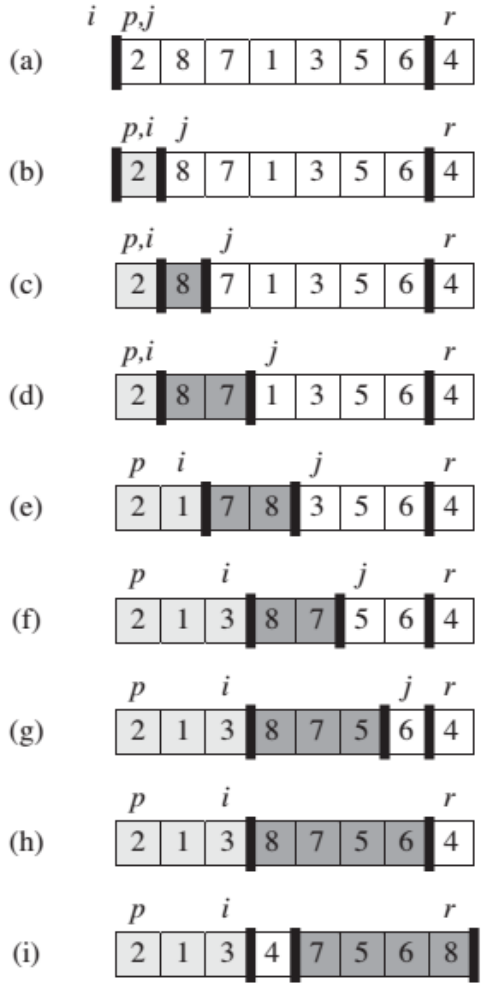
Laborator 9

1. *QuickSort*
Implementați în *Matlab* următorul algoritm de sortare a unui vector de numere A(p : r)
(unde p : r = p, p+1,..., r; p este poziția inițială, r este poziția finală, $1 \le p \le r$):
Divide: fie *x*=A(r) *pivotul;* se caută poziția q ($p \le q \le r$) și se sortează prin interschimbări de elemente vectorul A(p : r) astfel încât fiecare element din A(p : q-1) este mai mic sau egal decât *x*, A(q) este egal cu *x* și fiecare element din A(q+1 : r) este mai mare decât *x* (a se vedea exemplul de mai jos); se returnează poziția q.
Impera: se sortează vectorii A(p : q-1) și A(q+1 : r) prin apelări recursive ale procedurii *QuickSort.*

Realizați N=100, 1000,... de simulări ale algoritmului de mai sus pentru vectori de lungime *l*=10,100,... și afișați valoarea medie a timpului de execuție.

Exemplu ilustrativ:



pivotul = elementul de pe poziția *r*
j=poziția curentă
i = ultima poziție din vectorul cu elemente mai mici sau egale decât valoarea *pivotului*
j -1 = ultima poziție din vectorul cu elemente mai mari sau egale decât valoarea *pivotului*

2. *Randomized-QuickSort*
Implementați în *Matlab* următorul algoritm de sortare a unui vector de numere A(p : r):
Divide: se alege aleator un element *pivot* din vectorul A(p : r) care se interschimbă cu elementul A(r), iar apoi se aplică procedura **Divide** din algoritmul *QuickSort* și se returnează q.
Impera: se sortează vectorii A(p : q-1) și A(q+1 : r) prin apelări recursive ale procedurii *Randomized-QuickSort*.

Realizați N=100, 1000,... de simulări ale algoritmului de mai sus pentru vectori de lungime *l*=10,100,... și afișați valoarea medie a timpului de execuție.
Comparați timpul mediu de execuție de la problema **1** cu cel de la problema **2**, atunci când fiecare vector este sortat cu ambii algoritmi.

3. *QuickSelect*
Implementați în *Matlab* următorul algoritm de aflare a celui de-al i-lea element cel mai mic dintr-un vector de numere distincte A(p : r), adică exact i-1 elemente din A(p : r) sunt mai mici decât elementul căutat, unde $p \leq i \leq r$ (se caută elementul de rang i din vectorul dat):
Divide: se aplică procedura **Divide** din algoritmul *QuickSort* și se returnează q.
Impera: dacă i=k, atunci A(q) este elementul al i-lea cel mai mic cerut, unde k= length(A(p : q)); dacă i< k, atunci se caută al i-lea cel mai mic element din vectorul A(p : q-1), iar dacă i>k, atunci se caută al (i-k)-lea cel mai mic element din vectorul A(q+1 : r), apelând recursiv *QuickSelect* pentru vectorul corespunzător uneia dintre cele două situații.

Realizați N=100, 1000,... de simulări ale algoritmului de mai sus pentru vectori de numere distincte de lungime *l*=10,100,... și afișați valoarea medie a timpului de execuție.

4. *Randomized-QuickSelect*
Implementați în *Matlab* următorul algoritm de aflare a celui de-al i-lea element cel mai mic dintr-un vector de numere distincte A(p : r) (se caută elementul de rang i din vectorul dat):
Divide: se alege aleator un element *pivot* din vectorul A(p : r) care se interschimbă cu elementul A(r), iar apoi se aplică procedura **Divide** din algoritmul *QuickSort* și se returnează q.
Impera: se aplică procedura **Impera** din algoritmul *QuickSelect*, apelând recursiv de această dată *Randomized-QuickSelect*.

Realizați N=100, 1000,... de simulări ale algoritmului de mai sus pentru vectori de numere distincte de lungime *l*=10, 100,... și afișați valoarea medie a timpului de execuție.

Funcții Matlab: randi (se poate folosi doar la versiunile mai noi Matlab), tic, toc