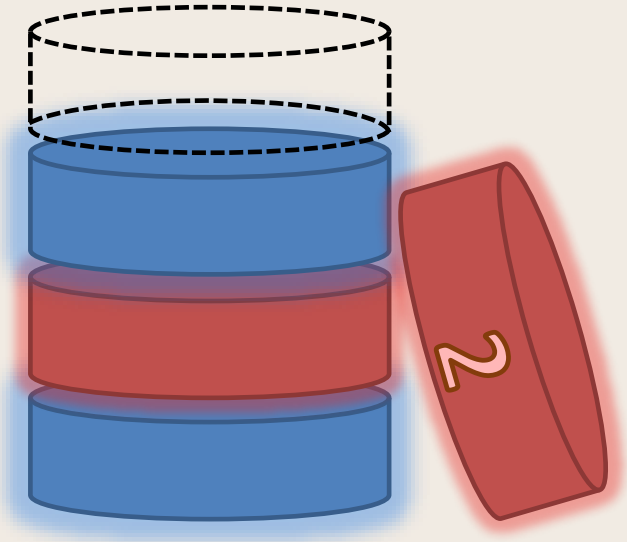
Sortare externă



Sortare

- O problemă clasică în informatică!
- Este critică pentru bazele de date:
  - Afișare informații într-o anumită ordine
  - Eliminarea duplicărilor
  - Grupare
  - Executarea *join*-ului între mai multe tabele

Sortare

- *Quick Sort, Heap Sort, Selection Sort,...*
  - Foarte eficienți, dar presupun că toate datele încap complet în memoria internă.

Sortare externă

- Problemă: sortare 1Tb de date având 1Gb de RAM.
- Sortare externă: ordonarea unei mulțimi de date ce nu încape în memoria internă
- Sortarea externă nu se realizează într-un singur pas

Sortare externă

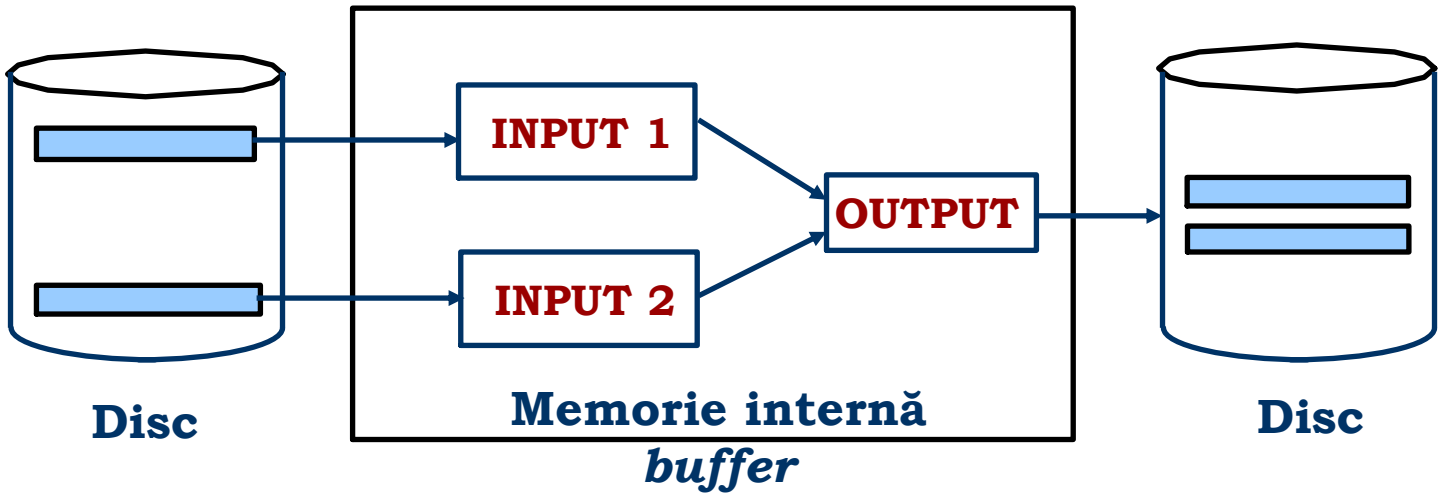
- Faze:
  1. Se împart datele în monotonii.
  2. Se interclasează monotoniile într-un singur şir complet sortat

Principii generale

- Monotoniile vor fi cât mai lungi posibil.
- În fiecare fază se va paraleliza cât mai mult posibil citirea datelor de intrare, procesarea şi salvarea datelor
- Se utilizează cât mai multă memorie internă posibil

Sortarea prin interclasarea a două şiruri

- Necesită exact 3 pagini în *buffer*
- Pas 0: citeşte o pagină → sortează → salvează pagina.
  - se foloseşte o singură pagină din *buffer*
- Paşi 1, 2, …, etc:
  - se folosesc 3 pagini:



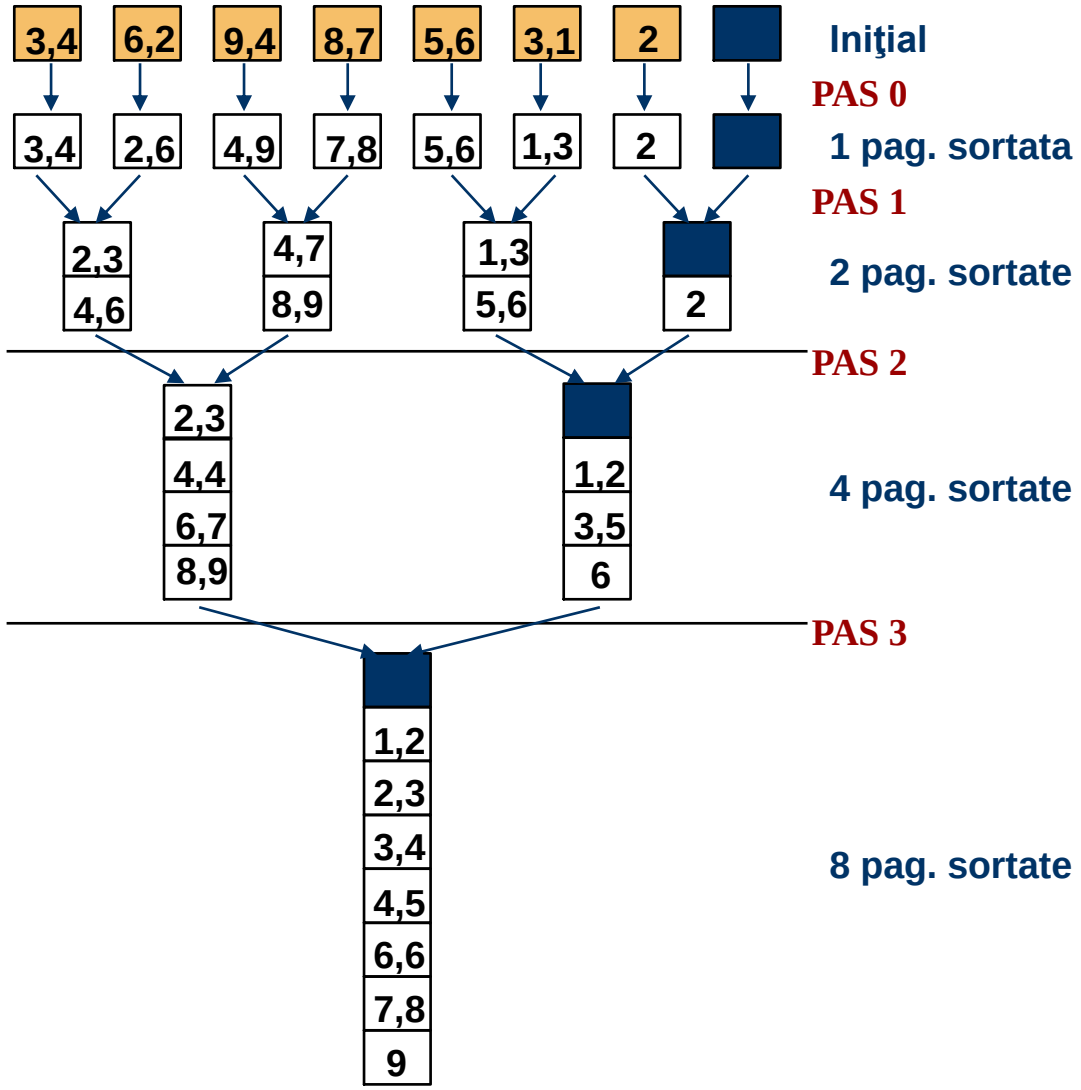
Detalierea interclasării

- La fiecare pas se citeşte şi scrie fiecare pagină
- N pagini de sortat => numărul de paşi = $\lceil \log_2 N \rceil + 1$
- Deci costul total este:

  $2N(\lceil \log_2 N \rceil + 1)$

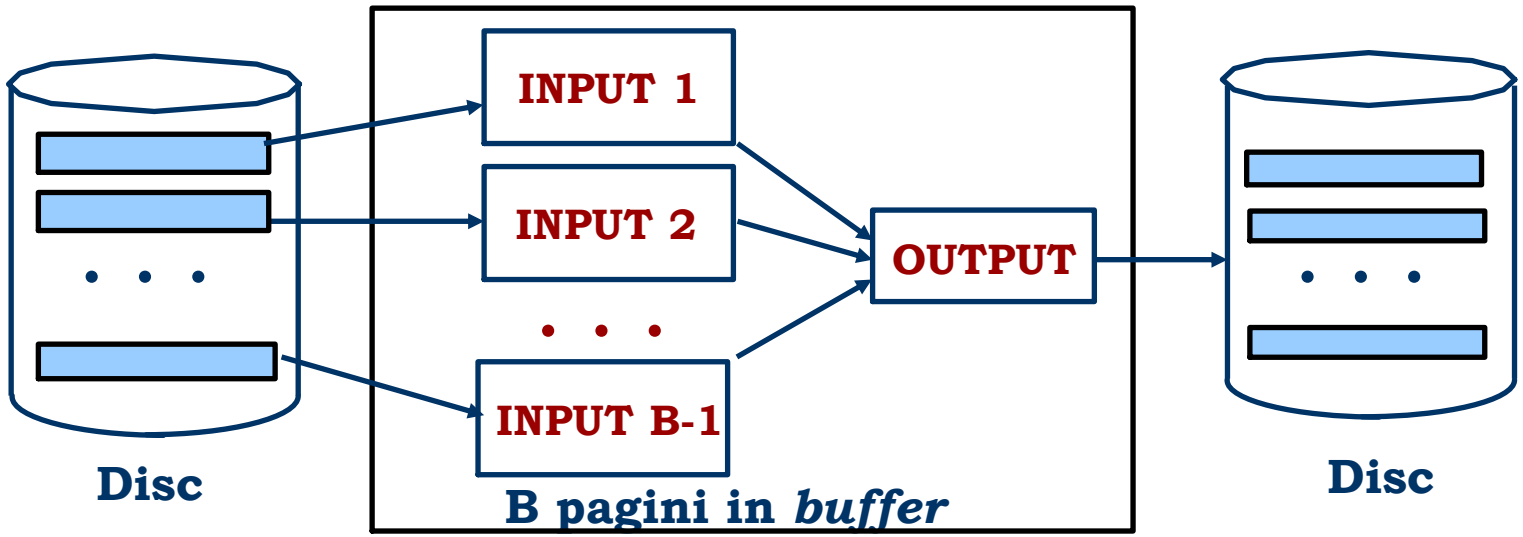
- *Ideea:*

  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* pagini fiecare.
  - Pas 1, 2, …, etc.: interclasează *B-1* monotonii.



Costul sortării externe

- Număr de paşi : $1+\lceil \log_{B-1} \lceil N/B \rceil \rceil$
- Cost = 2N * (număr de paşi)
- Ex. cu 5 pagini de *buffer* pentru a sorta 108 pagini de date:
  - Pas 0: $\lceil 108/5 \rceil = 22$ monotonii a câte 5 pagini fiecare (ultimul conține doar 3 pagini)
  - Pas 1: $\lceil 22/4 \rceil = 6$ monotonii a câte 20 pagini fiecare (ultimul conține doar 8 pagini)
  - Pas 2: 2 monotonii, 80 pagini şi 28 pagini
  - Pas 3: toate cele 108 pagini sortate

Număr de paşi 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 ale sortării externe

