2022

Κωδικοί Εκπτωτικών Κουπονιών

ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

ΧΡΗΣΤΟΣ ΚΑΤΣΑΝΔΡΗΣ - 1072755

Πίνακας περιεχομένων

1 Επεξήγηση του προβλήματος

Το πρόβλημα των κωδικών εκπτωτικών κουπονιών συνίσταται στον υπολογισμό του πλήθους από ένα σύνολο των κουπονιών που έχουν απόσταση Hamming 1.

Ως απόσταση Hamming μεταξύ δύο strings ίσου μήκους ορίζεται το πλήθος θέσεων στις οποίες οι αντίστοιχοι χαρακτήρες είναι διαφορετικοί (Κοινότητα Wikipedia, 2019). Για παράδειγμα, τα strings:

`Hello World!` και `Hello Earth!`

έχουν απόσταση Hamming 4, αφού οι χαρακτήρες στις θέσεις 6 (`W-E`), 7 (`o-a`), 9 (`l-t`) και 10 (`d-h`) είναι διαφορετικοί.

Ζητείται, λοιπόν, από ένα σύνολο n κωδικών της μορφής XXXX-YYYY-XXXX, όπου X ανεξάρτητοι χαρακτήρες του λατινικού αλφαβήτου και Y ανεξάρτητα ψηφία του δεκαδικού συστήματος αρίθμησης (χωρίς τις παύλες), να υπολογιστεί το πλήθος των ζευγαριών που έχουν απόσταση Hamming 1. Για παράδειγμα, στο σύνολο 6 κωδικών:

`WELC1234OTHE`

`IEEE1234MEXZ`

`AAAA0000ACAD`

`AAAA0000ACAF`

`AAAA0000ACAB`

`AAAA0000ABAB`

τα ζευγάρια που έχουν απόσταση Hamming 1 είναι συνολικά 4 (`AAAA0000ACAD` - `AAAA0000ACAF`, `AAAA0000ACAD` - `AAAA0000ACAB`, `AAAA0000ACAF` - `AAAA0000ACAB` και `AAAA0000ACAB` - `AAAA0000ABAB`).

Φαινομενικά, ο αλγόριθμος που υπολογίζει το ζητούμενο πλήθος είναι αρκετά απλός. Μπορούμε να συγκρίνουμε γραμμικά κάθε έναν κωδικό της λίστας με όλους τους υπόλοιπους και τα αθροίσουμε το πλήθος ομοιοτήτων. Ο αλγόριθμος αυτός επιστρέφει ικανοποιητικά γρήγορα το αποτέλεσμα για μία λίστα 6 κωδικών, ωστόσο αρκετά γρήγορα γίνεται μη αποδοτικός, αφού όσο μεγαλώνει το πλήθος της λίστας, ο χρόνος εκτέλεσης αυξάνεται τετραγωνικά (πολυπλοκότητα $\boldsymbol{O(n^2)}$).

Θα πρέπει, λοιπόν, να καταφύγουμε σε έναν αλγόριθμο πολυπλοκότητας μικρότερης του $\boldsymbol{O(n^2)}$ (ιδανικά $\boldsymbol{O(n)}$), ο οποίος θα είναι αποδοτικός για οποιοδήποτε μέγεθος λίστας κωδικών. Στον αλγόριθμο που ζητείται να υλοποιήσουμε, θα πρέπει να γίνει χρήση **πινάκων κατακερματισμού (hash tables)**.

Ο αλγόριθμος αυτός μπορεί να φανεί χρήσιμος σε μία εταιρεία που αποφασίζει να δημιουργήσει εκπτωτικά κουπόνια για τους πελάτες της, ώστε να γνωρίζει πόσοι κωδικοί είναι επικίνδυνο να χρησιμοποιηθούν εσφαλμένα από κάποιον κάτοχο άλλου κουπονιού, λόγω τυπογραφικού λάθους.

