Sortarea rapidă (Quick Sort)

Fie tab un tablou cu n componente. Sortarea prin aceasta metoda are la baza urmatorul principiu: practic tabloul este ordonat cand pentru fiecare element din tablou v[poz], elementele situalte la stanga sunt mai mici v[poz] iar cele din dreapta sunt mai mari sau egale decat v[poz]. Sortarea tabloului tab decurge astfel:
La un moment dat se prelucreaza o secventa din vector cu indecsi cuprinsi intre p si u

- se ia una din aceste componente, fie ea **piv=v[p],** care se consideră element pivot;
- se fac în tablou interschimbări de componente, astfel încât toate cele mai mici decât valoarea pivot sa treaca în stânga acesteia, iar elementele cu valoare mai mare decât pivot sa treacă în dreapta; prin această operaţie se va deplasa şi valoarea pivot, astfel ca ea nu se va mai gasi pe pozitia initiala ci pe o pozitie corespunzatoare relatiei de ordine. Fie aceasta k. Atentie! In urma acestui set de operatii elementele din stanga sunt mai mici decat pivotal dar nu neaparat si in ordine. La fel cele din dreapta sunt mai mari dar nu neaparat si in ordine

Se continuă setul de operatii similare, aplicând recursiv metoda pentru zona de tablou situată în stânga componentei pivot şi pentru cea din dreapta acesteia;

- oprirea recursiei se face când lungimea zonei de tablou care trebuie sortată devine egala cu 1.

Fie secventa de vector:
Piv=7
Se compara pivotal cu v[u]. Se interschimba

7	4	2	8	5	9	3	10	1	6

Avansez cu primul: (4<7. Nu se fac interschimbari)

6	4	2	8	5	9	3	10	1	7

Se avanseaza la 2. : (2<7. Nu se fac interschimbari)

6	4	2	8	5	9	3	10	1	7

Se avanseaza cu primul: (8>7. Se interschimba)

6	4	2	8	5	9	3	10	1	7

si se devanseaza in partea dreapta. 7>1.Se interschimba

6	4	2	7	5	9	3	10	1	8

Se avanseaza in stanga. 5<7. Nu se interchimba.

6	4	2	1	5	9	3	10	7	8

Se avanseaza la 9. 9>7 Se interschimba

6	4	2	1	5	9	3	10	7	8

Se devanseaza din dreapta. 7 si 10 sunt in ordine

6	4	2	1	5	7	3	10	9	8

Se trece la 3 care se schimba cu 7.

6	4	2	1	5	7	3	10	9	8

U si p ajung la aceeasi valoare caz in care se incheie secvanta.

6	4	2	1	5	3	7	10	9	8

P=u STOP!

K=p
In continuare se aplica aceeasi secventa pe portiunile din stanga respective dreapta pivotului:

6	4	2	1	5	3

Si

10	9	8

Etc….

Complexitatea algoritmului QuickSort este **O(n*log(n)).**

Pentru a stabili complexitatea algoritmului Quick Sort aplicată unui tablou tab cu n componente, notăm cu C(n) numărul de comparaţii efectuate şi remarcăm că, la fiecare pas al algoritmului au loc n comparaţii însoţite de interschimbări ale elementelor şi două invocari recursive ale metodei de sortare, deci

C(n) = n + C(k)+C(n-k)

unde k este numărul de componente din zona din stânga a tabloului, iar C(k) si C(n-k) sunt complexităţile pentru cele două subzone care urmează a fi sortate recursiv. Situaţia este asemanatoare cu cea întalnită în cazul metodei MergeSort, cu deosebirea că, în acest caz, tabloul nu se mai împarte în două părţi egale.

Cazul cel mai favorabil ar fi când, la fiecare recursie, cele două subzone (cu elemente mai mici şi respectiv mai mari decat elementul pivot) ar fi egale. În acest caz, calculul complexităţii se face la fel ca la MergeSort şi deci complexitatea este O(n.log(n)).

Cazul cel mai defavorabil ar fi cel în care, la fiecare recursie, elementul pivot ar fi ales în mod atat de nefericit, încat una din cele două subzone obţinute după interschimbări ar fi vida. In acest caz

C(n) = n + C(n-1) = n + (n-1) + C(n-2) = ...

sau, continuand pana la C(1)
C(n) = n + (n-1) + (n-2) + ... + 1 = **(n+1)n/2**

şi deci complexitatea algoritmului este O(n2). Acest caz "nefericit" este însă foarte puţin probabil. Cel mai probabil este că, în realitate, complexitatea sa fie situată în jurul valorii O(n.log(n)).