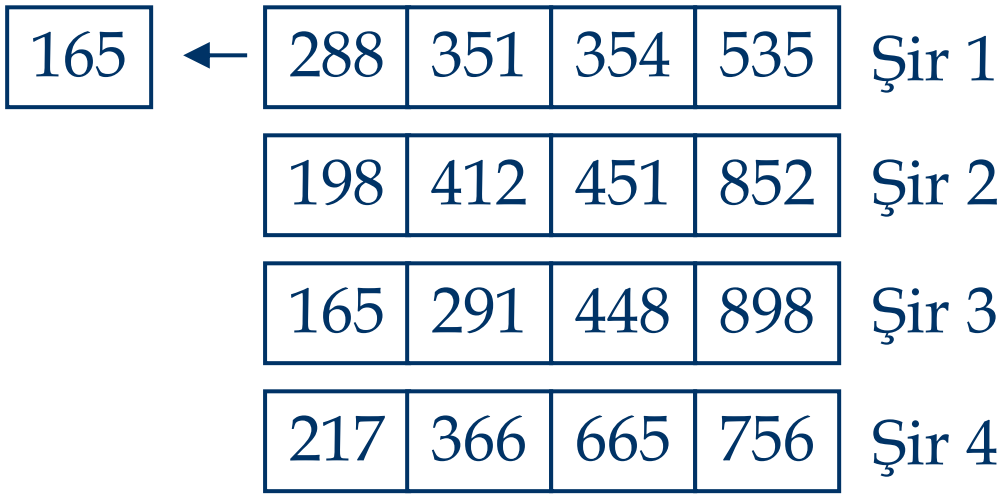
- **Optimizări:**
  - Noi algoritmi ca *Interclasare polifazică, Interclasare în cascadă*
  - Reducerea numărului de paşi intermediari prin implementarea unei interclasări a *n* monotonii deodată, cu valori mari pentru *n*.
  - Optimizări prin distribuirea perfectă a monotoniilor pe mediul de stocare.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

- **Dezavantaje:** costuri adiționale

Arbore de selecție

- **Problemă**: selectarea celui mai mic element este consumator de timp

Necesită (N / P) - 1 comparări când se utilizează algoritmul neoptimizat

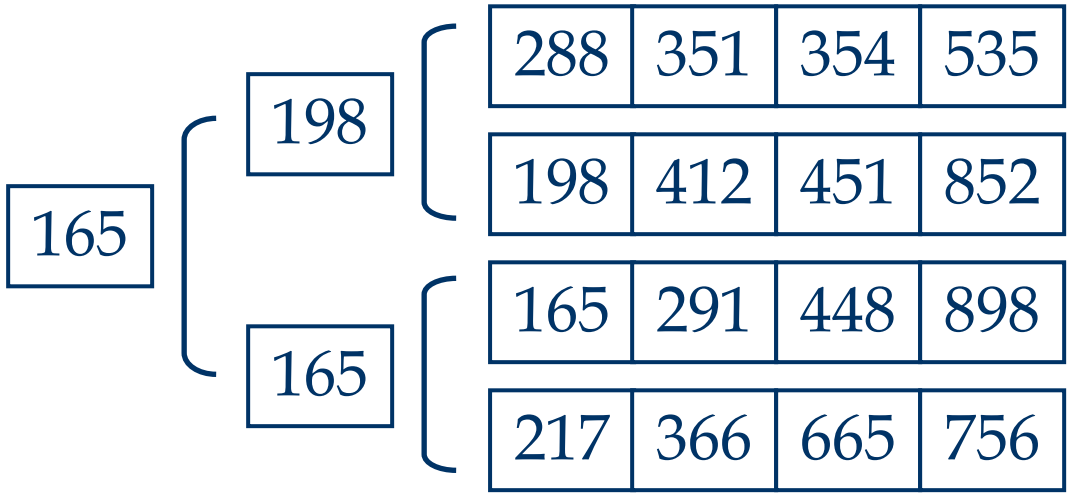


Primul element este comparat cu toate celelate *P-1* elemente

- **Soluție :** Construirea unui *arbore de selecție* elimină o bună parte din comparări şi grăbeşte procesul de selecție (doar $log_2 P$ comparări sunt necesare)

Arbore de selecție

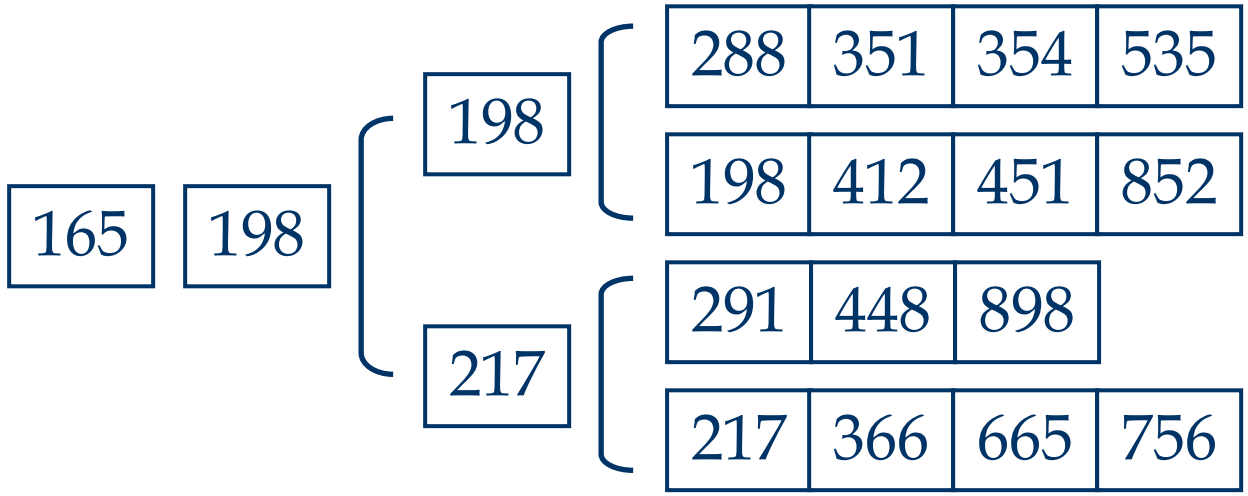
Start: Construirea unui arbore de selecție



Întotdeauna cele mai mici elemente sunt preluate din vârful arborelui
Elementele noi sunt “împinse” în față
Procesul se repetă până când tot arborele se goleşte

