Sortarea externă

Sortare externă = ordonarea unei mulțimi de date ce nu încape în memoria internă

- Nu se realizează într-un singur pas

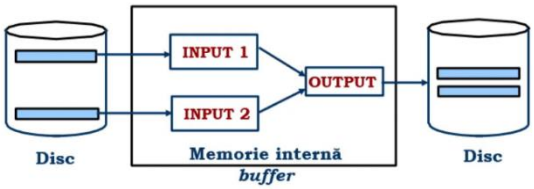
Faze:

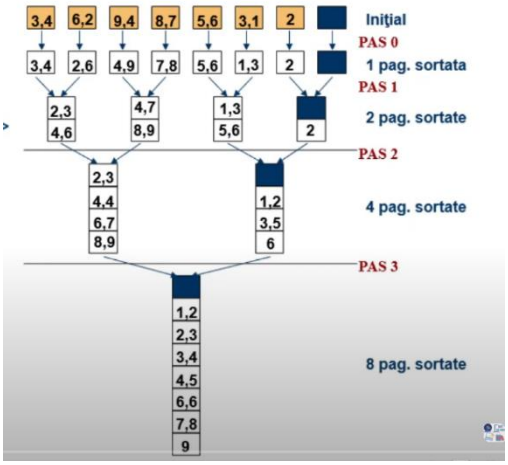
1. Se împart datele în monotonii – siruri ordonate
2. Se interclasează monotoniile într-un singur șir complet sortat

- Monotoniile vor fi cât mai lungi posibil – avem mai puțin de interclasat
- În fiecare fază se va paraleliza cât mai mult citirea datelor de intrare, procesarea și salvarea datelor
- Se utilizează cât mai multă memorie internă posibil

Sortarea prin interclasarea a două șiruri

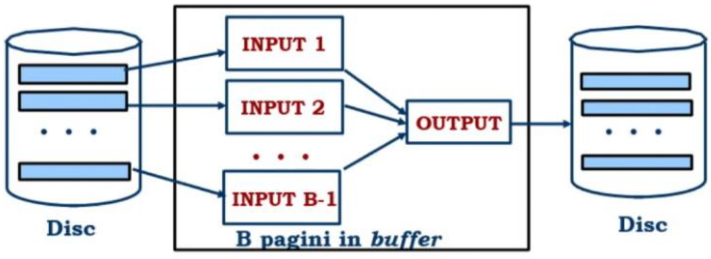
- Necesită exact 3 pagini în buffer
- Pas 0 (formare de monotonii): citește o pagină -> sorteaza -> salveaza pagina (se folosește o singură pagină din buffer)
- Pași 1,2,..., etc (interclasare): se folosesc 3 pagini:
  - Citește două pagini și în output o punem pe cea mai mică (comparăm datele)
- Dacă la prima trecere avem 100 de pagini, la a doua trecere vom avea 50 monotonii cu 2 pagini, la a treia 25 monotonii cu cate 4 pagini etc
- La fiecare pas se citește și scrie fiecare pagină
- N pagini de sortat => nr de pași = $[\log_2 N]+1$
- Costul total este $2N([\log_2 N]+1)$
- Ideea: Divide et impera





Sortare externă generalizată

- Sortarea a N pagini folosind B pagini din buffer:
  - Pas 0: se folosesc B pagini din buffer. Se produc [N/B] monotonii a câte B pagin i fiecare
  - Pas 1,2,...,etc: interclasează B-1 monotonii



Costul sortării externe:

- Număr de pași: $1+[\log_{B-1}[N/B]]$
- Cost = 2N * (număr pași)
- Ex: cu 5 pagini de buffer pt a sorta 108 pagini de date:
  - Pas 0: [108/5] = 22 monotonii a cate 5 pagini fiecare (ultimul contine doar 3 pagini)
  - Pas 1: [22/4] = 6 monotonii a cate 20 de pages fiecare (ultimul contine doar 8 pagini)
  - Pas 2: 2 monotonii, 80 de pagini si 28 pagini
  - Pas 3: toate cele 108 pagini sortate
  -
- Număr de pași în sortarea externă:

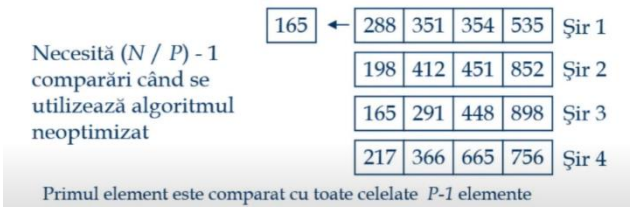
N	B=3	B=5	B=9	B=17	B=129	B=257
100	7	4	3	2	1	1
1,000	10	5	4	3	2	2
10,000	13	7	5	4	2	2
100,000	17	9	6	5	3	3
1,000,000	20	10	7	5	3	3
10,000,000	23	12	8	6	4	3
100,000,000	26	14	9	7	4	4
1,000,000,000	30	15	10	8	5	4