2 Υλοποίηση του προγράμματος

Το πρόγραμμα θα υλοποιηθεί σε γλώσσα Python, στο αρχείο `discount_voucher_codes.py` που παρέχεται στο αρχείο της εργασίας. Για την υλοποίηση του προγράμματος θα χρησιμοποιηθεί αντικειμενοστραφής προγραμματισμός, με χρήση κλάσεων της Python.

2.1 Απαιτούμενες βιβλιοθήκες

Ο Πίνακας 1 παρουσιάζει τις βιβλιοθήκες της Python που χρησιμοποιούνται στο πρόγραμμα. Ο χρήστης μπορεί να εγκαταστήσει τις βιβλιοθήκες αυτές, χρησιμοποιώντας το module `pip`.

Βιβλιοθήκη	**Χρήση**
`enum.Enum`	Υλοποίηση απαρίθμησης (enumeration)
`random`	Εκλογή τυχαίων αριθμών
`timeit`	Χρονομέτρηση
`math`	Υπολογισμός συνδυασμών
`sympy`	Εύρεση επόμενου πρώτου αριθμού

Πίνακας 1 – Απαιτούμενες βιβλιοθήκες

2.2 Διάρθρωση του προγράμματος

Αρχικά, ορίζεται η global μεταβλητή `ascii_letters` που θα χρησιμοποιηθεί για τη δημιουργία τυχαίων κωδικών.

Το πρόγραμμα διαρθρώνεται σε 3 κλάσεις.

- Η κλάση `Voucher` αντιπροσωπεύει ένα εκπτωτικό κουπόνι και αναλύεται στην Ενότητα 2.3.
- Η κλάση `HashTable` υλοποιεί έναν πίνακα κατακερματισμού με όλες τις απαραίτητες μεθόδους και αναλύεται στην Ενότητα 2.4.
- Η κλάση `Algorithm` αποτελεί στην κύρια κλάση του προγράμματος και αναλύεται στην Ενότητα 2.5.

Ο οδηγός κώδικας του προγράμματος βρίσκεται στο επίπεδο `<module>` και αναλύεται στην Ενότητα 2.6.

```
from enum import Enum
import random
import timeit
import math
import sympy

ascii_letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

Κώδικας 1. Imports και δήλωση global μεταβλητής.

2.3 Η κλάση Voucher

Η κλάση **Voucher** έχει μόνο μία ιδιότητα: το **code**, που αντιπροσωπεύει και τον κωδικό του εκπτωτικού κουπονιού.

Η μέθοδος **__init__** αρχικοποιεί τη μεταβλητή **code** με ένα κενό string.

Η μέθοδος **__str__** επιστρέφει τον κωδικό του εκπτωτικού κουπονιού χωρισμένο με παύλες ανά 4 χαρακτήρες, για να ταιριάζει με τη μορφή XXXX-YYYY-XXXX, που εξηγήθηκε στην Ενότητα 1. Αυτό θα χρησιμοποιηθεί για την εκτύπωση του κωδικού στην κονσόλα.

Η μέθοδος **generate** δημιουργεί έναν τυχαίο κωδικό για το εκπτωτικό κουπόνι. Με χρήση των **random.choice** και **random.randint**, επιλέγονται σειριακά 4 χαρακτήρες της global μεταβλητής **ascii_letters** (δηλαδή 4 γράμματα του λατινικού αλφαβήτου), 4 ψηφία του δεκαδικού συστήματος αρίθμησης και ξανά 4 γράμματα, με τυχαίο τρόπο. Το αποτέλεσμα αποθηκεύεται στη μεταβλητή **code**.

Η μέθοδος **load** φορτώνει στη μεταβλητή **code** απευθείας έναν έτοιμο κωδικό. Η μέθοδος αυτή χρησιμοποιείται, όταν επιθυμούμε να χρησιμοποιήσουμε έτοιμα εκπτωτικά κουπόνια από ένα εξωτερικό αρχείο.

