Α΄ Εργασία Εξαμήνου: Δομές Δεδομένων

Γ. Πρόκο
Π16119, Τμήμα Πληροφορικής
Πανεπιστήμιο Πειραιώς

Προθεσμία Παράδοσης:
20 Μαΐου 2022

Περίληψη

Σκοπός: Απόπειρα Βελτίωσης Βαθμολογίας Μαθήματος (Αναβαθμολόγηση).

Η παρούσα εργασία, αποτελεί υλοποίηση της α΄ υπολογιστικής εργασίας του μαθήματος **Δομές Δεδομένων**, του 1ου εξαμήνου σπουδών του τμήματος Πληροφορικής, Πανεπιστήμιο Πειραιώς (ακαδημαϊκό έτος 2021 - 2022, υπό την επίβλεψη του καθηγητή Χ. Κωνσταντόπουλου).

Στην εργασία αυτή, καλούμαστε να χρησιμοποιήσουμε την γλώσσα προγραμματισμού C/C++ προκειμένου να επιλύσουμε τα επιθυμητά ερωτήματα. Η εργασία αποτελείται απο 2 ερωτήματα, των οποίων η εκφώνηση παρατίθεται, καθώς και επιμέρους θέματα υλοποίησης που καλούμαστε να χρησιμοποιήσουμε.

Περιεχόμενα

1 Ερώτημα 1

1.1 Εκφώνηση

Έστω μια γραμμική λίστα L. Στα πλαίσια της εργασίας θα πρέπει να χρησιμοποιήσετε την μονά - συνδεδεμένη αναπαράσταση για όλες τις γραμμικές λίστες που θα χρειαστείτε. Για λεπτομέρειες δείτε το Κεφ. 3.4 του βιβλίου "Δομές Δεδομένων, Αλγόριθμοι και Εφαρμογές στη C++". Δεν επιτρέπεται η χρήση των έτοιμων δομών δεδομένων που προσφέρει η C++ (C++ Containers).

Τα δεδομένα σε κάθε κόμβο της λίστας L αποτελούνται από έναν ακέραιο αριθμό. Σκοπός του προγράμματος C++ που θα φτιάξετε είναι να δημιουργήσετε μια νέα λίστα histogram στην οποία θα αποθηκεύεται πόσες φορές εμφανίζεται κάθε αριθμός στην λίστα L. Τα στοιχεία της λίστας histogram θα πρέπει να δημιουργούνται δυναμικά κάθε φορά που συναντάμε έναν αριθμό στην λίστα L, για πρώτη φορά και να είναι ταξινομημένα κατά αύξουσα σειρά με βάση τους αριθμούς που περιέχει η λίστα L.

1.2 Θέματα Υλοποίησης

Παρακάτω παρατίθενται ειδικά θέματα υλοποίησης καθώς και ορισμένες χρήσιμες κατευθύνσεις.

- Το μέγεθος της λίστας L (δηλαδή το πλήθος των αριθμών που θα περιέχει) θα το διαβάζετε κατά την ώρα εκτέλεσης του προγράμματος από τον χρήστη.
- Στην συνέχεια το πρόγραμμα σας θα πρέπει να κατασκευάσει την λίστα L. Για κάθε κόμβο θα πρέπει να οριστεί ο αριθμός που θα περιέχει ως δεδομένο. Για τον σκοπό αυτό μπορεί να γίνει χρήση γεννήτριας τυχαίων αριθμών. Η C++ προσφέρει ένα σύνολο κλάσεων και μεθόδων για τον σκοπό αυτό.

1.3 Τεκμηρίωση Υλοποίησης

Στην παρούσα υποενότητα, θα παρουσιάσουμε την ακολουθία εκτέλεσης του προγράμματος προς επίλυση του ερωτήματος, καθώς και την συλλογιστική που ακολουθείται απο πίσω.

1.3.1 Είσοδος Μεγέθους n της Γραμμικής Λίστας L απο τον χρήστη

Κατά την έναρξη εκτέλεσης του προγράμματος, το πρόγραμμα ζητάει από τον χρήστη ως είσοδο, έναν θετικό ακεραίο αριθμό, $(n \geq 1)$ απο τον χρήστη, προκειμένου να κατασκευαστεί η γραμμική λίστα L όπως και η συσχετιζόμενη γραμμική λίστα H.

Σημειώνεται ότι ο ακέραιος αριθμός n, χαρακτηρίζει το μέγιστο εφικτό πλήθος κόμβων που δύναται να εμπεριέχει η γραμμική λίστα L σε κάθε δεδομένη χρονική στιγμή εκτέλεσης του προγράμματος. Κατά αντιστοιχία, δοθέντος ότι δυνειτικά ως είσοδος μπορεί να δοθεί στιγμιότυπο, I, ακολουθίας n διαφορετικών τιμών x (τιμη x που εμπεριέχει κάποιος κόμβος), θα πρέπει και το μέγεθος της γραμμικής λίστας H του ιστορογράμματος, να ακολυθεί την ίδια συλλογιστική.

Συνοψίζοντας, απαιτούμε, το μέγεθος της λίστας L και της λίστας H, να είναι κατέλάχιστον 1 και κατα μέγιστο n.

1.3.2 Γραμμική Λίστα L

Για την υλοποίηση της βασικής γραμμικής λίστας L, χρειάζεται να υλοποιήσουμε μια βοηθητική τάξη L_Node. Η βοηθητική αυτή τάξη αναπαριστά τον κόμβο της γραμμικής λίστας L και εμπεριέχει την ακέραια τιμή του

κόμβου. Επιπρόσθετα, εμπεριέχει δείκτη στον επόμενο κόμβο στον οποίο θα δείχνει.

Η βασική τάξη L_List, εμπεριέχει τα εξής βασικά δεδομένα, καθώς και λειτουργίες.

Δεδομένα:

- Δείκτη προς τον αρχικό κόμβο της γραμμικής λίστας
- Δείκτη προς τον τελικό κόμβο της γραμμικής λίστας
- Πλήθος Στοιχείων Λίστας
- Μέγιστο Πλήθος (Μέγιστη χωρητικότητα) Λίστας

Λειτουργικότητα (Συναρτήσεις):

- Constructor: $O(1)$
  Συνάρτηση η οποία ενεργοποιείται κατά την δημιουργία της τάξης. Κατά την διάρκεια εκτέλεσης αυτής της συνάρτησης, δημιουργούμε έναν κενό κόμβο στην Heap Memory και συσχετίζουμε κατάλληλα, ώστε ο δείκτης προς τον αρχικό κόμβο της γραμμικής λίστας, καθώς και ο δείκτης προς τον τελικό κόμβο της γραμμικής λίστας να δείχνουν προς τον καινούριο αυτόν αρχικό κόμβο. Ταυτόχρονα, αρχικοποιούμε το πλήθος στοιχείων της λίστας να είναι ίσο με το 0, καθώς και αρχικοποιούμε το μέγιστο πλήθος της λίστας να ισοδυναμεί με την αποδεκτή είσοδο του χρήστη.

- Destructor: $O(n)$
  Η συνάρτηση η οποία ενεργοποιείται είτε κατά την λήξη του πεδίου ορατότητας της τάξης στο πρόγραμμα, είτε κατά την χειροκίνητη καταστροφή του αντικειμένου απο το πρόγραμμα. Με την χρήση ενός βοηθητικού δείκτη, η συνάρτηση αυτή απελευθερώνει τους πόρους μνήμης που καταλαμβάνουν οι κόμβοι της λίστας και εν συνεχεία μηδενίζει τους δείκτες που δείχνουν στην αρχή και στο τέλος της λίστας.

