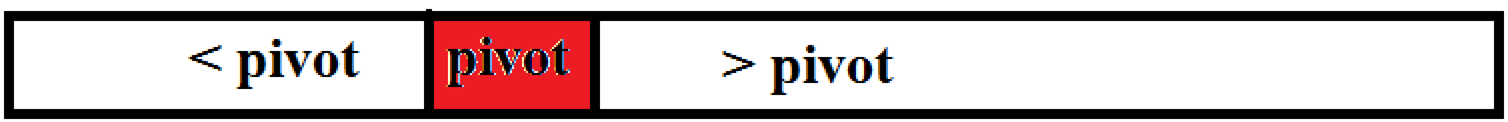
QUICKSORT (SORTAREA RAPIDĂ)

QUICKSORT

- In medie este cel mai rapid alg de sortare
- Sortare rapida = O(n log n)
- Inventat de C.A.R. Hoare in 1960.
- http://en.wikipedia.org/wiki/Quicksort
- http://ro.wikipedia.org/wiki/Quicksort

- Alege un element din lista, numit <u>pivot.</u>
- Rearanjeaza lista, prin interschimbari, astfel incat toate elementele mai mici decat pivotul sunt mutate inaintea lui, iar toate elementele mai mari sunt mutate dupa pivot.

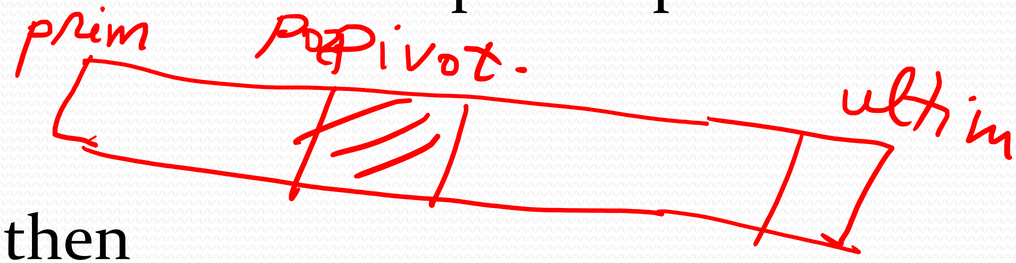


- Aplica recursiv algoritmul Quicksort pentru cele doua subliste: lista elementelor mai mici decat pivotul si lista elementelor mai mari decat pivotul.

- Exista mai multe metode de alegere a pivotului.
- Robert Sedgewick-(doctorand al lui Knuth) 1975 – a analizat mai multe metode.
- Alegem drept pivot primul element din lista ce trebuie sortata.
- Problema:
- Se dau $R_1, R_2, \ldots, R_N$ obiecte ce vor fi rearanjate astfel incat cheile lor $K_1, K_2, \ldots, K_N$ sa fie ordonate crescator.
- Algoritmul este recursiv pe principiul Divide et impera
- A sorta o lista data inseamna
  - a determina pozitia finala a pivotului,
  - a-l muta in aceasta pozitie in timp ce se fac interschimbarile necesare a.i. lista initiala sa fie aranjata dupa cum am precizat mai sus si
  - a sorta cele doua liste (L: {<pivot} si L : {>pivot})

Algoritmul Sortare rapida

- Quicksort (R, prim, ultim) /* Sorteaza elementele R_{prim}, ..., R_{ultim}. Apelarea initiala este pentru prim = 1 si ultim =N*/



- if prim < ultim then
- pozpivot=Partitie(R, prim, ultim)
- $R_{prim} \leftrightarrow R_{pozpivot}$
- call Quicksort (R, prim, pozpivot –1)
- call Quicksort (R, pozpivot +1, ultim)
- endif

Algoritmul Sortare rapida (Partitionarea)

Partitie(R, prim, ultim) /* Partitioneaza lista R_{prim}, …, R_{ultim} in doua subliste: o lista in care toate elementele au cheile <= Kprim si una in care toate au cheile >= Kprim si returneaza pozitia pivotului in lista ordonata. */

```
pivot = K_prim ; i=prim+1; j=ultim
while i<=j
        while i<=ultim and K_i <= pivot do    i++
        endwhile
        while j>=prim and K_j >pivot do  j--
        endwhile
        if i < j and i<=ultim and j>=prim then R_i ↔ R_j ,  i++,  j--
        endif
endwhile
return i-1
```

Exemplu

45 20 67 13 86 25 50

Prim =1

Ultim = 7

Pivot = K[1]= 45

i = 2, j=7

K[i]=20 <= pivot deci i=3

K[i]=67 > pivot deci nu se face i++

K[j]=50 >pivot deci j=6

K[j]=25 <pivot deci nu se face j--
cum i=3 < j=6 interschimbam K[3] cu K[6] -→ 45 20 25 13 86 67 50

Si i=4 j=5 i j

K[i]=13 <= pivot deci i=5

K[i]=86 > pivot deci nu se face i++

K[j]=86 >pivot deci j=4

K[j]=13 <pivot deci nu se face j--

Cum i=5 >j=4 se iese din while si se returneaza pozitia pivotului adica 4

Se interschima K[1] cu K[4] → 13 20 25 **45** 86 67 50

Se continua pentru listele 13, 20, 25

Si 86, 67, 50

Analiza algoritmului

- Se poate arata ca timpul mediu = O(N log N)
- Timpul maxim:

- Cand listele rezultate in urma partitiei au dimensiuni inegale.
- De exemplu daca lista este deja ordonata crescator atunci nu se face nicio interschimbare pivotul ramanand in prima pozitie iar listele rezultate sunt una vida si una cu N-1 elemente.

Analiza algoritmului (cont.)

- Operatiile pe care le luam in calcul sunt numai comparatiile desfasurate in functia Partitie, care pt o lista cu N elemente sunt in numar de N.
- Deci obtinem deci urmatoarea relatie de recurenta:

Pentru C_N = numarul de comparatii necesar sortarii unei liste cu N elemente

$C_N = N + C_{N-1}$, daca $N > 1$ si

$C_1 = 0$

- deci $C_N = N + (N-1) + \ldots + 1 = N(N+1)/2$

Deci timpul maxim este de ordin N^2.