```
class Voucher:
    def __init__(self):
        self.code = str()

    def __str__(self):
        return self.code[0:4] + '-' + self.code[4:8] + '-' + self.code[8:12]

    def generate(self):
        for _ in range(4):
            self.code += random.choice(ascii_letters)
        for _ in range(4):
            self.code += str(random.randint(0,9))
        for _ in range(4):
            self.code += random.choice(ascii_letters)

    def load(self, code):
        self.code = code

    def star_code(self, starred_codes_list):
        chars = list(self.code)
        for index in range(len(chars)):
            original = chars[index]
            chars[index] = "*"
            starred_codes_list.append("".join(chars))
            chars[index] = original
```

Κώδικας 2. Η κλάση Voucher.

Η μέθοδος `star_code` διατρέχει τον κωδικό του εκπτωτικού κουπονιού και για κάθε έναν χαρακτήρα του, τον αντικαθιστά με έναν χαρακτήρα `'*'`. Για παράδειγμα, ο κωδικός `WELC12340THE`, θα δημιουργήσει τα strings `*ELC12340THE`, `W*LC12340THE`, …, `WELC12340TH*`. Τα 12 strings που θα προκύψουν αποθηκεύονται σε μία λίστα, την οποία η μέθοδος δέχεται ως παράμετρο.

2.4 Η κλάση `HashTable`

Η κλάση `HashTable` έχει δύο ιδιότητες: το `size`, που αντιπροσωπεύει το μέγεθος του πίνακα και το `table`, δηλαδή τη λίστα που αντιπροσωπεύει τον πίνακα κατακερματισμού.

Η μέθοδος `__init__` αρχικοποιεί τη μεταβλητή `size` στην τιμή που δέχεται ως παράμετρο και τη μεταβλητή `table` σε μία λίστα που αποτελείται από `size` το πλήθος κενές λίστες. Κάθε μία από αυτές τις κενές λίστες αντιστοιχεί σε ένα κελί του πίνακα κατακερματισμού.

Η μέθοδος `insert` δέχεται ως παράμετρο έναν κωδικό (ο οποίος θα περιέχει και χαρακτήρα `'*'`) και τον τοποθετεί στο κατάλληλο κελί του hash table. Αρχικά, υπολογίζεται η μεταβλητή `key`, η οποία θα δώσει το κελί του πίνακα, στο οποίο πρόκειται να εισαχθεί ο κωδικός, σύμφωνα με τη συνάρτηση κατακερματισμού (hash function):

```
key = hash(starred_code) % self.size
```

όπου `starred_code` είναι ο κωδικός (με χαρακτήρα `'*'`) που πρόκειται να εισαχθεί, `self.size` είναι το μέγεθος του hash table και `hash()` είναι η εγγενής συνάρτηση της Python που επιστρέφει την τιμή hash του αντικειμένου που δέχεται ως παράμετρο. Το hash είναι ένας ακέραιος σταθερού μήκους, ο οποίος αντιπροσωπεύει μία συγκεκριμένη τιμή (value). Κάθε τιμή έχει το δικό της hash, οπότε δύο ίδιες τιμές έχουν το ίδιο hash, ακόμα και αν δεν είναι το ίδιο αντικείμενο (Regebro, 2013).

Επομένως, το κελί στο οποίο πρόκειται να εισαχθεί ο κωδικός βρίσκεται στη θέση που υπολογίζεται από το υπόλοιπο της διαίρεσης του hash του κωδικού δια το μέγεθος του hash table. Για να αποφευχθούν οι συγκρούσεις, υλοποιούμε το hash table με τη **μέθοδο ξεχωριστών αλυσίδων (separate chaining)**, δηλαδή κάθε κελί είναι μία λίστα. Τα περιεχόμενα κάθε κελιού είναι λίστες, καθεμία από τις οποίες περιέχει ένα string και έναν ακέραιο. Το string αποτελεί έναν κωδικό με χαρακτήρα `'*'` και ο ακέραιος το πλήθος των εμφανίσεων του συγκεκριμένου κωδικού. Για παράδειγμα, ένα κελί μπορεί να είναι το ακόλουθο:

```
[["AB*D1234ABCD", 1], ["EFGH5*78EFGH", 2], ["AAAA0000A*AA", 1]]
```

εννοώντας ότι τα strings **"AB*D1234ABCD"**, **"EFGH5*78EFGH"**, και **"AAAA0000A*AA"** έχουν το ίδιο κλειδί (αφού βρίσκονται στο ίδιο κελί) και επίσης το string **"EFGH5*78EFGH"** έχει εμφανιστεί δύο φορές (αφού ο ακέραιός του είναι 2), δηλαδή αντιστοιχεί σε δύο εκπτωτικά κουπόνια με απόσταση Hamming 1 (π.χ. **"EFGH5678EFGH"** και **"EFGH5778EFGH"**).

Για να εισάγουμε έναν κωδικό σε ένα κελί, ελέγχουμε πρώτα τα υπάρχοντα περιεχόμενα του κελιού, μήπως ο κωδικός υπάρχει ήδη. Αν ναι, αυξάνουμε τον ακέραιο της αντίστοιχης λίστας κατά 1. Αν όχι, εισάγουμε στο κελί μία νέα λίστα, που αποτελείται από τον κωδικό και τον ακέραιο 1.

Η μέθοδος `count_duplicates` μετράει τους συνδυασμούς εκπτωτικών κουπονιών που έχουν απόσταση Hamming 1. Για να γίνει αυτό, διατρέχει τα κελιά του πίνακα κατακερματισμού και ελέγχει τον ακέραιο κάθε περιεχομένου κάθε κελιού. Αν ο ακέραιος αυτός είναι μεγαλύτερος του 1 (εννοώντας ότι ο ίδιος κωδικός με χαρακτήρα '*' έχει εμφανιστεί περισσότερες από μία φορές), υπολογίζεται, με χρήση της `math.comb`, ο αριθμός $\binom{k}{2}$ (k ανά 2), δηλαδή το πλήθος των ζευγαριών που μπορούν να σχηματιστούν, και προστίθεται σε έναν μετρητή. Η τελική τιμή αυτού του μετρητή αποτελεί και το πλήθος των εκπτωτικών κουπονιών που έχουν απόσταση Hamming 1.

Για παράδειγμα, για τους κωδικούς:

`WELC1234OTHE`

`IEEE1234MEXZ`

`AAAA0000ACAD`

`AAAA0000ACAF`

`AAAA0000ACAB`

`AAAA0000ABAB`

βλέπουμε με debugging ότι το hash table διαμορφώνεται ως εξής:

```
class HashTable:
    def __init__(self, size):
        self.size = size
        self.table = [[] for _ in range(size)]

    def insert(self, starred_code):
        key = hash(starred_code) % self.size

        cell = self.table[key]
        for content in cell:
            if content[0] == starred_code:
                content[1] += 1
                return
        cell.append([starred_code, 1])

    def count_duplicates(self):
        counter = 0
        for cell in self.table:
            for content in cell:
                if content[1] > 1:
                    counter += math.comb(content[1], 2)
        return counter
```

Κώδικας 3. Η κλάση HashTable.