- Εισαγωγή Τιμής x: $O(1)$
  Μέσω της συνάρτησης εισαγωγής τιμής στην λίστα, γίνεται προσθήκη του στοιχείου x στο τέλος της λίστας. Κατασκευάζεται ένας καινούριος κόμβος λίστας, αρχικοποιείται με την τιμή x και γίνεται η προσθήκη του μετά τον δείκτη που δείχνει στον τελικό κόμβο της λίστας. Με την χρήση αυτού του επιπρόσθετου δείκτη, είμαστε σε θέση να κάνουμε εισαγωγή κόμβου σε $O(1)$ χρόνο.

  Επιπρόσθετα, προκειμένου να μην υπερβούμε το μέγιστο πλήθος στοιχείων που μπορούμε να εισαγάγουμε στην λίστα, για κάθε προσθήκη μελετάμε εαν το πλήθος των στοιχείων της λίστας είναι μικρότερο απο το μέγιστο. Εαν ισχύει η ισότητα, τότε αφαιρούμε το πρώτο στοιχείο της λίστας (σε μια όμοια με $F.I.F.O.$ συλλογιστική) και εν συνεχεία προσθέτουμε τον καινούριο κόμβο. Ας δούμε παρόλα-αυτά, δύο ειδικές περιπτώσεις που καλούμαστε να καλύψουμε κατά την εισαγωγή μιας καινούριας τιμής στην γραμμική μας λίστα.

  Ειδική Περίπτωση 1: Κατά την περίπτωση που το πλήθος των στοιχείων της λίστας είναι ίσο με το 0, η κατάσταση της λίστας περιγράφεται ως 1 κόμβος (ο πρώτος κόμβος κατασκευής) ο οποίος ωστόσο δεν έχει αρχικοποιηθεί με καμία τιμή μέχρι στιγμής. Το γεγονός αυτό υποδηλώνει ότι θα πρέπει να αρχικοποιήσουμε τον πρώτο κόμβο με την τιμή x, και εν συνεχεία να αυξήσουμε το πλήθος των στοιχείων κατά 1 (=1).

  Ειδική Περίπτωση 2: Κατά την ειδική περίπτωση που το μέγιστο πλήθος στοιχείων που δύναται να έχει η γραμμική λίστα Λ, ισούται με 1, τότε για κάθε καινούρια προσθήκη στοιχείων x, το μόνο που καλούμαστε να κάνουμε είναι αντικατάσταση της τιμής του 1-του κόμβου με την καινούρια τιμή.

- Εξαγωγή Πρώτου Κόμβου Λίστας (Κεφαλής): $O(1)$
  Κατά την κλήση της συνάρτησης αυτής, μέσω της χρήσης ενός βοηθητικού δείκτη, το μόνο που καλούμαστε να κάνουμε είναι να απελευθερώσουμε τους πόρους μνήμης που καταλαμβάνει ο πρώτος κόμβος της λίστας μας, και να τροποποιήσουμε την κεφαλή της λίστας να δείχνει στον αμέσως επόμενο κόμβο. Παράλληλα, οφείλουμε να μειώσουμε το πλήθος των στοιχείων της λίστας κατά 1.

- Ταυτότητα Λίστας: $O(n)$
  Κατά την κλήση της συνάρτησης αυτής, μέσω βοηθητικού δείκτη, γίνεται προσπέλαση όλων των κόμβων της λίστας και εμφάνιση των στοιχείων τους.

1.3.3 Γραμμική Λίστα H

Η γραμμική λίστα H, συγκεντρώνει τον βασικό όγκο της επιθυμητής λειτουργικότητας και προσοχής μας, λόγω των επιμέρους ιδιαιτεροτήτων ως προς την λειτουργικότητα που απαιτούνται, και ως εκ τούτου θα προσπαθήσουμε να τεκμηριώσουμε την όποια συλλογιστική πίσω απο την υλοποίηση της με τον βέλτιστο έφικτό τρόπο. Αφενός θα ξεκινήσουμε την ανάλυση μας με τα δεδομένα που εμπεριέχει η λίστα, και στην συνέχεια θα ακολουθήσει η τεκμηρίωση και των επιμέρους συναρτήσεων της. Παρόλαυτα, πριν την επιθυμητή ανάλυση, παραθέτουμε την βοηθητική τάξη κόμβου της γραμμικής λίστας.

Βοηθητική Τάξη Κόμβου: H_Node

Η βοηθητική τάξη κόμβου, είναι η τάξη που θα αναπαριστά τον εκάστοτε κόμβο της γραμμικής λίστας H. Βάση των απαιτήσεων υλοποίησης, η τάξη αυτή εμπεριέχει ως δεδομένα: (1) ακέραια τιμή η οποία αναπαριστά την βασική πληροφορία που αποθηκεύει ο κόμβος, (2) ακέραια τιμή η οποία αναπαριστά τον πληθάριθμο της βασικής πληροφορίας (δηλαδή πόσες φορές έχουμε συναντήσει την τιμή στην συσχετιζόμενη γραμμική λίστα L), καθώς και (3) δείκτη προς τον επόμενο κόμβο της γραμμικής λίστας H.

Δεδομένα Γραμμικής Λίστας H:

Η γραμμική λίστα H, εμπεριέχει τα εξής δεδομένα:

- Δείκτη προς τον αρχικό κόμβο της γραμμικής λίστας
- Πλήθος στοιχείων της γραμμικής λίστας
- Μέγιστη Δυνατή Χωρητικότητας της γραμμικής λίστας

Εν συνεχεία θα μελετήσουμε τις επιμέρους συναρτήσεις της τάξης που υλοποιούν την λειτουργικότητα της γραμμικής λίστα H, καθώς και τον χρόνο εκτέλεσης τους.

Constructor: $O(1)$

Όμοια με την γραμμική λίστα L, η συνάρτηση αυτή ενεργοποιείται κατά την δημιουργία της τάξης. Κατά την διάρκεια εκτέλεσης αυτής της συνάρτησης, δημιουργούμε έναν κενό κόμβο στην Heap Memory και συσχετίζουμε κατάλληλα ώστε ο δείκτης προς τον αρχικό κόμβο της γραμμικής λίστας, να δείχνει προς τον καινούριο αυτόν αρχικό κόμβο. Ταυτόχρονα, αρχικοποιούμε το πλήθος στοιχείων της λίστας να είναι ίσο με το 0, καθώς και αρχικοποιούμε το μέγιστο πλήθος της λίστας να ισοδυναμεί με την αποδεκτή είσοδο του χρήστη.

Destructor: $O(n)$

Η συνάρτηση η οποία ενεργοποιείται είτε κατά την λήξη του πεδίου ορατότητας της τάξης στο πρόγραμμα, είτε κατά την χειροκίνητη καταστροφή του αντικειμένου απο το πρόγραμμα. Με την χρήση ενός βοηθητικού

δείκτη, η συνάρτηση αυτή απελευθερώνει τους πόρους μνήμης που καταλαμβάνουν οι κόμβοι της λίστας και εν συνεχεία μηδενίζει τον δείκτη που δείχνει στην αρχή της λίστας.

Εισαγωγή στοιχείου x κατά αύξουσα σειρά:

Κατά την εισαγωγή οποιουδηποτε στοιχείου που παρατίθεται ως είσοδος, οφείλουμε να διατηρούμε ανά πάσα στιγμή την σχέση ισοδυναμίας των στοιχείων (ολική διάταξη - αύξουσα σειρά). Ταυτόχρονα, οφείλουμε να είμαστε προσεκτικοί ως προς το γεγονός ότι πρέπει να διαχωρίσουμε πότε χρειάζεται να επιτέλεσουμε καινούρια εισαγωγή στοιχείου (στοιχείο το οποίο δεν υπάρχει στην γραμμική μας λίστα H) καθώς και πότε πρέπει να κάνουμε ενημέρωση πληθαρίθμου στοιχείου. Για αυτόν τον λόγο, κατά την κλήση αυτής της συνάρτησης εισαγωγής στοιχείου, διακρίνουμε 4 περιπτώσεις εισαγωγής, για τις οποίες και υλοποιούμε 4 διαφορετικές βοηθητικές συναρτήσεις.

