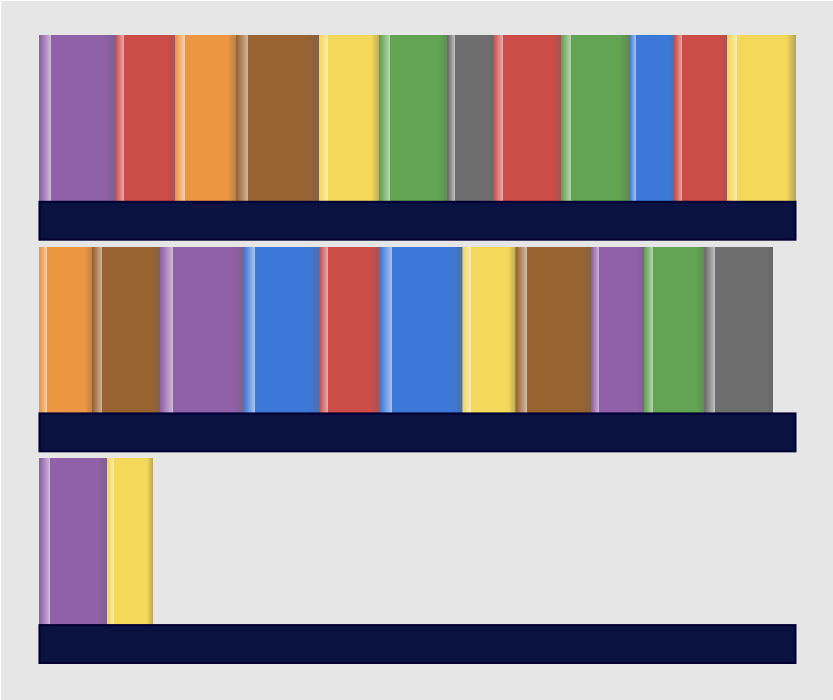
Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

Οικονομικό Πανεπιστήμιο Αθηνών

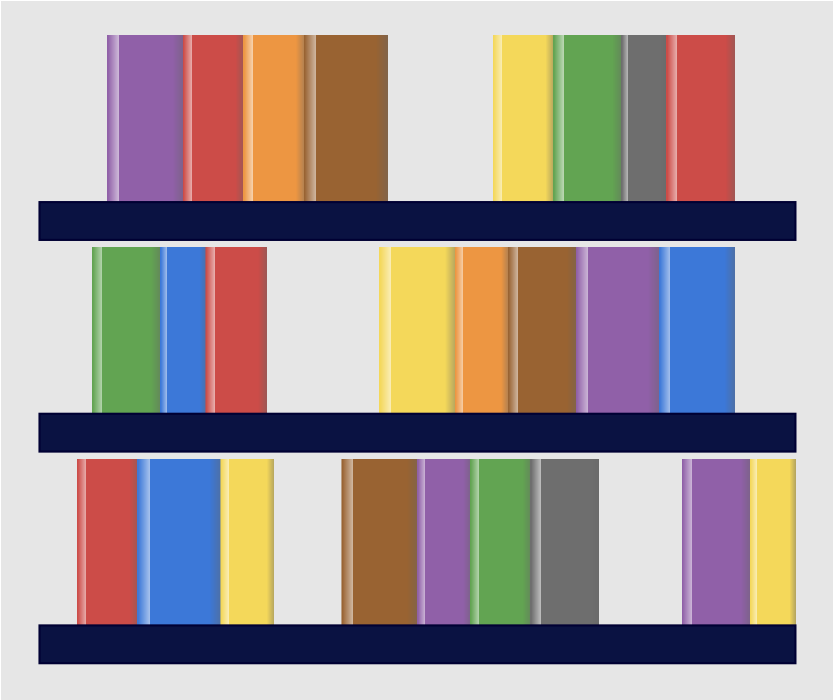
Ταξινόμηση Βιβλιοθήκης

Φανταστείτε ότι έχετε μια βιβλιοθήκη, στην οποία τοποθετείτε τα βιβλία ταξινομημένα. Αν λοιπόν τοποθετούσατε τα βιβλία σας κολλητά το ένα δίπλα στο άλλο, η βιβλιοθήκη σας μπορεί να ήταν όπως η παρακάτω:



Σκεφτείτε όμως τι θα συμβεί αν αποκτήσετε ένα βιβλίο, που η θέση που του αντιστοιχεί είναι στη μέση του επάνω ραφιού. Τότε, θα πρέπει να μεταφέρετε ένα ή περισσότερα βιβλία από το μεσαίο ράφι στο κάτι ράφι, και αντίστοιχα από το επάνω ράφι στο μεσαίο ράφι, ώστε να κάνετε χώρο για να τοποθετήσετε το βιβλίο στη σωστή του θέση. Επομένως, για την τοποθέτηση ενός και μόνο βιβλίου μπορεί αν απαιτηθούν πολλές μετακινήσεις—στη χειρότερη περίπτωση, αν το βιβλίο πρέπει να μπει στην αριστερή άκρη του επάνω ραφιού, όλα τα βιβλία στη βιβλιοθήκη θα πρέπει να μετακινηθούν.

Για να το αποφύγουμε αυτό, θα μπορούσαμε να είμαστε πιο προνοητικοί, και να βάλουμε τα βιβλία στη βιβλιοθήκη ταξινομημένα, αλλά αφήνοντας κάποια κενά μεταξύ τους, όπως για παράδειγμα στην παρακάτω εικόνα:



Η κατάσταση αυτή δεν εμφανίζεται μόνο στις βιβλιοθήκες. Αν θέλουμε να αποθηκεύουμε δεδομένα ταξινομημένα, έτσι ώστε σε αυτά να προστίθενται και άλλα τα οποία θέλουμε να τα τοποθετούμε στη σωστή τους θέση ελαχιστοποιώντας όμως τις μεταφορές δεδομένων κάθε φορά, μπορούμε να υιοθετήσουμε μια αντίστοιχη προσέγγιση με τη δημιουργία μιας κατάλληλης δομής δεδομένων, ενός *αραιού πίνακα* (sparse table).

Ο αραιός πίνακας θα έχει χωρητικότητα m, και σε αυτόν θα μπορούμε να αποθηκεύουμε n αυθεντικά κλειδιά, με $m > n$. Στις υπόλοιπες $m - n$ θέσεις του πίνακα θα αποθηκεύουμε *μαϊμούδες* (dummy keys). Θα περιγράφουμε τον πίνακα μας ως:

$$y = [y_0, y_1, \dots, y_{m-1}], \qquad y_0 \le y_1 \le \cdots \le y_{m-1}$$

Έτσι, το αυθεντικό κλειδί ή η μαϊμού y_i είναι στη θέση i του πίνακα, $0 \le i < m$.

Αν τα αυθεντικά κλειδιά του πίνακα είναι:

$$y_{i_0} < y_{i_1} < \dots < y_{i_{n-1}}, \qquad 0 \le i_0 < \dots < i_{n-1} = m - 1$$

τότε για κάθε αυθεντικό κλειδί y_{i_t} αποθηκεύουμε μαϊμούδες στις διαδοχικές θέσεις που μεσολαβούν από το προηγούμενο αυθεντικό κλειδί $y_{i_{t-1}}$ μέχρι το y_{i_t}:

$$i_{t-1} + 1, i_{t-1} + 2, \dots, i_t - 1, \qquad 0 \le t < n, i_{-1} = -1$$

Δηλαδή:

$$y_0 = \ldots = y_{i_0} < y_{i_0+1} = \ldots = y_{i_1} < \ldots < y_{i_{n-1}+1} = \ldots = y_{i_n-1}$$

Για παράδειγμα, στον παρακάτω πίνακα τα αυθεντικά κλειδιά είναι αποθηκευμένα στις θέσεις $2, 5, 6, 8$:

$$y = [2, 2, 2, 3, 3, 3, 4, 5, 5], \qquad n = 4, m = 9$$

Στη συνέχεια, θα θεωρήσουμε ότι ο πίνακας είναι κυκλικός, άρα μετά τη θέση $m - 1$ πηγαίνουμε στη θέση 0, και όλες οι σχετικές πράξεις γίνονται με υπόλοιπο m. Το πρώτο κλειδί του πίνακα δεν θα βρίσκεται υποχρεωτικά στη θέση 0· θα πρέπει να διατηρούμε έναν δείκτη που δείχνει κάθε στιγμή τη θέση του πρώτου κλειδιού. Τον δείκτη αυτόν θα τον ονομάζουμε *κεφαλή* (head).

