Arhitecturi Paralele

Abordarea algoritmilor în mod paralel

Dr. Ing. Cristian Chilipirea – cristian.chilipirea@gmail.com

Rank Sort

9	4	2	7	6	5	6	1

1	2	4	5	6	6	7	9

Rank Sort

9	4	2	7	3	5	6	1

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Rank Sort

9	4	2	7	3	5	6	1

0

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

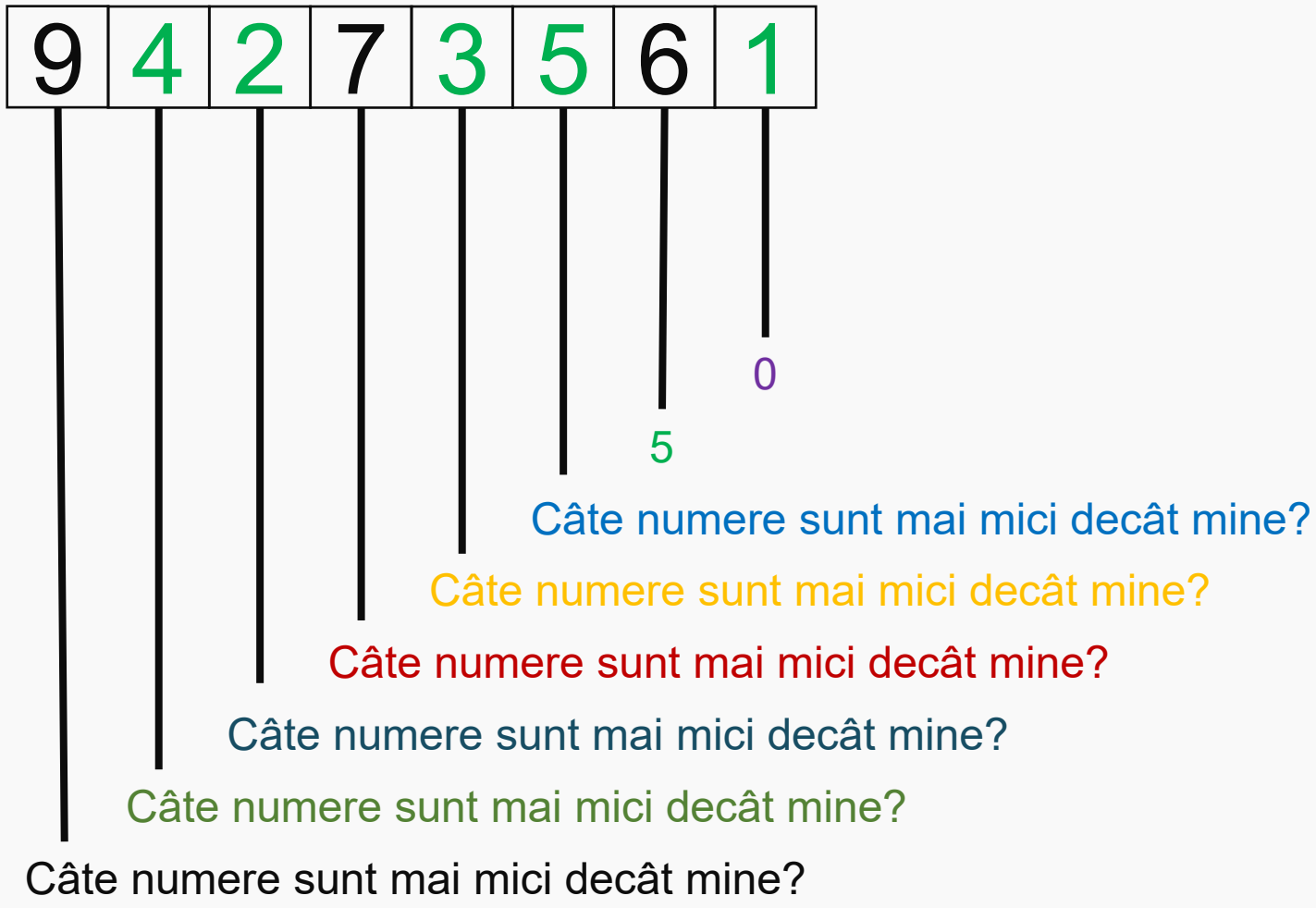
Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

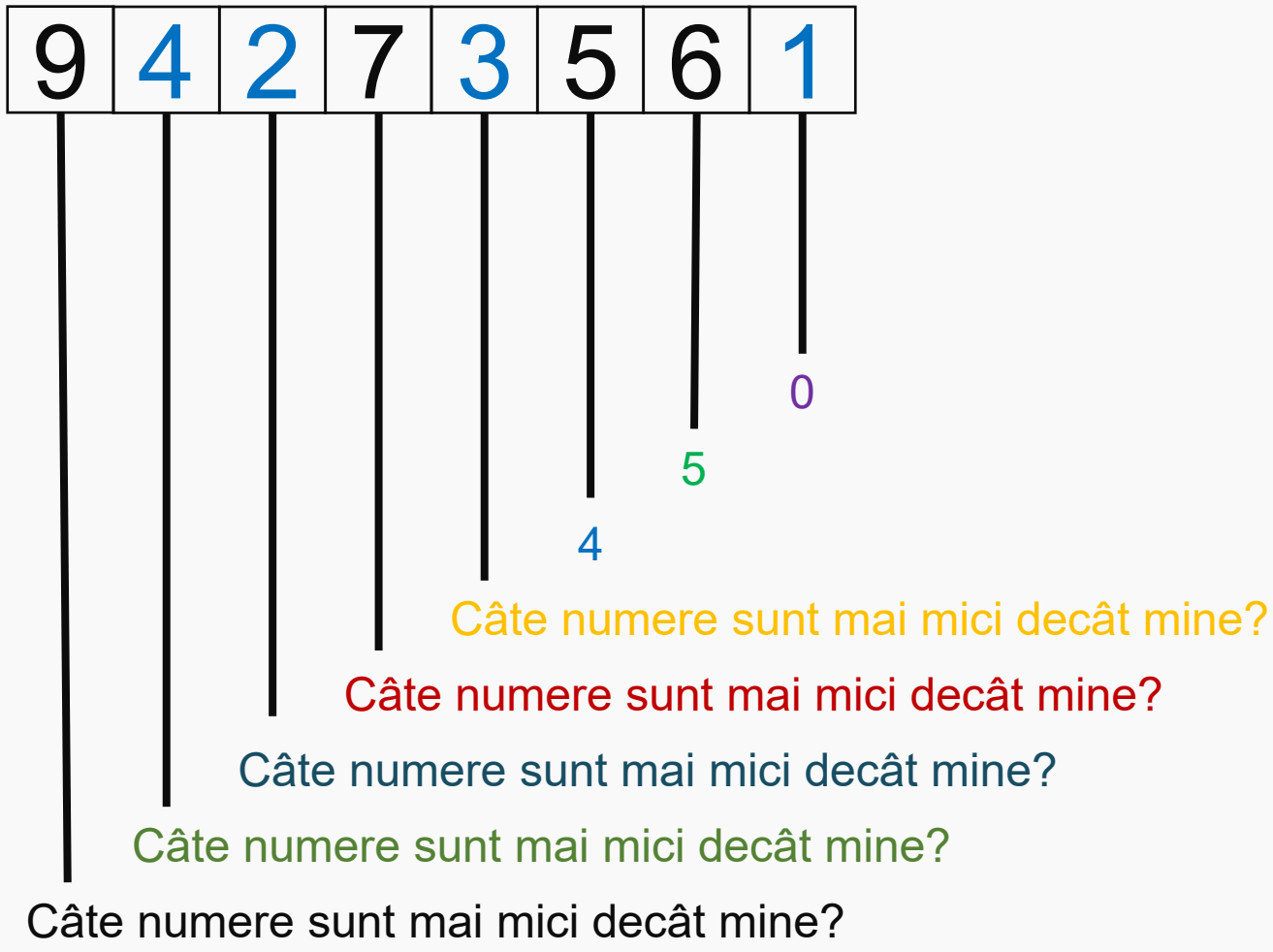
Câte numere sunt mai mici decât mine?

Câte numere sunt mai mici decât mine?