```
00:  [["AAAA0*00ACAF", 1]]
01:  [[]]
02:  [["*AAA0000ACAD", 1], ["*AAA0000ACAF", 1], ["AA*A0000ACAB", 1]]
     …
55:  [["*ELC1234OTHE", 1], ["AAAA0000A*AB", 2]]
     …
69:  [["WEL*1234OTHE", 1], ["AAAA0000ACA*", 3]]
     …
72:  [["IEEE1234M*XZ", 1]]
```

Έτσι, όταν η `count_duplicates` διατρέξει τον πίνακα κατακερματισμού, θα εστιάσει στα κελιά 55 και 69 (που έχουν περιεχόμενα με ακεραίους μεγαλύτερους του 1) και θα πραγματοποιήσει την πρόσθεση:

$$counter = \binom{2}{2} + \binom{3}{2} = 1 + 3 = 4$$

δηλαδή θα εντοπίσει 4 ζευγάρια κωδικών με απόσταση Hamming 1.

2.5 Η κλάση `Algorithm`

Η κλάση `Algorithm` διαθέτει την εμφωλευμένη (nested) κλάση `Mode`, η οποία ουσιαστικά αποτελεί μία απαρίθμηση (enumeration) με στοιχεία `GENERATE` και `IMPORT`.

Η κλάση `Algorithm` έχει 7 ιδιότητες:

- το `mode`, που μπορεί να πάρει μόνο τις τιμές `Mode.GENERATE` και `Mode.IMPORT` και καθορίζει το αν ο αλγόριθμος πρόκειται να δημιουργήσει νέα εκπτωτικά κουπόνια ή να εισάγει υπάρχοντα από κάποιο εξωτερικό αρχείο,
- το `number_of_vouchers`, που είναι το πλήθος των κουπονιών που πρόκειται να δημιουργηθούν, σε περίπτωση που `mode = Mode.GENERATE`,
- το `import_filename`, που είναι το όνομα του αρχείου απ' όπου πρόκειται να εισαχθούν τα υπάρχοντα κουπόνια, σε περίπτωση που `mode = Mode.IMPORT`,
- το `export`, που καθορίζει αν τα εκπτωτικά κουπόνια του αλγορίθμου θα εξαχθούν σε εξωτερικό αρχείο,
- το `export_filename`, που είναι το όνομα του αρχείου όπου πρόκειται να εξαχθούν τα κουπόνια, σε περίπτωση που `export = True`,
- το `vouchers_list`, που είναι η λίστα στην οποία είναι αποθηκευμένα τα αντικείμενα τύπου `Voucher`, που αντιπροσωπεύουν τα εκπτωτικά κουπόνια και
- το `starred_codes_list`, που είναι η λίστα στην οποία αποθηκεύονται οι κωδικοί με χαρακτήρα `'*'`.

Η μέθοδος `__init__` αρχικά ελέγχει τις παραμέτρους της για τυχόν ασυνέπειες. Συγκεκριμένα, κάνει τους ακόλουθους ελέγχους απαραίτητων παραμέτρων:

- Εάν ορίστηκε `mode = Mode.GENERATE` χωρίς να δοθεί επιθυμητό πλήθος κουπονιών `number_of_vouchers`.
- Εάν ορίστηκε `mode = Mode.GENERATE` και το `number_of_vouchers` δεν είναι θετικός ακέραιος.
- Εάν ορίστηκε `mode = Mode.IMPORT` χωρίς να δοθεί επιθυμητό όνομα αρχείου `import_filename`.
- Εάν ορίστηκε `export = True` χωρίς να δοθεί επιθυμητό όνομα αρχείου `export_filename`.

Σε οποιαδήποτε από αυτές τις 4 περιπτώσεις, εγείρεται `TypeError` και ο αλγόριθμος σταματά την εκτέλεσή του.

Επιπλέον, γίνονται οι ακόλουθοι έλεγχοι αχρησιμοποίητων παραμέτρων:

- Εάν ορίστηκε `mode = Mode.GENERATE` και δόθηκε όνομα αρχείου `import_filename`, το οποίο και θα απορριφθεί.
- Εάν ορίστηκε `mode = Mode.IMPORT` και δόθηκε πλήθος κουπονιών `number_of_vouchers`, ο οποίος και θα απορριφθεί.
