Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2023-2024

Εργασία 2

Ανακοινώθηκε την 15η Απριλίου 2024

Προθεσμία: 12 Μαΐου 2024 στις 23:59

1.6 μονάδες στις 10 του βαθμού στο μάθημα (Άριστα=450 μονάδες. Υπάρχουν και 100 μονάδες bonus)

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) και θα χρειαστείτε για την εργασία αυτή βρίσκεται στο παρακάτω private repository του classroom του μαθήματος: https://github.com/artioi-k08/2024-ergasia2. Για να συνδεθείτε στο repository αυτό, θα πρέπει να χρησιμοποιήσετε το λινκ https://classroom.github.com/a/AEoVPUWg.

Όταν συνδεθείτε θα μπορείτε να δείτε το προσωπικό σας repository https://github.com/artioi-k08/2024-ergasia2-<github-your-username> στο οποίο θα δουλέψετε για την Εργασία 2.

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-ergasia2 με υπο-φακέλους question1, question2, question3, question4 στους οποίους θα πρέπει να γράψετε

τον κώδικα ή τις απαντήσεις σας για τις 4 ασκήσεις αυτής της εργασίας που θα βρείτε παρακάτω. Παρακαλώ τηρείστε ευλαβικά αυτήν την οργάνωση αλλιώς θα χάσετε 20% του συνολικού βαθμού κατά την βαθμολόγηση. Για τα θεωρητικά ερωτήματα, οι απαντήσεις πρέπει να είναι σε ένα αρχείο τύπου pdf.

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. Επίσης, τα προγράμματα σας δεν θα πρέπει να έχουν memory errors ή leaks (θα ελεγχθούν με τη χρήση `valgrind`).

1. Θεωρήστε το πρόγραμμα που βρίσκεται στο αρχείο `insertion-sort.c` που βρίσκεται στο repository σας (φάκελος `insertion-sort-code`). Το πρόγραμμα αυτό ταξινομεί ένα πίνακα ακεραίων με τη μέθοδο της ταξινόμησης με εισαγωγή. Ποια είναι η χρονική υπολογιστική πολυπλοκότητα χειρότερης περίπτωσης της συνάρτησης `sort`; Ποια είναι η αντίστοιχη πολυπλοκότητα της `main`; Υποθέστε ότι η κλήση `rand()` εκτελείται σε σταθερό χρόνο. Να εξηγήσετε με λεπτομέρεια τις απαντήσεις σας.

   (10 μονάδες)

2. Υποθέστε ότι έχουμε 10 αλγόριθμους Α, Β, Γ, Δ, Ε, Ζ, Η, Θ, Ι και Κ με τις παρακάτω υπολογιστικές πολυπλοκότητες χρόνου (παραλείπουμε το $O(\ )$).

   Α. $200n$ Β. $50n + logn$ Γ. $2^{3000nlogn}$ Δ. $2^{300logn}$ Ε. n^4

   Ζ. $6n^4 + n$ Η. $(n + n^2)logn\, 2^{n+5}$ Θ. 2^{2^n} Ι. $(2 + n)^{2^{logn}}$ Κ. $2^{2^{n+logn}}$

   Να ταξινομήσετε τους αλγόριθμους από τον καλύτερο στον χειρότερο με βάση την υπολογιστική πολυπλοκότητα τους. Να δώσετε λεπτομερώς όσους μαθηματικούς υπολογισμούς χρειάζονται για να τεκμηριώσετε την απάντηση σας.

   (10 μονάδες)

3. Στην άσκηση αυτή θα επεκτείνετε την υλοποίηση του αφαιρετικού τύπου δεδομένων Δένδρο Δυαδικής Αναζήτησης (Binary Search Tree) που παρουσιάσαμε στην Ενότητα 9.

- Οι συναρτήσεις που θα προσφέρει αρχικά η διεπαφή σας (interface) είναι

```
void BSTinit(int);
int BSTcount();
void BSTinsert(Item);
Item BSTsearch(Key);
void BSTdelete(Item);
Item BSTselect(int);
void BSTsort(void (*visit)(Item));
```

Την υλοποίηση των παραπάνω συναρτήσεων μπορείτε να την πάρετε από τον κώδικα των διαφανειών της Ενότητας 9 (επίσης από την ιστοσελίδα https://cgi.di.uoa.gr/~k08/lectures-code.html) όπου παρουσιάζονται με ονόματα που ξεκινούν με `ST` (symbol table). Επιτρέπεται να αλλάξετε αυτόν τον κώδικα μόνο για τρέχει για τα δεδομένα της εφαρμογής που θα πούμε στην επόμενη παράγραφο. **Δεν** επιτρέπεται να κάνετε αλλαγές στον τρόπο υλοποίησης του δένδρου. Αν χρειαστεί όμως να κάνετε κάποια μικρή διόρθωση σε αλγόριθμους, παρακαλώ κάντε την.

