SORTAREA PRIN INTERCLASARE (MERGESORT)

- Sortarea prin interclasare = O(n log n)(chiar si timpul maxim) dar necesita un spatiu de memorie suplimentar de ordin O(n).
- Inventat de John von Neumann in 1945.
- http://en.wikipedia.org/wiki/Merge_sort
- http://ro.wikipedia.org/wiki/Mergesort

mergesort (L)

mijloc.

Descriere alg

mergesort L_1 L_2 mergesort (L_2)

- De tip metoda Divide et Impera

L_1 ordonată L_2 ordonată

- Lista inițială se împarte în două liste de dimensiuni egale prin determinarea mijlocului listei inițiale și alegerea celor două liste delimitate de acesta.

interclasare

Lista finală ordonată

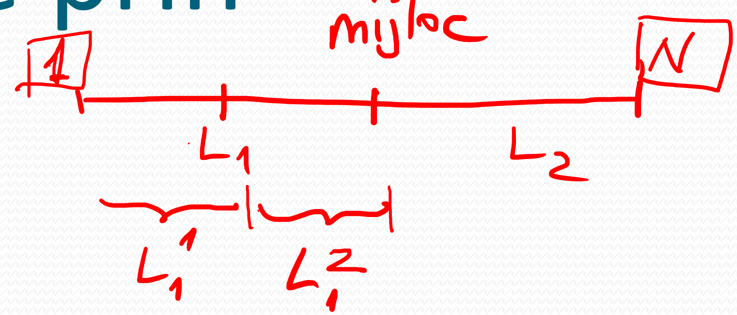
- Celor două liste li se aplică acelaşi algoritm Mergesort pentru a le ordona
- Se aplică algoritmul de interclasare a doua liste ordonate pentru combinarea lor și finalizarea ordonării listei inițiale.

Mergesort

- Problema:

  Se dau R_1, R_2, …, R_N obiecte ce vor fi rearanjate astfel incat cheile lor K_1, K_2, …, K_N sa fie ordonate crescator..
- Algoritmul este recursiv pe principiul Divide et impera.
- Se foloseste un vector suplimentar C de dimensiune N pentru stocarea partiala a elementelor ordonate in fiecare etapa.

Algoritmul de sortare prin interclasare



N = nr elem ale vectorului

- <u>Mergesort (K, prim, ultim)</u>
- // Sorteaza lista $K_{prim}, \ldots, K_{ultim}$. Initial prim = 1 , ultim =N
- // Se foloseste
-   if prim < ultim then
-     mijloc = [(prim +ultim)/2]
-     call Mergesort(K, prim, mijloc) ← L_1
-     call Mergesort(K, mijloc +1, ultim) ← L_2
-     call Interclasare($K_{prim} \leq \ldots \leq K_{mijloc}$ şi $K_{mijloc+1} \leq \ldots \leq K_{ultim}$)
-     $\rightarrow \boxed{C_{prim}, \ldots, C_{ultim}}$
-     for i = prim, ultim
-       $K_i = C_i$
-     endfor
-   endif