Ο αραιός πίνακας θα κατασκευάζεται καθώς εισάγουμε κλειδιά σε αυτόν. Το μέγεθος του πίνακα ορίζεται από δύο ακολουθίες αριθμών, της n_k που περιέχει φυσικούς, και της m_k που περιέχει πραγματικούς:

$$\{n_k \,:\, 0 \le k < \infty\} \quad \text{και} \quad \{m_k \,:\, 1 \le k < \infty\}$$

όπου

$$1 = n_0 < n_1 < \ldots < n_k < \ldots, \qquad n_k < n_{k-1} m_k, \qquad k = 1, 2, \ldots$$

Συγκεκριμένα, το μέγεθος m θα είναι $m = n_{k-1} m_k$ όταν ο αριθμός n των αυθεντικών κλειδιών στον πίνακα είναι $n_{k-1} < n \le n_k$ για $k > 1$ ή $n_{k-1} \le n \le n_k$ για $k = 1$.

Για παράδειγμα, έστω ότι:

$$n_k = \{1, 2, 5, 10, 15, 20\}, \qquad m_k = \{2, 5/2, 2, 3/2, 4/3\}$$

τότε το μέγεθος του πίνακα και τα κλειδιά που αποθηκεύονται σε αυτόν μπορεί να είναι:

k	n_k	m_k	$m = n_{k-1} m_k$	n
0	1			
1	2	2	2	$[1, 2]$
2	5	5/2	5	$(2, 5]$
3	10	2	10	$(5, 10]$
4	15	3/2	15	$(10, 15]$
5	20	4/3	20	$(15, 20]$

Για να εισάγουμε ένα κλειδί x σε έναν πίνακα μεγέθους $m = n_{k-1} m_k$, ο οποίος περιέχει $n_{k-1} < n < n_k$ αυθεντικά κλειδιά, εργαζόμαστε ως εξής:

1. Με τη χρήση δυαδικής αναζήτησης βρίσκουμε τη θέση s ώστε $y_{s-1} < x < y_s$.
2. Αν το y_s είναι μαϊμού, απλώς το αντικαθιστούμε με το x, και ο πίνακας γίνεται:

$$[y_0, \dots, y_{s-1}, x, y_{s+1}, \dots, y_{m-1}]$$

3. Αν το y_s είναι αυθεντικό, τα t αυθεντικά κλειδιά που ακολουθούν:

$$y_s \neq \underbrace{y_{n+1} \neq \dots y_{s+t-1} \neq y_{s+t}}_{t} = y_{s+t+1}$$

μετακινούνται κυκλικά δεξιά κατά μία θέση, ενημερώνουμε την κεφαλή του πίνακα αναλόγως, και το x εισάγεται στη διεύθυνση s δίνοντας τον πίνακα:

$$[y_0, \dots, y_{s-1}, x, y_s, \dots, y_{s+t-1}, y_{s+t+1}, \dots, y_{m-1}]$$

Αν θέλουμε να εισάγουμε ένα κλειδί x σε έναν πίνακα $m = n_{k-1}m_k$, ο οποίος περιέχει n_k αυθεντικά κλειδιά, τότε *αναδομούμε* τον πίνακα. Αυξάνουμε το μέγεθος του πίνακα σε $m' = n_k m_{k+1}$ και εισάγουμε τα n_k κλειδιά $y_{i_0} < y_{i_1} < \dots < y_{i_{n_k-1}}$ ομοιόμορφα στον επαυξημένο πίνακα. Στη συνέχεια, εκτελούμε μια δυαδική αναζήτηση στον επαυξημένο πίνακα και εισάγουμε το κλειδί x όπως προηγουμένως.