Variații are sortării externe

- Optimizări:
  - Noi algoritmi ca Interclasare polifazică, Interclasare în cascadă
  - Reducerea numărului de paşi intermediari prin implementarea unei interclasări a n monotonii deodată, cu valori mari pentru n.
  - Optimizări prin distribuirea perfectă a monotoniilor pe mediul de stocare.
  - Maximizarea vitezei prin creşterea numărului de dispozitive de stocare (pentru a minimiza timpul de acces).
- Dezavantaje: costuri adiţionale

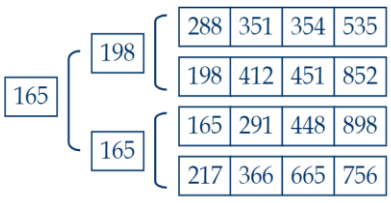
Arbore de selecție

- **Problemă:** selectarea celui mai mic element este consumator de timp
- **Soluție:** construirea unui arbore de selecție elimină o bună parte din comparări și grăbește procesul de selecție (doar $\log_2 P$ comparări sunt necesare)

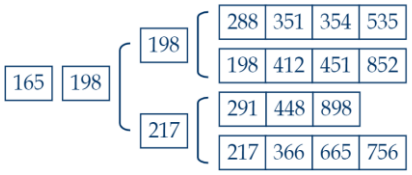


- Întotdeauna cele mai mici elemente sunt preluate din vârful arborelui. Elementele noi sunt "împinse" în faţă. Procesul se repetă până când tot arborele se goleşte

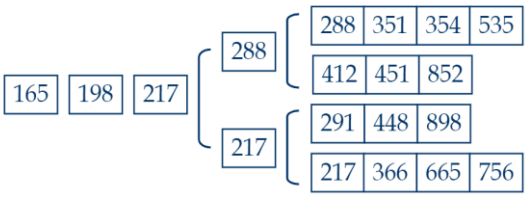
Start: Construirea unui arbore de selecție



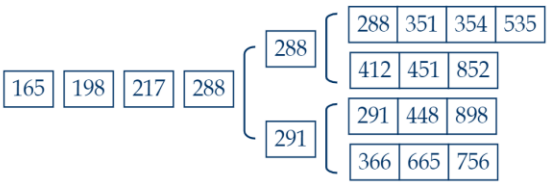
Pas 1: Extragerea celui mai mic element



Pas 2: Extragerea celui mai mic element

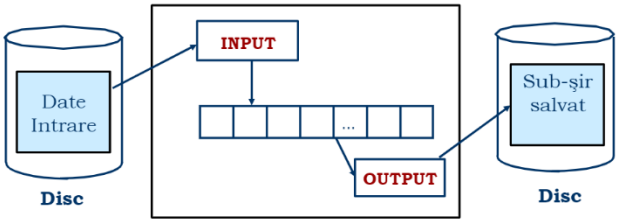


Pas 3: Extragerea celui mai mic element



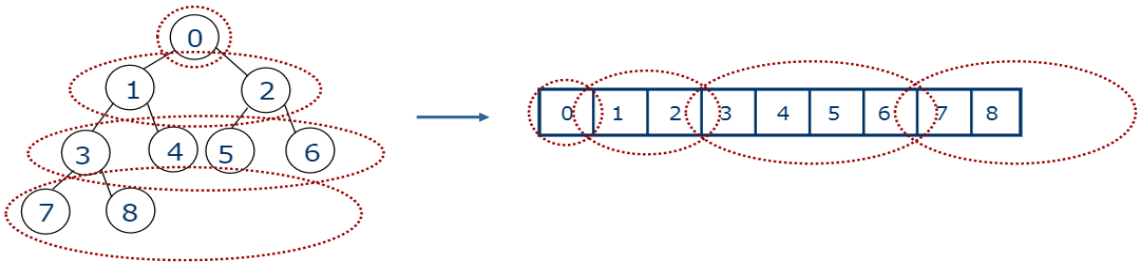
Algoritm de sortare internă

- Pentru sortarea internă poate fi utilizat Quicksort
- **Replacement selection**
  - Bazat pe folosirea unor arbori binari completi unde valoarea fiecărui nod e mai mică decât valoarea nodurilor fiu (min-heap)
  - Memoria internă conţine spaţiu pentru min-heap, o pagină de intrare şi una pentru rezultat.