- Εάν ορίστηκε `export = False` και δόθηκε όνομα αρχείου `export_filename`, το οποίο και θα απορριφθεί.

Σε οποιαδήποτε από αυτές τις 3 περιπτώσεις, τυπώνεται μία προειδοποίηση (warning) και ο αλγόριθμος συνεχίζει κανονικά την εκτέλεσή του.

Έπειτα, η `__init__` αρχικοποιεί τις ιδιότητες της κλάσης και καλεί τη μέθοδο `main`, η οποία αναλύεται παρακάτω.

Η μέθοδος `generate_vouchers` δημιουργεί `number_of_vouchers` το πλήθος αντικείμενα τύπου `Voucher`, καλεί τη μέθοδό τους `generate` και τα τοποθετεί στη λίστα `vouchers_list`. Εκτυπώνει στην κονσόλα μήνυμα επιτυχίας που επιβεβαιώνει το πλήθος των κουπονιών που δημιουργήθηκαν.

Η μέθοδος `import_vouchers` διαβάζει το αρχείο με όνομα `import_filename`. Δημιουργεί αντικείμενα τύπου `Voucher`, καλεί τη μέθοδό τους `load` με παράμετρο το string που διάβασε από το αρχείο και τα τοποθετεί στη λίστα `vouchers_list`. Ενημερώνει τη μεταβλητή `number_of_vouchers` με το πλήθος των κουπονιών που εισήχθησαν και εκτυπώνει στην κονσόλα μήνυμα επιτυχίας.

Η μέθοδος `export_vouchers` εκτυπώνει στο αρχείο με όνομα `export_filename` τους κωδικούς των εκπτωτικών κουπονιών της λίστας `vouchers_list` και εκτυπώνει στην κονσόλα μήνυμα επιτυχίας.

Η μέθοδος `main` αποτελεί τον πυρήνα του αλγορίθμου. Ανάλογα με την τιμή του `mode`, καλεί μία εκ των `generate_vouchers` και `import_vouchers`. Έπειτα, ξεκινά τη χρονομέτρηση, με χρήση της βιβλιοθήκης `timeit`. Για κάθε ένα εκπτωτικό κουπόνι της λίστας `vouchers_list`, καλεί τη μέθοδό τους `star_code` (βλ. Ενότητα 2.3), ώστε να δημιουργηθούν οι κωδικοί με χαρακτήρα `'*'` που προκύπτουν από το εκάστοτε εκπτωτικό κουπόνι και να τοποθετηθούν στη λίστα `starred_codes_list`. Δημιουργεί ένα αντικείμενο τύπου `HashTable` μεγέθους ίσο

με τον επόμενο πρώτο (prime) αριθμό του πλήθους των κωδικών με χαρακτήρα **'*'**. Στον πίνακα κατακερματισμού πρόκειται να εισαχθούν όλοι οι κωδικοί με χαρακτήρα **'*'**, ωστόσο επιλέγουμε μέγεθος πίνακα ίσο με κάποιον πρώτο αριθμό, αφού όπως γνωρίζουμε από τη θεωρία του κατακερματισμού, τέτοια μεγέθη οδηγούν σε πιο αποδοτικούς πίνακες κατακερματισμού. Για κάθε κωδικό με χαρακτήρα **'*'**, καλεί τη μέθοδο **`insert`** του hash table (βλ. Ενότητα 2.4), ώστε αυτός να τοποθετηθεί στο κατάλληλο κελί. Όταν τοποθετηθούν όλοι οι κωδικοί με χαρακτήρα **'*'**, καλεί τη μέθοδο **`count_duplicates`** του hash table (βλ. Ενότητα 2.4), ώστε να γίνει καταμέτρηση των ζευγαριών που έχουν απόσταση Hamming 1. Τότε, σταματά η χρονομέτρηση και εκτυπώνεται στην κονσόλα μήνυμα που περιλαμβάνει το πλήθος των κουπονιών με απόσταση Hamming 1 που βρέθηκαν και τον χρόνο εκτέλεσης. Τέλος, εάν έχει επιλεγεί **`export = True`**, καλεί τη μέθοδο **`export_vouchers`**, ώστε τα κουπόνια να αποθηκευτούν σε εξωτερικό αρχείο.