Για να μοιράσουμε τα n_k αυθεντικά κλειδιά ομοιόμορφα στον πίνακα, υπολογίζουμε το πηλίκο $q = \lfloor m'/n_k \rfloor$ και το υπόλοιπο $r = m' \bmod n_k$ της διαίρεσης του νέου μεγέθους του πίνακα m' με τον αριθμό των κλειδιών. Αυτό σημαίνει ότι μπορούμε να μοιράσουμε τα κλειδιά σε n_k διαστήματα στον πίνακα, το μέγεθος του καθενός διαστήματος θα είναι q, αλλά θα μας μείνουν στο τέλος r θέσεις στον πίνακα. Θα θέλαμε αυτές οι r θέσεις να κατανεμηθούν όσο γίνεται πιο ομοιόμορφα στα διαστήματα.

Αν βρισκόμαστε στο διάστημα i, τότε θα πρέπει μέχρι την αρχή του διαστήματος να έχουμε κατανείμει $\lfloor i \times r/n_k \rfloor$ από τις r θέσεις ενώ στο τέλος του διαστήματος θα πρέπει να έχουμε κατανείμει $\lfloor (i+1) \times r/n_k \rfloor$ από τις r θέσεις. Συνεπώς, αν $\lfloor i \times r/n_k \rfloor < \lfloor (i+1) \times r/n_k \rfloor$ τότε το διάστημα i θα έχει μέγεθος $q+1$ αντί για q.

Για παράδειγμα, αν έχουμε $m' = 10$ και $n_k = 4$, τότε $q = 2, r = 2$, και μπορούμε να υπολογίσουμε:

i	$\lfloor i \times r/n_k \rfloor$	$\lfloor (i+1) \times r/n_k \rfloor$	$q+1$	μέγεθος
0	0	0	όχι	2
1	0	1	ναι	3
2	1	1	όχι	2
3	1	2	ναι	3

Για να διαγράψουμε ένα κλειδί από τον πίνακα, θα πρέπει πάλι να προνοήσουμε για τον χειρισμό των μαϊμούδων και να αναδομήσουμε τον πίνακα, αν χρειάζεται. Έστω ότι θέλουμε να διαγράψουμε το κλειδί x από τον πίνακα:

$$y = [y_0, y_1, \dots, y_{n_{k-1}m_k-1}]$$

Κάνουμε μια δυαδική αναζήτηση και βρίσκουμε τη θέση s της πρώτης εμφάνισης του κλειδιού x στον πίνακα (γιατί μπορεί να ακολουθούν μαϊμούδες):

$$y_{s-1} < x = y_s$$

Έστω ότι ακολουθούν μαϊμούδες, έτσι ώστε το επόμενο αυθεντικό κλειδί είναι L θέσεις (κυκλικά) μετά:

$$\underbrace{y_s = y_{s+1} = \cdots = y_{s+L-1}}_{L-1} \neq y_{s+L}$$

Τότε αντικαθιστούμε τις μαϊμούδες αυτές:

$$[\underbrace{y_s, y_{s+1}, \cdots, y_{s+L-1}}_{L-1}]$$

με την τιμή του επόμενου αυθεντικού κλειδιού:

$$[\underbrace{y_{s+L}, y_{s+L}, \cdots, y_{s+L}}_{L-1}]$$

παίρνοντας τον πίνακα:

$$y' = [y_0, y_1, \cdots, y_{s-1}, \underbrace{y_{s+L}, y_{s+L}, \cdots, y_{s+L}}_{L-1}, y_{s+L+1+}, \cdots, y_{n_{k-1}m_k-1}]$$

Αν τώρα σβήνοντας ένα κλειδί ο αριθμός των αυθεντικών κλειδιών μετά τη διαγραφή του κλειδιού πέσει στο n_{k-2}, τότε θα πρέπει να αναδομήσουμε τον πίνακα. Μειώνουμε το μέγεθος του πίνακα σε $n_{k-2}m_{k-1}$ και εισάγουμε τα n_{k-2} αυθεντικά κλειδιά ομοιόμορφα στον ελάσσωνα πίνακα που έχει προκύψει.

Σκοπός της εργασίας αυτής είναι να υλοποιήσετε αραιούς πίνακες σύμφωνα με την παραπάνω περιγραφή.

Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα

ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*

- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2025-2`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `library_sorting.py`.
- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
  - `json`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Η εργασία είναι αποκλειστικά ατομική. Δεν επιτρέπεται συνεργασία μεταξύ φοιτητών στην εκπόνησή της, με ποινή το μηδενισμό. Επιπλέον η εργασία δεν μπορεί να είναι αποτέλεσμα συστημάτων Τεχνητής Νοημοσύνης (όπως ChatGPT). Ειδικότερα όσον αφορά το τελευταίο σημείο προσέξτε ότι τα συστήματα αυτά χωλαίνουν στην αλγοριθμική σχέση, άρα τυχόν προτάσεις που κάνουν σε σχετικά θέματα μπορεί να είναι λανθασμένες. Επιπλέον, αν θέλετε να χρησιμοποιήσετε τέτοια συστήματα, τότε αν και κάποιος άλλος το κάνει αυτό, μπορεί οι προτάσεις που θα λάβετε να είναι παρόμοιες, οπότε οι εργασίες θα παρουσιάσουν ομοιότητες και άρα θα μηδενιστούν.
- Η έξοδος του προγράμματος θα πρέπει να περιλαμβάνει μόνο ό,τι φαίνεται στα παραδείγματα που παρατίθενται. *Η φλυαρία δεν επιβραβεύεται.*

Χρήση του GitHub

Όπως αναφέρθηκε το πρόγραμμά σας για να αξιολογηθεί θα πρέπει να αποθηκευθεί στο GitHub. Επιπλέον, θα πρέπει το GitHub να χρησιμοποιηθεί καθόλη τη διάρκεια της ανάπτυξης του.

Αυτό σημαίνει ότι *δεν θα ανεβάσετε στο GitHub απλώς την τελική λύση του προβλήματος* μέσω της λειτουργίας "Upload files". Στο GitHub θα πρέπει να φαίνεται *το ιστορικό της συγγραφής του προγράμματος.* Αυτό συνάδει και με τη φιλοσοφία του εργαλείου: λέμε "commit early, commit often". Εργαζόμαστε σε ένα πρόγραμμα και κάθε μέρα, ή όποια στιγμή έχουμε κάνει κάποιο σημαντικό βήμα, αποθηκεύουμε την αλλαγή στο GitHub. Αυτό έχει σειρά ευεργετικών αποτελεσμάτων:

- Έχουμε πάντα ένα αξιόπιστο εφεδρικό μέσο στο οποίο μπορούμε να ανατρέξουμε αν κάτι πάει στραβά στον υπολογιστή μας (μας γλιτώνει από πανικούς του τύπου: ένα βράδυ πριν από την παράδοση ο υπολογιστής μας ή ο δίσκος του πνέει τα λοίσθια, και εμείς τι θα υποβάλουμε στον Λουρίδα;).
- Καθώς έχουμε πλήρες ιστορικό των σημαντικών αλλαγών και εκδόσεων του προγράμματός μας, μπορούμε να επιστρέψουμε σε μία προηγούμενη αν συνειδητοποιήσουμε κάποια στιγμή ότι πήραμε λάθος δρόμο (μας γλιτώνει από πανικούς του τύπου: μα το πρωί δούλευε σωστά, τι στο καλό έκανα και τώρα δεν παίζει τίποτε;).
- Καθώς φαίνεται η πρόοδος μας στο GitHub, επιβεβαιώνουμε ότι το πρόγραμμα δεν είναι αποτέλεσμα της όποιας επιφοίτησης (μας γλιτώνει από πανικούς του τύπου: καλά, πώς το έλυσες το πρόβλημα αυτό με τη μία, μόνος σου;).
- Αν δουλεύετε σε μία ομάδα *που βεβαίως δεν είναι καθόλου η περίπτωση μας εδώ*, μπορούν όλοι να εργάζονται στα ίδια αρχεία στο GitHub εξασφαλίζοντας ότι ο ένας δεν γράφει πάνω στις αλλαγές του άλλου. Παρά το ότι η εργασία αυτή είναι ατομική, καλό είναι να αποκτάτε τριβή με το εργαλείο git και την υπηρεσία GitHub μιας και χρησιμοποιούνται ευρέως και όχι μόνο στη συγγραφή κώδικα.

Άρα επενδύστε λίγο χρόνο στην εκμάθηση των git / GitHub, των οποίων ο σωστός τρόπος χρήσης είναι μέσω γραμμής εντολών (command line), ή ειδικών προγραμμάτων (clients) ή μέσω ενοποίησης στο περιβάλλον ανάπτυξης (editor, IDE).

