**Πανεπιστήμιο Ιωαννίνων – Τμήμα Μηχανικών Η/Υ και Πληροφορικής**
**Δομές Δεδομένων [ΜΥΥ303] – Χειμερινό Εξάμηνο 2021**

**2η Εργαστηριακή Άσκηση**
**Αναδρομικοί Αλγόριθμοι**
Παράδοση έως Τετάρτη 10/11, 12:00 από το eCourse

**ΠΡΟΣΟΧΗ: Γράψτε σε κάθε αρχείο που παραδίδετε τα ονόματα και τους Α.Μ. των μελών της ομάδας σας. Συμπεριλάβετε όλα τα αρχεία σας (κώδικας Java και lab2results.txt) σε ένα zip αρχείο το οποίο να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.**

Θέλουμε να υλοποιήσουμε μια μέθοδο εύρεσης του $k$-οστού μικρότερου αριθμού σε ένα μη ταξινομημένο πίνακα $A[0:n-1] = [a_0 \quad a_1 \quad \dots \quad a_{n-1}]$, χωρίς να προηγηθεί η ταξινόμησή του. Έστω $a_i$ ο $k$-οστός μικρότερος αριθμός του $A$. Ακόμα, έστω $A' = [a'_0 \quad a'_1 \quad \dots \quad a'_{n-1}]$ ο πίνακας που προκύπτει από την ταξινόμηση του $A$ σε αύξουσα σειρά, δηλαδή $a'_0 \leq a'_1 \leq \cdots \leq a'_{n-1}$. Τότε $a_i = a'_{k-1}$. Για παράδειγμα, ο 5ος μικρότερος αριθμός του πίνακα $A = [12 \quad 2 \quad 43 \quad 15 \quad 50 \quad 14 \quad 88 \quad 75 \quad 7 \quad 20]$ είναι ο 15. Δηλαδή, έχουμε $a_3 = a'_4 = 15$, αφού ο αντίστοιχος ταξινομημένος πίνακα είναι ο $A' = [2 \quad 7 \quad 12 \quad 14 \quad \mathbf{15} \quad 20 \quad 43 \quad 50 \quad 75 \quad 88]$.

**Αναδρομικός αλγόριθμος υπολογισμού του *k*-οστού μικρότερου αριθμού**

Επιλέγουμε το τελευταίο στοιχείο $a_{n-1}$ του πίνακα $A$ ως στοιχείο διαμέρισης και χωρίζουμε τον πίνακα σε δύο μέρη: το αριστερό το οποίο περιλαμβάνει τους αριθμούς που είναι μικρότεροι του $a_{n-1}$ και το δεξί το οποίο περιλαμβάνει τους αριθμούς που είναι μεγαλύτεροι του $a_{n-1}$. (Για λόγους απλότητας, θεωρούμε ότι όλοι οι αριθμοί του $A$ είναι διαφορετικοί.) Έστω ότι το αριστερό μέρος περιλαμβάνει $j-1$ στοιχεία. Τότε, το στοιχείο διαμέρισης είναι ο $j$-οστός μικρότερος αριθμός του $A$, δηλαδή $a_{n-1} = a'_{j-1}$. Έτσι διακρίνουμε τις εξής περιπτώσεις:

- Αν $j-1 = k-1$, τότε το στοιχείο διαμέρισης $a_{n-1} = a'_{k-1}$ είναι ο ζητούμενος αριθμός.
- Αν $j-1 > k-1$, τότε ο ζητούμενος αριθμός είναι ο $k$-οστός μικρότερος αριθμός του υποπίνακα $A[0:j-2]$.
- Αν $j-1 < k-1$, τότε ο ζητούμενος αριθμός είναι ο $(k-j)$-οστός μικρότερος αριθμός του υποπίνακα $A[j:n-1]$.

Στο παράδειγμά μας, όπου $A = [12 \quad 2 \quad 43 \quad 15 \quad 50 \quad 14 \quad 88 \quad 75 \quad 7 \quad 20]$ και $k = 5$, επιλέγουμε ως στοιχείο διαμέρισης το 20. Τότε, ο νέος πίνακας $A$, μετά τη διαμέριση των στοιχείων του με βάση το 20, μπορεί να έχει τη μορφή $A = [12 \quad 2 \quad 15 \quad 14 \quad 7 \quad \mathbf{20} \quad 88 \quad 75 \quad 50 \quad 43]$. Άρα, το στοιχείο διαμέρισης είναι ο 6ος μικρότερος αριθμός του $A$ (δηλαδή $j = 6$), οπότε θα πρέπει να αναζητήσουμε τον 5ο μικρότερο αριθμό του υποπίνακα $A[0:4] = [12 \quad 2 \quad 15 \quad 14 \quad 7]$. Επιλέγουμε ως νέο στοιχείο διαμέρισης τον αριθμό 7, οπότε μετά τη διαμέριση έχουμε $A[0:4] = [2 \quad \mathbf{7} \quad 15 \quad 14 \quad 12]$. Άρα, το στοιχείο διαμέρισης είναι ο 2ος μικρότερος αριθμός του $A[0:4]$ (δηλαδή $j = 2$), οπότε θα πρέπει να αναζητήσουμε τον $(k-j) = 3$ο μικρότερο αριθμό του υποπίνακα $A[2:4] = [15 \quad 14 \quad 12]$. Συνεχίζοντας με αυτό τον τρόπο, βρίσκουμε ότι ο 5ος μικρότερος αριθμός του αρχικού πίνακα $A$ είναι ο 15.