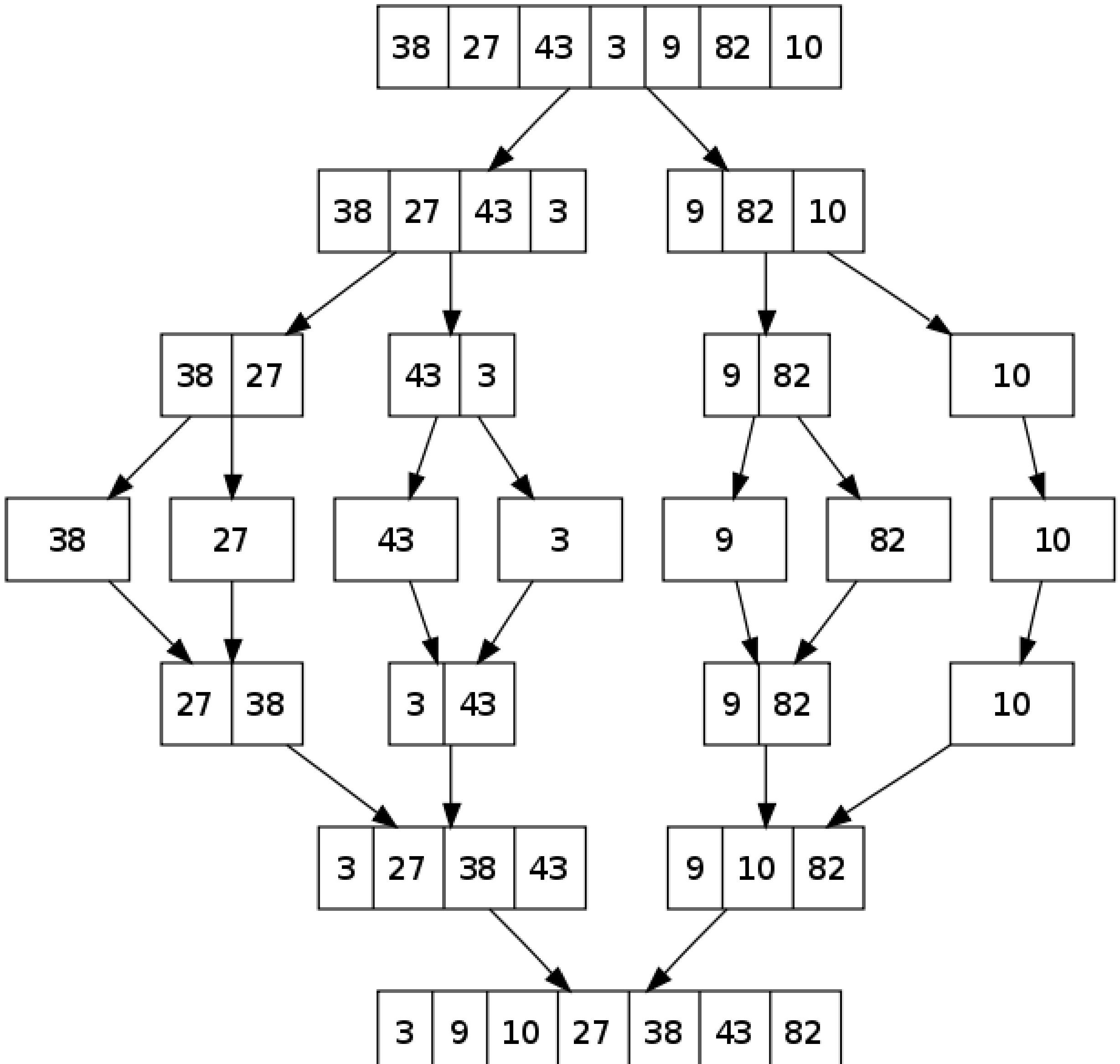
$mijloc = \left[\frac{prim+ultim}{2}\right]$

Algoritmul Interclasare(continuare)

- Algoritm Interclasare: Se dau listele A: $A_1 <= A_2 <= ... <= A_n$ și B: $B_1 <= B_2 <= ... <= B_m$. Se combină cele două liste și se obține lista C astfel încât $C_1 <= C_2 <= ... <= C_{n+m.}$

```
iterA = 1, iterB = 1, iterC = 1
// atat timp cat nici una din liste nu s-a terminat de parcurs
while iterA ≤ n and iterB ≤ m
        if A_iterA < B_iterB then C_iterC = A_iterA
                                  iterA = iterA +1
                          else  C_iterC = B_iterB
                                  iterB = iterB +1
    endif
        iterC = iterC +1
endwhile
// daca lista A nu s-a terminat se copiaza restul de elemente in lista C
while iterA ≤ n
        C_iterC = A_iterA
    iterA = iterA +1
        iterC = iterC +1
endwhile
// daca lista B nu s-a terminat se copiaza restul de elemente in lista C
while iterB ≤ m
        C_iterC = B_iterB
    iterB = iterB +1
        iterC = iterC +1
endwhile
```

Exemplu



38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82

Analiza algoritmului Interclasare

- Operaţia principală = comparaţia dintre elementele celor două liste.
- Să observăm întâi că algoritmul Interclasare va compara un element al listei A cu un element al listei B până când una din liste se termină.
- Ce se întâmplă dacă toate elementele listei A sunt mai mici decât cele din lista B? În acest caz, fiecare element al lui A va fi comparat cu B_1 deci numărul de comparaţii efectuate va fi n.
- Similar, dacă toate elementele listei B sunt mai mici decât cele din lista A atunci numărul de comparaţii efectuate va fi m.

