Γλωσσική Τεχνολογία

Δομές Δεδομένων – File & Process Handling

Επιλογή δομής δεδομένων

- Κριτήρια:
  - Μέγεθος του προβλήματος
  - Πως θα χρησιμοποιηθεί
- Ενέργειες που καθορίζουν το κόστος:
  - Lookup: αναζήτηση/έλεγχος ύπαρξης δεδομένων στη δομή.
  - Insert: εισαγωγή δεδομένων στη δομή.

Συνήθεις κλάσεις

- Οι νεότερες γλώσσες προγραμματισμού προσφέρουν συνήθως δύο βασικές κλάσεις:
  - Dictionary
    - Δυναμικού μεγέθους.
    - Κάθε εγγραφή είναι του τύπου <key, value>.
    - Το κλειδί εισάγεται σε hashtable.
    - Οι εγγραφές δεν ταξινομούνται.
  - List:
    - Δυναμικού μεγέθους.
    - Κάθε εγγραφή είναι μόνο value.
    - Δεν υπάρχει οργάνωση.
    - Υποστηρίζει κλήσεις ταξινόμησης.

Υπέρ-Κατά

- Dictionaries:
  - Πολύ γρήγορα lookups.
  - Πιο αργή προσθήκη εγγραφών (hashing process).
  - Για να ταξινομηθεί πρέπει να εξαχθεί η λίστα των κλειδιών.
- Lists:
  - Αργά lookups. Συνήθως γίνεται σειριακή αναζήτηση.
  - Γρήγορη προσθήκη εγγραφών.
  - Μπορεί να ταξινομηθεί.

Τι χρησιμοποιούμε?

- Κριτήρια:
  - Πόσες θα είναι οι εγγραφές? Πχ για < 10 δεν αξίζει το κόστος του hashing.
  - Πόσο συχνά θα ψάχνουμε στη δομή? Πχ αν σε κάθε insert αντιστοιχεί και ένα lookup ή περισσότερο το hashing συμφέρει.
  - Θα ταξινομήσουμε? Αν και οι περισσότερες γλώσσες υποστηρίζουν απευθείας μετατροπή των κλειδιών του Dictionary σε λίστα για να γίνει ταξινόμηση.
- Πως επιλέγουμε?
  - Μαντεύοντας. Ο γενικός κανόνας είναι "rule of thumb". Κάθε πρόβλημα είναι διαφορετικό και μέχρι να το δεις να δουλεύει δεν ξέρεις με σιγουριά.

Python - Lists

- list.sort()
  - Ταξινόμηση λίστας
- list.reverse()
  - Αντιστροφή στοιχείων λίστας

```
>>> a = [66.25, 333, 333, 1, 1234.5]
>>> a
[66.25, 333, 333, 1, 1234.5]
>>> a.sort()
>>> a
[1, 66.25, 333, 333, 1234.5]
>>> a.reverse()
>>> a
[1234.5, 333, 333, 66.25, 1]
>>>
```

Python - Lists as stacks

- list.append()
  - Προσθήκη στο τέλος της λίστας
- list.pop()
  - Αφαίρεση από το τέλος της λίστας

```
>>> stack = [1,2,3,4,5]
>>> stack
[1, 2, 3, 4, 5]
>>> stack.append(6)
>>> stack.append(7)
>>> stack
[1, 2, 3, 4, 5, 6, 7]
>>> stack.pop()
7
>>> stack.pop()
6
>>> stack
[1, 2, 3, 4, 5]
>>>
```

Python – List Comprehensions

```
>>> a = [1,2,3,4,5]
>>> b = [6,7,8]
>>> [x*2 for x in a]
[2, 4, 6, 8, 10]
>>> [x*3 for x in a if x > 2]
[9, 12, 15]
>>> [x+y for x in a for y in b]
[7, 8, 9, 8, 9, 10, 9, 10, 11, 10, 11, 12, 11, 12, 13]
>>> [x for x in a if x in b]
[]
>>> [x+y for x in a for y in b if x+y<10]
[7, 8, 9, 8, 9, 9]
>>>
```

Python - Dictionaries

```
>>> d = {'t1':5, 't2':3, 't3':0, 't4':4}
>>> keys = d.keys()
>>> keys.sort()
>>> keys
['t1', 't2', 't3', 't4']
>>> pairs = d.items()
>>> pairs
[('t4', 4), ('t2', 3), ('t3', 0), ('t1', 5)]
>>> pairs.sort()
>>> pairs
[('t1', 5), ('t2', 3), ('t3', 0), ('t4', 4)]
>>>
```

