Αναφορά 3ης εργαστηριακής άσκησης

ΜΥΕ041 – Διαχείριση Σύνθετων Δεδομένων
Ονοματεπώνυμο: Γεωργαλλή Αλέξανδρος
Αριθμός Μητρώου: 5135

Μέρος 1:

Στο 1ο μέρος, υλοποιώ τον αλγόριθμο top-k join. Τα δεδομένα που χρησιμοποιώ είναι από το https://kdd.ics.uci.edu/databases/census-income/census-income.html και αποτελούν δημογραφικά δεδομένα από τις ΗΠΑ. Τα δεδομένα έχουν χωριστεί σε δύο αρχεία τα οποία είναι ταξινομημένα με βάση το πεδίο instance weight σε φθίνουσα σειρά: το αρχείο males_sorted έχει όλες τις εγγραφές που αντιστοιχούν σε άντρες (πεδίο sex = Male) και το αρχείο females_sorted έχει όλες τις εγγραφές που αντιστοιχούν σε γυναίνες (πεδίο sex = Female).

Ο αλγόριθμος αρχικά διαβάζει την επόμενη **έγκυρη γραμμή** από το males_sorted ή το females_sorted, εναλλάξ.
Έγκυρη γραμμή: περιέχει πληροφορίες για ένα άτομο που δεν είναι παντρεμένο και είναι τουλάχιστον 18 ετών.

Αφού διαβάσει την επόμενη έγκυρη γραμμή, ενημερώνει τα αντίστοιχα hash_tables που δημιούργησα για κάθε αρχείο. Αρχικοποιώ τις μεταβλητές p1_max = p1_cur = p2_max = p2_cur = 0 και την count = 0, η οποία είναι βοηθητική μεταβλητή έτσι ώστε να τρέχει εναλλάξ ο αλγόριθμος για males_sorted & females_sorted. Το threshold, ενημερώνεται μετά από κάθε αλλαγή στο hash_table του κάθε αρχείου.

Έγκυρη γραμμή:

```
def gen_next_female():
    with open('females_sorted', 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            age = int(parts[1])
            marital_status = parts[8]
            if age < 18 or marital_status.startswith(" Married"):
                continue
            yield parts


def gen_next_male():
    with open('males_sorted', 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            age = int(parts[1])
            marital_status = parts[8]
            if age < 18 or marital_status.startswith(" Married"):
                continue
            yield parts
```

Αλγόριθμος top_k_join:

```
def top_k_join():
    males_gen = gen_next_male()
    females_gen = gen_next_female()

    males_dict = {}
    females_dict = {}
    max_heap = []

    p1_max = p1_cur = p2_max = p2_cur = 0
    counter = 0

    while True:
        if counter % 2 == 0:
            try:
                male = next(males_gen)
            except StopIteration:
                break
            age = int(male[1])
            instance_weight = float(male[25])
            p1_cur = instance_weight
            if p1_cur > p1_max:
                p1_max = p1_cur
            if age not in males_dict:
                males_dict[age] = []
            males_dict[age].append(male)
            if age in females_dict:
                for female in females_dict[age]:
                    score = float(male[25]) + float(female[25])
                    heapq.heappush(max_heap, (-score, male, female))
        else:
            try:
                female = next(females_gen)
            except StopIteration:
                break
            age = int(female[1])
            instance_weight = float(female[25])
            p2_cur = instance_weight
            if p2_cur > p2_max:
                p2_max = p2_cur
            if age not in females_dict:
                females_dict[age] = []
            females_dict[age].append(female)
            if age in males_dict:
                for male in males_dict[age]:
                    score = float(male[25]) + float(female[25])
                    heapq.heappush(max_heap, (-score, male, female))

        T = max((p1_max + p2_cur), (p2_max + p1_cur))

        while len(max_heap) > 0 and -max_heap[0][0] >= T:
            yield heapq.heappop(max_heap)

        counter += 1
```

Usage of meros1.py:
python3 meros1.py <K>

Μέρος 2:

Ο αλγόριθμος στο μέρος 2, είναι μια παραλλαγή του top-k join. Ο αλγόριθμος διαβάζει εξολοκλήρου το αρχείο males_sorted και βάζει τις πλειάδες (μόνο τις έγκυρες) σε ένα hash table (με κλειδί το age). Μετά, διαβάζει μία προς μία τις πλειάδες από το females_sorted και για καθεμιά από αυτές βρίσκει τις πλειάδες με τις οποίες κάνει join χρησιμοποιώντας το hash table. Από τα αποτελέσματα του join που προκύπτουν κρατάμε σε ένα min-heap τα μέχρι στιγμής κορυφαία K. Μόλις ολοκληρωθεί ο αλγόριθμος το heap θα πρέπει να έχει τα κορυφαία K ζευγάρια.