Analiza algoritmului Interclasare

- Se poate arăta că <u>cel mai bun caz</u> al acestui algoritm este chiar unul din cele două cazuri și anume cel pentru care

$\{ n \le m$ și $A_1 <= A_2 <= \ldots <= A_n \le B_1 <= B_2 <= \ldots <= B_m\}$

sau cel pentru care $\{ m \le n$ și $B_1 <= B_2 <= \ldots <= B_m \le A_1 <= A_2 <= \ldots <= A_n\}$.

Deci numărul de comparaţii în cel mai bun caz este min(n, m).

Analiza algoritmului Interclasare

- Să considerăm acum cazul în care elementele listei A sunt printre elementele listei B, cu alte cuvinte
- $B_1 <= A_1 <= B_2 <= A_2 <= B_3$, etc.
- În acest caz numărul comparaţiilor este n + m – 1, caz ce corespunde <u>celui mai rău caz.</u> Deci numărul maxim de comparaţii este n + m – 1.
- Observăm de asemenea că algoritmul Interclasare necesită spaţiu de memorie suplimentar de mărime m + n.

Analiza algoritmului Mergesort

- Operaţia principală = comparaţia dintre elementele listei ce se efectuează în cadrul algoritmului de interclasare.

- Cunoscând ordinul de complexitate al algoritmului Interclasare, putem analiza algoritmul Mergesort.

- Notăm $C^{min}(N)$ = numărul de comparaţii efectuate în cel mai bun caz
  $C^{max}(N)$ = numărul de comparaţii efectuate în cel mai rău caz.
- Conform analizei anterioare, algoritmul Interclasare, aşa cum este aplicat în algoritmul de sortare Mergesort, va efectua un număr minim de comparaţii =

$$\min\left(\left\lfloor\frac{n+1}{2}\right\rfloor, n-\left\lfloor\frac{n+1}{2}\right\rfloor\right)=\left\lfloor\frac{n}{2}\right\rfloor$$

min (m, n)

- şi un număr maxim de comparaţii =

n+m-1.

$$\left\lfloor\frac{n+1}{2}\right\rfloor+\left(n-\left\lfloor\frac{n+1}{2}\right\rfloor\right)-1=n-1$$

- Din descrierea algoritmului Mergesort ce implică două apelări recursive ale sale pentru liste mai mici şi o apelare a algoritmului Interclasare, se obţin următoarele relaţii de recurenţă :

Analiza algoritmului (cont.)

nr. min comp pt interclasare

$$C^{min}(n) = C^{min}\left(\left\lceil\frac{n}{2}\right\rceil\right) + C^{min}\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + \left\lfloor\frac{n}{2}\right\rfloor \text{ pentru } n \geq 1 \text{ și } C^{min}(1) = 0$$

$$C^{max}(n) = C^{max}\left(\left\lceil\frac{n}{2}\right\rceil\right) + C^{max}\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n - 1 \text{ pentru } n \geq 1 \text{ și } C^{max}(1) = 0$$

$$C^{min}(n) = 2C^{min}\left(\frac{n}{2}\right) + \left\lfloor\frac{n}{2}\right\rfloor, \quad C^{min}(1) = 0.$$

$\boxed{k = \log_2 n}$

- Se poate arăta că *$C^{max}(n)$* = *$O(n \log n)$* şi că *$C^{min}(n)$* = *$O(n \log n)$*. Deci timpul mediu este tot $O(n \log n)$.

$\boxed{n = 2^k}$

$$C^{min}(2^k) = 2C^{min}(2^{k-1}) + 2^{k-1} = 2\left[2C^{min}(2^{k-2}) + 2^{k-2}\right] = 2^2 C^{min}(2^{k-2}) + 2^{k-1} = \ldots\ldots$$