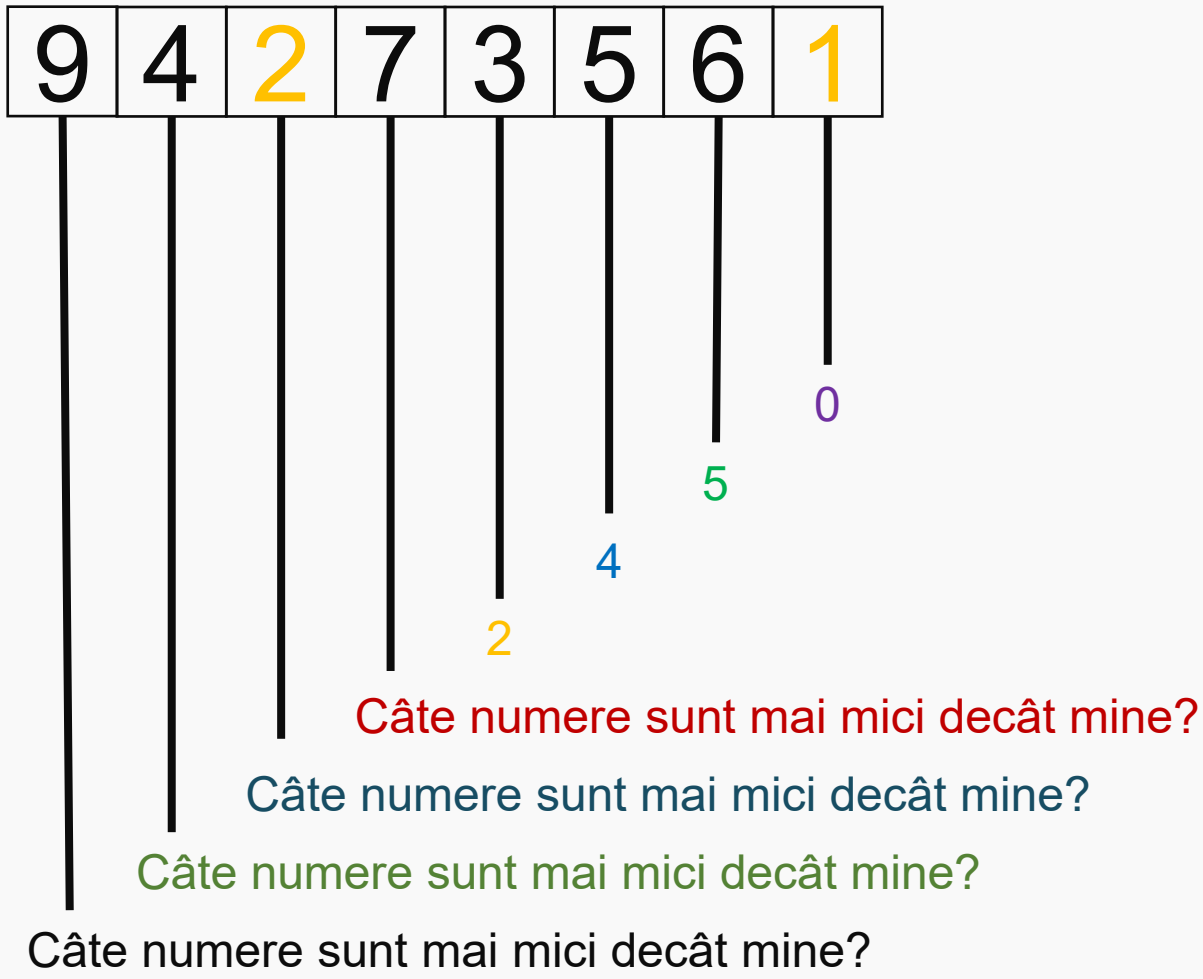
Rank Sort


9 4 2 7 3 5 6 1
0
5
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

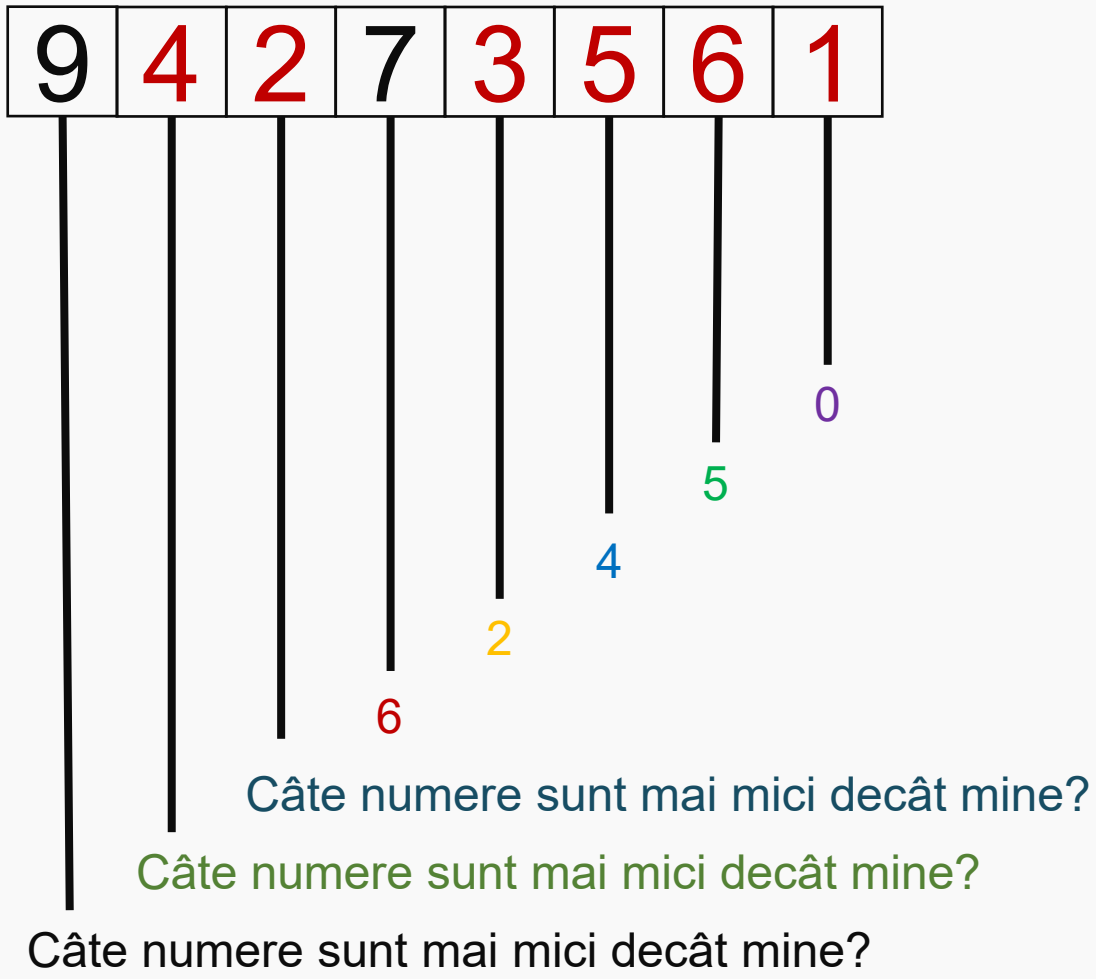
Rank Sort


9 4 2 7 3 5 6 1
0
5
4
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

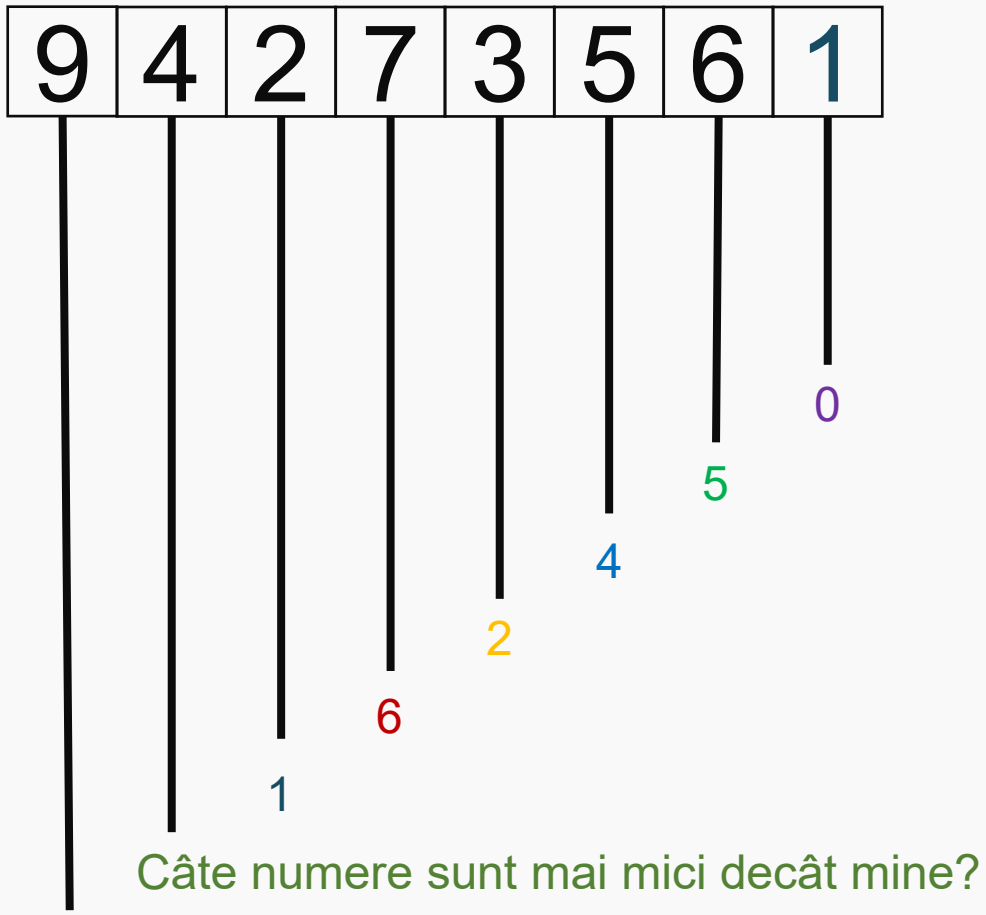
Rank Sort


9 4 2 7 3 5 6 1
0
5
4
2
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

Rank Sort


9 4 2 7 3 5 6 1
0
5
4
2
6
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

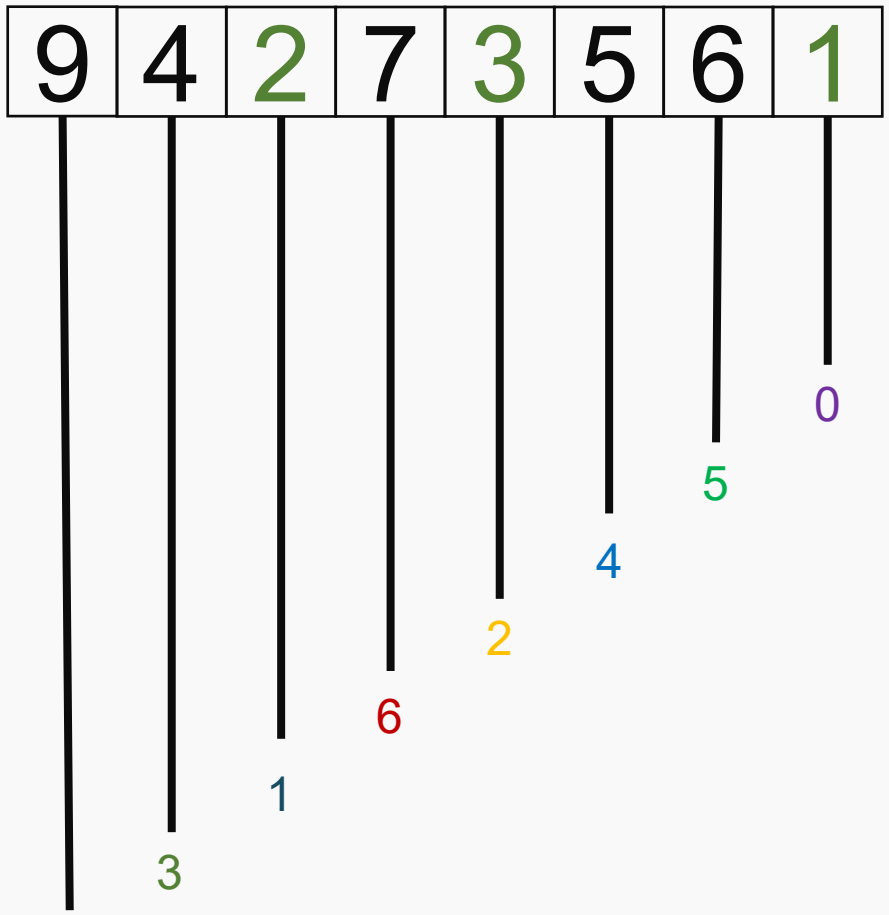
Rank Sort



Câte numere sunt mai mici decât mine?

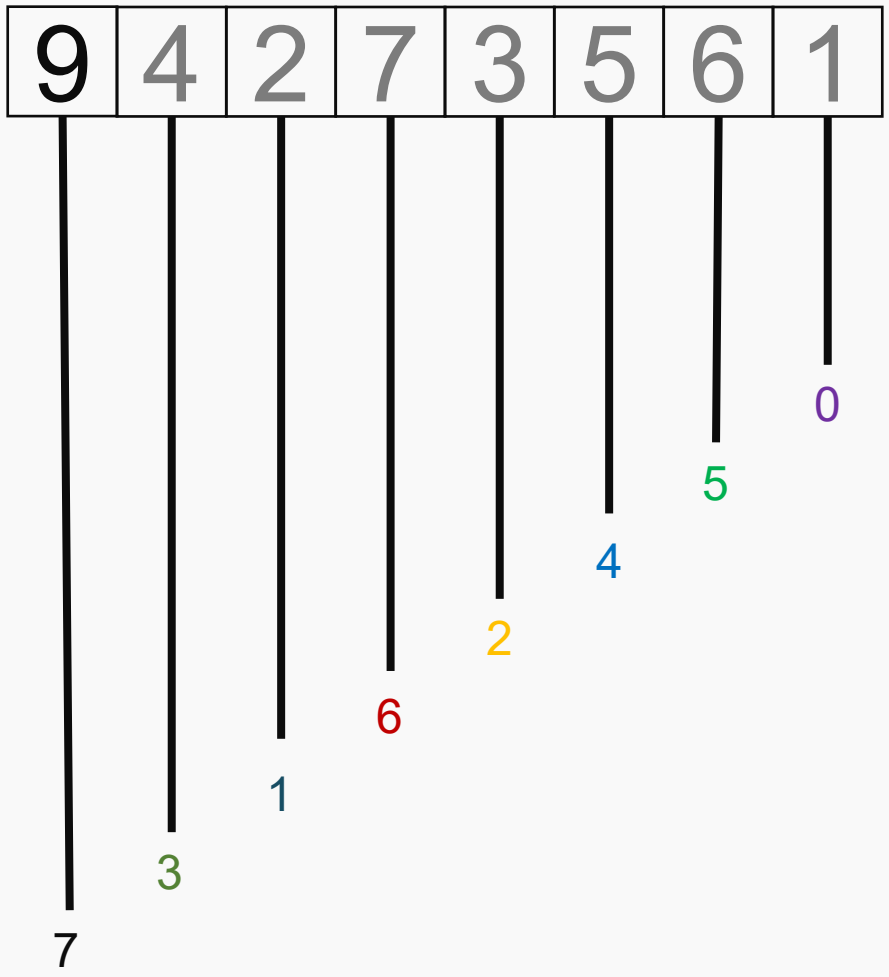
Câte numere sunt mai mici decât mine?

Rank Sort



Câte numere sunt mai mici decât mine?

Rank Sort



Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

Acestea sunt pozițiile în vectorul final

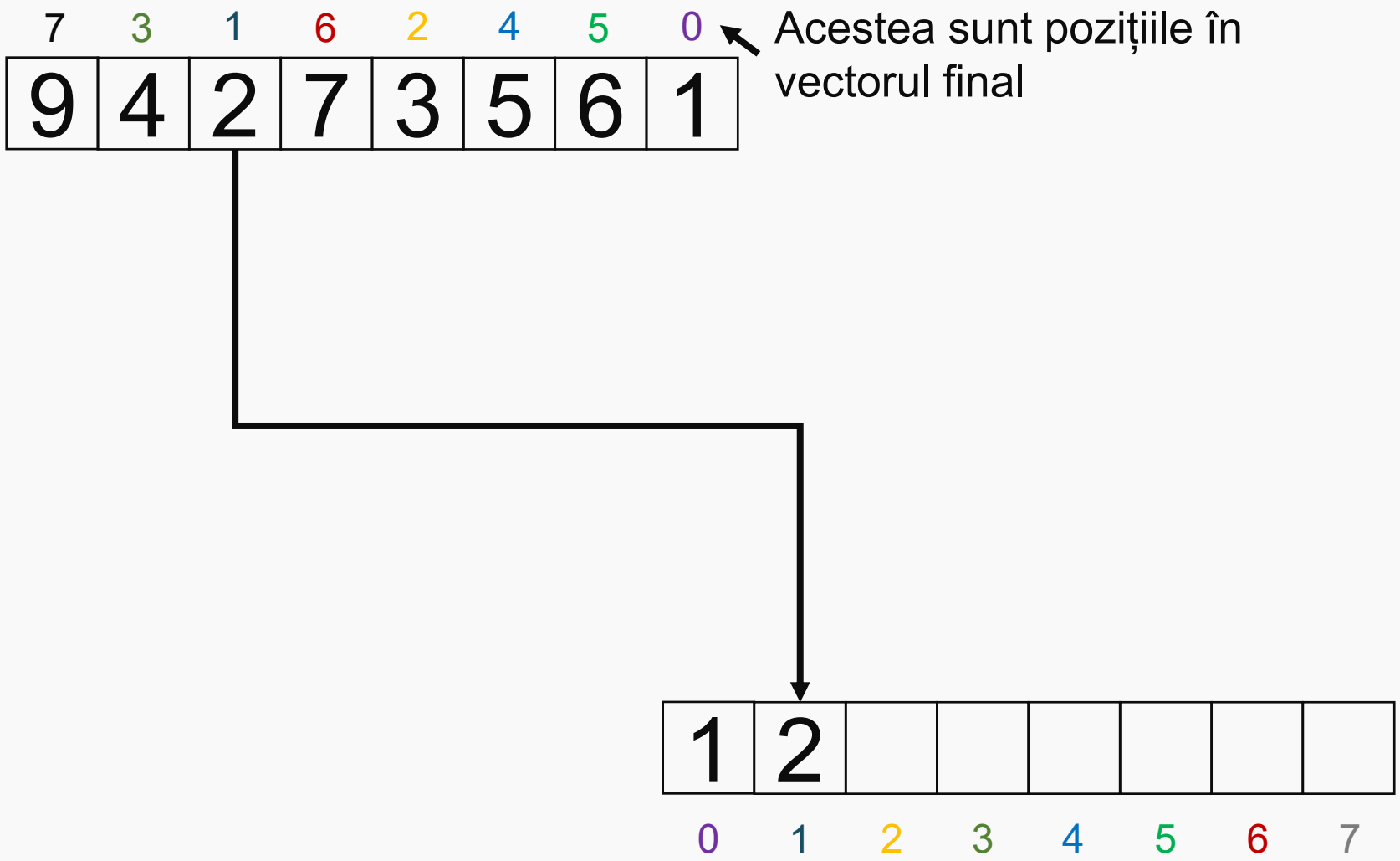
Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