Το module os – Διαχείριση Paths

- import os
  - Module για την διαχείρηση αρχείων και φακέλων.
  - Είναι cross-platform!!
- os.path.join(path1 [, path2[, ...]])
  - Συνένωση μονοπατιών.
  - π.χ: os.path.join("c:\\music\\ap", "mahadeva.mp3")
  - Παρατηρείστε την έλλειψη "\". Το αποτέλεσμα είναι σωστό!! Προστίθεται από μόνο του!!
- os.join.expanduser(~)
  - Μεταφορά στον "home" folder ανάλογα με το ΛΣ.
    - HOME (linux)
    - My Documents (Windows)

Το module os – Διαχείριση Paths

- os.path.split(path)
  - Επιστρέφει ένα tuple της μορφής (head, tail) όπου head το path μέχρι το τελευταίο "/" και tail ότι το ακολουθεί.
  - π.χ.: (filepath, filename) = os.path.split("c:\\music\\ap\\mahadeva.mp3")
    - filepath = 'c:\\music\\ap'
    - filename = 'mahadeva.mp3'

Το module os – Διαχείριση Paths

- os.path.splitext(path)
  - Επιστρέφει ένα tuple της μορφής (root, ext) όπου root το filename και ext η κατάληξη.
  - π.χ.: (shortname, extension) = os.path.splitext(filename)
  - shortname = 'mahadeva'
  - extension = '.mp3'

Το module os – Αρχεία & Φάκελοι

- os.path.isfile(path)
  - Επιστρέφει boolean τιμή ανάλογα με το όρισμα.
- os.path.isdir(path)
  - Επιστρέφει boolean τιμή ανάλογα με το όρισμα.
- os.listdir(path)
  - Επιστρέφει μία λίστα με τα περιεχόμενα του φακέλου που δίνεται ως όρισμα.
- Χρήσιμα list comprehensions:

```
[f for f in os.listdir('/bin') if os.path.isfile(os.path.join('/bin',f))]
[f for f in os.listdir('/bin') if os.path.isdir(os.path.join('/bin',f))]
```

Το module os – Κλήση Διεργασιών

- os.system(command)
  - Εκτέλεση εξωτερικής διεργασίας. Η παράμετρος command πρόκειται για ένα string που καλεί στην ουσία ένα εξωτερικό πρόγραμμα.
  - π.χ.: os.system("ls")
  - Ως επιστρεφόμενη τιμή, λαμβάνεται το exit status της διεργασίας.
  - Η υλοποίηση της παραπάνω συνάρτησης γίνεται μέσω της αντίστοιχης συνάρτησης system() της C και ακολουθείται γενικά το πρότυπο POSIX.

Το module os – Κλήση Διεργασιών

- os.popen(*command*[, *mode*[, *bufsize*]])¶
  - Κλήση εξωτερικής διεργασίας μέσω του command
  - Επιστρέφει file handler: μπορείτε να ανοίξετε ρεύμα ανάγνωσης του stdout ή εγγραφής στο stdin
  - Πχ

```
f=os.popen('ls ~')
output = f.read()
f.close()
print output
```

  - FYI: στις νεότερες εκδόσεις της python, η χρήση της αντικαθίσταται από το subprocess module

Το module glob I

- import glob
  - Ανάκτηση full paths με χρήση wildcart.
- Παραδείγματα:
  - glob.glob('c:\\music_singles*.mp3')
  - ['c:\\music_singles\\a_time_long_forgotten_con.mp3',
  - 'c:\\music_singles\\hellraiser.mp3',
  - 'c:\\music_singles\\kairo.mp3',
  - 'c:\\music_singles\\long_way_home1.mp3',
  - 'c:\\music_singles\\sidewinder.mp3',
  - 'c:\\music_singles\\spinning.mp3']

To module glob II

- glob.glob('c:\\music_singles\\s*.mp3')
  - ['sidewinder.mp3', 'spinning.mp3']
- glob.glob('c:\\music**.mp3')
  - Θα επιστρέψει μία λίστα με όλα τα mp3s που περιέχονται σε όλους τους υποφακέλους του φακέλου music!!