Περίπτωση 0: Αρχικοποίηση του πρώτου κόμβου της λίστας - $O(1)$

Η περίπτωση αυτή ανακύπτει κατά το πρώτο στοιχείο που βλέπουμε ως είσοδο, δηλαδή όταν το πλήθος των στοιχείων της λίστας είναι ίσο με το 0. Σε αυτήν την περίπτωση, αρχικοποιούμε την τιμή του αρχικού κόμβου της λίστας με την τιμή x, αρχικοποιούμε τον πληθάριθμο του αριθμού ως 1, και αυξάνουμε το πλήθος των στοιχείων της λίστας απο 0 σε 1.

Περίπτωση 1: Μέγιστο πλήθος στοιχείων $= 1$ - $O(1)$

Η περίπτωση αυτή ανακύπτει όταν το μέγιστο πλήθος στοιχείων που μπορούμε να αποθηκεύσουμε στην γραμμική λίστα, ισούται με 1. Σε αυτήν την περίπτωση καλούμαστε να διαχωρίσουμε δύο υποπεριπτώσεις: (α) Όταν η τιμή x ισούται με την τιμή του αρχικού κόμβου της λίστας, καθώς και όταν (β) η τιμή x είναι διαφορετική απο την τιμή του αρχικού κόμβου της λίστας.

Στην πρώτη περίπτωση, το μόνο που κάνουμε είναι να αυξήσουμε τον πληθάριθμο του αρχικού κόμβου της λίστας κατά 1, ενώ στην δεύτερη περίπτωση, επανα-αρχικοποιούμε τα δεδομένα του αρχικού κόμβου κατάλληλα.
Περίπτωση 2: Βασική Περίπτωση Εισαγωγής (Πλήθος Στοιχείων Λίστας $<$ Μεγίστου Πλήθους Στοιχείων Λίστας) - $O(n)$

Η περίπτωση αυτή συγκαταλέγει την βασική περίπτωση εισαγωγής κάποιου δεδομένου στην γραμμική λίστα H ιστορογράμματος. Το πλήθος των στοιχείων που εμπεριέχει η λίστα είναι γνησίως μικρότερο του μεγίστου, και ως εκ τούτου, είναι εφικτή η εισαγωγή καινούριου κόμβου στην κατάλληλη θέση στην λίστα. Παράλληλα ωστόσο, δεν θα πρέπει να ξεχνάμε ότι το στοιχείο x μπορεί να υπάρχει ήδη στην γραμμική λίστα, και ως εκ τούτου δεν θα πρέπει να κατασκευάσουμε καινούριο κόμβο, αλλά θα πρέπει να ενημερώσουμε κατάλληλα τα δεδομένα του υπάρχοντος κόμβου.

Για την λειτουργικότητα αυτής της υποπερίπτωσης, καλούμαστε να κάνουμε χρήση 2 βοηθητικών δεικτών προς κόμβους. Ο ένας βοηθητικός δείκτης, θα δείχνει κάθε φορά προς τον κόμβο που προσπελαύνουμε την i-οστή χρονική στιγμή, ενώ ο δεύτερος βοηθητικός δείκτης (ακόλουθος), θα δείχνει κάθε φορά προς τον κόμβο που προσπελάσαμε στην $(i-1)$-οστή χρονική στιγμή. Αρχικοποιούμε αρχικά τον πρώτο δείκτη να δείχνει στον πρώτο κόμβο της λίστας ενώ τον δεύτερο να μην δείχνει πουθενά.

Προσπελαύνουμε τους κόμβους της γραμμικής μας λίστας, εως ότου μια απο τις δύο συνθήκες τερματισμού ενεργοποιηθούν: είτε ο βασικός μας βοηθητικός δείκτης δείχνει στο 0, δηλαδή έχουμε προσπελάσει όλους τους κόμβους της γραμμικής μας λίστας, είτε η τιμή του κόμβου που δείχνει ο βασικός μας βοηθητικός δείκτης είναι μεγαλύτερη της δοθείσας τιμής x. Η δεύτερη αυτή υπο-περίπτωση υποδηλώνει ότι ο κόμβος στον οποίο

δείχνει ο δείκτης ακόλουθος, εμπεριέχει τιμή $k \leq x$.

Πρωτού μελετήσουμε ωστόσο τις δύο συνθήκες τερματισμού που αναφέραμε προηγουμένως, καλούμαστε να μελετήσουμε μια ειδική περίπτωση που μπορεί να προκύψει και δεν έχουμε σχολιάσει. Η περίπτωση αυτή προκύπτει ότι ο βασικός βοηθητικός μας δείκτης, δείχνει προς τον αρχικό μας κόμβο και η τιμή του κόμβου είναι $k \geq x$. Σε αυτήν την περίπτωση, δράμε ως εξής:

Εαν ο υφίσταται κόμβος στον οποίο δείχνει ο βασικός βοηθητικός δείκτης και η τιμή του ισούται με x, αυξάνουμε τον πληθάριθμο της τιμής του βασικού βοηθητικού δείκτη κατά 1. Αν ο βασικός βοηθητικός δείκτης δείχνει προς τον πρώτο κόμβο της γραμμικής λίστας, αυτό υποδηλώνει ότι η τιμή του πρώτου κόμβου είναι $\geq q$, οπότε και δημιουργούμε έναν καινούριο κόμβο, θέτουμε τις κατάλληλες τιμές και τον εισαγάγουμε πρίν τον πρώτο κόμβο της γραμμικής λίστας. Εν συνεχεία τροποποιούμε κατάλληλα τους δείκτες προς τους επόμενους κόμβους, ώστε ο καινούριος κόμβος να δείχνει στον πρώτο κόμβο της λίστας, και ο δείκτης προς τον πρώτο κόμβο της λίστας να δείχνει στον καινούριο κόμβο. Παράλληλα αυξάνουμε το πλήθος των στοιχείων της γραμμικής λίστας κατά 1.

Αν αυτή η ειδική περίπτωση δεν έχει ενεργοποιηθεί, τότε αυτό υποδηλώνει ότι είτε πρέπει να τοποθετήσουμε κάποιον καινούριο κόμβο, μετά απο κάποιον υπάρχον κόμβο $k \in [2, m]$, όπου m το πλήθος των κόμβων που έχει η λίστα την παρούσα χρονική στιγμή, είτε θα χρειαστεί να τροποποιήσουμε τα δεδομένα κάποιου υπάρχοντος κόμβου ανάλογα.

Παρατηρούμε ότι την στιγμή που ενεργοποιείται μια απο τις συνθήκες τερματισμού της βασικής επαναληπτικής διαδικασία προσπέλασης κόμβων, είτε η τιμή του κόμβου $(i-1)$ είναι $=$ με την τιμή x, είτε η τιμή του κόμβου $(i-1)$ είναι $<$ απο την τιμή x. Σε κάθε περίπτωση, η τιμή του κόμβου i είναι $>$ απο την τιμή x.

