Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Προηγμένη Σχεδίαση Αλγορίθμων και Δομών Δεδομένων [ΜΥΕ028]
Εαρινό Εξάμηνο 2020

2ο σύνολο ασκήσεων. Ημερομηνία παράδοσης: Πέμπτη 7/5/2020

Παράδοση εργασιών μέσω eCourse

Άσκηση 1 (Αντισταθμιστική ανάλυση δομής FIFO ουράς)

Ο παρακάτω κώδικα υλοποιεί μια FIFO (first-in first-out) ουρά, για την αποθήκευση αντικειμένων ενός αφηρημένου τύπου Item, κάνοντας χρήση δύο LIFO (last-in first-out) στοιβών $S1$ και $S2$. Η ρουτίνα $put(x)$ τοποθετεί ένα νέο στοιχείο x στην ουρά, η ρουτίνα $get()$ επιστρέφει το παλαιότερο στοιχείο της ουράς, ενώ η βοηθητική ρουτίνα $move$ μεταφέρει τα αντικείμενα της $S1$ στη $S2$.

```
void move() {
    while (!S1.isEmpty()) { S2.push(S1.pop()); }
}
void put(Item x) {  S1.push(x); }
Item get() {
    if (S1.isEmpty() && S2.isEmpty()) return null;
    if (S2.isEmpty()) move();
    return S2.pop();
}
```

Οι $S.push/S.pop$ είναι οι λειτουργίες ώθησης/απώθησης σε μια στοίβα S και η $S.isEmpty$ επιστρέφει $true$ αν η S είναι κενή.

Θεωρήστε ότι η ουρά είναι αρχικά κενή (δηλαδή, οι στοίβες $S1$ και $S2$ είναι αρχικά κενές). Υπολογίστε τους χρόνους εκτέλεσης χειρότερης περίπτωσης, καθώς και τους αντισταθμιστικούς χρόνους εκτέλεσης των put και get. Ποιος είναι ο χρόνος εκτέλεσης χειρότερης περίπτωσης μιας αυθαίρετης ακολουθίας n εντολών put και get;

Άσκηση 2 (Υπολογισμός μέγιστης ροής)

Θεωρήστε την ακόλουθη μέθοδο υπολογισμού μιας ροής f δικτύου $G = (V, E)$ με αφετηριακό κόμβο s, τερματικό κόμβο t και συνάρτηση χωρητικότητας ακμών c. Ξεκινάμε με μηδενική ροή $|f| = 0$ και βρίσκουμε σε κάθε επανάληψη μια αυξητική διαδρομή από τον s στον t χρησιμοποιώντας μόνο της ακμές του αρχικού δικτύου G. Δηλαδή ο αλγόριθμος δεν λαμβάνει υπόψη το υπολειπόμενο δίκτυο και εξετάζει μόνο τις ακμές $(u, v) \in E$ για τις οποίες $f(u, v) < c(u, v)$. Έχουμε δει στο μάθημα ότι αυτή η μέθοδος δε βρίσκει πάντα τη μέγιστη ροή f^* ενός δικτύου. Δείξτε με ένα κατάλληλο παράδειγμα ότι ο λόγος $|f^*|/|f|$ μπορεί να είναι αυθαίρετα μεγάλος, δηλαδή συνάρτηση του αριθμού των κόμβων του δικτύου.

Υπόδειξη: Γενικεύστε το παράδειγμα που είδαμε στις διαφάνειες #27-#30 της παρουσίασης MaxFlow1.

Άσκηση 3 (Υπολογισμός ανεξάρτητων μονοπατιών)

Έστω $G = (V, E)$ ένα μη κατευθυνόμενο γράφημα με αφετηριακό κόμβο s και τερματικό κόμβο t (χωρίς βάρη στις ακμές). Λέμε ότι δύο μονοπάτια P και Q είναι δύο *ανεξάρτητα* s-t αν συνδέουν τον s με τον t και δεν έχουν κοινές ακμές.

α. Περιγράψτε ένα αποδοτικό αλγόριθμο ο οποίος απλά ελέγχει αν το G έχει δύο ανεξάρτητα s-t μονοπάτια P και Q (χωρίς να χρειάζεται να τα κατασκευάσει).

β. Τροποποιήστε τον αλγόριθμο του 3α έτσι ώστε να επιστρέφει τα μονοπάτια P και Q.

Ποιος είναι ο χρόνος εκτέλεσης του αλγόριθμου σας;

Άσκηση 4 (Αντισταθμιστική ανάλυση δομής διατεταγμένων συνόλων)

Θέλουμε να αναλύσουμε την απόδοση μιας δομής δεδομένων που χειρίζεται διατεταγμένες λίστες ακεραίων από το σύνολο $\{1,2,3,\ldots,n\}$ και υποστηρίζει την πράξη της συγχώνευσης δύο λιστών. Αρχικά κάθε ακέραιος αποτελεί μια ξεχωριστή λίστα. Η πράξη συγχώνευσης, $merge(A, B)$, δημιουργεί μια νέα λίστα C με τα στοιχεία των λιστών A και B. Μετά τη συγχώνευση οι δύο λίστες A και B έχουν καταστραφεί, δηλαδή, παύουν να υπάρχουν ως ξεχωριστές λίστες. Για παράδειγμα, αν έχουμε $A = \langle 2, 5, 8, 12, 21\rangle$ και $B = \langle 1, 15, 18, 22, 34, 55\rangle$ τότε η συγχώνευση τους δίνει μια νέα λίστα $C = \langle 1, 2, 5, 8, 12, 15, 18, 21, 22, 34, 55\rangle$.

