Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού (Τμήμα Φοιτητών/τριών με Άρτιο ΑΜ)

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2021-2022

Εργασία 2 (ανακοινώθηκε στις 11 Απριλίου 2022, προθεσμία παράδοσης: 24 Μαϊου 2022, ώρα 23:59)

16% του συνολικού βαθμού στο μάθημα. Άριστα=280 μονάδες (+30 μονάδες μπόνους)

Προσοχή: Πριν διαβάσετε παρακάτω, διαβάστε παρακαλώ προσεκτικά τις οδηγίες υποβολής των ασκήσεων που βρίσκονται στην ιστοσελίδα http://cgi.di.uoa.gr/~k08/homework.html, ειδικά ότι αναφέρεται στο github και τα σχετικά αρχεία.

Απορίες: Αν έχετε απορίες σχετικά με την εργασία, ρωτήστε στο piazza και όχι στέλνοντας e-mail στον διδάσκοντα. Τέτοια e-mails ΔΕΝ θα λαμβάνουν απάντηση.

Κώδικας: Ο κώδικας που παρουσιάσαμε στις διαλέξεις του μαθήματος (Ενότητες 6-12) βρίσκεται στο παρακάτω private repository του classroom του μαθήματος: https://github.com/artioi-k08/2022-ergasia2. Για να συνδεθείτε στο repository αυτό, θα πρέπει να χρησιμοποιήσετε το λινκ https://classroom.github.com/a/TSyiep_E. Μπορείτε να χρησιμοποιήσετε αυτόν τον κώδικα στην εργασία αυτή όμως θα πρέπει να το γράψετε ξεκάθαρα στα σχόλια του προγράμματος σας.

Όταν συνδεθείτε θα μπορείτε να δείτε το προσωπικό σας repository https://github.com/artioi-k08/2022-ergasia1-<github-your-username> στο οποίο θα δουλέψετε για την Εργασία 1.

Μετά τη σύνδεση σας, στο προσωπικό σας repository, θα βρείτε τον κώδικα που καλύπτει τις Ενότητες 6-12 του μαθήματος οργανωμένο σε κατάλληλους φακέλους. Θα βρείτε επίσης και ένα φάκελο solutions-ergasia1 με υπο-φακέλους question1, question2, …, question4, question5 στους οποίους θα πρέπει να

γράψετε τον κώδικα σας για τα 5 ερωτήματα της εργασίας που θα βρείτε παρακάτω. Παρακαλώ τηρείστε ευλαβικά αυτήν την οργάνωση αλλιώς θα χάσετε 20% του συνολικού βαθμού κατά την βαθμολόγηση.

Κύριο πρόγραμμα: Σε όλες τις παρακάτω προγραμματιστικές ασκήσεις θα πρέπει να υλοποιήσετε και ένα κύριο πρόγραμμα (συνάρτηση `main`) το οποίο θα διαβάζει τα δεδομένα εισόδου, θα επιδεικνύει τη λειτουργικότητα της συνάρτησης σας για κατάλληλα επιλεγμένες εισόδους, και θα πείθει τον βαθμολογητή ώστε να σας βαθμολογήσει με τον υψηλότερο δυνατό βαθμό.

1. Να υλοποιήσετε τον αφαιρετικό τύπο δεδομένων `BinaryTree` χρησιμοποιώντας ενότητες (modules) της C και κάνοντας καλή απόκρυψη πληροφορίας. Ο τύπος `BinaryTree` προσφέρει μια διεπαφή με τις παρακάτω πράξεις:
    - `link MakeTree(Item item)`: creates a binary tree with `Root` as the root element and `Item` as value (children should be null).
    - `Item Root(link root)`: returns the element at the root of the tree
    - `Item Parent(link tree, link node)`: returns the element at the parent of a node
    - `Item Sibling (link tree, link node)`: returns the element at the sibling of a node
    - `bool IsInternal (link tree, link node)`: returns true if a node is an internal node
    - `bool IsExternal (link tree, link node)`: returns true if a node is an external node
    - `link InsertLeft (link node, Item item)`: inserts a left child at a given node. If an old child exists, it should be disregarded (and free the corresponding memory). It returns the new child node.
    - `link InsertRight (link node, Item item)`: inserts a right child at a given node. If an old child exists, it should be disregarded (and free the corresponding memory). It returns the new child node.