Ελέγχουμε κατάλληλα αυτές τις δύο υπο-περιπτώσεις. Αν η τιμή του κόμβου $(i-1)$ είναι ίση με x αυξάνουμε τον πληθάριθμο τιμής του κόμβου, ενώ σε αντίθετη περίπτωση, δημιουργούμε έναν καινούριο κόμβο (αρχικοποιόντας με κατάλληλες τιμές), του οποίου ο επόμενος κόμβος θα δείχνει όπου δείχνει ο δείκτης i, ενώ ο επόμενος κόμβος του κόμβου $(i-1)$ θα είναι ο καινούριος κόμβος. Στην δεύτερη αυτή περίπτωση και μόνο, αυξάνουμε το πλήθος των στοιχείων της λίστας κατά 1.

Αξιοσημείωτο αναφοράς είναι επίσης το γεγονός ότι η συλλογιστική αυτή καλύπτει και την περίπτωση κατά την οποία πρέπει να γίνει εισαγωγή καινούριου κόμβου στο τέλος της υπάρχουσας γραμμικής λίστας.

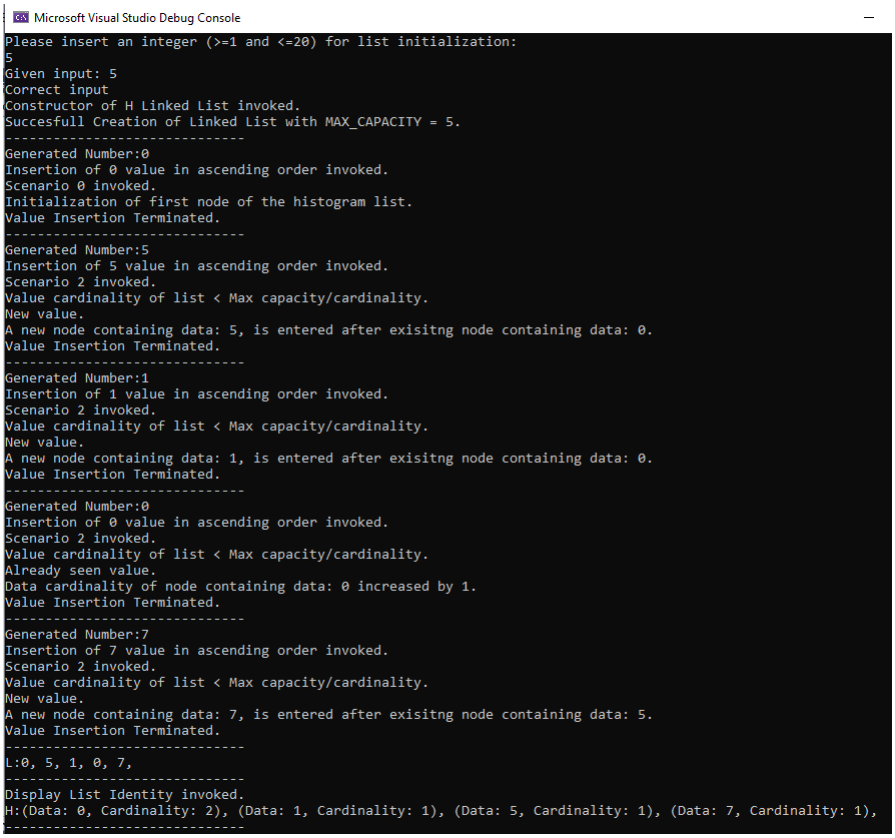
Περίπτωση 3: Πλήθος Στοιχείων Λίστας = Μεγίστο Πλήθος Στοιχείων Λίστας) - $O(n)$

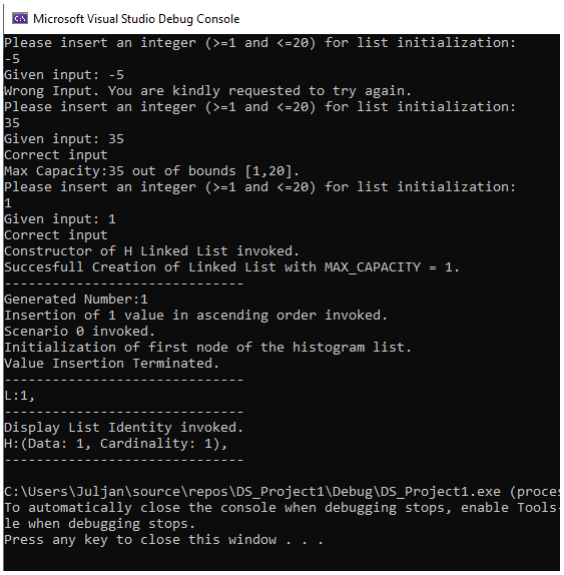
Σε αυτή την περίπτωση, χρειάζεται να είμαστε προσεκτικοί ως προς το γεγονός εάν επιθυμούμε να προσθέσουμε έναν καινούριο κόμβο, ή απλά να ενημερώσουμε κάποιον υπάρχον κόμβο. Στην περίπτωση που θέλουμε να ενημερώσουμε έναν υπάρχον κόμβο, δηλαδή η τιμή x υπάρχει σε κάποιον κόμβο της λίστας μας, τότε το μόνο που καλούμαστε να κάνουμε, είναι με την χρήση ενός βοηθητικού δείκτη να κάνουμε γραμμική αναζήτηση της υπάρχουσας λίστας και να ελέγχξουμε εάν υπάρχει ήδη η τιμή x σε κάποιον κόμβο.

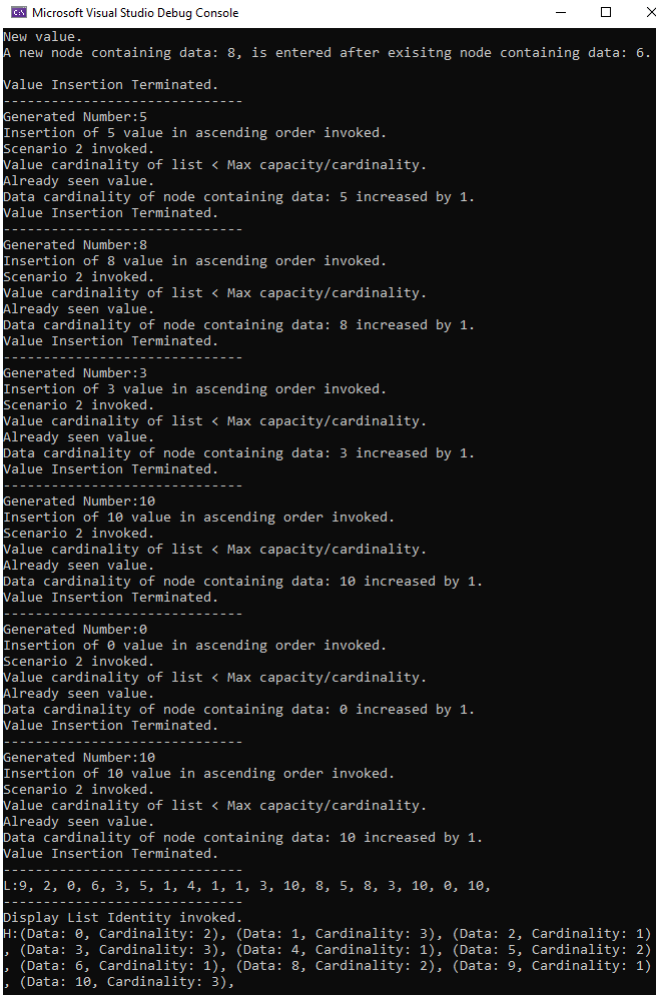
Σε περίπτωση που η περίπτωση αυτή αποτύχει, διαγράφουμε το πρώτο στοιχείο της λίστας κατάλληλα (όμοια με την γραμμική λίστα L), και ξανακαλούμε την συνάρτηση εισαγωγής στοιχείου σε αύξουσα σειρά της τάξης με την τιμή x, η οποία θα αποφανθεί κατάλληλα ποιο απο τα υπόλοιπα 3 σενάρια θα ενεργοποιηθεί.