Arbore de selecție

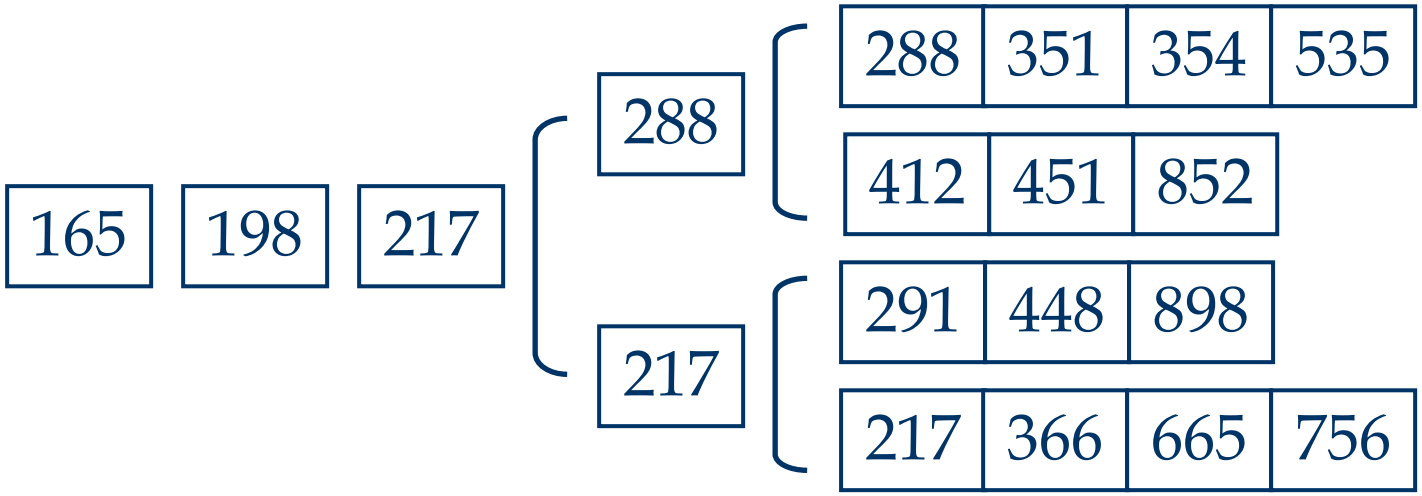
Pas 1: Extragerea celui mai mic element



Întotdeauna cele mai mici elemente sunt preluate din vârful arborelui
Elementele noi sunt “împinse” în față
Procesul se repetă până când tot arborele se goleşte

Arbore de selecție

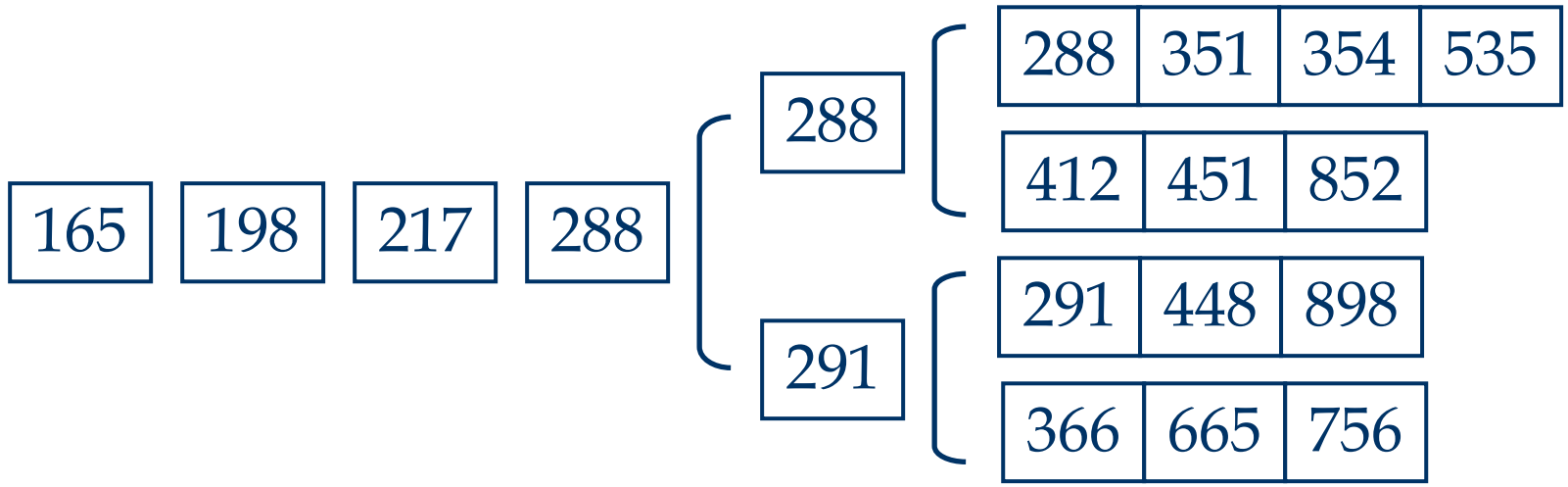
Pas 2: Extragerea celui mai mic element



Întotdeauna cele mai mici elemente sunt preluate din vârful arborelui
Elementele noi sunt “împinse” în față
Procesul se repetă până când tot arborele se goleşte