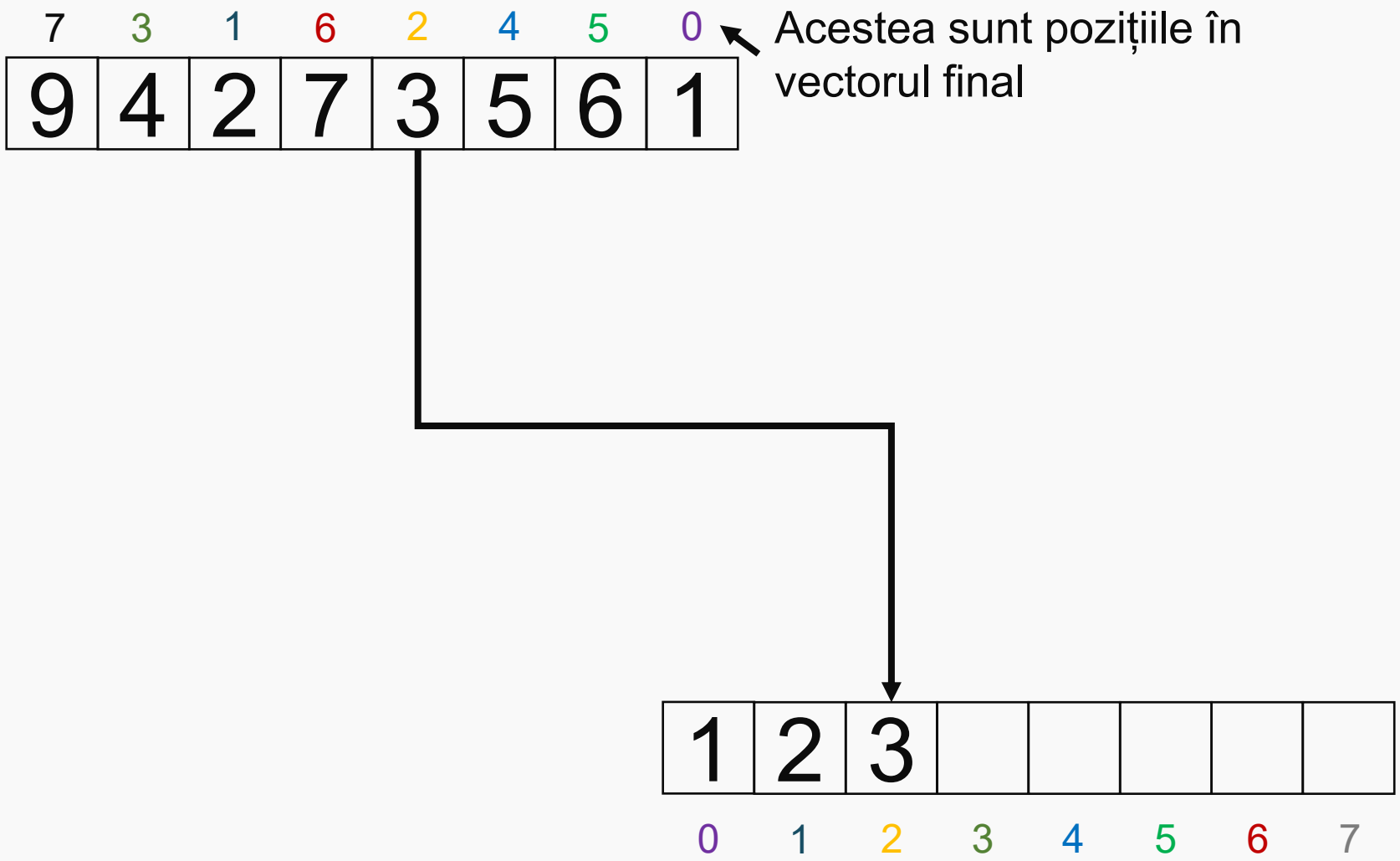
Acestea sunt pozițiile în vectorul final