## Ζητούμενες Μέθοδοι

Συμπληρώστε στο πρόγραμμα Select.java τις παρακάτω μεθόδους:

| | |
|---|---|
| double rselect(double[] $A$, int $l$, int $r$, int $k$) | Υλοποιεί τον αναδρομικό αλγόριθμο υπολογισμού του $k$-οστού μικρότερου αριθμού του πίνακα $A[l:r]$. |
| double select(double[] $A$, int $l$, int $r$, int $k$) | Υπολογίζει τον $k$-οστό μικρότερο αριθμό του πίνακα $A[l:r]$, όπως η παραπάνω μέθοδος, αλλά *χωρίς* τη χρήση αναδρομής. |

Μπορεί να σας φανεί χρήσιμο να υλοποιήσετε βοηθητικές μεθόδους, όπως η παρακάτω:

| | |
|---|---|
| int partition(double[] $A$, int $l$, int $r$) | Διαμερίζει τον πίνακα $A[l:r]$ με βάση το τελευταίο του στοιχείο $p = A[r]$. Αν ο πίνακας $A[l:r]$ έχει $j-1$ στοιχεία μικρότερα του $p$, τότε τα στοιχεία του πίνακα θα τοποθετηθούν ως εξής: το $p$ τοποθετείται στη θέση $A[l+j-1]$, ο πίνακας $A[l:l+j-2]$ περιέχει τα στοιχεία που είναι μικρότερα του $p$ και ο πίνακας $A[l+j:r]$ περιέχει τα στοιχεία που είναι μεγαλύτερα του $p$. |

## Εκτέλεση Προγράμματος

Η main() μέθοδος του προγράμματος Select.java λαμβάνει ως είσοδο το πλήθος των στοιχείων $n$ του πίνακα $A$ (τύπου double), θέτει $k = \lfloor n/2 \rfloor$ και καλεί τη μέθοδο randomArray($Acopy$, $seed$), η οποία συμπληρώνει ένα βοηθητικό πίνακα $Acopy$ με τυχαίες τιμές στο διάστημα [0,100], χρησιμοποιώντας τον ακέραιο $seed$ για την αρχικοποίηση της γεννήτριας (ψευδο)τυχαίων αριθμών.

Στη συνέχεια εκτελεί τις μεθόδους rselect$(A, 0, n-1, k)$ και select$(A, 0, n-1, k)$, και συγκρίνει την απόδοσή τους (καθώς και το αποτέλεσμα που επιστρέφουν) με τη μέθοδο Arrays.sort της Java. Πριν από κάθε κλήση, ο πίνακας $A$ επαναφέρεται στην αρχική του κατάσταση $Acopy$, το οποίο είναι απαραίτητο αφού τόσο οι rselect και select, όσο και η sort αλλάζουν τη θέση των στοιχείων του πίνακα.

Για την εκτέλεση του προγράμματος με $n = 1000$ στοιχεία και $seed = 1$ γράψτε java Select 1000 1. Αποθηκεύστε τα αποτελέσματα των εκτελέσεων για $n = 1000, 10000, 100000$ και $1000000$, χρησιμοποιώντας ως $seed$ τον Α.Μ. κάποιου μέλους της ομάδας σας.

## Χείριστη Είσοδος

Υλοποιήστε τη μέθοδο badArray(double[] $A$), η οποία κάνει τη χειρότερη δυνατή αρχικοποίηση του πίνακα $A$ έτσι ώστε να μεγιστοποιείται ο χρόνος εκτέλεσης των rselect και select.

Για να δοκιμάσετε τη μέθοδό σας, βγάλτε τα σχόλια από την εντολή badArray($Acopy$) στη main μέθοδο. Ποιος είναι ο ασυμπτωτικός χρόνος χειρότερης εκτέλεσης των μεθόδων rselect και select για πίνακα με $n$ στοιχεία;

### Παραδοτέα

Ανεβάστε στο eCourse ένα zip αρχείο με το τελικό σας πρόγραμμα Select.java, καθώς και το αρχείο των αποτελεσμάτων lab2results.txt. Το zip αρχείο πρέπει να έχει το όνομα ενός μέλους της ομάδας σας με λατινικούς χαρακτήρες.