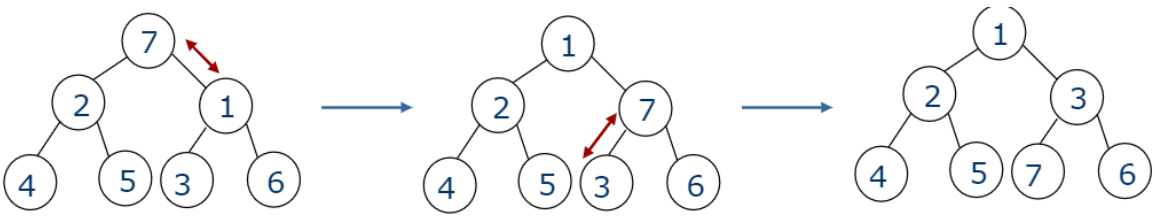


Min-Heap

- Arbore binar complet: dacă înălţimea arborelui este d, atunci toate nivelele arborelui sunt complete (excepţie ar putea face nivelul d). Nivelul d conţine toate nodurile din partea stângă
- Valorile sunt ordonate parţial.
- Reprezentarea în memorie: sub formă de şir de elemente ce conţine secvenţa de noduri pe nivele de la stânga la dreapta

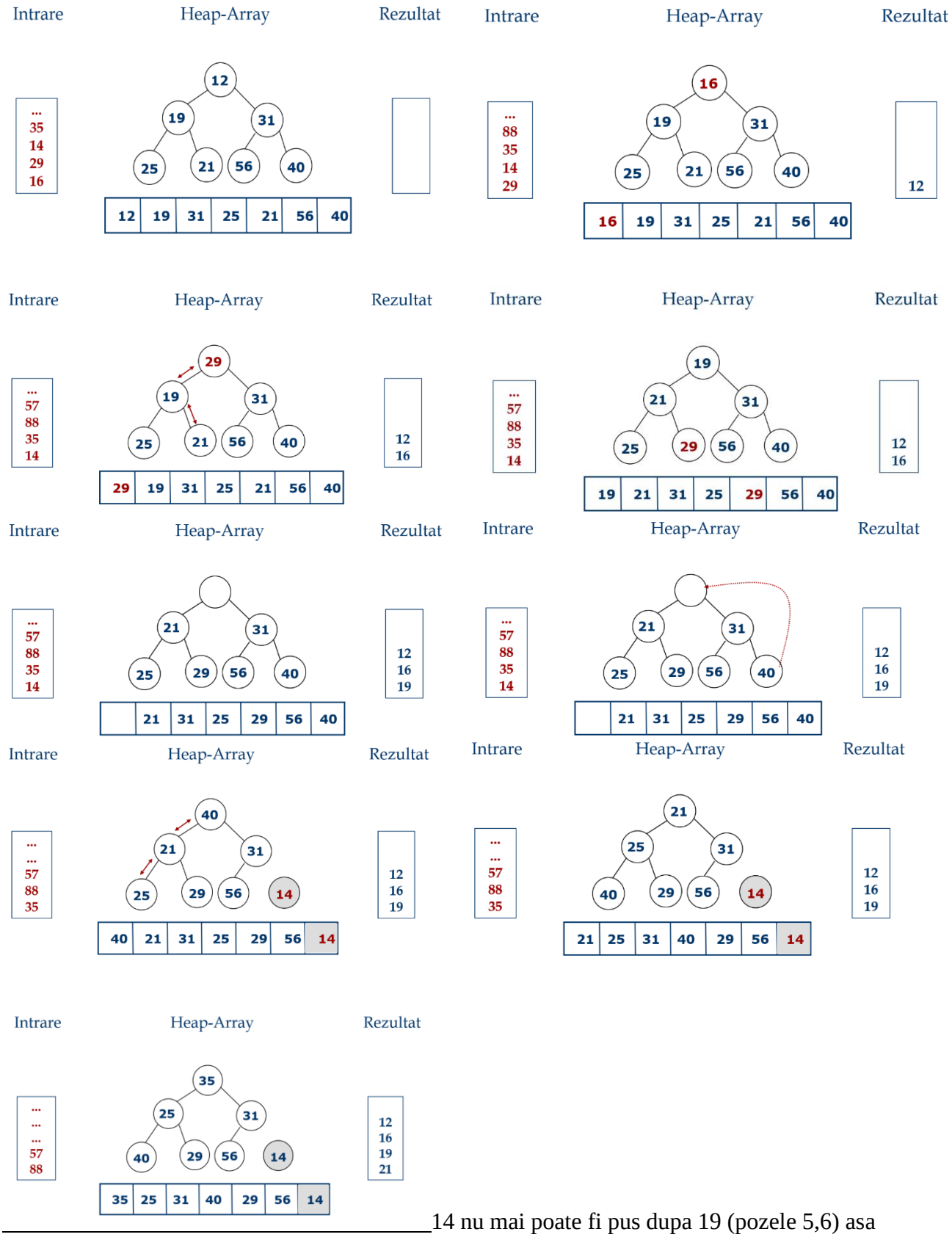


- Construcţie:
  - De la nivelul superior la cel inferior.
  - Poziţionează fiecare element la locul său (siftdown).
  - Operaţiile de poziţionare se termină când avem ordine parţială sau când am ajuns la frunze
  - Exemplu: poziţionare corectă a elementului 7