Να γράψετε μια `main` η οποία θα επιδεικνύει τη λειτουργία των παραπάνω συναρτήσεων με δεδομένα από βιβλία από το ηλεκτρονικό κατάστημα https://www.amazon.co.uk/ . Για κάθε βιβλίο, το κλειδί θα είναι ο κωδικός ISBN και τα υπόλοιπα στοιχεία θα είναι συγγραφέας (ή συγγραφείς), τίτλος και έτος έκδοσης (οπότε τα κλειδιά δεν θα είναι τύπου `int` όπως στον κώδικα της Ενότητας 9). Επειδή τα ISBNs είναι μοναδικοί αριθμοί αποτελούμενοι από 13 ψηφία, δεν θα έχετε διπλότυπα στο δένδρο σας. Σαν παράδειγμα, δείτε το πρόσφατο βιβλίο της ομάδας μου 😉. Ο ορισμός τύπου για τα δένδρα δυαδικής αναζήτησης που θα χρησιμοποιήσετε δίνεται στην σελίδα 21 των διαφανειών της Ενότητας 9 (θα χρειαστεί όμως να ορίσετε κατάλληλο interface file `Item.h`).

- Να προσθέσετε στο παραπάνω interface και να υλοποιήσετε τις ακόλουθες συναρτήσεις:
    - `int BSTheight()` η οποία υπολογίζει το ύψος του δένδρου.
    - `void BSTclear()` η οποία θα διασχίζει ένα δυαδικό δέντρο αναζήτησης και θα το «καθαρίζει» ανακυκλώνοντας όλες τις θέσεις μνήμης που καταλαμβάνουν οι κόμβοι του.
    - `int BSTcountLeaves()` η οποία μετράει πόσα φύλλα έχει το δένδρο και επιστρέφει το πλήθος τους.
    - `void BSTlevelOrder()` η οποία τυπώνει τα κλειδιά των κόμβων του δένδρου σε διάταξη επιπέδου. Μπορείτε να υλοποιήσετε αυτή τη λειτουργία σε γραμμικό χρόνο;
    - `Item BSTfloor(Key)` η οποία δέχεται σαν όρισμα ένα κλειδί και επιστρέφει το μεγαλύτερο κλειδί που είναι λεξικογραφικά μικρότερο από το κλειδί που δόθηκε σαν όρισμα (ή την τιμή `NULLitem` αν δεν υπάρχει τέτοιο κλειδί).
    - `Item BSTceiling(Key)` η οποία δέχεται σαν όρισμα ένα κλειδί και επιστρέφει το λεξικογραφικά μικρότερο κλειδί που είναι μεγαλύτερο από το κλειδί που δόθηκε σαν όρισμα (ή την τιμή `NULLitem` αν δεν υπάρχει τέτοιο κλειδί).
    - `void BSTrangeSearch(Low, High)` η οποία δέχεται σαν όρισμα δύο κλειδιά `Low` και `High` και τυπώνει τα στοιχεία του δένδρου με κλειδιά λεξικογραφικά μεγαλύτερα ή ίσα του `Low` και μικρότερα ή ίσα του `High`.
    - `int BSTCountRange(Low, High)` η οποία δέχεται σαν όρισμα δύο κλειδιά `Low` και `High` και επιστρέφει το πλήθος των στοιχείων του δένδρου με κλειδιά λεξικογραφικά μεγαλύτερα ή ίσα του `Low` και μικρότερα ή ίσα του `High`.
    - `int isBST()` η οποία επιστρέφει `1` αν το δοσμένο δένδρο είναι δυαδικό δένδρο αναζήτησης και `0` αν δεν είναι.

  Να επεκτείνετε την `main` σας ώστε να επιδεικνύετε την λειτουργία των παραπάνω συναρτήσεων.
- Παρατηρήστε ότι ο κώδικας που δώσαμε στην Ενότητα 9, και τον οποίο χρησιμοποιήσατε και επεκτείνατε στα παραπάνω ερωτήματα, μας

επιτρέπει να έχουμε ένα μόνο δένδρο δυαδικής αναζήτησης. Να υλοποιήσετε μια νέα έκδοση του λογισμικού των παραπάνω ερωτημάτων (μόνο το πρώτο ερώτημα-bullet) που να επιτρέπει τη δημιουργία πολλών δένδρων και την χρήση τους από ένα κύριο πρόγραμμα που θα επιδεικνύει τη λειτουργικότητα του λογισμικού σας. Να σκεφτείτε μια εφαρμογή στην οποία θα έχει νόημα να έχετε δύο δένδρα δυαδικής αναζήτησης (για παράδειγμα, ένα δένδρο για βιβλία και ένα για συγγραφείς). Σε αυτό το ερώτημα καλείστε να υλοποιήσετε unit tests για την νέα έκδοση του δένδρου δυαδικής αναζήτησης.

Σε όλα τα παραπάνω ερωτήματα, θα πρέπει ο κώδικας των modules που θα αναπτύξετε να κάνει την καλύτερη δυνατή απόκρυψη πληροφορίας.