Arbore de selecție

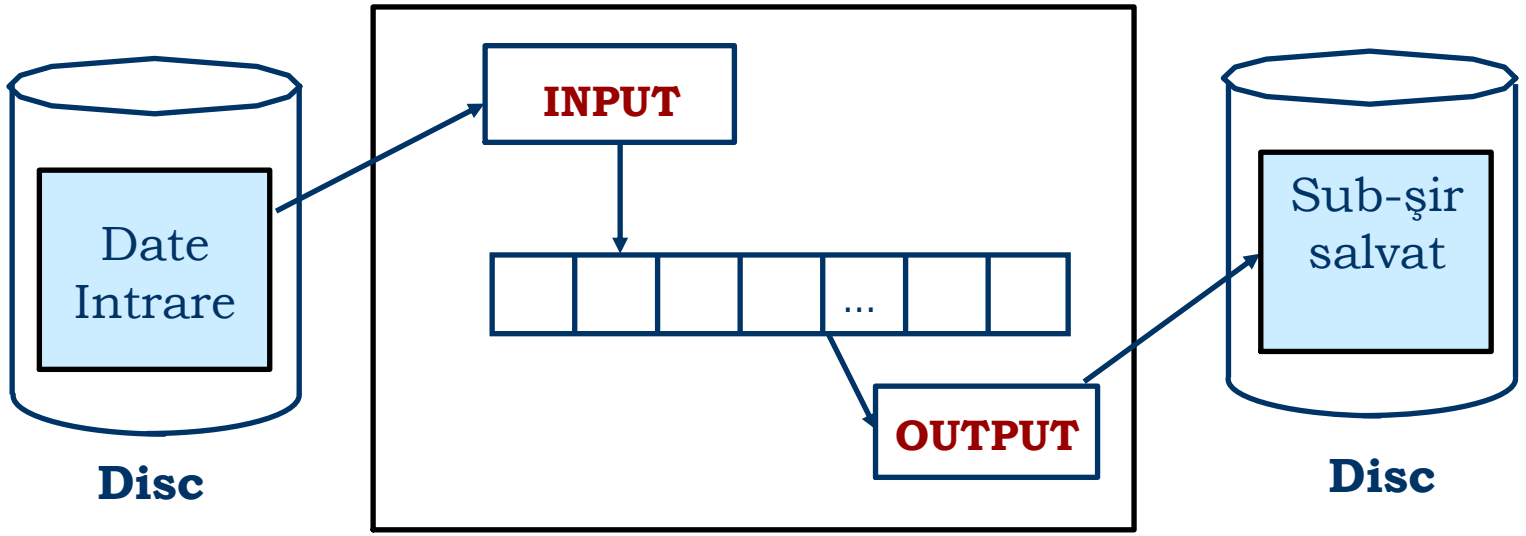
Pas 3: Extragerea celui mai mic element



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

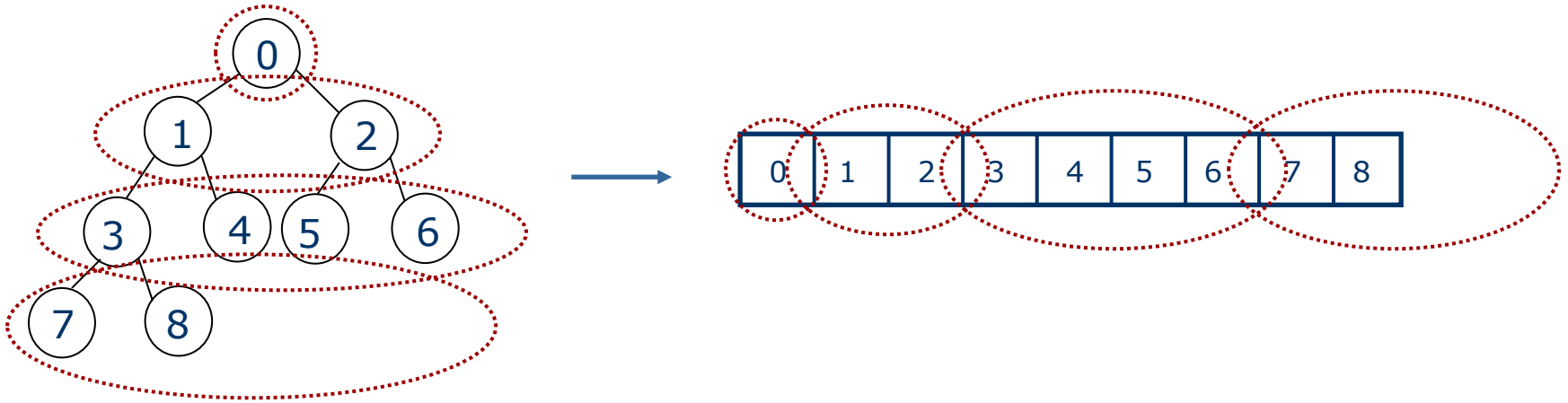
Algoritm de sortare internă

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



Min-Heap

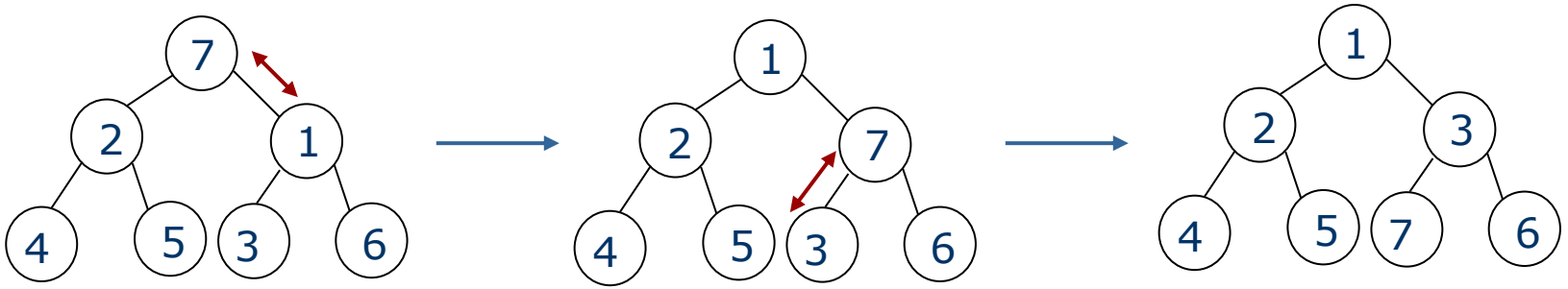
- Arbore binar complet: dacă înălțimea arborelui este *d*, atunci toate nivelele arborelui sunt complete (excepție ar putea face nivelul *d*). Nivelul *d* conține toate nodurile din partea stângă
- Valorile sunt ordonate parțial.
- Reprezentarea în memorie: sub formă de şir de elemente ce conține secvența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



Min-Heap

- Construcție:
  - De la nivelul superior la cel inferior.
  - Poziționează fiecare element la locul său (*siftdown*).
  - Operațiile de poziționare se termină când avem ordine parțială sau când am ajuns la frunze