Διάβασμα έγκυρων γραμμών από το males_sorted:

```python
def gen_next_male():
    global males_dict

    with open('males_sorted', 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            age = int(parts[1])
            marital_status = parts[8]
            if age < 18 or marital_status.startswith(" Married"):
                continue
            if age not in males_dict:
                males_dict[age] = []
            males_dict[age].append(parts)
    return males_dict
```

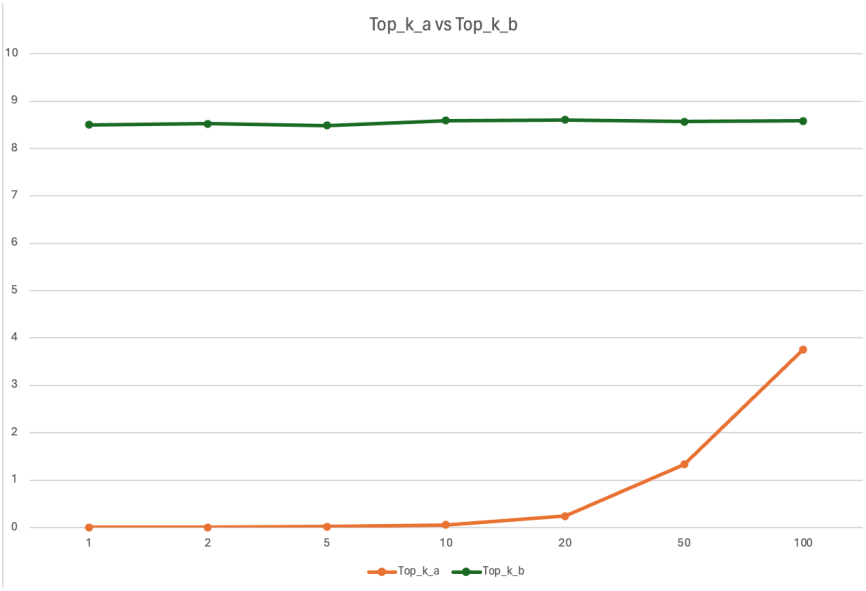
Αλγόριθμος top_k_join b:

```python
def top_k_join(k):
    global males_dict
    males_dict = gen_next_male()
    females_gen = gen_next_female(k)

    min_heap = []

    for female in females_gen:
        age = int(female[1])
        if age in males_dict:
            for male in males_dict[age]:
                instance_weight_sum = float(male[25]) + float(female[25])
                if len(min_heap) < k:
                    heapq.heappush(min_heap, (instance_weight_sum, male, female))
                else:
                    if instance_weight_sum > min_heap[0][0]:
                        heapq.heappushpop(min_heap, (instance_weight_sum, male, female))

    return sorted(min_heap, reverse=True)
```

Usage of meros2.py:
python3 meros2.py <K>

Μέρος 3:

Σε αυτό το μέρος, έτρεξα τον κάθε ένα από τους αλγορίθμους με τιμές K = 1,2,5,10,20,50,100. Τα αποτελέσματα φαίνονται στην παρακάτω φωτογραφία με τον κατακόρυφο άξονα να αντιστοιχεί στα δευτερόλεπτα (sec) και τον οριζόντιο να αντιστοιχεί στις τιμές του K.



Τα αποτελέσματα που πήρα από αυτή τη γραφική παράσταση είναι τα αναμενόμενα καθώς στον Αλγόριθμο A (top_k_a) χρησιμοποιώ 2 hash tables και τα αποτελέσματα παράγονται μέσω της generator function, ενώ στον Αλγόριθμο B (top_k_b), χρησιμοποιώ ένα hash table για να αποθηκεύσω τις έγκυρες γραμμές του males_sorted και μετά ψάχνω για ταιριάσματα για κάθε έγκυρη γραμμή που επιστρέφεται από την gen_next_female.

Τα πλεονεκτήματα του αλγορίθμου A έναντι του B είναι ότι για λίγες τιμές είναι πολύ πιο γρήγορος, αλλά όσο αυξάνονται οι τιμές του K τόσο θα αυξάνεται και ο χρόνος εκτέλεσης του. Ο Αλγόριθμος B από την άλλη, έχει σταθερό χρόνο για όλες τις τιμές του K.

Συμπερασματικά, αν γνωρίζουμε ότι οι τιμές του K είναι πολύ μεγάλες, τότε θα χρησιμοποιήσουμε τον αλγόριθμο B.

<u>Περισσότερα στατιστικά εκτέλεσης:</u>

K	Valid lines read by algorithm A	Valid lines read by algorithm B
1	88	57182
2	268	57182
5	874	57182
10	2290	57182
20	5312	57182
50	12060	57182
100	20278	57182