(20+8*20+40+50=220 μονάδες)

- Να υλοποιήσετε τον αφαιρετικό τύπο δεδομένων Ουρά Προτεραιότητας χρησιμοποιώντας σωρούς μεγίστων όπως δείξαμε στην Ενότητα 8 των διαφανειών. Με βάση τον κώδικα των διαλέξεων, να υλοποιήσετε την παρακάτω διεπαφή που σας επιτρέπει να έχετε παραπάνω από μια ουρές προτεραιότητας:

```
typedef struct priority_queue{
    int Count;
    Item *ItemArray;
}PriorityQueue;
typedef PriorityQueue *PQPointer;
PQPointer QUEUEinit(int maxN);
int QUEUEempty(PQPointer);
void QUEUEput(PQPointer, Item);
Item QUEUEget(PQPointer);
```

Ο τύπος `Item` μπορεί να είναι `int`. Να γράψετε και ένα κύριο πρόγραμμα το οποίο επιδεικνύει τη συμπεριφορά της ουράς προτεραιότητας. Μαζί με το κύριο

πρόγραμμα, καλείστε να υλοποιήσετε unit tests για τις συναρτήσεις της διεπαφής. Μπορείτε να χρησιμοποιήσετε κώδικα από την Ενότητα 8 των διαφανειών.

(50 μονάδες)

4. Να υλοποιήσετε τον αφαιρετικό τύπο δεδομένων QuadTree. Τα quadtrees είναι μια χρήσιμη δομή για διδιάστατα δεδομένα (π.χ., πόλεις του κόσμου και οι συντεταγμένες τους στο χάρτη). Έχετε να κάνετε τα εξής:
    - Διαβάστε σχετικά με quadtrees στο βιβλίο «Foundations of Multidimensional and Metric Data Structures» του Hanan Samet (κεφ. 1.4, σελίδες 28-37). Θα ανακοινώσω στο piazza ένα link από το οποίο θα μπορείτε να κατεβάσετε το pdf του βιβλίου.
    - Υλοποιήστε τη λειτουργικότητα εισαγωγής και αναζήτησης στο quadtree (κεφ. 1.4.1 και 1.4.3 στο βιβλίο) και επιδείξτε τη συμπεριφορά της υλοποίηση σας στα διδιάστατα δεδομένα που αναφέραμε παραπάνω (χρησιμοποιήστε πραγματικά δεδομένα πόλεων και των συντεταγμένων τους από την Wikipedia). Θα πρέπει να γράψετε συναρτήσεις για δύο ειδών αναζητήσεις: (α) Είναι το σημείο *(x,y)* στο δένδρο και ποια είναι η πόλη που αντιστοιχεί σε αυτό; και (β) Επέστρεψε όλα τα σημεία που βρίσκονται στον κύκλο με κέντρο το σημείο *(x,y)* και ακτίνα *r*.
    - **(Bonus)** Υλοποιήστε τη λειτουργικότητα της διαγραφής στο quadtree (κεφ. 1.4.2 στο βιβλίο) και επιδείξτε την στην ίδια εφαρμογή.

    Θα πρέπει να χρησιμοποιήσετε απαραίτητα τους παρακάτω ορισμούς στο αρχείο διεπαφής (interface file), το οποίο επίσης θα περιέχει ό,τι άλλο χρειάζεστε.

```
typedef struct Point {
    float x, y;
    char* name; // City name
} Point;

typedef struct QuadTreeNode {
```

```
    Point* point;
    float x, y;
    float width, height;
    struct QuadTreeNode *ne, *nw, *se, *sw;
} QuadTreeNode;

typedef struct QuadTree {
    QuadTreeNode* root;
} QuadTree;

QuadTreeNode* initNode(float x, float y, float width, float
height);

QuadTree* initQuadTree(float width, float height);

bool QuadTreeInsert(QuadTreeNode* node, Point* point);

Point* QuadTreeSearchPoint(QuadTreeNode* node, float x,
float y);

void QuadTreeSearchWithinRadius(QuadTreeNode* node, float
centerX, float centerY, float radius, Point* points[], int*
count, int maxPoints);
```

Ο τύπος `Point` είναι μία πόλη με συντεταγμένες, ο τύπος είναι ένας κόμβος τους δένδρου και το `QuadTree` είναι ο τύπος που αποθηκεύει τη ρίζα και πληροροφορίες του δένδρου. Η εισαγωγή με την `QuadTreeInsert` επιστρέφει true αν εισάχθηκε σωστά ο νέος κόμβος, διαφορετικά false. Για την αναζήτηση-(α) θα υλοποιήσετε την `QuadTreeSearchPoint` ενώ για την αναζήτηση-(β) την `QuadTreeSearchWithinRadius`. Αναλόγως με τα παραπάνω καλείστε να υλοποιήσετε και το bonus της delete.

Δεν απαιτείται η υλοποίηση unit tests σε αυτή την άσκηση.

(30+30+50+100 (Bonus)=210 μονάδες)

Καλή Επιτυχία!