  - Exemplu: poziționare corectă a elementului 7

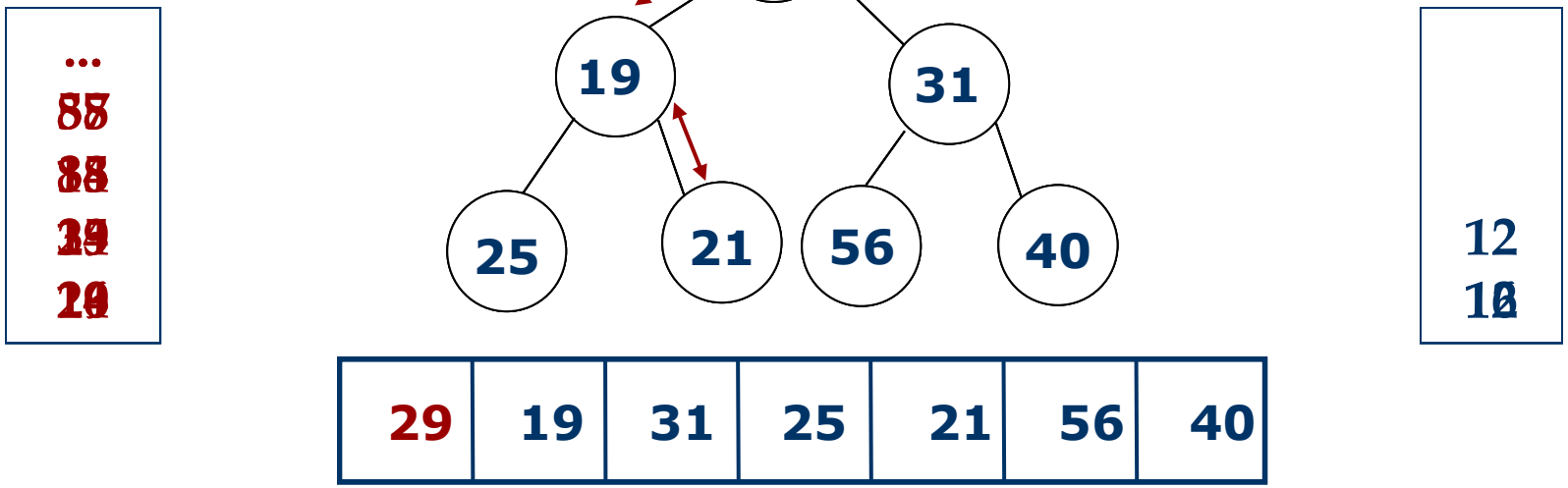


Algoritmul *Replacement Selection*

Intrare

Heap-Array

Rezultat

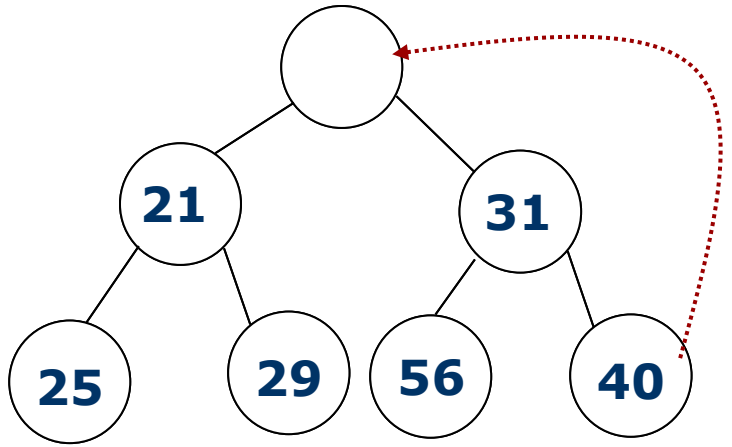


Algoritmul *Replacement Selection*

Intrare

...
57
88
35
14

Heap-Array



	21	31	25	29	56	40

Rezultat

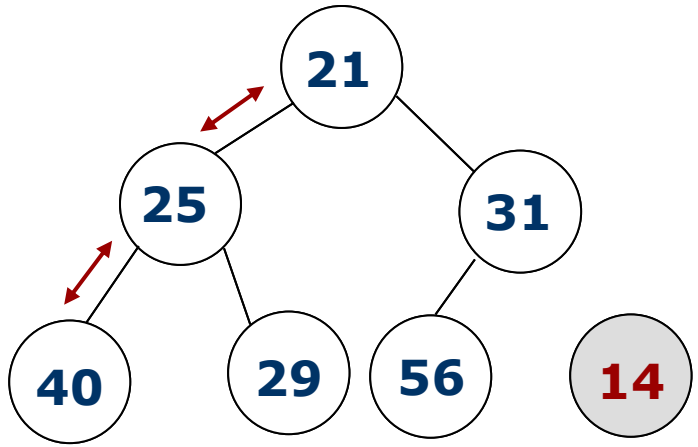
12
16
19

Algoritmul *Replacement Selection*

Intrare

...
...
57
88
35

Heap-Array



21	25	31	40	29	56	14

Rezultat

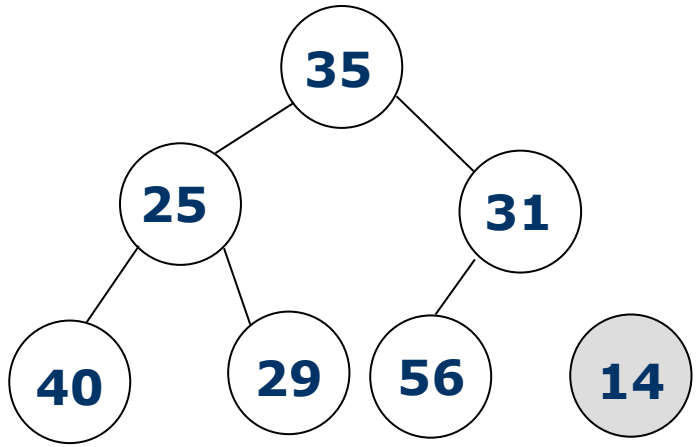
12
16
19

Algoritmul *Replacement Selection*

Intrare

...
...
...
57
88

Heap-Array



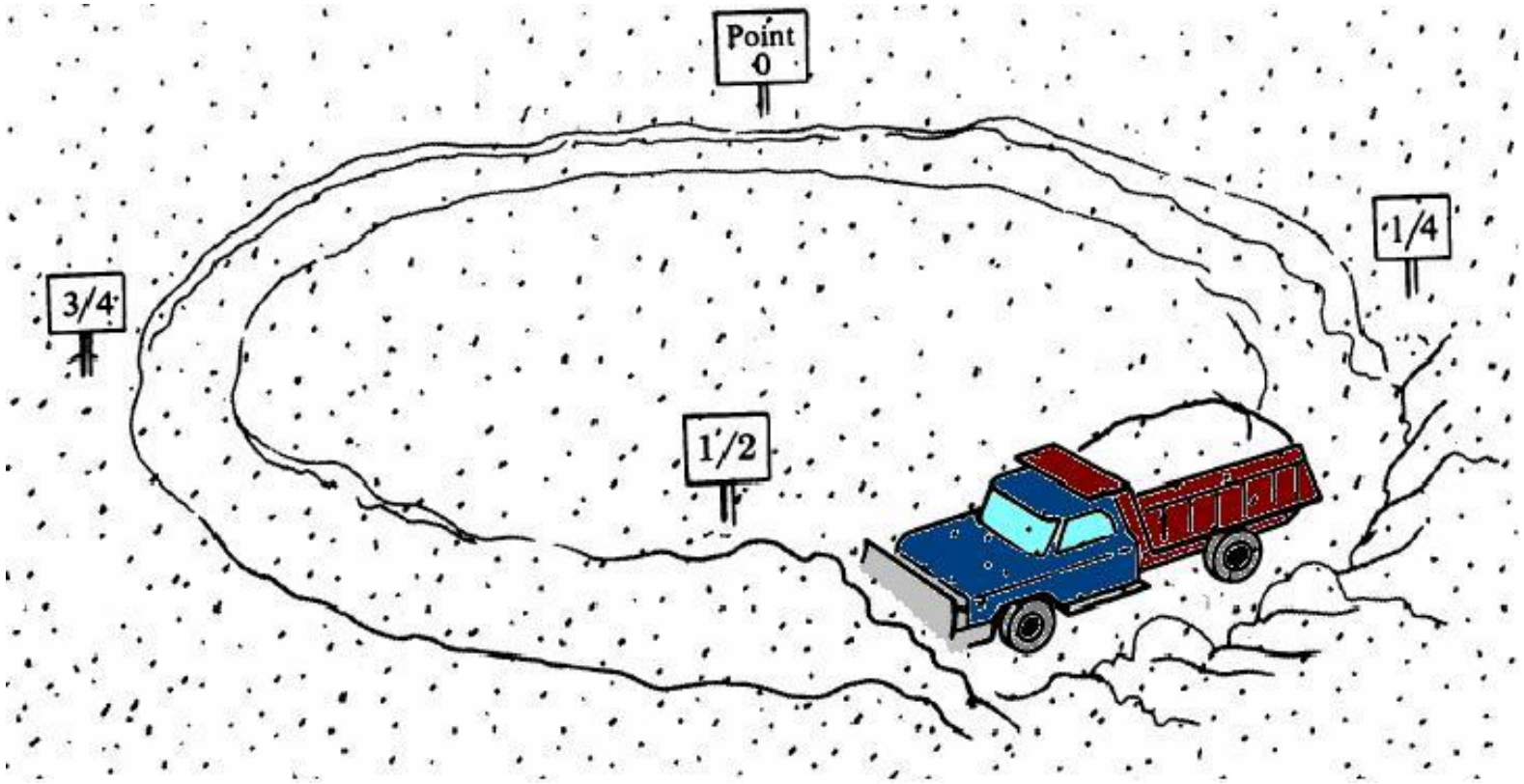
35	25	31	40	29	56	14

Rezultat

12
16
19
21

Replacement Selection – analogia cu plug de zăpadă

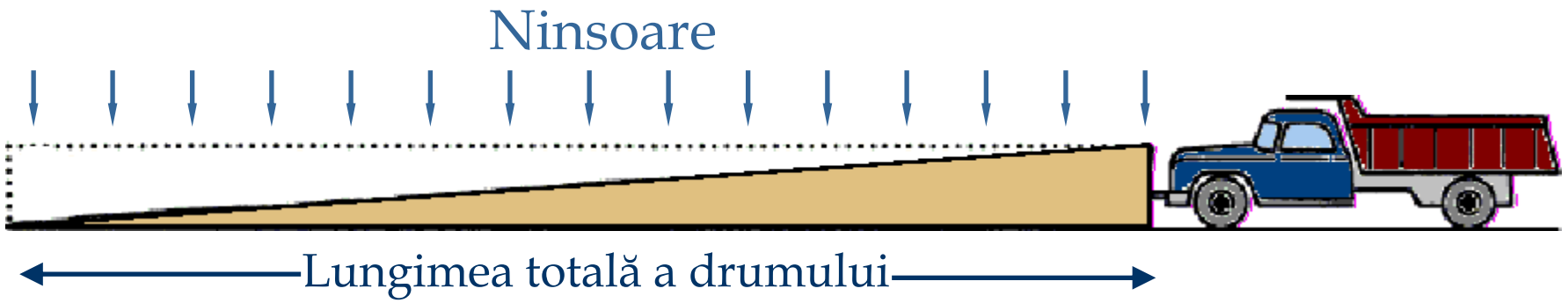
- Se pot forma monotonii inițiale de lungime $2 * q$, unde q e dimensiune *buffer*-ului



Un plug de zăpadă curăță drumul de zăpada ce cade aleator peste tot

Replacement Selection – analogia cu plug de zăpadă

- Deoarece zăpada cade cu viteză constantă, această situație stabilă nu se va modifica:



- Dreptunghiul este tăiat pe jumătate de linia ce reprezintă nivelul actual al zăpezii
- Nivelul actual al zăpezii reprezintă elementele din memorie
- După o parcurgere nu mai este zăpadă din tura precedentă
- Un subşir s-a sortat, urmează sortarea unui nou subşir.
- Volumul de zăpadă îndepărtat la o trecere (adică, lungimea unui subşir) este de două ori cantitatea de zăpadă existentă pe drum în orice moment de timp.

I/O pentru sortarea externă

- … monotonii mai lungi înseamnă mai puțini paşi de sortare
- Am considerat că se citeşte/scrie o pagină la un moment dat. În realitate se citeşte un *bloc* de pagini secvențiale!
- 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 .