1							
0	1	2	3	4	5	6	7

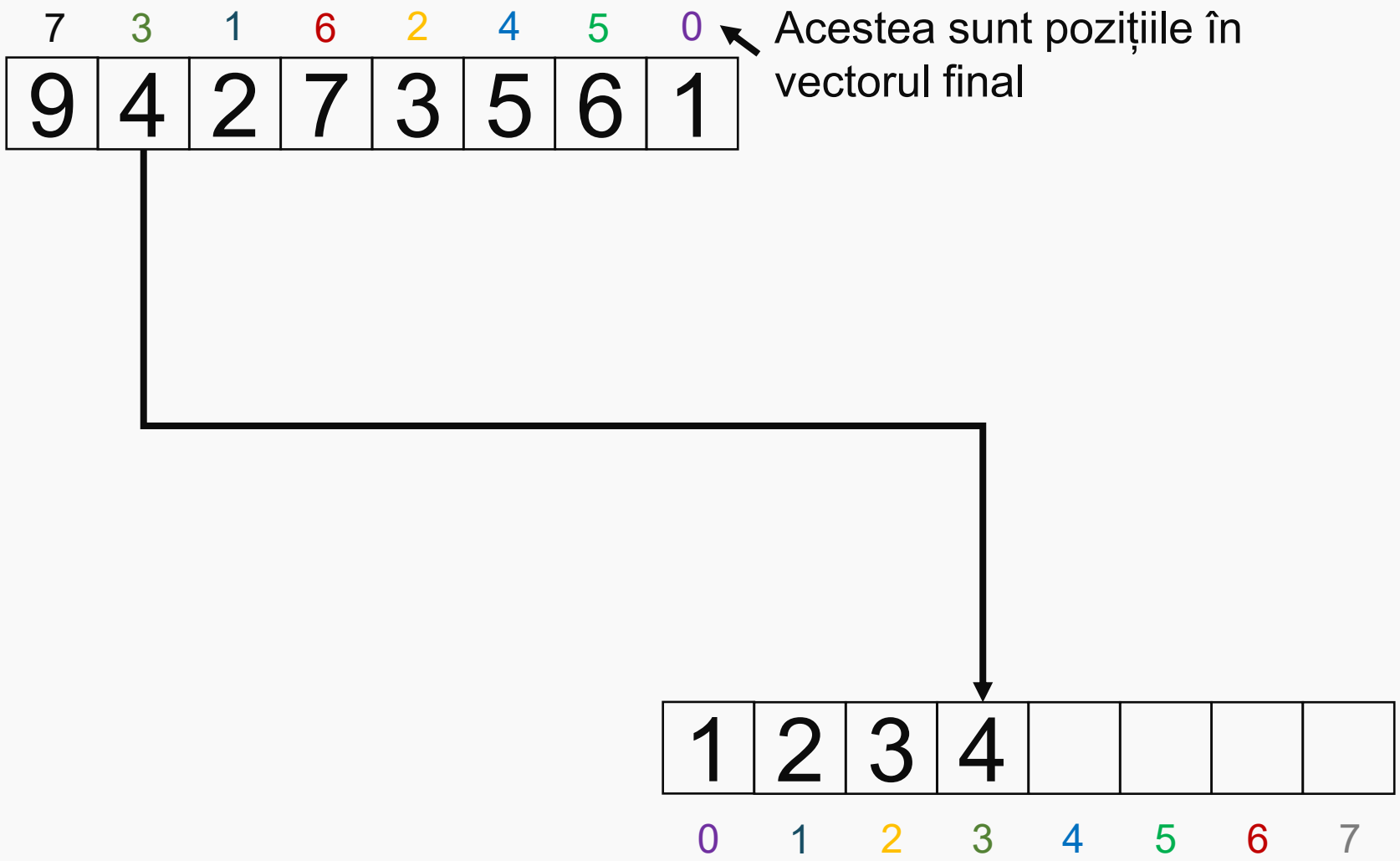
Rank Sort


7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2
0 1 2 3 4 5 6 7

Rank Sort


7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2 3
0 1 2 3 4 5 6 7

Rank Sort


7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2 3 4
0 1 2 3 4 5 6 7

Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

Acestea sunt pozițiile în vectorul final

1	2	3	4	5			
0	1	2	3	4	5	6	7

Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

Acestea sunt pozițiile în vectorul final

1	2	3	4	5	6		
0	1	2	3	4	5	6	7

Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

Acestea sunt pozițiile în vectorul final

1	2	3	4	5	6	7	
0	1	2	3	4	5	6	7

Rank Sort

7	3	1	6	2	4	5	0
9	4	2	7	3	5	6	1

Acestea sunt pozițiile în vectorul final

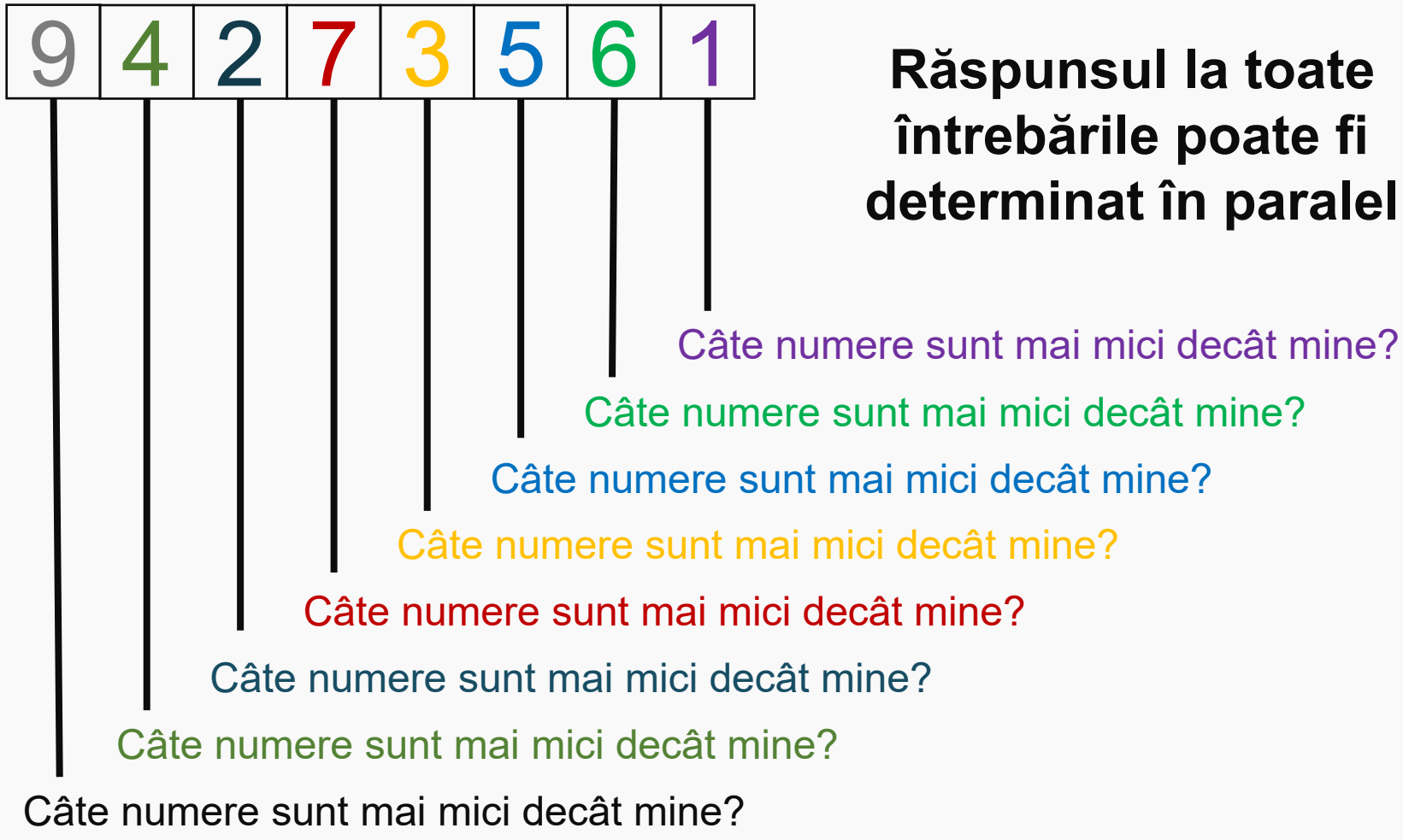
1	2	3	4	5	6	7	9
0	1	2	3	4	5	6	7

Rank Sort Paralel

9	4	2	7	6	5	6	1

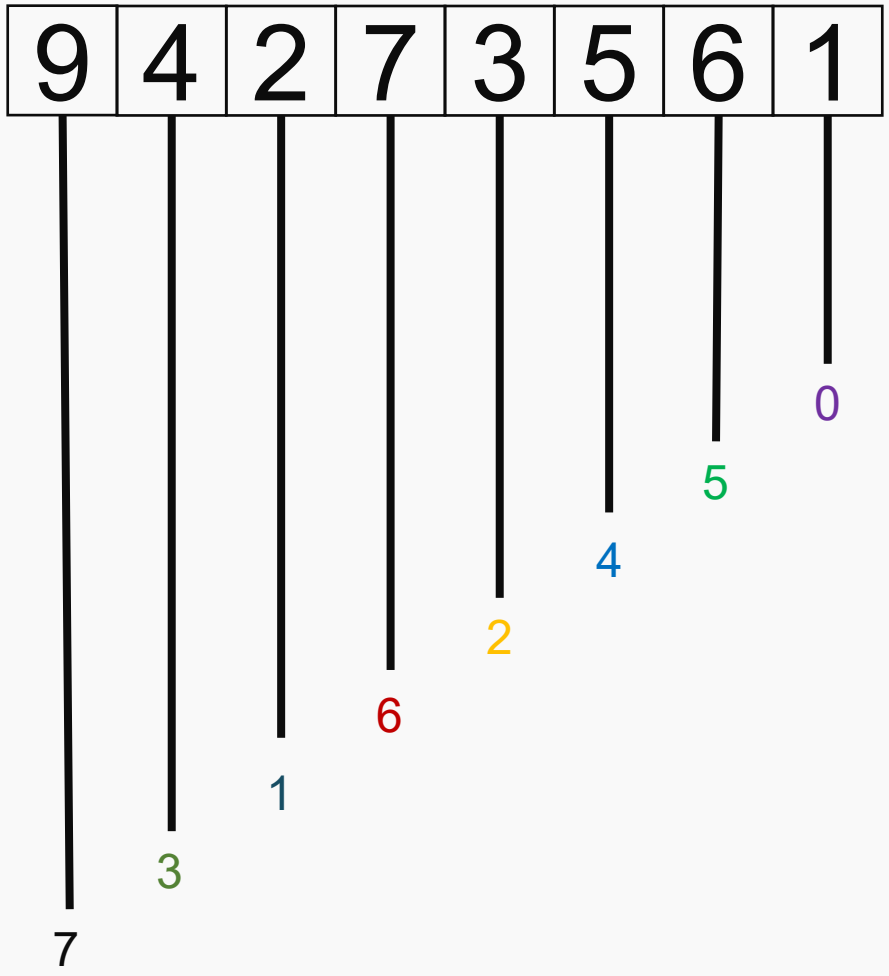
1	2	4	5	6	6	7	9

Rank Sort Paralel



Răspunsul la toate întrebările poate fi determinat în paralel

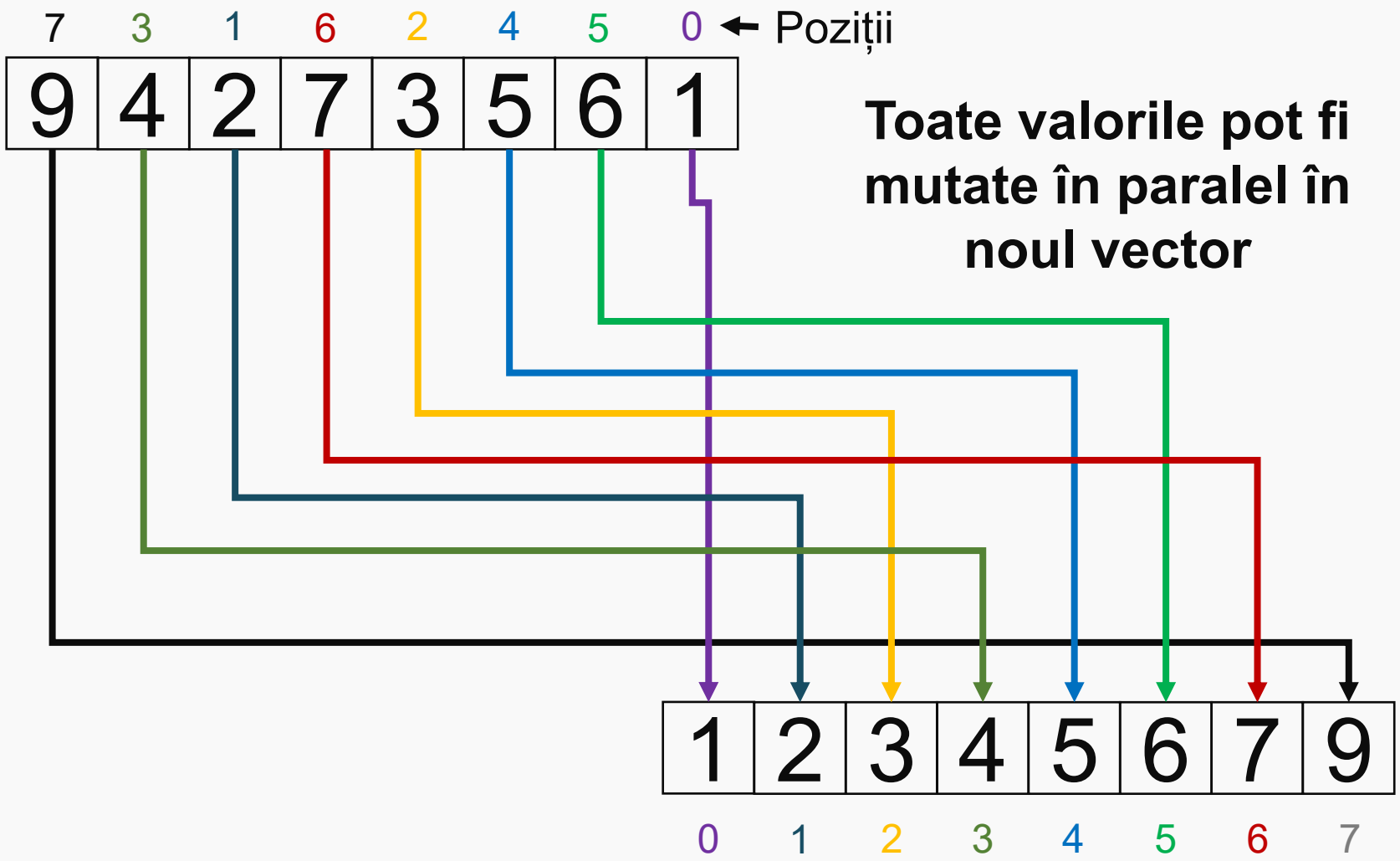
Rank Sort Paralel



Răspunsul la toate întrebările poate fi determinat în paralel

Rank Sort Paralel

Rank Sort Paralel



Toate valorile pot fi mutate în paralel în noul vector

Algoritm merge

2	3	4	5	7

1	2	4	4	6

Avem ca intrare două liste **sortate** dorim să le unim într-o listă **sortată**

1	2	2	3	4	4	4	5	6	7

Algoritm merge

2	3	4	5	7

1	2	4	4	6

Soluție:

Se extrage mereu cel mai mic element (Garantat să fie pe prima poziție în una din cele două liste)