```
class Algorithm:
    class Mode(Enum):
        GENERATE = 1,
        IMPORT = 2

    def __init__(
        self,
        mode = Mode.GENERATE,
        number_of_vouchers = None,
        import_filename = None,
        export = False,
        export_filename = None
    ):
        """
        Parameters
        ----------
        mode : Algorithm.Mode
            Choose Algorithm.Mode.GENERATE should you want to generate random
voucher codes.
            Choose Algorithm.Mode.IMPORT should you want to import voucher codes
from a file.
            default: Algorithm.Mode.GENERATE

        number_of_vouchers : int
            Choose how many vouchers should be generated.
            Number should be a positive integer.
            Required argument if GENERATE mode has been selected.
            default: None
```

Κώδικας 4. Η κλάση Algorithm.

```
        import_filename : str
            Choose the file from which the vouchers should be imported.
            Required argument if IMPORT mode has been selected.
            default: None

        export : boolean
            Choose whether the voucher codes should be exported to a file.
            default: False

        export_filename : str
            Choose the file to which the vouchers should be exported.
            Required argument if export has been set True.
            default: None
        """

        # Required arguments validation
        if mode == self.Mode.GENERATE and number_of_vouchers == None:
            raise TypeError("Please specify how many vouchers should be
generated.")
        if mode == self.Mode.GENERATE and ((not isinstance(number_of_vouchers,
int)) or number_of_vouchers < 1):
            raise TypeError("Invalid number of vouchers requested.")
        if mode == self.Mode.IMPORT and import_filename == None:
            raise TypeError("Please specify the file name from which the vouchers
should be imported.")
        if export and export_filename == None:
            raise TypeError("Please specify the file name to which the vouchers
should be exported.")

        # Unused arguments validation
        if mode == self.Mode.GENERATE and import_filename != None:
            print("WARNING: Vouchers will be generated. Import filename will be
discarded.")
        if mode == self.Mode.IMPORT and number_of_vouchers != None:
            print("WARNING: Vouchers will be imported. Requested number of
vouchers will be discarded.")
        if (not export) and export_filename != None:
            print("WARNING: Vouchers will not be exported. Export filename will
be discarded.")

        self.mode = mode
        self.export = export
        self.import_filename = import_filename
```

Κώδικας 5. Η κλάση Algorithm (συνέχεια).

```
        self.export_filename = export_filename
        self.number_of_vouchers = number_of_vouchers

        self.vouchers_list = []
        self.starred_codes_list = []

        self.main()

    def generate_vouchers(self):
        for _ in range(self.number_of_vouchers):
            new_voucher = Voucher()
            new_voucher.generate()
            self.vouchers_list.append(new_voucher)
        print(f"Generated {self.number_of_vouchers} vouchers.")

    def import_vouchers(self):
        with open(self.import_filename, "rt") as file:
            while True:
                new_line = file.readline().strip()
                if new_line == "":
                    break
                new_voucher = Voucher()
                new_voucher.load(new_line)
                self.vouchers_list.append(new_voucher)
        self.number_of_vouchers = len(self.vouchers_list)
        print(f"Imported {self.number_of_vouchers} vouchers.")

    def export_vouchers(self):
        with open(self.export_filename, "wt") as file:
            for voucher in self.vouchers_list:
                print(voucher.code, file=file)
        print(f"Exported {self.number_of_vouchers} vouchers.")

    def main(self):
        if self.mode == self.Mode.GENERATE:
            self.generate_vouchers()
        else:
            self.import_vouchers()

        initial_timestamp = timeit.default_timer()
        for voucher in self.vouchers_list:
            voucher.star_code(self.starred_codes_list)
```

Κώδικας 6. Η κλάση Algorithm (συνέχεια).

