ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Μυλωνάκης Αλέξανδρος
ΑΜ: 3045

Εργασία 3η Διαχείριση Σύνθετων δεδομένων

Γλώσσα προγραμματισμού: Python

Προεργασία:

Αρχικά δημιούργησα τα αρχεία main.py, file_manager.py(το οποίο είναι υπεύθυνο για την διαχείριση του αρχείου rnd.txt

Επίσης για να αντιμετωπίσω ένα πρόβλημα που είχα με τις float μεταβλητές (μετά από κάποιες πράξεις με float είχα ένα error που προσέθετε ένα πολύ μικρό δεκαδικό ψηφίο στον αριθμό χρησιμοποίησα την inbuilt βιβλιοθήκη decimal)

Για να πάρω input χρησιμοποιώ τα arguments που παίρνω από το τερματικό με

```
arg1 = sys.argv[1]
arg2 = sys.argv[2]
arg3 = sys.argv[3]
arg4 = sys.argv[4]
```

Agr1 = rnd.txt

Arg2 = seq1.txt

Arg3 = seq2.txt

Arg4 = k

Για το rnd.txt:

```
...
class file_manager:
    def __init__(self,path):
        self.path = path

    def create_structure(self):
        R = []
        with open(self.path, 'r') as f:
            contents = f.read()
        lines = contents.splitlines()

        for i in range(len(lines)):
            split_line = lines[i].split()
            R.append(Decimal(split_line[1]))
        return R        You, 2 days ago • Initial
```

Δημιουργώ μια δομή R στην οποία αποθηκεύω τα στοιχεία του rnd.txt

Υλοποίηση:

Για την υλοποίηση του top-k query δημιούργησα μια κλάση

round-robin η οποία παίρνει ως είσοδο τα 2 αρχεία seq1.txt, seq2.txt και το k και κάνει σειριακές προσπελάσεις τους.

Αρχικά κάνω σειριακές επαναλήψεις για να δημιουργήσω την λίστα που θα μετατρέψω σε στοίβα για να διαβάσω τα πρώτα k αντικείμενα και μόλις τα εντοπίσω φτιάχνω την στοίβα ελαχίστου (min_heap)

```
def do_the_sequence(self):
    obj_counter = 0
    file_index = 1
    with open(self.path1, 'r') as file1, open(self.path2, 'r') as file2:
        while obj_counter<self.k:
            if file_index == 1:
                line = file1.readline()
                if not line:
                    line = file2.readline()
                    if not line:
                        break
                file_index = 2
            else:
                line = file2.readline()
                if not line:
                    line = file1.readline()
                file_index = 1
            index,val = self.split_line(line)
            in_full_score =self.check_dicts(index,val,file_index)
            if not in_full_score:
                obj_counter+=1
            self.sequential_accesses +=1
        self.create_min_heap()

        while self.check_loop():
            if file_index == 1:
                line = file1.readline()
                if not line:
                    line = file2.readline()
                    if not line:
                        break
                file_index = 2
            else:
                line = file2.readline()
                if not line:
                    line = file1.readline()
                    if not line:
                        break       You, 1 second ago • Uncommitted changes
                file_index = 1
            index,val = self.split_line(line)
            self.check_dicts(index,val,file_index)
            self.check_heap(index)
            self.sequential_accesses +=1
    return self.Wk
```

Η μέθοδος για την δημιουργία της στοίβας:

```
def create_min_heap(self):
    for key, value in self.lower_bound.items():
        self.Wk.append([value[0],key])
    if len(self.full_score)>0:
        for key , value in self.full_score:
            self.Wk.append([value,key])
```

Όσο αναφορά την διαχείριση των δυο αρχείων έχω δημιουργήσει δυο dictionaries

```
self.lower_bound = {}
self.full_score = {}
```

Τα οποία διαχειρίζομαι κατά την προσπέλαση των αρχείων με την μέθοδο check_dicts:

```
def check_dicts(self,index,val,file_index):
    if file_index == 1:
        self.val1 = val
    else:
        self.val2 = val
    if index in self.lower_bound:
        self.full_score[index]=self.lower_bound[index][0]+val
        del self.lower_bound[index]
        in_full_score = True
    else:
        lower_bound = val+self.R[index]
        self.lower_bound[index]= [lower_bound,file_index]
        in_full_score=False
    return in_full_score
```

Αυτή παίρνει ως ορίσματα το δείκτη για το στοιχείο το score(value) του και ένα δείκτη για το αρχείο για το οποίο βρέθηκε. Αν ο δείκτης δεν είναι ήδη κλειδί του πρώτου λεξικού(lower_bound) δηλαδή του λεξικού που έχει πληροφορία για το score από το round.txt και ενός από τα αρχεία seq1.txt,seq2.txt καθώς και ποιο αρχείο είναι αυτό. Το δεύτερο λεξικό έχει ως κλειδί το δείκτη του στοιχείου και ως τιμή το πλήρες score του. Αυτή η μέθοδος επιστρέφει μια Boolean τιμή (το οποίο μας βοηθάει στην αρχικοποίηση της στοίβας).

Στην συνέχεια έφτιαξα μια συνάρτηση check loop η οποία είναι υπεύθυνη για την συνέχιση της σειριακής επανάληψης βάση του threshold_T και των ήδη υπάρχων values

```
def check_loop(self):
    threshold_T = self.val1 + self.val2 + 5
    if threshold_T > self.Wk[0][0]:
        return True
    elif len(self.lower_bound)>0:
        for key,value in self.lower_bound.items():
            if value[1] == 1:
                if self.Wk[0][0] < value[0]+self.val2:
                    return True
            if value[1] == 2:
                if self.Wk[0][0] < value[0]+self.val1:
                    return True
    return False
```

Και χρησιμοποιώντας την check_heap ενημερώνω την στοίβα μου

```
def check_heap(self,index):
    heapq.heapify(self.Wk)
    if index in self.full_score:
        value = self.full_score[index]
    else:
        value = self.lower_bound[index][0]
    if self.Wk[0][0] < value:
        heapq.heappop(self.Wk)
        heapq.heappush(self.Wk,[value,index])
```

Τέλος για τον έλεγχο της ορθότητας έφτιαξα μία κλάση brute_force η οποία διαβάζει όλα τα δεδομένα και βρίσκει τα όλα τα αποτελέσματα.

```
You, 4 hours ago | 1 author (You)
class brute_force:
    def __init__(self,R,path1,path2):
        self.R = R
        self.path1 = path1
        self.path2 = path2
        self.lower_bound = {}
        self.final_list = []

    def calc_all(self):
        with open(self.path1, 'r') as f:
            contents = f.read()
        lines = contents.splitlines()

        for i in range(len(lines)):
            split_line = lines[i].split()
            value = self.R[int(split_line[0])] + Decimal(split_line[1])
            self.lower_bound[int(split_line[0])]   =  value

        with open(self.path2, 'r') as f:
            contents2 = f.read()
        lines2 = contents2.splitlines()

        for i in range(len(lines2)):
            split_line2 = lines2[i].split()
            self.R[int(split_line2[0])] += Decimal(split_line2[1])
            value =  self.lower_bound[int(split_line2[0])] + Decimal(split_line2[1])
            self.final_list.append([value,int(split_line2[0])])

        return self.final_list
```