Complexitate: $\boldsymbol{O}(\boldsymbol{N})$

1	2	2	3	4	4	4	5	6	7

Algoritm merge

2	3	4	5	7
1	2	4	4	6

Algoritm merge

2	3	4	5	7
	2	4	4	6

1

Algoritm merge

3	4	5	7
2	4	4	6

1	2

Algoritm merge

3	4	5	7

4	4	6

1	2	2

Algoritm merge

4	5	7

4	4	6

1	2	2	3

Algoritm merge

	5	7
4	4	6

1	2	2	3	4

Algoritm merge

5	7

4	6

1	2	2	3	4	4

Algoritm merge

5	7

6

1	2	2	3	4	4	4

Algoritm merge

7
6

1	2	2	3	4	4	4	5

Algoritm merge

7

1	2	2	3	4	4	4	5	6

Algoritm merge

1	2	2	3	4	4	4	5	6	7

Algoritm merge

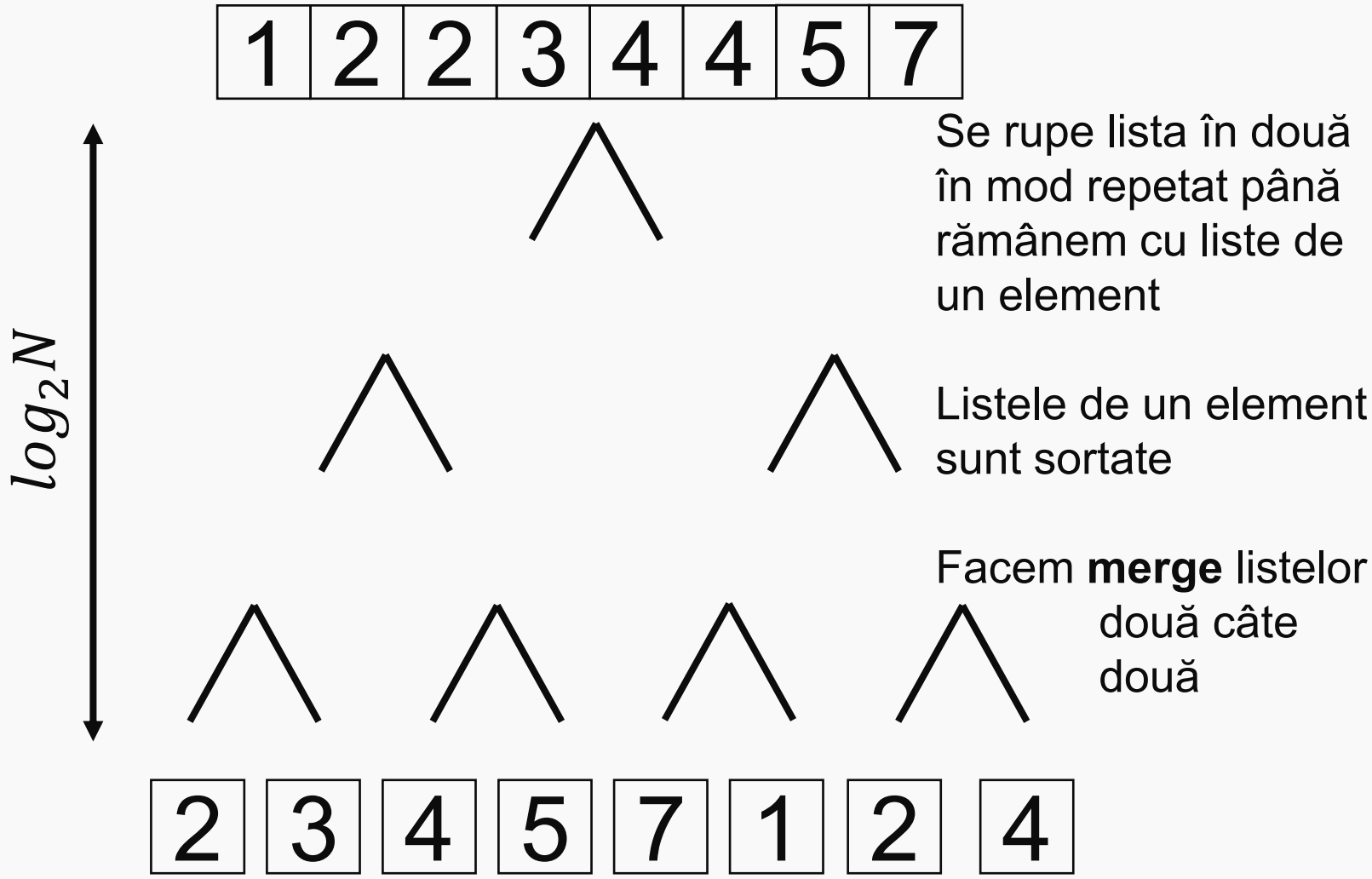
1	2	2	3	4	4	4	5	6	7

Folosim acest semn pentru a reprezenta operația **MERGE** $\wedge$

2	3	4	5	7

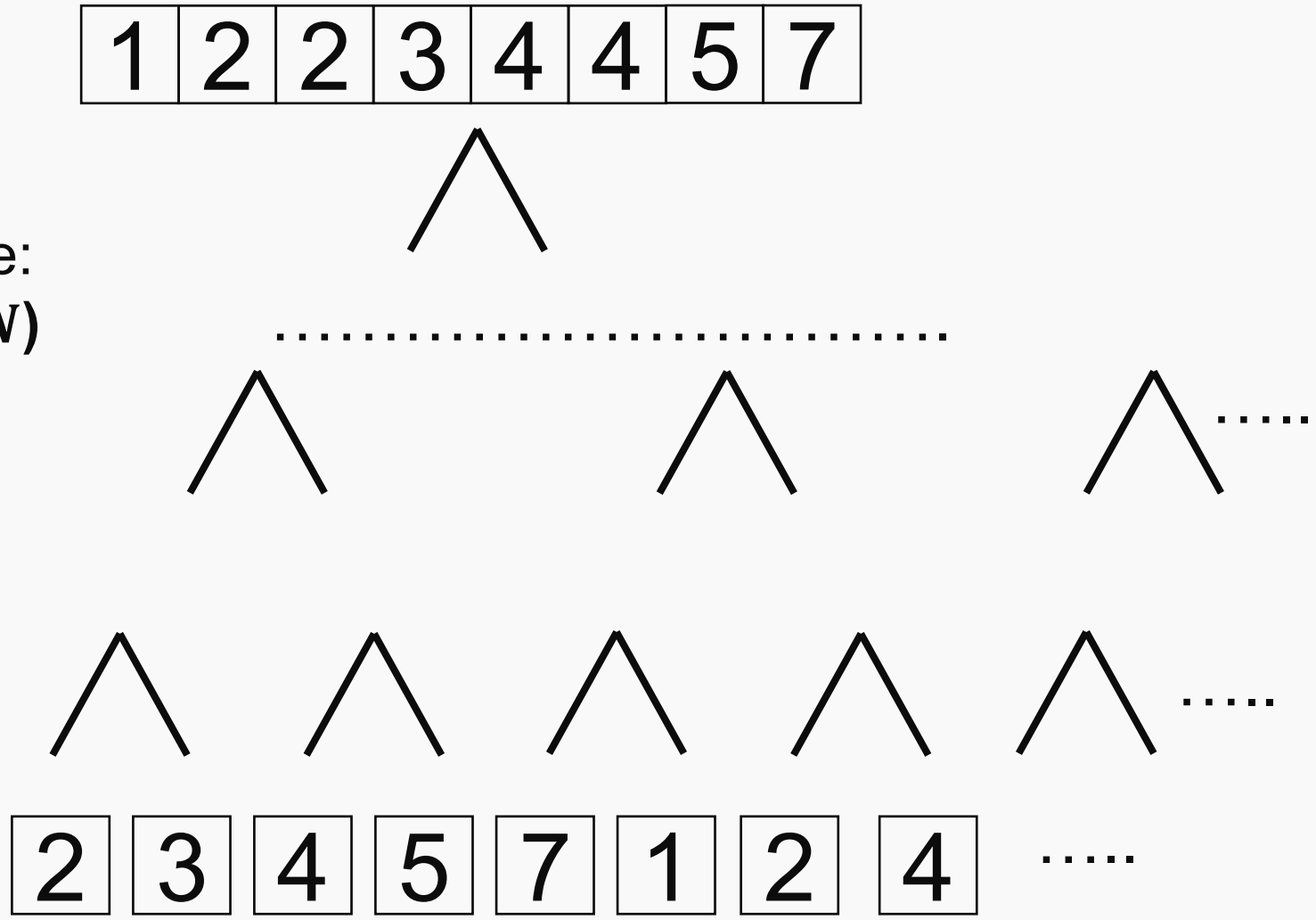
1	2	4	4	6

Merge sort

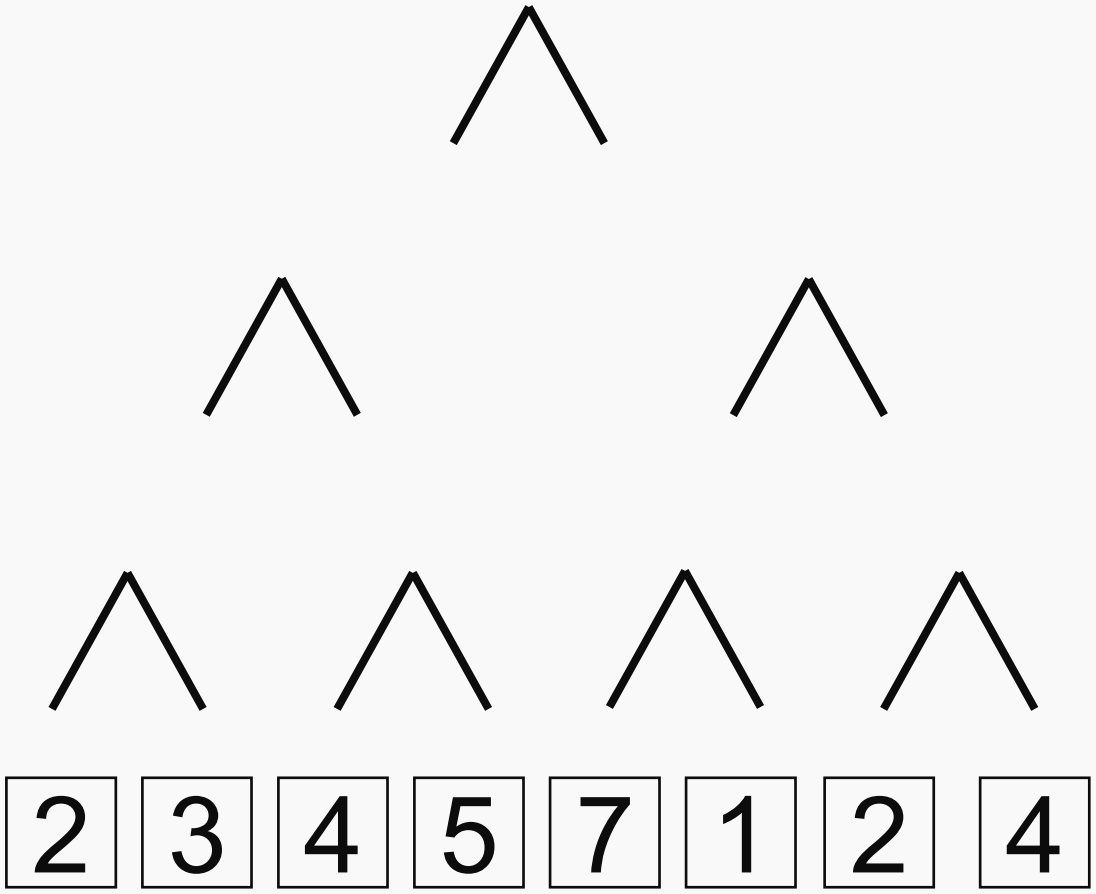

1 2 2 3 4 4 5 7
$log_2 N$
Se rupe lista în două în mod repetat până rămânem cu liste de un element
Listele de un element sunt sortate
Facem **merge** listelor două câte două
2 3 4 5 7 1 2 4