Algoritmul Replacement Selection

- Formăm monotonii inițiale mai lungi (in Rezultat putem avea mai multe decat in heap)



14 nu mai poate fi pus dupa 19 (pozele 5,6) asa ca aducem valoarea cea mai din dreapta in radacina si punem 14 in locul lui (pozele 6, 7,8)

I/O pentru sortarea externă

- ... monotonii mai lungi înseamnă mai puţini paşi de sortare
- Am considerat că se citeşte/scrie o pagină la un moment dat. În realitate se citeşte un bloc de pagini secvenţiale!
- În buffer se poate rezerva câte un bloc de pagini pentru intrări şi rezultate.
    - Acest lucru va reduce numărul de pagini disponibile pentru sortare internă
    - În practică, majoritatea tabelelor se sortează în 2-3 paşi

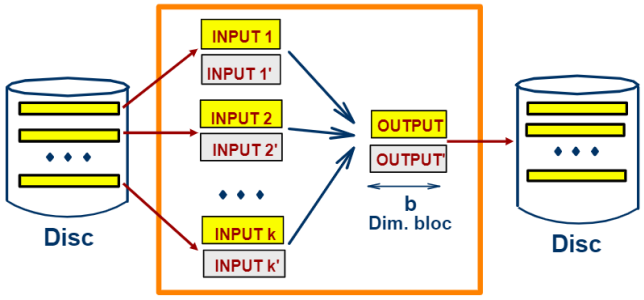
N	B=1,000	B=5,000	B=10,000
100	1	1	1
1,000	1	1	1
10,000	2	2	1
100,000	3	2	2
1,000,000	3	2	2
10,000,000	4	3	3
100,000,000	5	3	3
1,000,000,000	5	4	3

* *Dimensiune bloc = 32, pasul iniţial produce subşiruri de dimensiune 2B.*

Număr de paşi pentru sortarea optimizată

Double Buffering

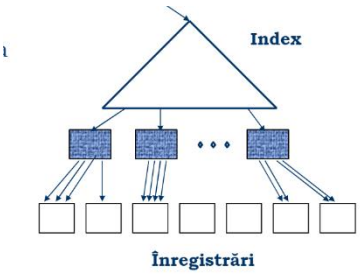
- Pentru a reduce timpul de citire/scriere, se pot folosi pagini suplimentare pentru citire/scriere în avans (prefetch).
    1. Potential, mai mulţi paşi; înpractică, majoritatea tabelelor continuă să se sorteze în 2-3 paşi.

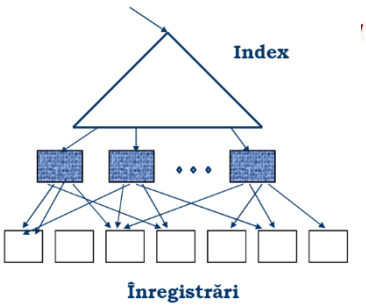


B locaţii in buffer, interclasare generală

Utilizarea arborilor B+ pentru sortare

- Scenariu: tabela se sortează pe baza unui index pe câmpurile de sortare, structurat ca un B-arbore.
- Ideea: Obţinerea înregistrărilor prin traversarea valorilor din frunze.
- Cazuri:
    1. B-arborele este grupat→Perfect!
        - Cost: parcurgerea arborelui până la cea mai din stânga frunză
        - Fiecare pagină e parcursă o singură dată
    2. B-arborele nu este grupat→f ineficient
        - În general, o citire pe pagină de înregistrare!





Sortare externă vs. index neclusterizat

N	Sortare	p=1	p=10	p=100
100	200	100	1,000	10,000
1,000	2,000	1,000	10,000	100,000
10,000	40,000	10,000	100,000	1,000,000
100,000	600,000	100,000	1,000,000	10,000,000
1,000,000	8,000,000	1,000,000	10,000,000	100,000,000
10,000,000	80,000,000	10,000,000	100,000,000	1,000,000,000

* *p*: # număr de înregistrări pe pagină
* *B=1,000 şi dimensiune bloc=32 pt sortare*
* *p=100 este o valoare mai mult decât realistă*