```
        hash_table = HashTable(sympy.nextprime(len(self.starred_codes_list)))

        for starred_code in self.starred_codes_list:
            hash_table.insert(starred_code)

        duplicates = hash_table.count_duplicates()

        final_timestamp = timeit.default_timer() - initial_timestamp
        print(f"Found {duplicates} similar codes in {final_timestamp} seconds.")

        if self.export:
            self.export_vouchers()

        print("-------------------------------------")
```

Κώδικας 7. Η κλάση Algorithm (συνέχεια).

2.6 Ο οδηγός κώδικας

Ο οδηγός κώδικας του προγράμματος καλείται στην έναρξη της εκτέλεσης του προγράμματος και δημιουργεί τρία αντικείμενα τύπου **Algorithm**, δηλαδή εκτελεί τον αλγόριθμο για τρία διαφορετικά σενάρια:

- Στο πρώτο σενάριο, ο αλγόριθμος δημιουργεί 100000 εκπτωτικά κουπόνια.
- Στο δεύτερο σενάριο, ο αλγόριθμος εισάγει από εξωτερικό αρχείο δοκιμαστικούς κωδικούς.
- Στο τρίτο σενάριο, ο αλγόριθμος δημιουργεί 200000 εκπτωτικά κουπόνια και τα αποθηκεύει σε εξωτερικό αρχείο.

Τα αποτελέσματα εκτέλεσης του αλγορίθμου φαίνονται στον παρακάτω πίνακα:

```
if __name__ == "__main__":
    Algorithm(
        mode = Algorithm.Mode.GENERATE,
        number_of_vouchers = 100000
    )
    Algorithm(
        mode = Algorithm.Mode.IMPORT,
        import_filename = "voucher_list.txt",
    )
    Algorithm(
        mode = Algorithm.Mode.GENERATE,
        number_of_vouchers = 200000,
        export = True,
        export_filename = "exported_vouchers.txt"
    )
```

Κώδικας 8. Ο οδηγός κώδικας.

Σενάριο	Πλήθος εκπτωτικών κουπονιών	Κουπόνια με απόσταση Hamming 1	Χρόνος εκτέλεσης (sec)
1	100000	0	3.8719947
2	6	4	0.0001379
3	200000	0	7.7126406

Πίνακας 2. Αποτελέσματα εκτελέσεων αλγορίθμου.

3 Συμπεράσματα

Η χρήση πινάκων κατακερματισμού για την εύρεση των κωδικών με απόσταση Hamming 1 στο πρόβλημα κωδικών εκπτωτικών κουπονιών αποδεικνύεται εξαιρετικά χρήσιμη, μιας και καταφέρνει να μειώσει την πολυπλοκότητα του αλγορίθμου σε $\boldsymbol{O(n)}$. Η γραμμική πολυπλοκότητα είναι εμφανώς πιο αποδοτική, όταν διαχειρίζονται μεγάλα σύνολα δεδομένων, όπως για παράδειγμα από μία μεγάλη εταιρεία που δημιουργεί εκπτωτικά κουπόνια για τους πελάτες της.

4 Βιβλιογραφία – Αναφορές

Chapagain, M. (2018, Ιανουαρίου 29). *Hash Table implementation in Python [Data Structures & Algorithms]*. Ανάκτηση από Mukesh Chapagain Blog: https://blog.chapagain.com.np/hash-table-implementation-in-python-data-structures-algorithms/

Regebro, L. (2013, Ιουλίου 11). *What does hash do in python?* Ανάκτηση από Stack Overflow: https://stackoverflow.com/questions/17585730/what-does-hash-do-in-python

Κοινότητα Wikipedia. (2019, Οκτωβρίου 31). *Απόσταση Χάμινγκ*. Ανάκτηση από Ελληνική Wikipedia: https://el.wikipedia.org/wiki/Απόσταση_Χάμινγκ