Merge sort

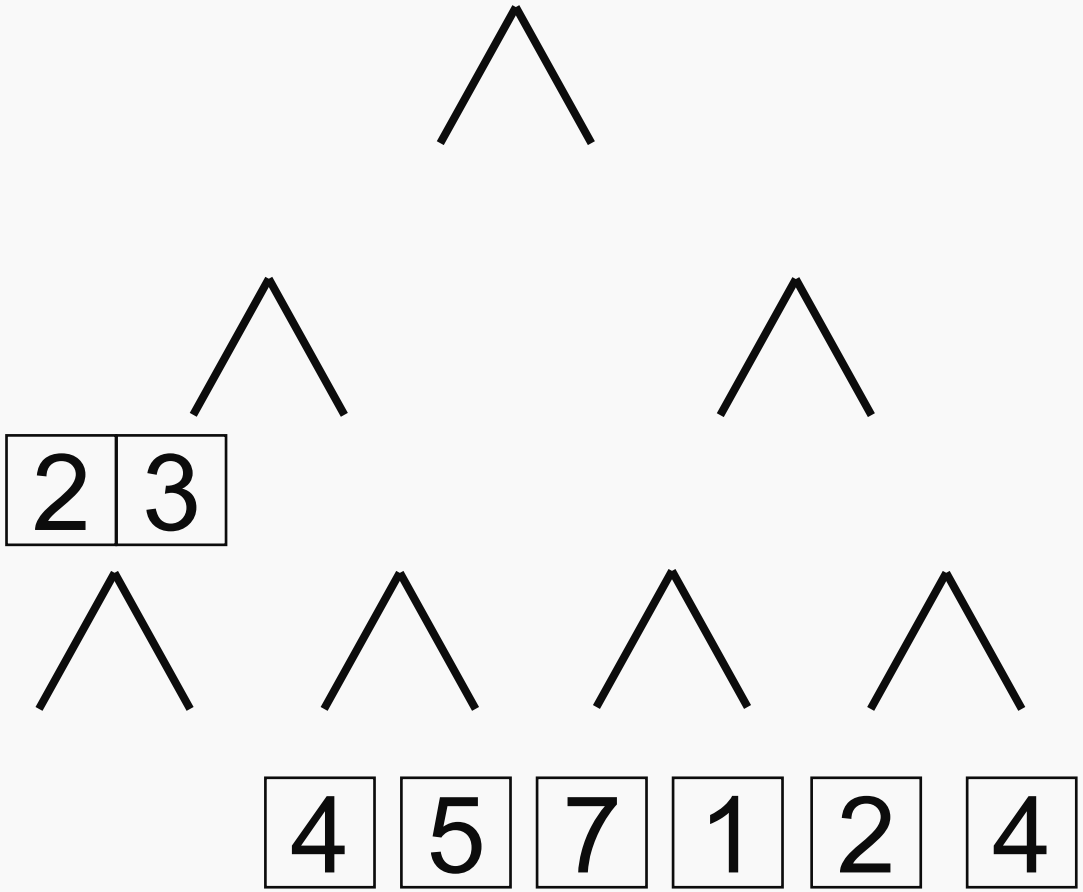
Complexitate:
O$(N * log_2 N)$



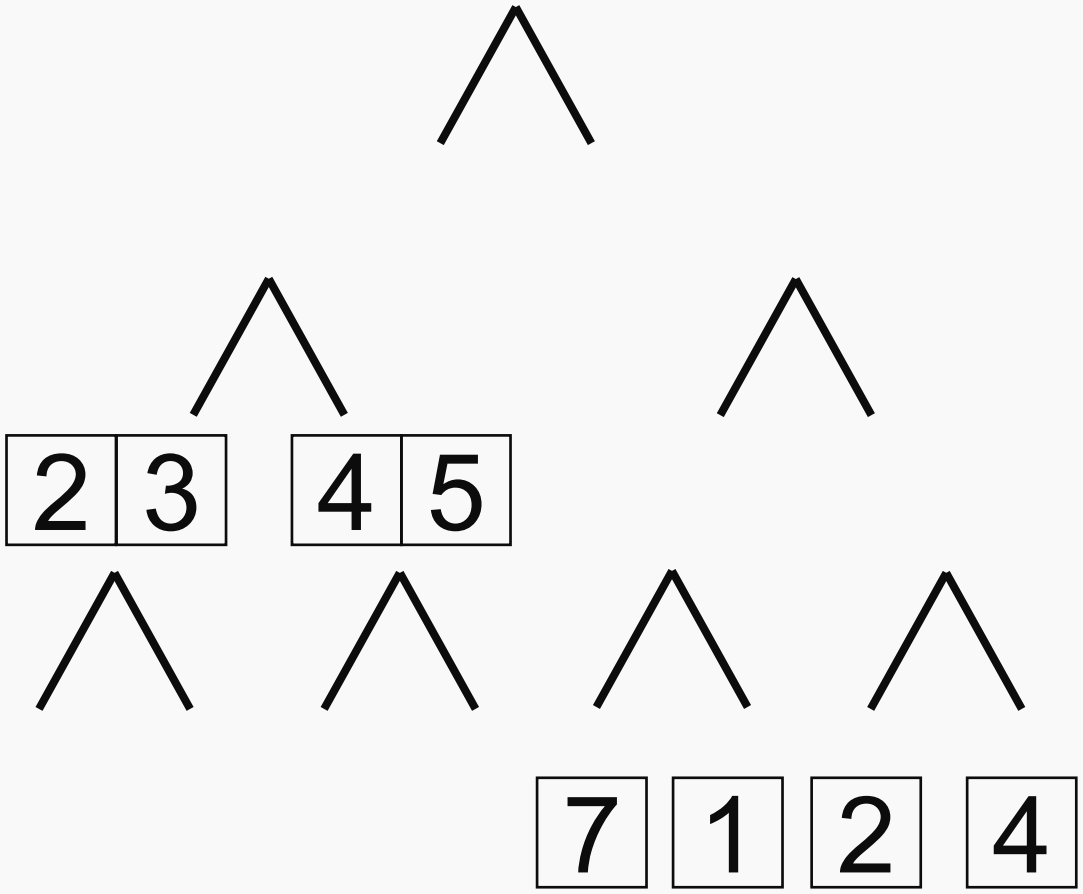
Merge sort



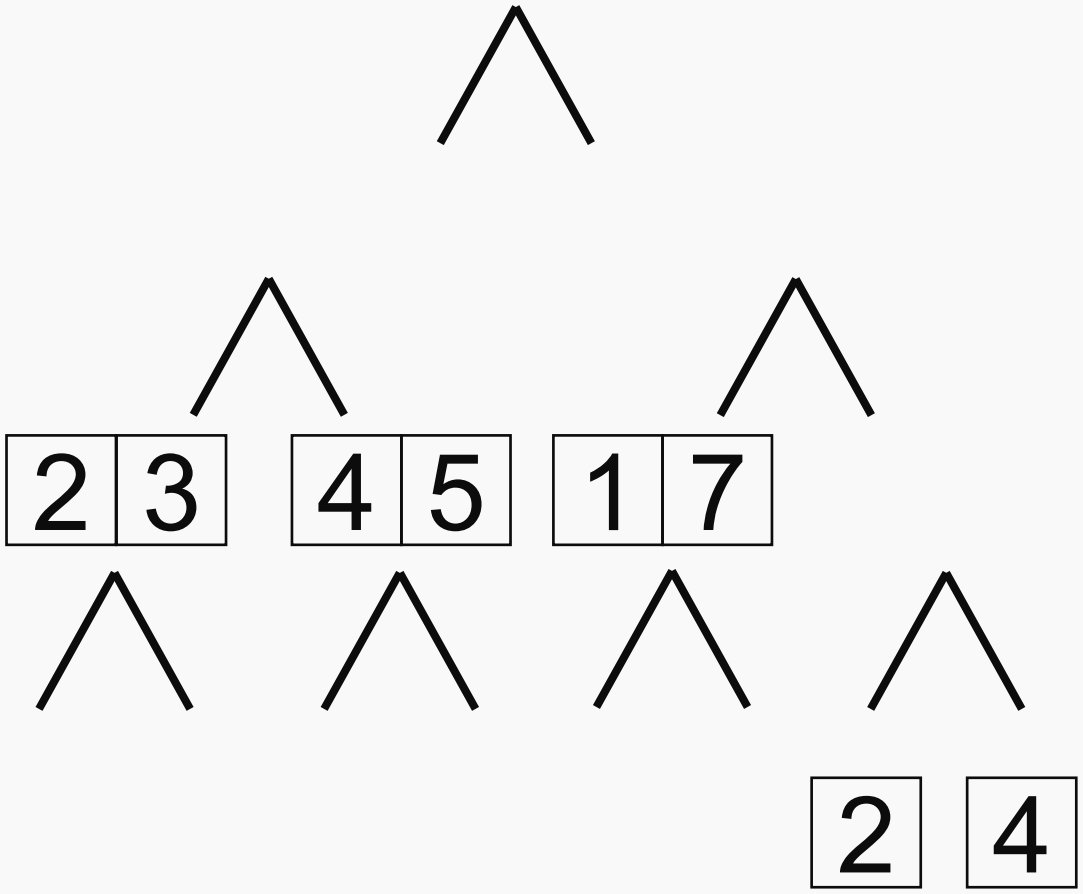
Merge sort



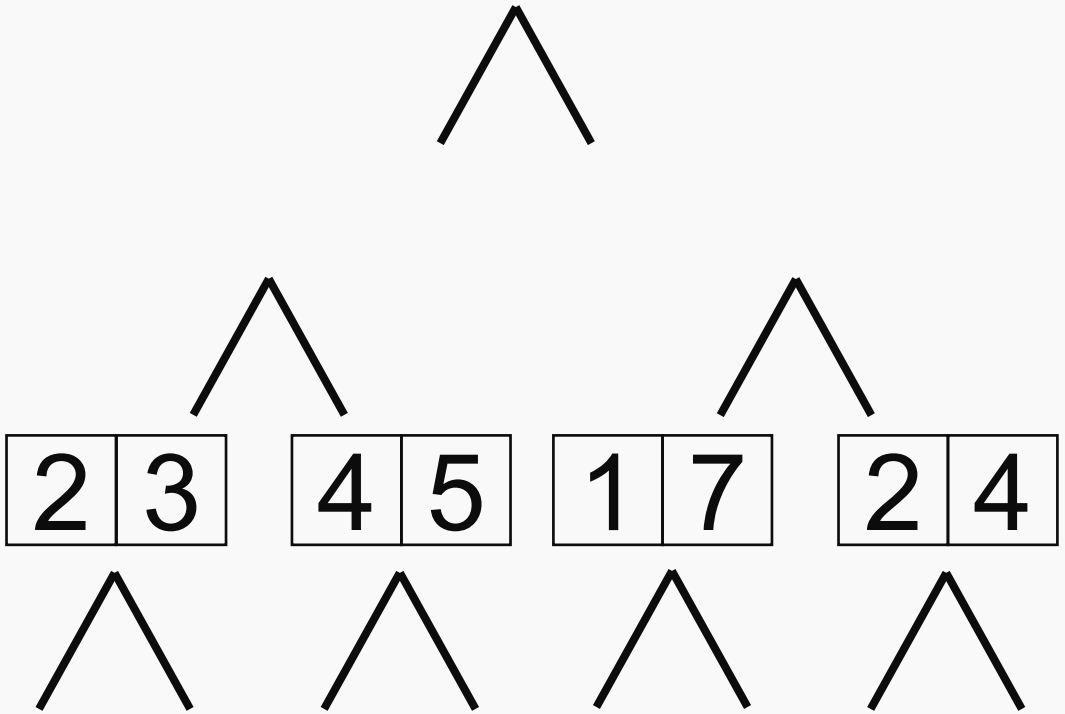
Merge sort



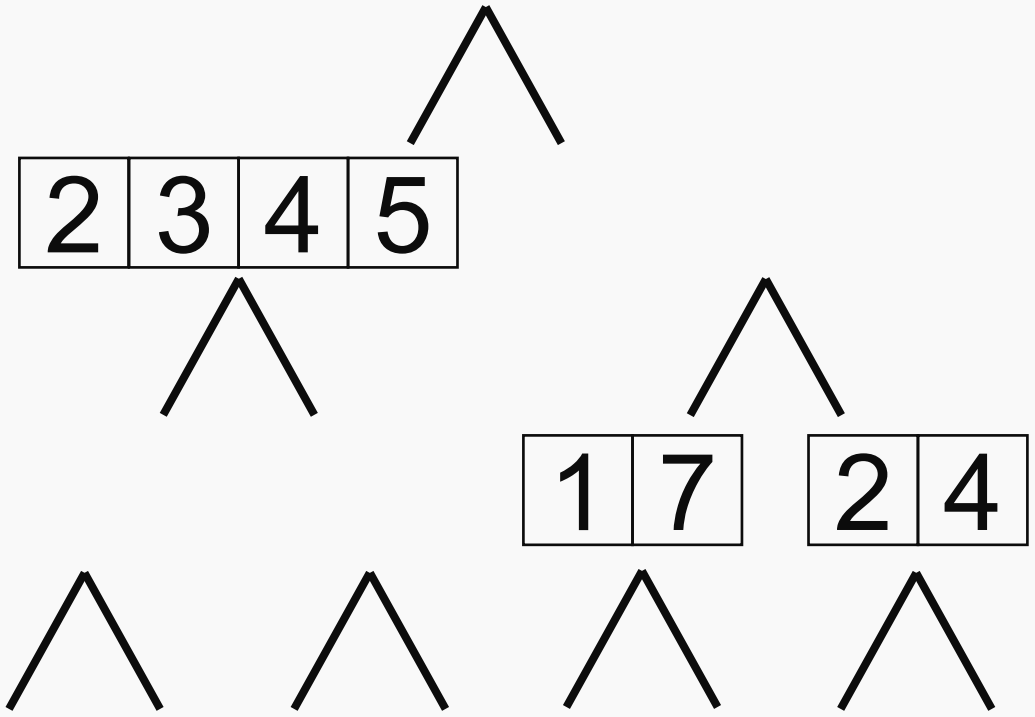
Merge sort



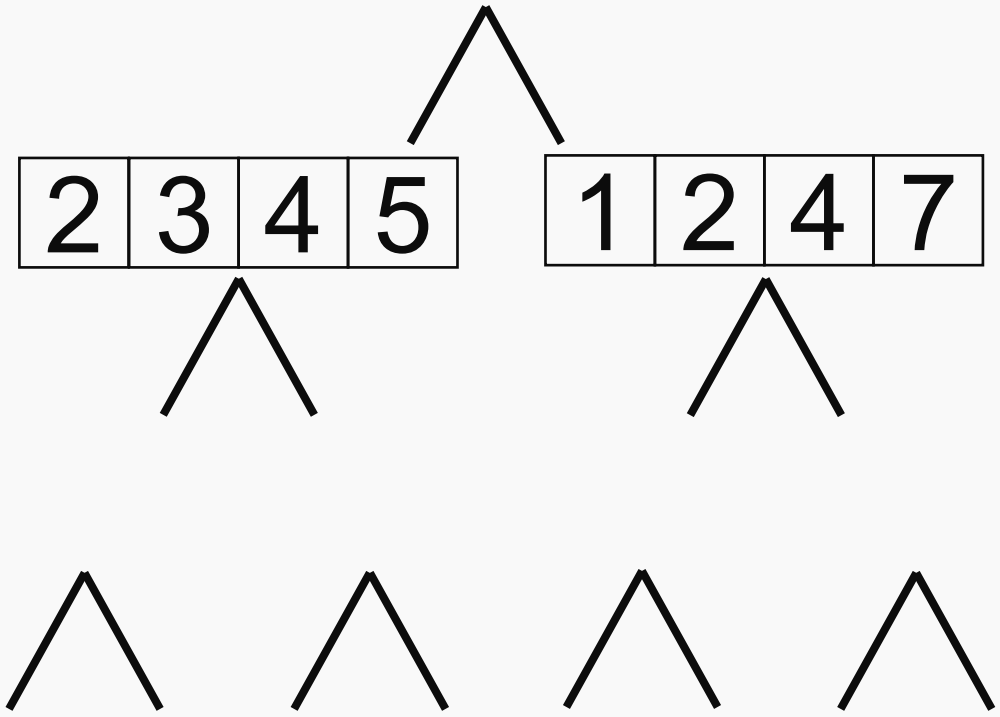
Merge sort



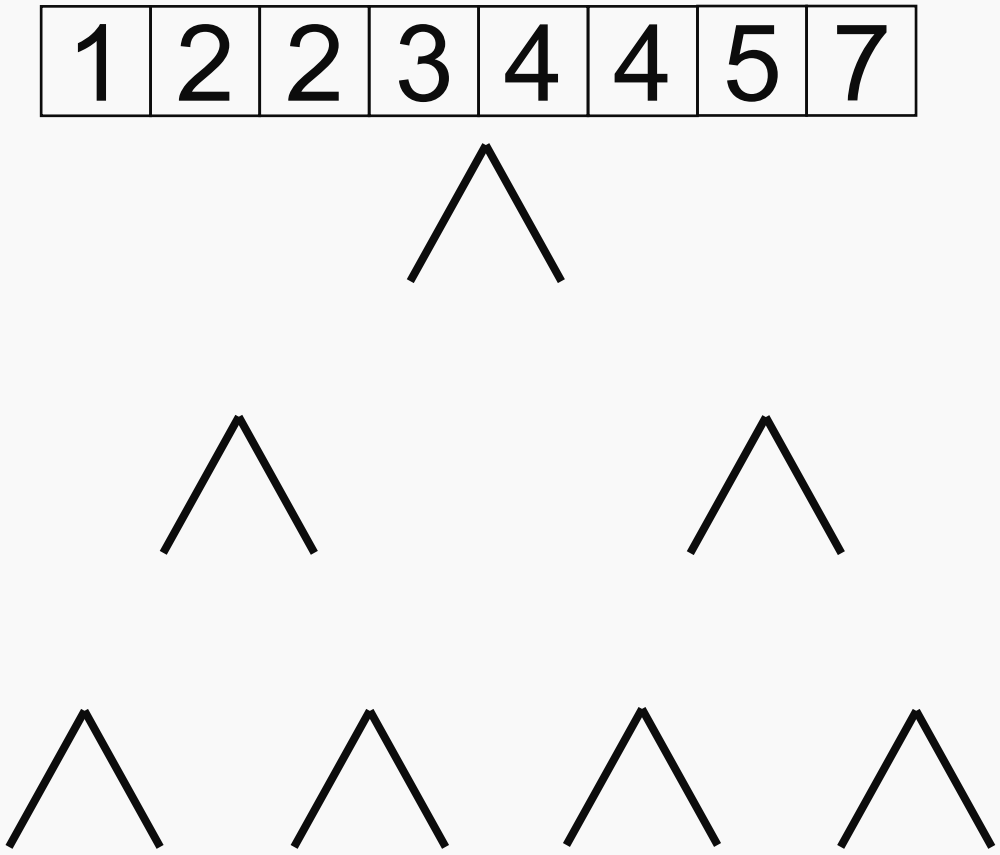
Merge sort



Merge sort



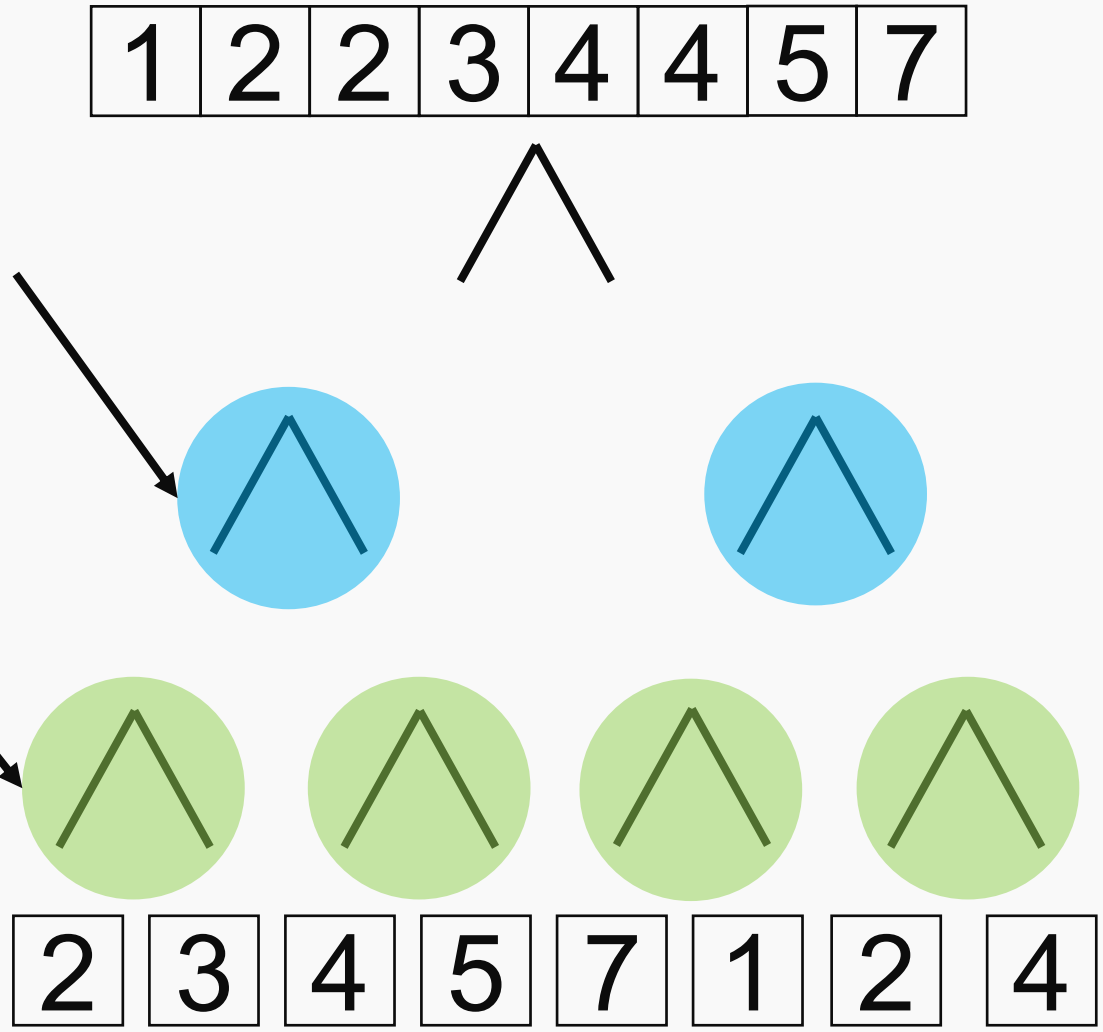
Merge sort



Merge sort paralel

Operațiile **albastre** pot fi executate în paralel

