενικά λειτουργεί παρόμοια με την mergeJoin() της προηγούμενης άσκησης, με κάποιες συμπληρωματικές μεταβλητές και κάποιους ελέγχους.

Αρχικοποιούμε την μεταβλητή sumE που θα κρατάει την πρόσθεση των τιμών που θέλουμε. Η μεταβλητή temp είναι μία προσωρινή μεταβλητή που θα ανανεώνεται κάθε φορά πριν γράψουμε στο αρχείο με την γραμμή που θα γραφτεί, και θα αδειάζει αφού γράψουμε. Στην μεταβλητή currentR κρατάμε την γραμμή του R που πρόκειται να γραφτεί στο τελικό αρχείο, αφού τελειώσει πρώτα ο έλεγχος ότι δεν υπάρχει άλλη τιμή για το τρέχων attribute. Ο έλεγχος αυτός γίνεται στο while-loop, όπου συγκρίνουμε τις

τιμές του A των δύο αρχείων. Η εκτέλεση είναι ίδια με της προηγούμενης άσκησης, μόνο που εδώ ελέγχουμε και το R.C attribute και γράφουμε στο αρχείο μόνο αφού τελειώσει ο υπολογισμός της πρόσθεσης της τρέχουσας τιμής του A.
Ο κώδικας της συνάρτησης είναι ο εξής:

```
def compositeQuery(reader1, reader2, output):
    writer = csv.writer(output)
    R = next(reader1, None)
    S = next(reader2, None)

    currentR = []
    temp = []
    sumE = 0

    while R is not None and S is not None:
        if not R or not S:
            continue
        if int(R[0]) == int(S[1]):
            if R[2] == '7':
                sumE = sumE + int(S[2])
                currentR = R[0]
            else:
                if sumE!=0:
                    temp.append(currentR)
                    temp.append(str(sumE))
                    writer.writerow(temp)
                    currentR = temp = []
                    sumE = 0
            S = next(reader2, None)

        elif int(R[0]) < int(S[1]):
            if sumE!=0:
                temp.append(currentR)
                temp.append(str(sumE))
                writer.writerow(temp)
                currentR = temp = []
                sumE = 0
            R = next(reader1, None)
        else:
            if sumE!=0:
                temp.append(currentR)
                temp.append(str(sumE))
                writer.writerow(temp)
                currentR = temp = []
                sumE = 0
            S = next(reader2, None)
```

Τελικά αποτελέσματα “O3.csv”:

	A	B
1	45	63
2	47	44
3	78	60
4	187	70
5	501	55
6	505	73
7	538	48
8	577	68
9	616	32
10	656	55

Από το αρχείο O2.csv που δημιουργήθηκε στο μέρος 2, ελέγχουμε ότι τα αποτελέσματα είναι σωστά. Οι τιμές του R.A = S.A για τις οποίες ισχύει και R.C = 7 είναι μόνο οι 10 παραπάνω.

Το αρχείο O2.csv σε αυτές τις τιμές:

	A	B	C	D	E
1	45	41	7	1	3
2	45	41	7	2	9
3	45	41	7	3	7
4	45	41	7	4	6
5	45	41	7	5	4
6	45	41	7	6	2
7	45	41	7	7	1
8	45	41	7	8	4
9	45	41	7	9	7
10	45	41	7	10	10
11	45	41	7	11	10
12	47	43	7	12	5
13	47	43	7	13	6
14	47	43	7	14	9
15	47	43	7	15	5
16	47	43	7	16	9
17	47	43	7	17	10

499	505	45	7	499	2
500	505	45	7	500	7
501	505	45	7	501	7
502	505	45	7	502	2
503	505	45	7	503	6
504	505	45	7	504	2
505	505	45	7	505	2
506	505	45	7	506	9
507	505	45	7	507	5
508	505	45	7	508	2
509	505	45	7	509	1
510	505	45	7	510	7
511	505	45	7	511	6
512	505	45	7	512	8
513	505	45	7	513	7

57	78	17	7	57	4
58	78	17	7	58	7
59	78	17	7	59	1
60	78	17	7	60	10
61	78	17	7	61	8
62	78	17	7	62	6
63	78	17	7	63	10
64	78	17	7	64	7
65	78	17	7	65	7

529	538	3	7	529	7
530	538	3	7	530	5
531	538	3	7	531	9
532	538	3	7	532	7
533	538	3	7	533	6
534	538	3	7	534	10

212	187	5	7	212	6
213	187	5	7	213	5
214	187	5	7	214	2
215	187	5	7	215	9
216	187	5	7	216	10
217	187	5	7	217	7
218	187	5	7	218	4
219	187	5	7	219	5
220	187	5	7	220	5
221	187	5	7	221	9
222	187	5	7	222	8

555	577	15	7	555	3
556	577	15	7	556	9
557	577	15	7	557	5
558	577	15	7	558	8
559	577	15	7	559	6
560	577	15	7	560	7
561	577	15	7	561	6
562	577	15	7	562	5
563	577	15	7	563	7
564	577	15	7	564	1
565	577	15	7	565	1
566	577	15	7	566	10

489	501	42	7	489	9
490	501	42	7	490	6
491	501	42	7	491	6
492	501	42	7	492	8
493	501	42	7	493	8
494	501	42	7	494	6
495	501	42	7	495	3
496	501	42	7	496	2
497	501	42	7	497	5
498	501	42	7	498	2

594	616	97	7	594	4
595	616	97	7	595	8
596	616	97	7	596	9
597	616	97	7	597	5
598	616	97	7	598	2
599	616	97	7	599	4

669	656	65	7	669	10
670	656	65	7	670	7
671	656	65	7	671	7
672	656	65	7	672	9
673	656	65	7	673	6

674	656	65	7	674	2
675	656	65	7	675	3
676	656	65	7	676	8
677	656	65	7	677	2
678	656	65	7	678	1

Πηγές:

- Διαφάνειες του μαθήματος (background, evaluation)
- https://en.wikipedia.org/wiki/Merge_sort
- https://www.geeksforgeeks.org/merge-sort/
- https://en.wikipedia.org/wiki/Sort-merge_join
- https://sqlserverfast.com/epr/merge-join/