Τελικό Πρόγραμμα

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python library_sorting.py test
```

Η παράμετρος `test` δίνει το όνομα αρχείου JSON το οποίο περιέχει τις οδηγίες για την κατασκευή και τη χρήση του αραιού πίνακα.

Το αρχείο αυτό ακολουθεί το ακόλουθο σχήμα (δείτε JSON Schema για λεπτομέρειες). Προσέξτε ότι ο πίνακας `nn` αντιστοιχεί στον n_k και ο πίνακας `mm` αντιστοιχεί στον m_{k-1}, ώστε να μην έχουμε κενό στην αρχή του.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "description": "Schema for describing sparse table test runs.",
  "properties": {
    "nn": {
      "description": "Array of n_k integers.",
      "type": "array",
      "items": {
        "type": "integer"
```

```
      }
    },
    "mm": {
      "description": "Array of m_{k-1} real numbers.",
      "type": "array",
      "items": {
        "type": "number"
      }
    },
    "k": {
      "description": "Integer indexing nn and mm.",
      "type": "integer"
    },
    "x": {
      "description": "Initial key to insert in the table.",
      "type": "integer"
    },
    "actions": {
      "type": "array",
      "description": "List of actions to perform on the sparse table.",
      "items": {
        "type": "object",
        "properties": {
          "action": {
            "type": "string",
            "enum": ["insert", "lookup", "delete"]
          },
          "key": {
            "type": "number"
          }
        },
        "required": ["action", "key"],
        "additionalProperties": false
      }
    }
  },
  "required": ["nn", "mm", "k", "x", "actions"],
  "additionalProperties": false
}
```

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python library_sorting.py example_1.json
```

το πρόγραμμά σας θα διαβάσει το αρχείο example_1.json και θα πρέπει να εμφανίσει ακριβώς τα παρακάτω. Προσέξτε ότι ο πίνακας εμφανίζεται με την κεφαλή του ανάμεσα στους χαρακτήρες >< .

```
CREATE with k=1, n_k=[1, 2, 5, 10, 15], m_k=[2, 2.5, 2, 1.5,
↪  1.33], key=3
[>3<, 3]
INSERT 5
[5, >3<]
INSERT 6
[6, >3<, 5, 5, 5]
INSERT 3
[6, >3<, 5, 5, 5]
INSERT 4
[6, >3<, 4, 5, 5]
INSERT 10
[6, 10, >3<, 4, 5]
INSERT 8
[>3<, 3, 4, 4, 5, 5, 6, 6, 8, 10]
INSERT 7
[10, >3<, 4, 4, 5, 5, 6, 6, 7, 8]
LOOKUP 10
Key 10 found at position 0.
[10, >3<, 4, 4, 5, 5, 6, 6, 7, 8]
LOOKUP 15
Key 15 not found. It should be at position 1.
[10, >3<, 4, 4, 5, 5, 6, 6, 7, 8]
LOOKUP 5
Key 5 found at position 5.
[10, >3<, 4, 4, 5, 5, 6, 6, 7, 8]
LOOKUP 0
Key 0 not found. It should be at position 1.
[10, >3<, 4, 4, 5, 5, 6, 6, 7, 8]
DELETE 3
[10, >4<, 4, 4, 5, 5, 6, 6, 7, 8]
DELETE 10
[>4<, 4, 4, 4, 5, 5, 6, 6, 7, 8]
DELETE 10
[>4<, 4, 4, 4, 5, 5, 6, 6, 7, 8]
DELETE 4
[>5<, 5, 5, 5, 5, 5, 6, 6, 7, 8]
DELETE 5
[>6<, 6, 6, 6, 6, 6, 6, 6, 7, 8]
LOOKUP 6.5
Key 6.5 not found. It should be at position 8.
[>6<, 6, 6, 6, 6, 6, 6, 6, 7, 8]
LOOKUP 6
```

```
Key 6 found at position 4.
[>6<, 6, 6, 6, 6, 6, 6, 7, 8]
DELETE 6
[>7<, 7, 8, 8, 8]
DELETE 7
[>8<, 8]
```

Καλή Επιτυχία!