Operațiile **verzi** pot fi executate în paralel

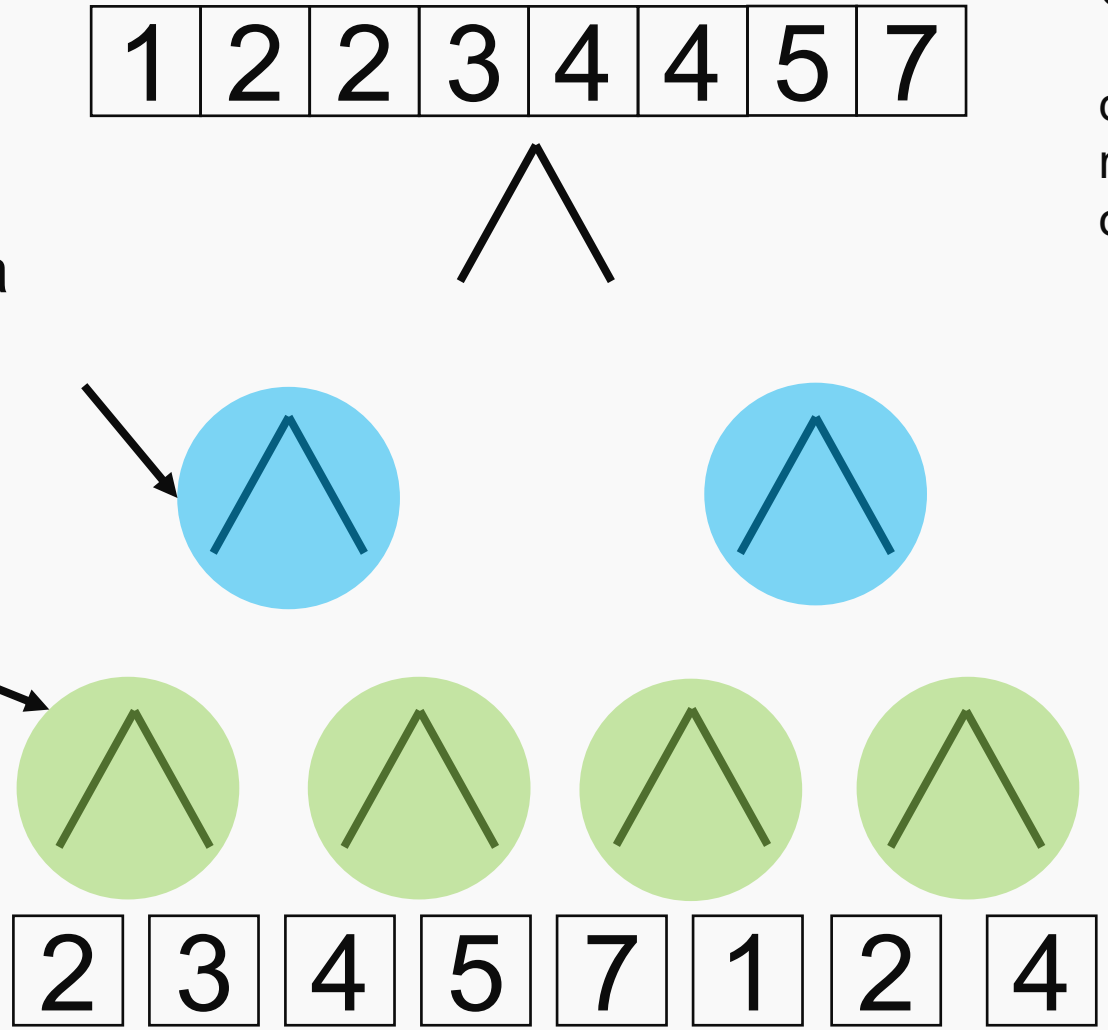


Merge sort paralel

Operația

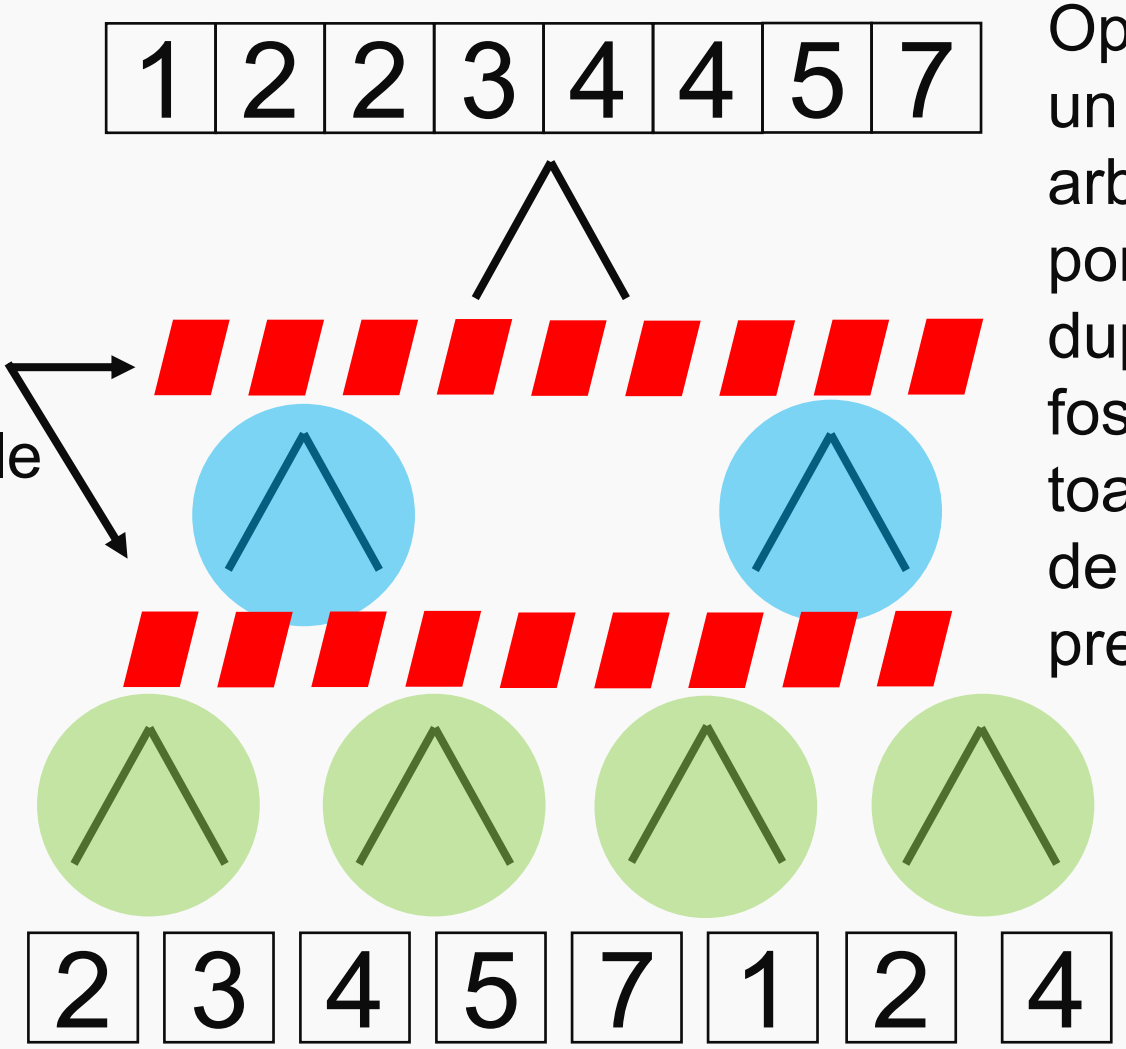
depinde de rezultatul operațiilor

O operație **verde** cu una **albastră** nu poate fi executată în paralel



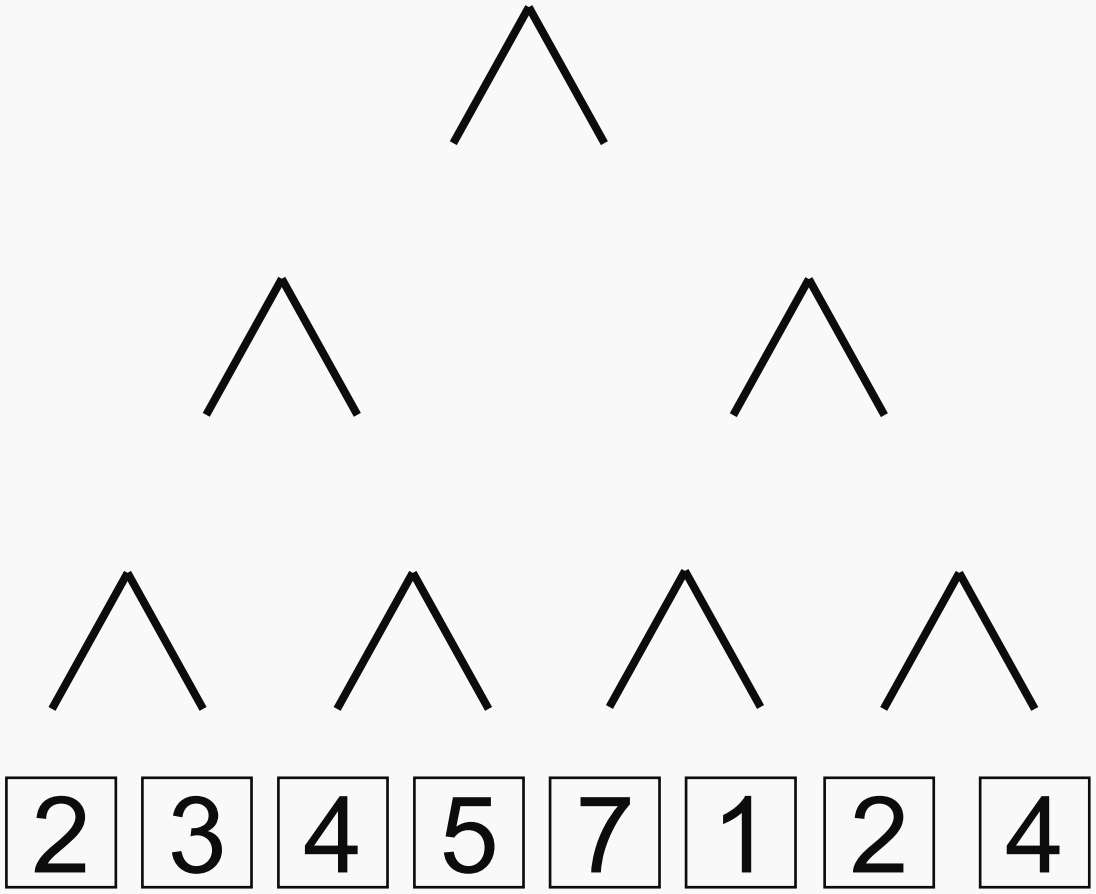
Merge sort paralel

Soluție:
Barrier
Între nivelele arborelui

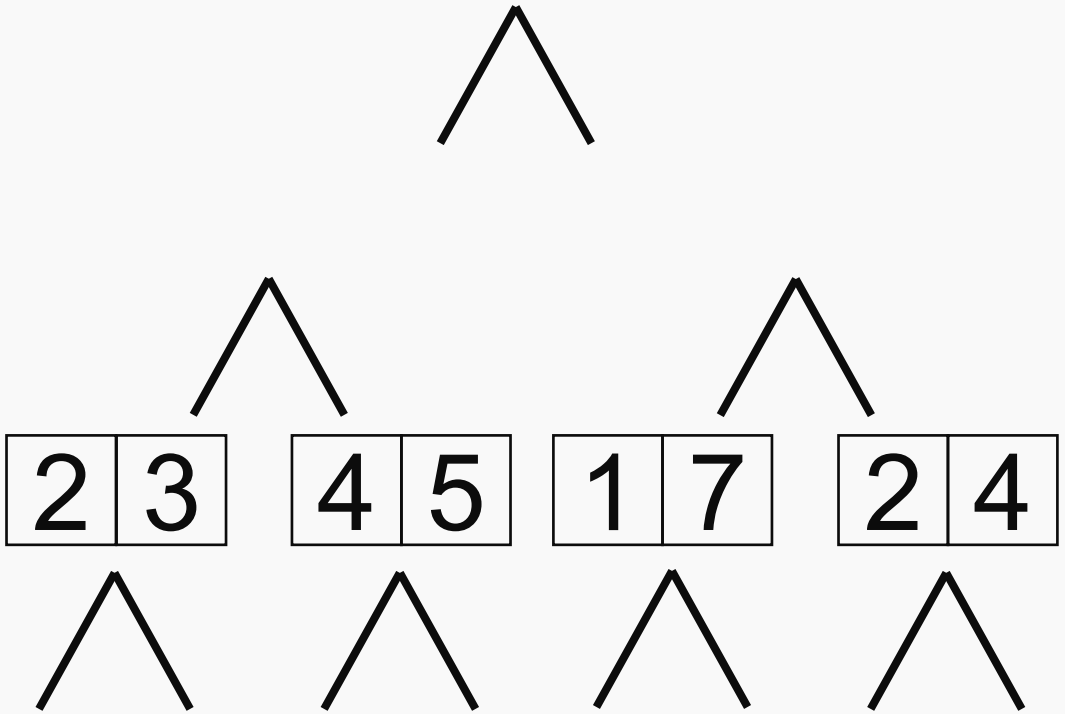


Operațiile după un nivel al arborelui pot porni doar după ce au fost terminate toate operațiile de pe nivelul precedent.