Για να λύσουμε το παραπάνω πρόβλημα αποθηκεύουμε κάθε λίστα σε ένα ισορροπημένο δένδρο δυαδικής αναζήτησης (π.χ. AVL ή κοκκινόμαυρο), το οποίο υποστηρίζει τις παρακάτω εντολές:

- $size(A)$: Επιστρέφει το πλήθος των στοιχείων της λίστας A.
- $min(A)$: Επιστρέφει το ελάχιστο στοιχείο της λίστας A.
- $search(A, x)$: Βρίσκει τη θέση του ακέραιου x στη λίστα A. Δηλαδή, επιστρέφει τη θέση του μεγαλύτερου ακέραιου $y \in A$, τέτοιου ώστε $y \leq x$. (Αν το x βρίσκεται στη λίστα A, τότε $y = x$.)
- $split(A, x)$: Χωρίζει τη λίστα A στη θέση του ακέραιου x. Επιστρέφει δύο νέες λίστες B και C όπου η B περιέχει τους ακέραιους της A που είναι $\leq x$ και η C περιέχει τους ακέραιους της A που είναι $> x$.
- $join(A, B)$: Προϋποθέτει ότι όλοι οι ακέραιοι της λίστας A είναι μικρότεροι από τους ακέραιους της λίστας B. Επιστρέφει μια νέα λίστα C με την ένωση των A και B.

Υποθέστε ότι οι $size$ και min εκτελούνται σε σταθερό χρόνο χειρότερης περίπτωσης και ότι όλες οι υπόλοιπες εντολές εκτελούνται σε $O(\log n)$ χρόνο χειρότερης περίπτωσης.

α) Δείξτε πως υλοποιείται η πράξη $merge(A, B)$ συνδυάζοντας τις παραπάνω πράξεις. Ποιος είναι ο χρόνος εκτέλεσης στη χειρότερη περίπτωση;

β) Τώρα θέλουμε να δείξουμε ότι ο αντισταθμιστικός χρόνος της συγχώνευσης είναι $O(\log^2 n)$, ξεκινώντας από n λίστες με ένα ακέραιο η κάθε μια. Για το σκοπό αυτό θα ορίσουμε το δυναμικό $\varphi(a_j)$ ενός ακέραιου a_j ως εξής. Έστω ότι ο a_j ανήκει στη λίστα

$A = \langle a_1, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_k \rangle$. Αν το a_j είναι το μοναδικό στοιχείο της λίστας τότε $\varphi(a_j) = 2\log n$. Διαφορετικά έχουμε τις ακόλουθες περιπτώσεις. Αν το a_j είναι το πρώτο στοιχείο της λίστας ($j = 1$) τότε $\varphi(a_j) = \log n + \log(a_{j+1} - a_j)$. Αν το a_j είναι το τελευταίο στοιχείο της λίστας ($j = k$) τότε $\varphi(a_j) = \log(a_j - a_{j-1}) + \log n$. Διαφορετικά ($1 < j < k$) έχουμε $\varphi(a_j) = \log(a_j - a_{j-1}) + \log(a_{j+1} - a_j)$.

Το ολικό δυναμικό Φ της δομής είναι το άθροισμα των επιμέρους δυναμικών $\varphi(a_j)$. Αρχικά, αφού κάθε λίστα έχει μόνο ένα στοιχείο, έχουμε $\Phi = 2n\log n$. Στη συνέχεια κάθε πράξη συγχώνευσης έχει ως αποτέλεσμα τη μείωση του δυναμικού.

Με τη βοήθεια της παραπάνω συνάρτησης δείξτε ότι σε οποιαδήποτε ακολουθία m συγχωνεύσεων θα γίνουν το πολύ $O(m\log n)$ πράξεις αναζήτησης, διαχωρισμού και ένωσης. Καταλήξτε στο συμπέρασμα ότι ο αντισταθμιστικός χρόνος της συγχώνευσης είναι $O(\log^2 n)$.

Υπόδειξη: Μπορείτε να δείξετε ότι η εκτέλεση δύο διαδοχικών βημάτων της διαδικασίας $merge(A, B)$ *ελαττώνουν τη συνάρτηση δυναμικού* Φ *τουλάχιστον κατά μία μονάδα. Έστω ότι η υπολίστα* $\langle u, \ldots, v \rangle$ *της* B *εισάγεται μεταξύ των στοιχείων* x *και* y *της* A*. Αυτό σημαίνει ότι* $x < u < v < y$ *και άρα ισχύει* $u \leq (x + y)/2$ *ή* $v \geq (x + y)/2$ *ή και τα δύο.*