ΥΣ02 Τεχνητή Νοημοσύνη

Χειμερινό Εξάμηνο 2018-2019

Εργασία 0 (0.25 μονάδες του συνολικού βαθμού στο μάθημα)

Ημερομηνία Παράδοσης: 9 Οκτωβρίου 2018, ώρα 24:00

Η εργασία αυτή έχει σκοπό την εξοικείωση σας με την γλώσσα προγραμματισμού python που θα χρησιμοποιήσουμε στις υπόλοιπες εργασίες. Η εργασία αποτελείται από τα παρακάτω δύο μέρη:

1. Να κάνετε το Project 0 από τα Pacman projects (http://ai.berkeley.edu/tutorial.html). Θα παραδώσετε μόνο τα αρχεία `addition.py`, `buyLotsOfFruits.py` και `shopSmart.py`
2. Στο δεύτερο μέρος της εργασίας αυτής θα υλοποιήσουμε μια δομή δεδομένων που θα χρειαστούμε συχνά στις υλοποιήσεις αλγορίθμων αναζήτησης που θα μελετήσουμε σύντομα στο μάθημα.

   Να ορίσετε μια κλάση `PriorityQueue` η οποία ορίζει μια ουρά προτεραιότητας υλοποιημένη σαν σωρό (heap). Οι έννοιες της ουράς προτεραιότητας και του σωρού θεωρούνται γνωστές από το μάθημα «Δομές Δεδομένων και Τεχνικές Προγραμματισμού». Αν όχι, θα χρειαστείτε λίγο διάβασμα από τις διαφάνειες του μαθήματος όπως δόθηκε πέρυσι:

   - Για τον ορισμό της δομής ουρά προτεραιότητας και δύο απλές υλοποιήσεις σε C, δείτε τις διαφάνειες της Ενότητας 3 από την ιστοσελίδα http://cgi.di.uoa.gr/~k08/lectures.htm .
   - Για τον ορισμό της δομής σωρός, την υλοποίηση του σε C και την υλοποίηση ουράς προτεραιότητας χρησιμοποιώντας σωρό, δείτε τις διαφάνειες της Ενότητας 8 από την ίδια ιστοσελίδα.

   Ευτυχώς δεν θα χρειαστεί να υλοποιήσετε τη δομή σωρός διότι την προσφέρει η standard library της python. Δείτε την ιστοσελίδα https://docs.python.org/2/library/heapq.html .

   Η κλάση `PriorityQueue` που θα ορίσετε θα έχει τα εξής χαρακτηριστικά:
   - `heap`: υλοποιεί τον αντίστοιχο σωρό. Αρχικοποιείται σε `[]`(κενή λίστα).
   - `count`: είναι ένας μετρητής που μετράει πόσα στοιχεία βρίσκονται στην ουρά. Αρχικοποιείται σε `0`.

   Η κλάση `PriorityQueue` θα έχει επίσης τις παρακάτω μεθόδους:

   - `push(pq, item, priority)`: Εισάγει το στοιχείο `item` με προτεραιότητα `priority` στην ουρά `pq`.
   - `pop(pq)`: επιστρέφει το στοιχείο της ουράς προτεραιότητας `pq` με την μικρότερη προτεραιότητα.
   - `isEmpty(pq)`: επιστρέφει `True` αν η ουρά προτεραιότητας `pq` είναι άδεια, αλλιώς επιστρέφει `False`.

- `update(pq,item,priority)`: Αν το `item` περιέχεται στην ουρά προτεραιότητας `pq` με προτεραιότητα μικρότερη ή ίση από `priority`, τότε η μέθοδος δεν κάνει τίποτα. Αν το item περιέχεται στην `pq` με προτεραιότητα μεγαλύτερη από `priority`, τότε αλλάζουμε την προτεραιότητα του `item` ώστε να γίνει priority (η μικρότερη). Αν το `item` δεν περιέχεται στην `pq`, τότε εισάγεται με προτεραιότητα `priority` (εδώ η `update` λειτουργεί όπως η `push`).

Αφού ορίσετε την κλάση `PriorityQueue` και φορτώσετε το αντίστοιχο αρχείο στον διερμηνευτή της python, θα μπορείτε να χρησιμοποιήσετε την κλάση όπως παρακάτω:

```
>>> q.push("task1", 1)
>>> q.push("task1", 2)
>>> q.push("task0", 0)
>>> t=q.pop()
>>> t
'task0'
>>> t=q.pop()
>>> t
'task1'
>>> q.push("task3", 3)
>>> q.push("task3", 4)
>>> q.push("task2", 0)
>>> t=q.pop()
>>> t
'task2'
>>>
```

Αφού ελέγξετε την λειτουργία της κλάσης `PriorityQueue` όπως παραπάνω, να ορίσετε μια συνάρτηση `PQSort` η οποία παίρνει σαν είσοδο μια λίστα ακεραίων και την επιστρέφει ταξινομημένη κατά αύξουσα τάξη χρησιμοποιώντας μια ουρά προτεραιότητας.

Θα πρέπει να παραδώσετε ένα αρχείο `priorityQueue.py` όπου θα περιέχεται όλος ο

σχετικός κώδικας μαζί με σχόλια που επεξηγούν τον κώδικα σας και θα βοηθήσουν τους βαθμολογητές να σας βαθμολογήσουν δίκαια.