Άνοιγμα αρχείων

- open(filename, mode)
  - Το πρώτο όρισμα είναι τύπου string και περιέχει το όνομα του αρχείου (ή και το path κάτω από το οποίο αυτό υπάρχει).
  - Το δεύτερο όρισμα είναι επίσης τύπου string και υποδηλώνει τον τρόπο με τον οποίο θα χρησιμοποιηθεί το αρχείο.
  - 'r' (read - default)
  - 'w' (write)
  - 'a' (append)
  - 'r+' (both read/write)

Ανάγνωση περιεχομένου Ι

- f.read(size)
  - Η παράμετρος size είναι προεραιτική και υποδηλώνει πόσα bytes θα διαβαστούν από το αρχείο f (file object).
  - Είναι ευθύνη του προγραμματιστή να καθορίσει την τιμή της size. Αν δεν δοθεί τιμή, η read διαβάζει ολόκληρο το περιεχόμενο!!
  - Μεγάλη προσοχή στην χρήση της!! Η μνήμη δεν είναι άπειρη...
  - Στο τέλος του αρχείου επιστρέφει empty string

Ανάγνωση περιεχομένου II

- f.readline()
  - Ανάγνωση μίας γραμμής από το αρχείο. Ως delimeter χρησιμοποιείται ο χαρακτήρας νέας γραμμής ('\n').
  - Το επιστρεφόμενο string περιέχει τον χαρακτήρα νέας γραμμής εκτός και αν πρόκειται για το τέλος του αρχείου, το οποίο δεν τελειώνει με νέα γραμμή!!
  - Προσοχή στην επιστρεφόμενη τιμή!! Ένα κενό string είναι το τέλος του αρχείου ενώ ένα string της μορφής '\n' είναι μία κενή γραμμή!!

Ανάγνωση περιεχομένου III

- f.readlines(sizehint)
  - Επιστρέφει μία λίστα, όπου κάθε στοιχείο της είναι μία γραμμή του αρχείου.
  - Η παράμετρος sizehint είναι προεραιτική και συνίσταται για πραγματικά μεγάλα αρχεία, με σκοπό την καλύτερη διαχείρηση μνήμης. Αναφέρεται σε bytes, αλλά η συνάρτηση θα επιστρέψει μόνο ολόκληρες γραμμές.
  - Η καλύτερη προσέγγιση!! (so far..)

Ανάγνωση περιεχομένου IV

```
for line in f:
        print line
```

- Απλοϊκή προσέγγιση
- Προσπέλαση ανά γραμμή χωρίς την χρήση συνάρτησης.
- Διαφορετικό buffering στην μνήμη!! Δεν πρέπει να χρησιμοποιείται σε συνδυασμό με τις προηγούμενες μεθόδους.
- Καλύτερη διαχείριση μνήμης (streaming)

Εγγραφή

- f.write(string)
  - Τόσο απλά!!
  - Για εγγραφή άλλου τύπου δεδομένων, π.χ. int, πρέπει να γίνει πρώτα μετατροπή σε string, π.χ.:
  - a = 5
  - s = str(a)
  - f.write(s)
  - Hint: Προσοχή κατά την αντίστοιχη ανάκτηση και χρήση αριθμών από αρχεία. Πρέπει να γίνει μετατροπή σε int πριν την χρήση τους σε μαθηματικές πράξεις..

Προσπέλαση περιεχομένου

- f.tell()
  - Επιστρέφει έναν int που δείχνει σε ποιο byte (ξεκινώντας απ'την αρχή του αρχείου) βρίσκεται η προσπέλαση.
- f.seek(offset, from_what)
  - Ρητή αλλαγή στην θέση του δείκτη.
  - Η παράμετρος offset υποδηλώνει τα bytes που προστίθενται στην from_what για την μετακίνησή του κάθε φορά.
  - Η παράμετρος from_what υποδηλώνει την θέση του δείκτη.
    - 0 (απ'την αρχή του αρχείου - default)
    - 1 (απ'την τρέχουσα θέση του μέσα στο αρχείο)
    - 2 (από το τέλος του αρχείου)

Κλείσιμο αρχείων

- f.close()
  - Τόσο απλά!!
  - Όταν τελειώσουμε με το αρχείο εκτελούμε την παραπάνω εντολή για την αποδέσμευση μνήμης και την αποφυγή περίεργων καταστάσεων…

Exceptions I

- Κατά το άνοιγμα/κλείσιμο ή κατά την ανάγνωση/εγγραφή ενός αρχείου οτιδήποτε μπορεί να πάει στραβά.. (“Νόμος του Murphy για το I/O.”)
- Χρησιμοποιούμε χειρισμό εξαιρέσεων για την αποφυγή “βίαιου” τερματισμού της εκτέλεσης του προγράμματός μας.
- Ο τύπος exception που γίνεται “throw” σε αυτές τις περιπτώσεις είναι ο IOError.

Exceptions II

- Γενική μορφή try block:

```
try:
        ......
except IOError:
        pass
finally:
        ......
```

Exceptions III

- Παράδειγμα:

```
try:
        f = open(path,mode)
        f.readlines()
        ......
except IOError:
        pass
finally:
        f.close()
```