1.4 Στιγμιότυπα Εκτέλεσης Προγράμματος

Παρακάτω, παρατίθενται στιγμιότυπα εκτέλεσης του προγράμματος συσχετιζόμενα με τις κατάλληλες αναφορές.


Microsoft Visual Studio Debug Console
Please insert an integer (>=1 and <=20) for list initialization:
5
Given input: 5
Correct input
Constructor of H Linked List invoked.
Succesfull Creation of Linked List with MAX_CAPACITY = 5.
Generated Number:0
Insertion of 0 value in ascending order invoked.
Scenario 0 invoked.
Initialization of first node of the histogram list.
Value Insertion Terminated.
Generated Number:5
Insertion of 5 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
New value.
A new node containing data: 5, is entered after exisitng node containing data: 0.
Value Insertion Terminated.
Generated Number:1
Insertion of 1 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
New value.
A new node containing data: 1, is entered after exisitng node containing data: 0.
Value Insertion Terminated.
Generated Number:0
Insertion of 0 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 0 increased by 1.
Value Insertion Terminated.
Generated Number:7
Insertion of 7 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
New value.
A new node containing data: 7, is entered after exisitng node containing data: 5.
Value Insertion Terminated.
L:0, 5, 1, 0, 7,
Display List Identity invoked.
H:(Data: 0, Cardinality: 2), (Data: 1, Cardinality: 1), (Data: 5, Cardinality: 1), (Data: 7, Cardinality: 1),


Microsoft Visual Studio Debug Console
Please insert an integer (>=1 and <=20) for list initialization:
-5
Given input: -5
Wrong Input. You are kindly requested to try again.
Please insert an integer (>=1 and <=20) for list initialization:
35
Given input: 35
Correct input
Max Capacity:35 out of bounds [1,20].
Please insert an integer (>=1 and <=20) for list initialization:
1
Given input: 1
Correct input
Constructor of H Linked List invoked.
Succesfull Creation of Linked List with MAX_CAPACITY = 1.
Generated Number:1
Insertion of 1 value in ascending order invoked.
Scenario 0 invoked.
Initialization of first node of the histogram list.
Value Insertion Terminated.
L:1,
Display List Identity invoked.
H:(Data: 1, Cardinality: 1),
C:\Users\Juljan\source\repos\DS_Project1\Debug\DS_Project1.exe (proces
To automatically close the console when debugging stops, enable Tools
le when debugging stops.
Press any key to close this window . . .


Microsoft Visual Studio Debug Console
New value.
A new node containing data: 8, is entered after exisitng node containing data: 6.
Value Insertion Terminated.

Generated Number:5
Insertion of 5 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 5 increased by 1.
Value Insertion Terminated.

Generated Number:8
Insertion of 8 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 8 increased by 1.
Value Insertion Terminated.

Generated Number:3
Insertion of 3 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 3 increased by 1.
Value Insertion Terminated.

Generated Number:10
Insertion of 10 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 10 increased by 1.
Value Insertion Terminated.

Generated Number:0
Insertion of 0 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 0 increased by 1.
Value Insertion Terminated.

Generated Number:10
Insertion of 10 value in ascending order invoked.
Scenario 2 invoked.
Value cardinality of list < Max capacity/cardinality.
Already seen value.
Data cardinality of node containing data: 10 increased by 1.
Value Insertion Terminated.

L:9, 2, 0, 6, 3, 5, 1, 4, 1, 1, 3, 10, 8, 5, 8, 3, 10, 0, 10,
---
Display List Identity invoked.
H:(Data: 0, Cardinality: 2), (Data: 1, Cardinality: 3), (Data: 2, Cardinality: 1)
, (Data: 3, Cardinality: 3), (Data: 4, Cardinality: 1), (Data: 5, Cardinality: 2)
, (Data: 6, Cardinality: 1), (Data: 8, Cardinality: 2), (Data: 9, Cardinality: 1)
, (Data: 10, Cardinality: 3),

2 Ερώτημα 2

2.1 Εκφώνηση

Η αναπαράσταση των αριθμών κινητής υποδιαστολής (floating point numbers) σε έναν υπολογιστή χρησιμοποιεί συγκεκριμένο πλήθος από βψτες: 4 bytes για τον τύπο δεδομένων float και 8 bytes για τον τύπο δεδομένων double της C++. Αυτό έχει ως αποτέλεσμα να μπορεί να αποθηκευτεί μέχρι ένα μέγιστο πλήθος δεκαδικών ψηφίων κάθε αριθμού. Αν ο αριθμός έχει περισσότερα δεκαδικά ψηφία αυτά αναγκαστικά αποκόπτονται. Το ίδιο συμβαίνει στο αποτέλεσμα που προκύπτει μετά από οποιαδήποτε αριθμητική πράξη μεταξύ αριθμών κινητής υποδιαστολής. Έτσι εισάγονται αριθμητικά σφάλματα στους υπολογισμούς, τα οποία προφανώς πρέπει να ελαχιστοποιηθούν.

Στην εργασία θα μας απασχολήσει η πράξη της πρόσθεσης αριθμών τύπου float. Δεν θα χρησιμοποιήσουμε αριθμούς τύπου double, απλώς για να φανεί πιο εύκολα η επίδραση των αριθμητικών σφαλμάτων. Αν προσπαθήσουμε να προσθέσουμε έναν πολύ μεγάλο και έναν πολύ μικρό αριθμό κινητής υποδιαστολής, λόγω των αριθμητικών σφαλμάτων ο μικρότερος αριθμός μπορεί να «χαθεί» λόγω της αποκοπής των δεκαδικών ψηφίων. Αν θέλουμε να προσθέσουμε μεταξύ τους περισσότερους αριθμούς κινητής υποδιαστολής τότε θα πρέπει να ξεκινήσουμε προσθέτοντας τους δύο μικρότερους αριθμούς μεταξύ τους, το αποτέλεσμα να το προσθέσουμε με τον αμέσως μικρότερο αριθμό, κλπ. Με αυτόν τον τρόπο το άθροισμα μεγαλώνει και πλησιάζει περισσότερο προς τον επόμενο αριθμό που πρέπει να προστεθεί και το αποτέλεσμα έχει μεγαλύτερη ακρίβεια.

Στα πλαίσια της εργασίας θα πρέπει να υλοποιήσετε σε C++ έναν σωρό ελαχίστων για να βρίσκετε με αποδοτικό τρόπο τους μικρότερους αριθμούς που πρέπει να προστεθούν κάθε φορά. Θα υλοποιήσετε επίσης την πρόσθεση αριθμών κινητής υποδιαστολής ξεκινώντας από τους μεγαλύτερους αριθμούς (επομένως θα πρέπει να υλοποιήσετε και έναν σωρό μεγίστων), για να δείτε τις διαφορές στα αριθμητικά αποτελέσματα.