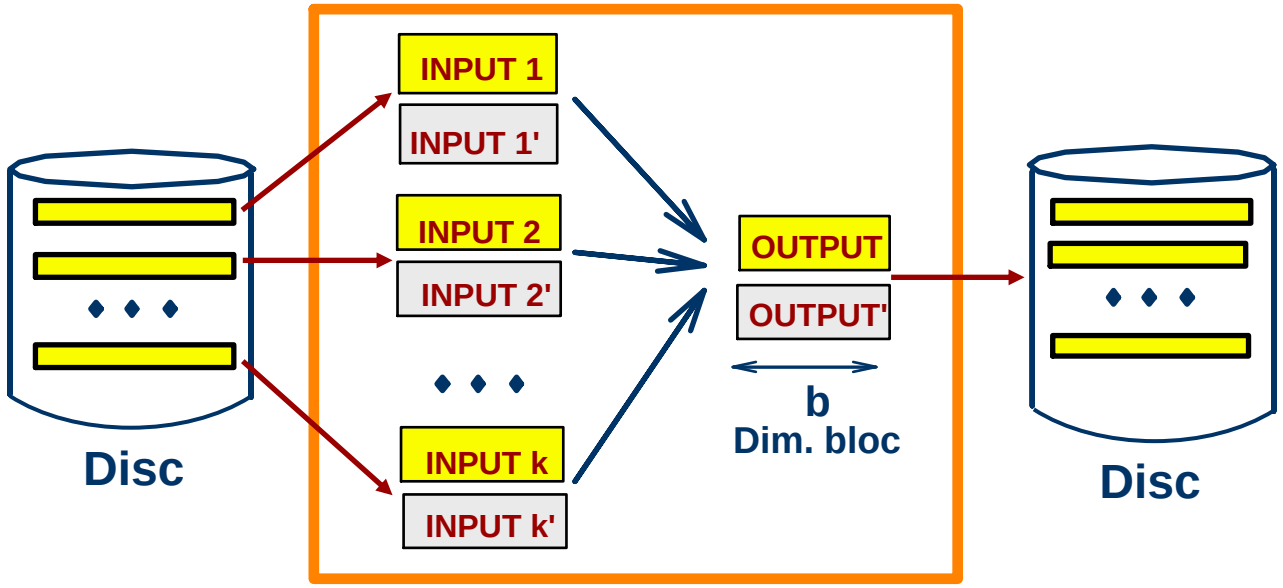
Număr de paşi pentru sortarea optimizată

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 inițial produce subşiruri de dimensiune 2B.*

Double Buffering

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



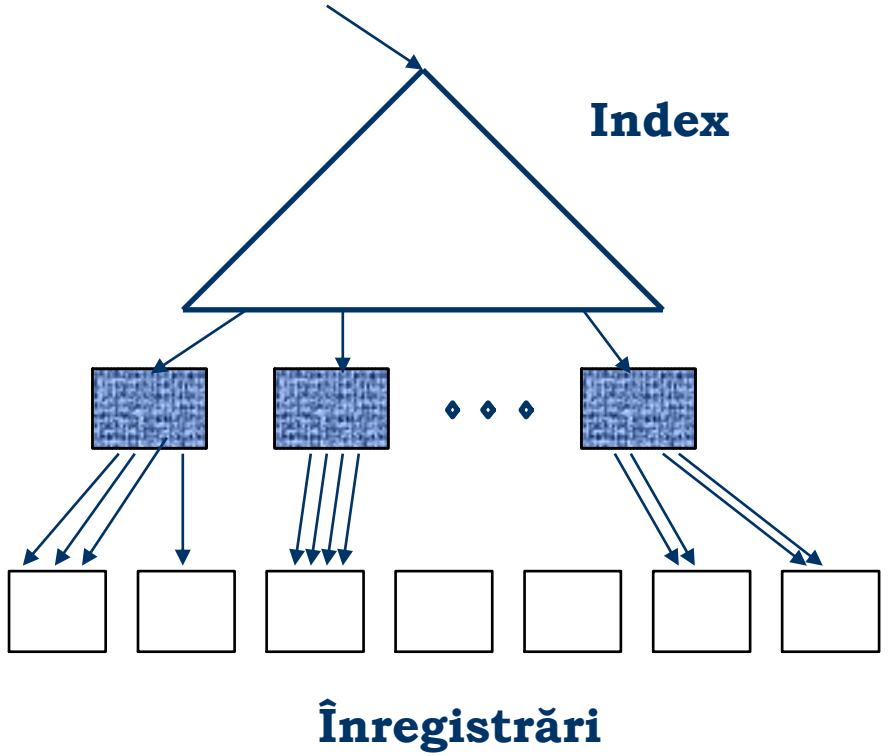
B locaţii in buffer, interclasare generală

Utilizarea arborilor B+ pentru sortare

- Scenariu: tabela se sortează pe baza unui index pe câmpurile de sortare, structurat ca un B-arbore.
- Ideea: Obținerea înregistrărilor prin traversarea valorilor din frunze.
- Cazuri:
  - B-arborele este grupat → ***Perfect!***
  - B-arborele nu este grupat → ***f ineficient***

B-arbore grupat folosit la sortare

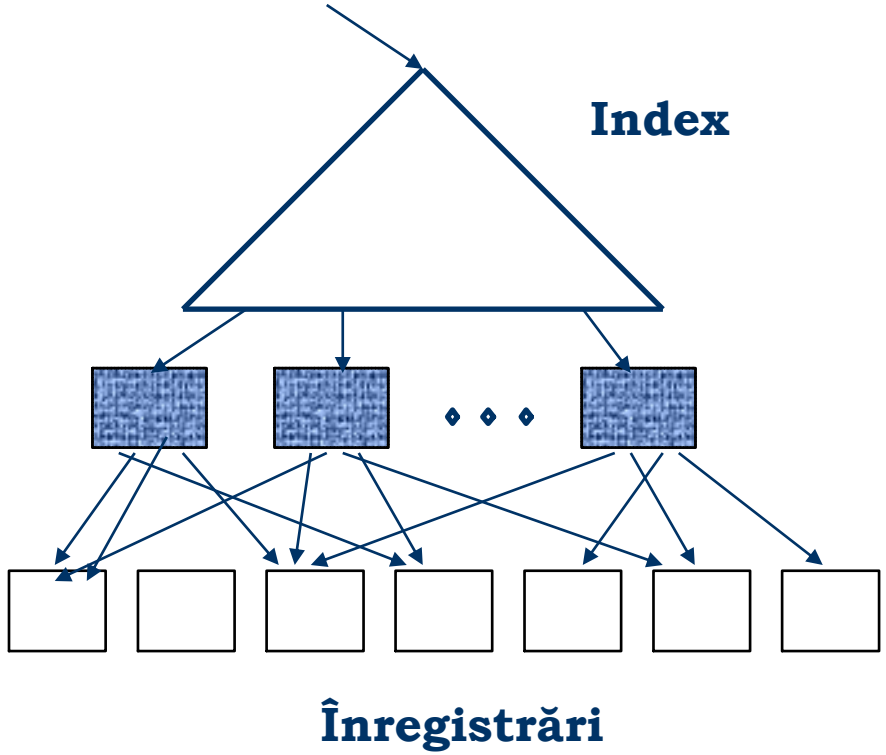
- Cost: parcurgerea arborelui până la cea mai din stânga frunză
- Fiecare pagină e parcursă o singură dată



* *variantă superioară sortării externe!*

B-arbore negrupat folosit la sortare

- În general, o citire de pagină pe 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***