- `void Attach (link node, link left, link right)`: attaches two trees to be subtrees of an external node
- `void Remove (link tree, link node)`: removes a node with zero or one child which is an external node
- `void Destroy (link tree)`: deletes the tree by freeing all its nodes.
- `void PreOrder (link tree, void (*visit)(Item))`, `InOrder`, `PostOrder`, `LevelOrder`: traverses the tree and visit its nodes in the respective order.
- `int Height (link tree)`: returns the height of the tree.
- `int Size (link tree)`: returns the number of elements in the tree.
- `link Search (link tree, Key key)`: returns the node (pointer) in the tree of a given key.

```
typedef struct STnode* link;

struct STnode { Item item; link l, r; int N; };

typedef int Item;

typedef int Key;

#define NULLitem -1 /* NULLitem is a constant */

#define key(A) (A)

#define less(A, B) (key(A) < key(B))

#define eq(A, B) (!less(A, B) && !less(B, A))
```

Θα πρέπει να χρησιμοποιήσετε structs και pointers για την υλοποίηση κάθε κόμβου του δένδρου (δηλαδή, δεν θα πρέπει να ακολουθήσετε την υλοποίηση με πίνακα της διαφάνειας 36). Προτείνετε να χρησιμοποιήσετε το `typedef` της διαφάνειας 110 της Ενότητας 8. Το δένδρο δεν χρειάζεται να είναι δένδρο

δυαδικής αναζήτησης ή extended (εκτεταμένο). Θα πρέπει να γράψετε και μια συνάρτηση `main` η οποία θα παρουσιάζει τη λειτουργικότητα του αφαιρετικού τύπου δεδομένων. Ένα καλό παράδειγμα μπορεί να είναι η υλοποίηση ενός δένδρου εκφράσεων. Η υλοποίηση σας θα πρέπει να είναι τέτοια ώστε να μπορεί κανείς να κατασκευάσει περισσότερα του ενός δυαδικά δένδρα από τη `main`.

Σημείωση: Έχετε την ελευθερία να ορίσετε με διαφορετικό τρόπο τα prototypes των συναρτήσεων της διεπαφής, όμως πρέπει να κρατήσετε τη φιλοσοφία τους (τι κάνουν οι συναρτήσεις).

(50 μονάδες + 30 μονάδες μπόνους)

2. Στην άσκηση αυτή θα επεκτείνετε την υλοποίηση του αφαιρετικού τύπου δεδομένων Πίνακας Συμβόλων (Symbol Table) με δένδρα δυαδικής αναζήτησης η οποία παρουσιάζεται στις διαφάνειες της Ενότητας 9 (Δένδρα δυαδικής αναζήτησης).

Να οργανώσετε τον κώδικα του αρχείου `symbol-table.txt` που βρίσκεται στο repository σας (φάκελος `symbol-table-code`) σε ένα module της C το οποίο υλοποιεί τον αφαιρετικό τύπο δεδομένων Πίνακας Συμβόλων με την διεπαφή που δίνεται στη σελίδα 5 των σχετικών διαφανειών. Μπορείτε να χρησιμοποιήσετε και να επεκτείνετε τη συνάρτηση `main` της σελίδας 65 των διαφανειών για να επιδείξετε τη λειτουργία του πίνακα συμβόλων με κατάλληλες κλήσεις στις συναρτήσεις που προσφέρονται από τη διεπαφή. Είναι δική σας ευθύνη να επεκτείνετε τον κώδικα που σας δίνεται, ώστε να δουλεύουν πλήρως οι διάφορες συναρτήσεις. Για παράδειγμα, θα χρειαστεί να επεκτείνετε κάποιες συναρτήσεις που δεν ενημερώνουν το πεδίο `N` του `struct STnode` ώστε να το κάνουν.