2.2 Θέματα Υλοποίησης

Παρακάτω παρατίθενται τα επιμέρους ειδικά θέματα υλοποίησης του ερωτήματος:

- Θα δημιουργήσετε έναν πίνακα heap_min με μέγεθος N, στον οποίο θα αποθηκεύσετε αριθμούς κινητής υποδιαστολής. Το μέγεθος N του πίνακα θα το διαβάζετε κατά την ώρα εκτέλεσης του προγράμματος από τον χρήστη και ο πίνακας heap_min θα πρέπει να δεσμεύεται δυναμικά.
- Το μέγεθος του πίνακα N μπορεί στην αρχή να είναι μικρό για να μπορείτε να επαληθεύσετε τα αποτελέσματα, αλλά θα πρέπει στο τέλος να τρέξετε και να παρουσιάσετε αποτελέσματα για μεγάλα N, π.χ. $> 10.000.000$.
- Οι αριθμοί που θα αποθηκεύονται στον πίνακα heap_min θα παράγονται με μια γεννήτρια τυχαίων αριθμών.
- Στην συνέχεια θα δεσμεύετε δυναμικά έναν δεύτερο πίνακα heap_max με ίδιο μέγεθος N και θα αντιγράφετε όλα τα στοιχεία του heap_min στον heap_max. Θα έχετε επομένως δύο πίνακες με το ίδιο μέγεθος και τα ίδια περιεχόμενα.
- Θα μετατρέψετε τον πίνακα heap_min σε σωρό ελαχίστων και τον πίνακα heap_max σε σωρό μεγίστων, χρησιμοποιώντας ως βάση το Πρόγραμμα 9.5, σελ. 428 του βιβλίου "Δομές Δεδομένων, Αλγόριθμοι και Εφαρμογές στη C++" και προσαρμόζοντας το κατάλληλα.
- Σε ότι αφορά τον σωρό ελαχίστων, θα εξάγετε τα δύο μικρότερα στοιχεία, θα τα προσθέτετε και το αποτέλεσμα θα το εισάγετε στον σωρό ελαχίστων. Αυτό θα επαναλαμβάνεται έως ότου ο σωρός

αδειάσει, οπότε το τελευταίο στοιχείο που θα εξάγετε θα είναι το άθροισμα όλων των αριθμών κινητής υποδιαστολής.

- Θα επαναλάβετε τον παραπάνω αλγόριθμο για τον σωρό μεγίστων αυτή τη φορά, εξάγοντας σε κάθε βήμα τα δύο μεγαλύτερα στοιχεία και επανεισάγοντας στον σωρό το άθροισμα τους.
- Συγκρίνετε στο τέλος τα αποτελέσματα από τους δύο αυτούς τρόπους πρόσθεσης.

2.3 Τεκμηρίωση Υλοποίησης

Στην παρούσα υποενότητα, θα παρουσιάσουμε την ακολουθία εκτέλεσης του προγράμματος προς επίλυση του ερωτήματος, καθώς και την συλλογιστική που ακολουθείται απο πίσω.

2.3.1 Τάξη FloatAddition

Η τάξη FloatAddition, συστήνει την βασική τάξη υλοποίησης του ερωτήματος. Σε αυτήν την υποενότητα, παραθέτουμε τα επιμέρους δεδομένα της τάξης, καθώς και τις συναρτήσεις λειτουργικότητας.

Δεδομένα:

Δείκτης προς μεταβλητή τύπου Float, η οποία θα δείχνει στο πρώτο στοιχείο ενός σωρού ελαχίστων.

Δείκτης προς μεταβλητή τύπου Float, η οποία θα δείχνει στο πρώτο στοιχείο ενός σωρού μεγίστων.

Ακέραια τιμή η οποία δείχνει το πλήθος των στοιχείων που εμπεριέχουν και οι δύο σωροί (καθώς επιτελούν την λειτουργικότητα τους παράλληλα).

Ακέραια τιμή η οποία δείχνει το μέγιστο πλήθος των στοιχείων που δύναται να εμπεριέχουν οι σωροί.

Συνολικό άθροισμα των στοιχείων του σωρού ελαχίστων.

Συνολικό άθροισμα των στοιχείων του σωρού μεγίστων.

Συναρτήσεις Λειτουργικότητας:

Παρακάτω, παρατίθενται οι συναρτήσεις λειτουργικότητας της τάξης καθώς και η επιμέρους επεξήγηση ως προς την συλλογιστική επίλυσης τους.

Constructor()

Η συνάρτηση αυτή ενεργοποιείται κατά την δημιουργία ενός αντικειμένου τύπου FloatAddition, όπου δεν δίδεται καμία παράμετρος ως όρισμα εισόδου. Για σκοπούς επείδηξης της ορθής λειτουργικότητας και πειραματισμού, στην συνάρτηση αυτή αρχικοποιούνται δυναμικά 2 πίνακες μεγέθους 7, αριθμών κινητής υποδιαστολής, με κάποια σταθερά δεδομένα εισόδου. Αξίζει να σημειωθεί, ότι στην προκειμένη φάση, τα δεδομένα τα οποία βρίσκονται στους πίνακες, δεν ικανοποιούν τις ιδιότητες σωρού (ούτε μεγίστων ούτε ελαχίστων), και ως εκ τούτου, μπορούμε να θεωρήσουμε ως σωρό μόνο το 1 στοιχείο του εκάστοτε πίνακα. Ακολούθως, αρχικοποιούμε το πλήθος των στοιχείων που εμπεριέχουν και οι 2 σωροί ως 1. Σε αυτο το σημείο αρχικοποιούμε το άθροισμα των στοιχείων του σωρού ελαχίστων με 0 και το άθροισμα των στοιχείων του σωρού μεγίστων με 0. Το μέγιστο πλήθος στοιχείων των σωρών ισούται με 7.

Constructor(int N)

Η συνάρτηση αυτή αποτελεί την δεύτερη μέθοδο δημιουργίας ενός αντικειμένου τύπου FloatAddition, όπου δίδεται ως παράμετρος το πλήθος N των στοιχείων. Χρησιμοποιώντας γεννήτρια τυχαίων συναρτήσεων, γεμίζουμε με τυχαίες τιμές και τους δύο πίνακες. Αξίζει να σημειωθεί, ότι στην προκειμένη φάση, τα δεδομένα τα οποία βρίσκονται στους πίνακες, δεν ικανοποιούν τις ιδιότητες σωρού (ούτε μεγίστων ούτε ελαχίστων), και ως εκ τούτου, μπορούμε να θεωρήσουμε ως σωρό μόνο το 1 στοιχείο του εκάστοτε πίνακα. Ακολούθως, αρχικοποιούμε το πλήθος των στοιχείων που εμπεριέχουν και οι 2 σωροί ως 1. Σε αυτο το σημείο αρχικοποιούμε το άθροισμα των στοιχείων του σωρού ελαχίστων με 0 και το άθροισμα των στοιχείων του σωρού μεγίστων με 0. Το μέγιστο πλήθος στοιχείων των σωρών ισούται με N.

Ταυτότητα Σωρών

Κατά την κλήση της συνάρτησης αυτής, εκτυπώνονται οι δύο σωροί (μεγίστων και ελαχίστων), λαμβάνοντας υπόψιν το πλήθος στοιχείων που μπορούν να θεωρηθούν ως σωρός.