Να επεκτείνετε την υλοποίηση σας ώστε να μπορεί να χρησιμοποιηθεί στην περίπτωση που θέλουμε ο πίνακας συμβόλων να λειτουργεί σαν κατάλογος βιβλίων της amazon 😊. Υποθέστε ότι σε κάθε `item` τώρα δεν αποθηκεύουμε ένα ακέραιο αλλά τις εξής πληροφορίες: το κλειδί που είναι ο αριθμός ISBN του

βιβλίου (https://www.nlg.gr/static-page/isbn/), ο τίτλος του και τα ονόματα των συγγραφέων.

Παρατηρήστε ότι ο κώδικας που δώσαμε στο μάθημα, και τον οποίο χρησιμοποιήσατε και επεκτείνατε όπως προτείναμε παραπάνω, μας επιτρέπει να έχουμε ένα μόνο πίνακα συμβόλων. Να υλοποιήσετε μια νέα έκδοση του λογισμικού σας που να επιτρέπει τη δημιουργία πολλών πινάκων συμβόλων και την χρήση τους από ένα κύριο πρόγραμμα που θα επιδεικνύει τη λειτουργικότητα του λογισμικού σας (υλοποιήστε το μόνο για την περίπτωση που το `item` είναι ακέραιος).

(50 μονάδες)

3. Να ορίσετε τον αφαιρετικό τύπο δεδομένων `RedBlackTree` με συναρτήσεις `Initialize`, `InsertKey`, `RemoveKey`, `Search`, `PrintElements` που ορίσαμε στην ενότητα διαφανειών 12 (η συνάρτηση `PrintElements` τυπώνει τα κλειδιά του δένδρου σε αύξουσα σειρά). Να υλοποιήσετε αυτές τις συναρτήσεις και να γράψετε κατάλληλη `main` που τις επιδεικνύει. Μπορείτε να υποθέσετε ότι τα κλειδιά του δένδρου είναι ακέραιοι. Δεν μπορείτε να χρησιμοποιήσετε κώδικα από το κεφάλαιο 13.4 του βιβλίου του Sedgewick.

(150 μονάδες)

4. Υποθέστε ότι έχουμε έξι αλγόριθμους Α, Β, Γ, Δ, Ε και Ζ με τις παρακάτω υπολογιστικές πολυπλοκότητες χρόνου.
Α. $2000n$ Β. $50n + \log n$ Γ. $2^{3000\, n\log n}$ Δ. $2^{300\log n}$ Ε. n^4 Ζ. $6n^4 + n$
Να ταξινομήσετε τους αλγόριθμους από τον καλύτερο στον χειρότερο με βάση την υπολογιστική πολυπλοκότητα τους. Να δώσετε λεπτομερώς όσους υπολογισμούς χρειάζονται για να τεκμηριώσετε την απάντηση σας.

(10 μονάδες)

5. Θεωρείστε τον παρακάτω αλγόριθμο ταξινόμησης `BubbleSort`. Πόση είναι η χρονική πολυπλοκότητα χειρίστης περίπτωσης του αλγόριθμου αυτού; Αποδείξτε το με λεπτομέρεια όπως κάναμε στις διαφάνειες της ενότητας 6!

```
void BubbleSort(SortingArray A)
{
    int i; KeyType Temp; Boolean NotDone;
```

```
    do {
       NotDone=false;
       for (i=0; i<n-1; ++i){
          if (A[i]>A[i+1]){
            Temp=A[i]; A[i]=A[i+1]; A[i+1]=Temp;
            NotDone=true;
          }
        }
    } while (NotDone);
}
```

(20 μονάδες)

Καλή Επιτυχία!