Επέκταση Σωρού

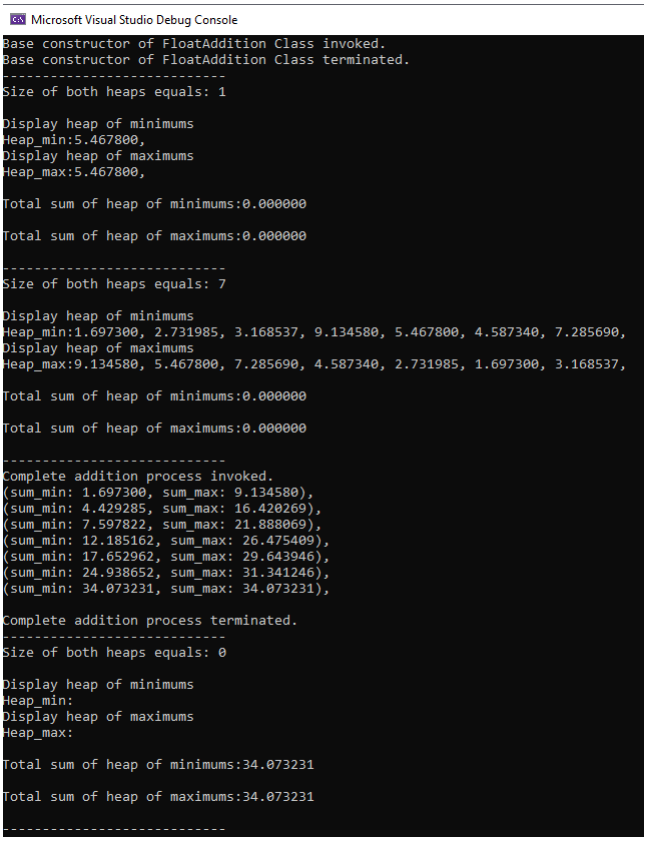
Μέχρις ώτου το πλήθος των στοιχείων του εκάστωτε σωρού να είναι ίσο με το μέγιστο πλήθος των στοιχείων του εκάστωτε σωρού, παράλληλα για τις σωρούς μεγίστου καθώς και για τις σωρούς ελαχίστου, επεκτείνουμε το μέγεθος του σωρού κατά 1 και μειώνοντας το μέγεθος των υπολοίπων δεδομένων του πίνακα κατά 1, μελετάμε διαδοχικά την τιμή του στοιχείου με την τιμή των γονέων, και επιτελούμε τις απαραίτητες μεταφορές στοιχείων στον πίνακα. Το αποτέλεσμα, συστήνει έναν σωρό μεγίστων και ελαχίστων αντίστοιχα.

Εξαγωγή πρώτου στοιχείου

Κατή την κλήση αυτής της συνάρτησης, το πρόγραμμα εισέρχεται σε μια επαναληπτική διαδικασία εξαγωγής του πρώτου στοιχείου απο τον σωρό, της πρόσθεσης του στοιχείου στο γενικό άθροισμα, το μηδενισμού της τελευταίας θέσης του σωρού και της μείωσης του πλήθους των στοιχείων του σωρού κατά 1. Με αυτόν τον τρόπο, στο πέρας της διαδικασίας για όλα τα στοιχεία, μέχρις ώτου δηλαδή το πλήθος των στοιχείων του σωρού να είναι 0, θα έχουμε συγκεντρώσει το άθροισμα του σωρού στην μεταβλητή μας. Εκτελούμε παράλληλα και για τους δύο σωρούς.

2.4 Στιγμιότυπα Εκτέλεσης Προγράμματος

Σε αυτήν την υποενότητα, παραθέτουμε στιγμιότυπα εκτέλεσης του προγράμματος τα οποία δείχνουν αφενός την ορθότητα εκτέλεσης, και αφετέρου της διαφορές ως προς την ακρίβεια του αποτελέσματος. Όπως φαίνεται απο τα στιγμιότυπα, έχει επιλεχθεί η είσοδος να είναι είτε σταθερή (1), είτε τυχαία με πλήθος στοιχείων $N =$ 15, 100, 1,000, 10,000, 100,000, 1,000,000, 10,000,000, καθώς και 100,000,000. Στα μεγάλα στιγμιότυπα, έχει επιλεχθεί να παρατίθενται μόνο οι άκρως απαραίτητες πληροφορίες.


Microsoft Visual Studio Debug Console
Base constructor of FloatAddition Class invoked.
Base constructor of FloatAddition Class terminated.

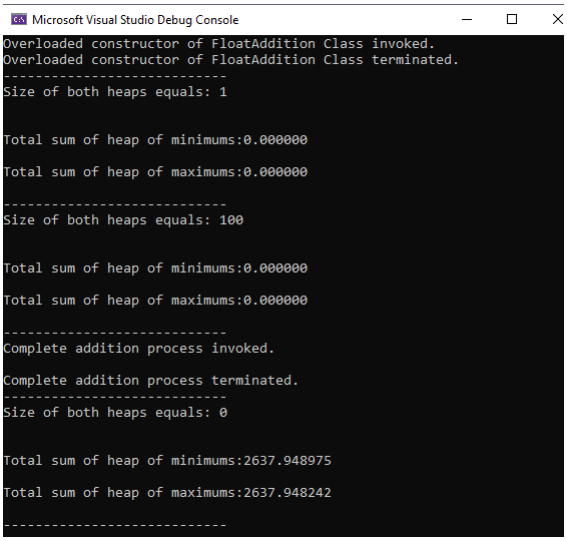
Size of both heaps equals: 1
Display heap of minimums
Heap_min:5.467800,
Display heap of maximums
Heap_max:5.467800,
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

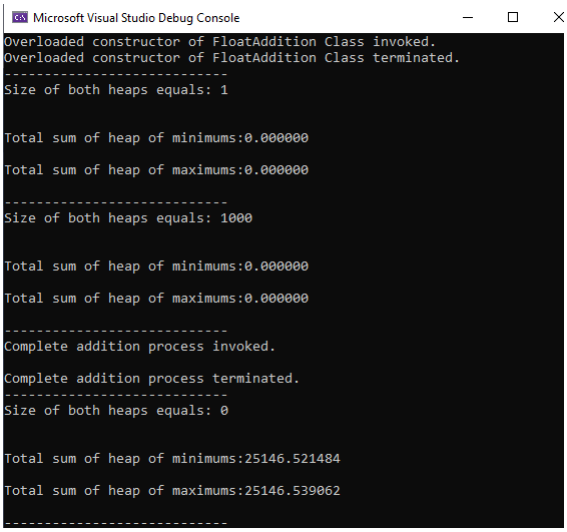
Size of both heaps equals: 7
Display heap of minimums
Heap_min:1.697300, 2.731985, 3.168537, 9.134580, 5.467800, 4.587340, 7.285690,
Display heap of maximums
Heap_max:9.134580, 5.467800, 7.285690, 4.587340, 2.731985, 1.697300, 3.168537,
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Complete addition process invoked.
(sum_min: 1.697300, sum_max: 9.134580),
(sum_min: 4.429285, sum_max: 16.420269),
(sum_min: 7.597822, sum_max: 21.888069),
(sum_min: 12.185162, sum_max: 26.475409),
(sum_min: 17.652962, sum_max: 29.643946),
(sum_min: 24.938652, sum_max: 31.341246),
(sum_min: 34.073231, sum_max: 34.073231),
Complete addition process terminated.

Size of both heaps equals: 0
Display heap of minimums
Heap_min:
Display heap of maximums
Heap_max:
Total sum of heap of minimums:34.073231
Total sum of heap of maximums:34.073231

```
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.
----------------------------
Size of both heaps equals: 1

Display heap of minimums
Heap_min:29.410480,
Display heap of maximums
Heap_max:29.410480,

Total sum of heap of minimums:0.000000

Total sum of heap of maximums:0.000000

----------------------------
Size of both heaps equals: 15

Display heap of minimums
Heap_min:3.015365, 3.720076, 4.291935, 29.410480, 26.614958, 46.651199
, 5.964401, 45.004570, 41.679478, 47.770061, 27.103712, 49.133507, 46.
980034, 38.308830, 13.276772,
Display heap of maximums
Heap_max:49.133507, 45.004570, 47.770061, 41.679478, 27.103712, 46.980
034, 13.276772, 5.964401, 38.308830, 3.015365, 26.614958, 29.410480, 4
6.651199, 3.720076, 4.291935,

Total sum of heap of minimums:0.000000

Total sum of heap of maximums:0.000000

----------------------------
Complete addition process invoked.
(sum_min: 3.015365, sum_max: 49.133507),
(sum_min: 6.735441, sum_max: 96.903564),
(sum_min: 11.027376, sum_max: 143.883606),
(sum_min: 16.991777, sum_max: 190.534805),
(sum_min: 30.268551, sum_max: 235.539368),
(sum_min: 56.883507, sum_max: 277.218842),
(sum_min: 83.987221, sum_max: 315.527679),
(sum_min: 113.397705, sum_max: 344.938171),
(sum_min: 151.706543, sum_max: 372.041870),
(sum_min: 193.386017, sum_max: 398.656830),
(sum_min: 238.390594, sum_max: 411.933594),
(sum_min: 285.041809, sum_max: 417.897980),
(sum_min: 332.021851, sum_max: 422.189911),
(sum_min: 381.155365, sum_max: 425.909973),
(sum_min: 428.925415, sum_max: 428.925323),

Complete addition process terminated.
----------------------------
Size of both heaps equals: 0

Display heap of minimums
Heap_min:
Display heap of maximums
Heap_max:

Total sum of heap of minimums:428.925415

Total sum of heap of maximums:428.925323

----------------------------
```



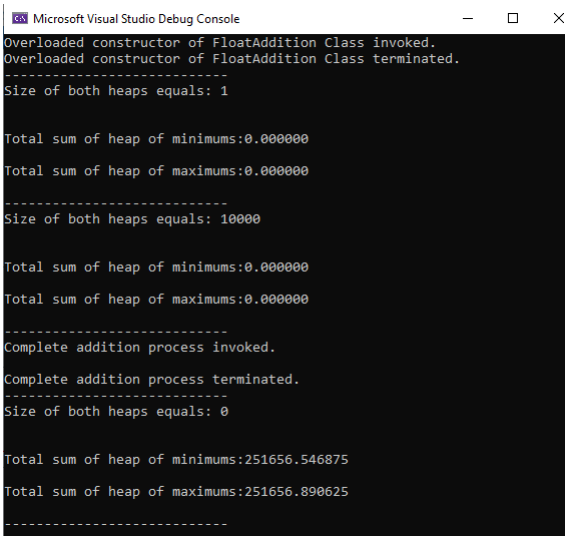

Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.

Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Size of both heaps equals: 1000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Complete addition process invoked.
Complete addition process terminated.

Size of both heaps equals: 0
Total sum of heap of minimums:25146.521484
Total sum of heap of maximums:25146.539062

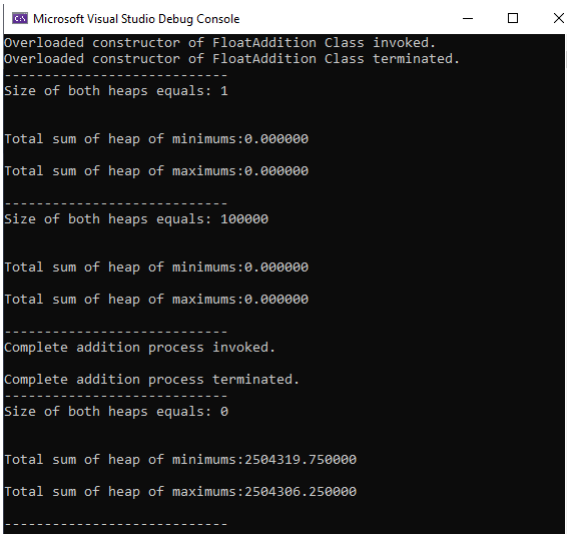

Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.

Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Size of both heaps equals: 10000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Complete addition process invoked.
Complete addition process terminated.

Size of both heaps equals: 0
Total sum of heap of minimums:251656.546875
Total sum of heap of maximums:251656.890625

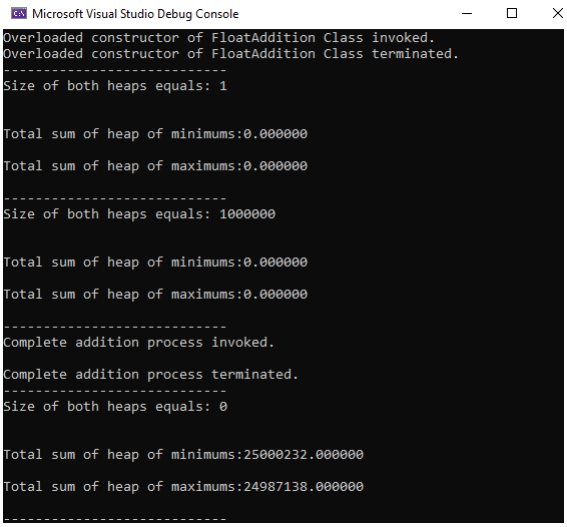

Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.

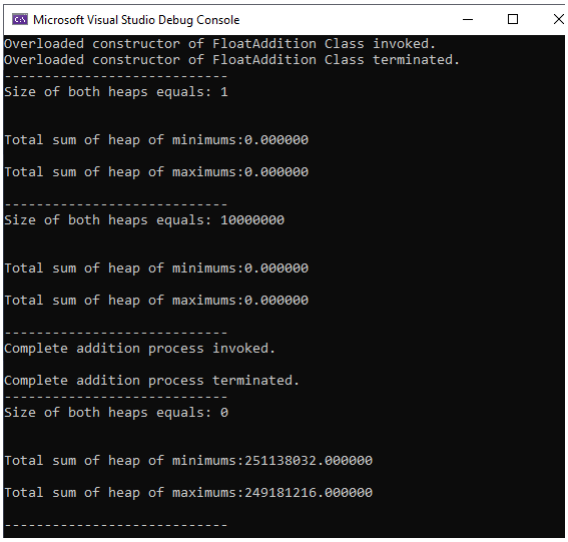
Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

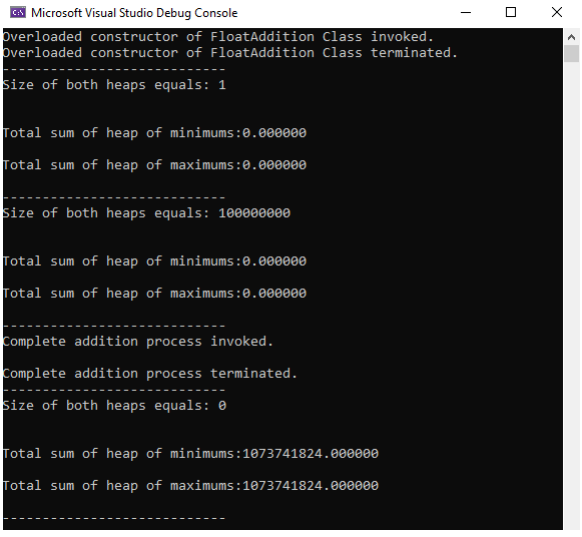
Size of both heaps equals: 100000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000

Complete addition process invoked.
Complete addition process terminated.

Size of both heaps equals: 0
Total sum of heap of minimums:2504319.750000
Total sum of heap of maximums:2504306.250000


Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.
Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Size of both heaps equals: 1000000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Complete addition process invoked.
Complete addition process terminated.
Size of both heaps equals: 0
Total sum of heap of minimums:25000232.000000
Total sum of heap of maximums:24987138.000000


Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.
Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Size of both heaps equals: 10000000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Complete addition process invoked.
Complete addition process terminated.
Size of both heaps equals: 0
Total sum of heap of minimums:251138032.000000
Total sum of heap of maximums:249181216.000000


Microsoft Visual Studio Debug Console
Overloaded constructor of FloatAddition Class invoked.
Overloaded constructor of FloatAddition Class terminated.
Size of both heaps equals: 1
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Size of both heaps equals: 100000000
Total sum of heap of minimums:0.000000
Total sum of heap of maximums:0.000000
Complete addition process invoked.
Complete addition process terminated.
Size of both heaps equals: 0
Total sum of heap of minimums:1073741824.000000
Total sum of heap of maximums:1073741824.000000