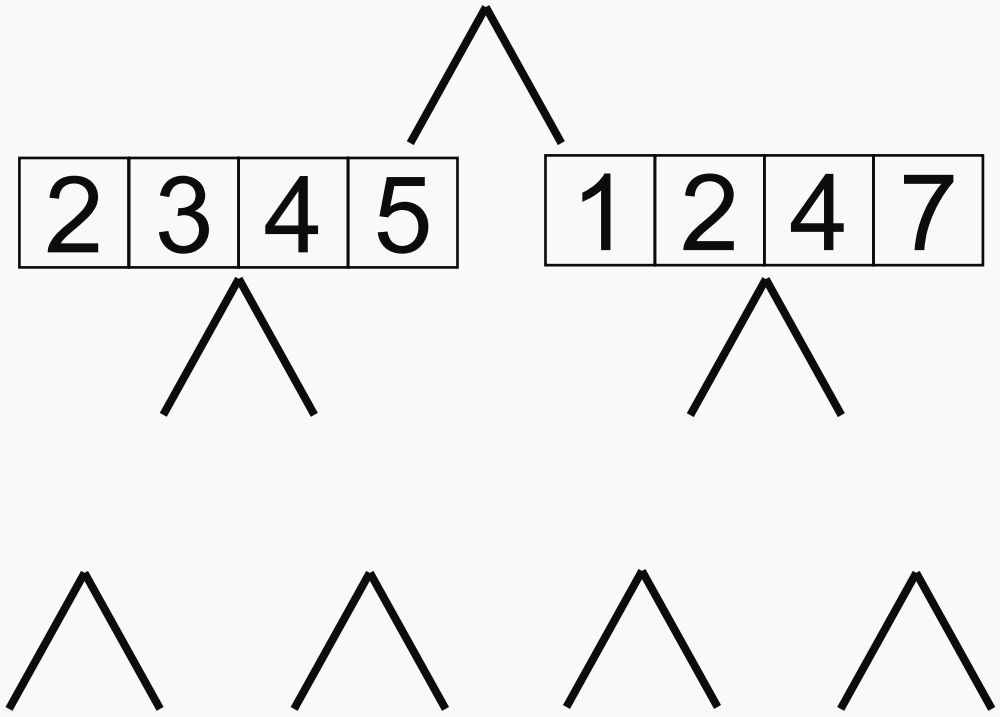
Merge sort paralel



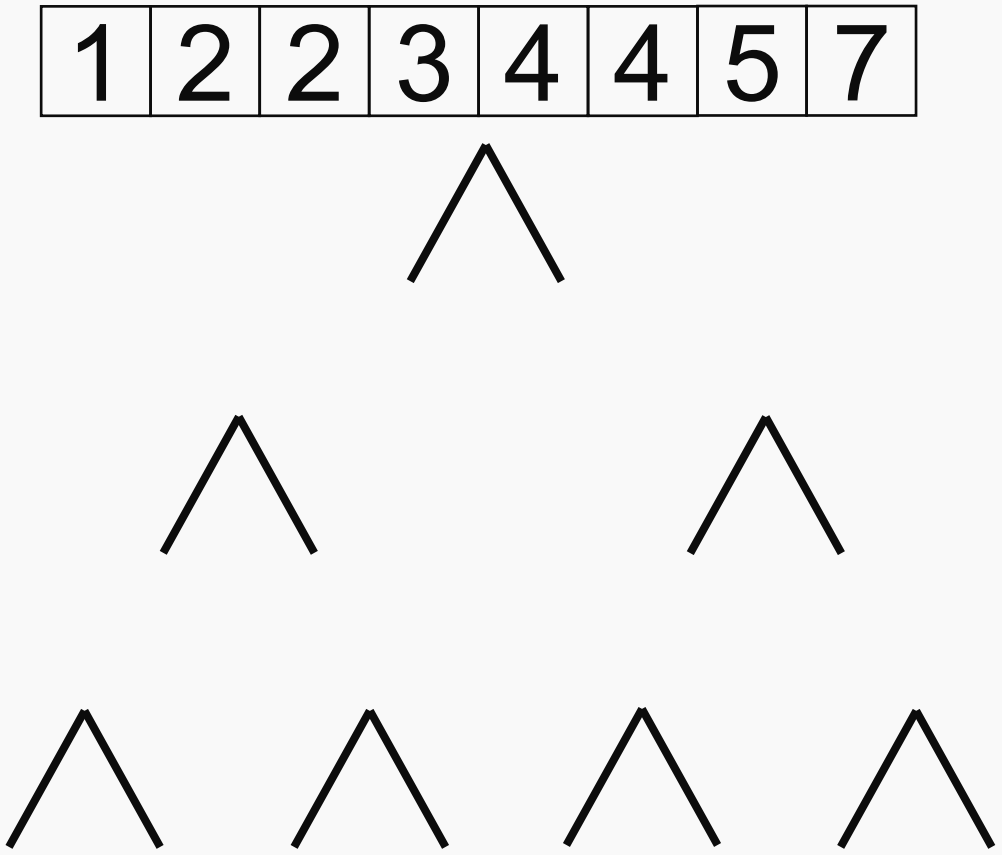
Merge sort paralel



Merge sort paralel



Merge sort paralel

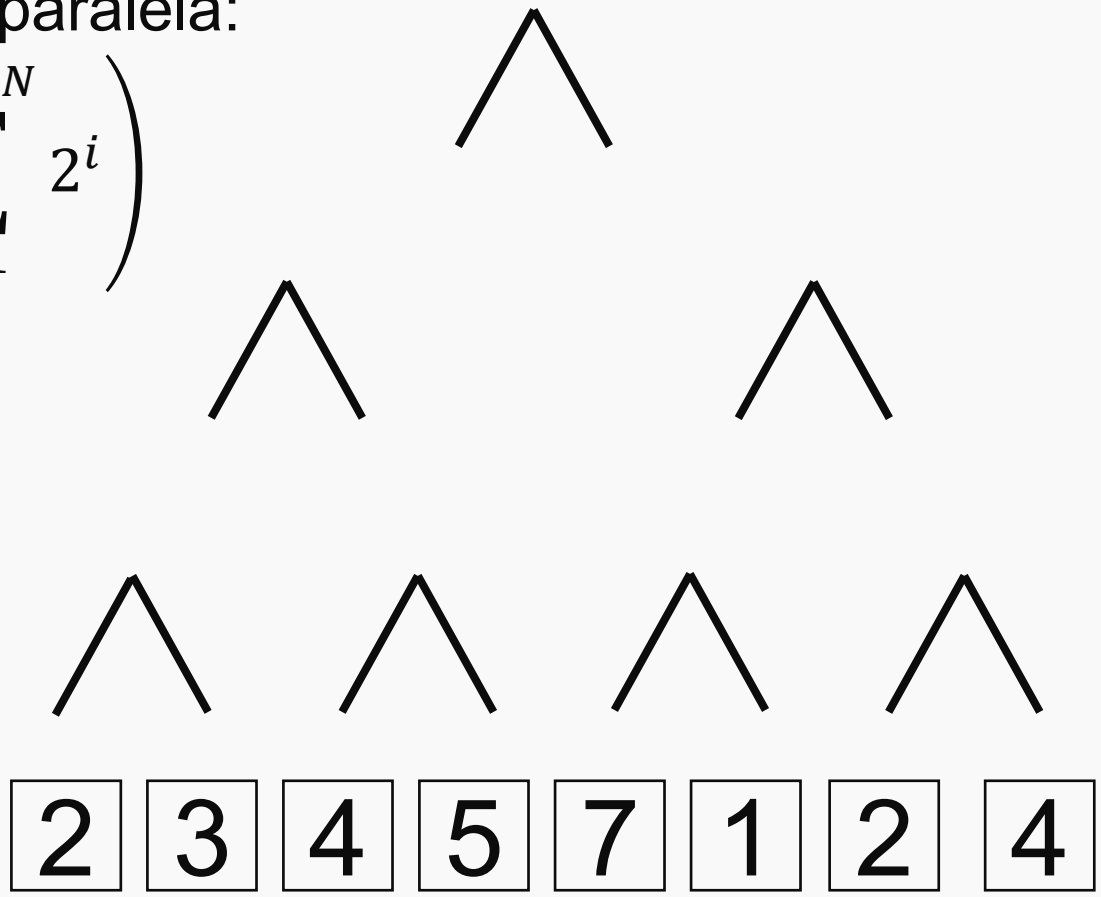


Merge sort paralel - complexitate

Complexitate paralelă:

$$O\left(\sum_{i=1}^{log_2 N} 2^i\right)$$

Dacă P = N



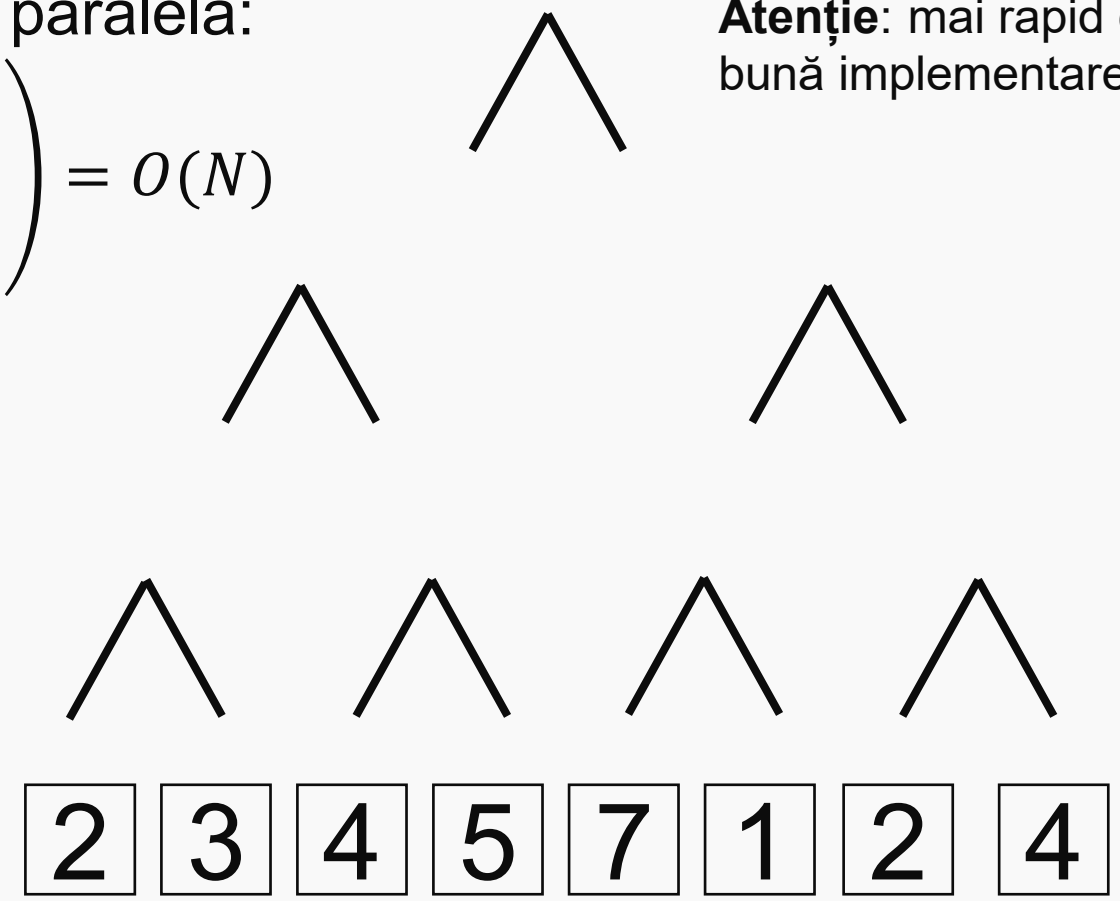
Merge sort paralel - complexitate

Complexitate paralelă:

$$O\left(\sum_{i=1}^{log_2 N} 2^i\right) = O(N)$$

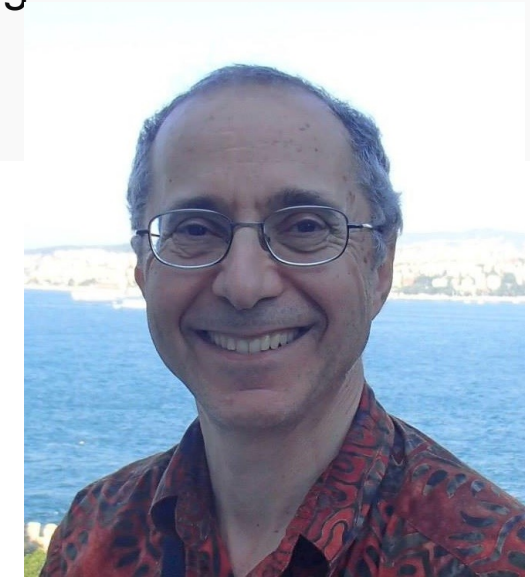
Dacă P = N

Atenție: mai rapid decât cea mai bună implementare secvențială



Merge sort paralel - complexitate

Cea mai bună soluție paralelă: paralelizează și operația merge
Articol
[Parallel Merge Sort – Richard Cole]



Parallel Merge Sort

Richard Cole
New York University

Abstract. We give a parallel implementation of merge sort on a CREW PRAM that uses n processors and $O(\log n)$ time; the constant in the running time is small. We also give a more complex version of the algorithm for the EREW PRAM; it also uses n processors and $O(\log n)$ time. The constant in the running time is still moderate, though not as small.

1. Introduction

1975]; this procedure merges two sorted arrays, each of length at most n, in time $O(\log\log n)$ using a linear number of processors. When used in the obvious way, Valiant's procedure leads to an implementation of merge sort on n processors using $O(\log n \log\log n)$ time. More recently, Kruskal [K, 1983] improved this sorting algorithm to obtain a sorting algorithm that

Căutare binară

Căutăm 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Căutare binară

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 < 7

Căutare binară

Căutăm 3

Între pozițiile 0 7

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 < 5

Căutare binară

Căutăm 3

Între pozițiile 0 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 > 2

Căutare binară

Căutăm 3

Între pozițiile 3 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 = 3

end

Complexitate $O(log_2(N))$

Căutare binară

Căutăm 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Fiecare thread este responsabil de o zonă

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Elementul căutat este în bucata mea

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Elementul nu este la mine, mă opresc

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Elementul nu este la mine, mă opresc

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 1 4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Caut binar

Căutare paralelă – implementare naivă

Căutăm 3

Între pozițiile 1 4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Caut binar

Doar un singur pas s-a executat în paralel.
Am pornit thread-uri degeaba.

Complexitate: $O(log_2(N))$ la fel ca secvențial

Căutare paralelă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Fiecare thread este responsabil de o zonă.
Când trecem la pasul următor toate thread-urile se mută în noua zonă

Căutare paralelă

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Elementul căutat este în bucata mea

Căutare paralelă

Căutăm 3

Între pozițiile 1 4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Am găsit elementul

Am găsit elementul

Toate thread-urile caută în noua zonă

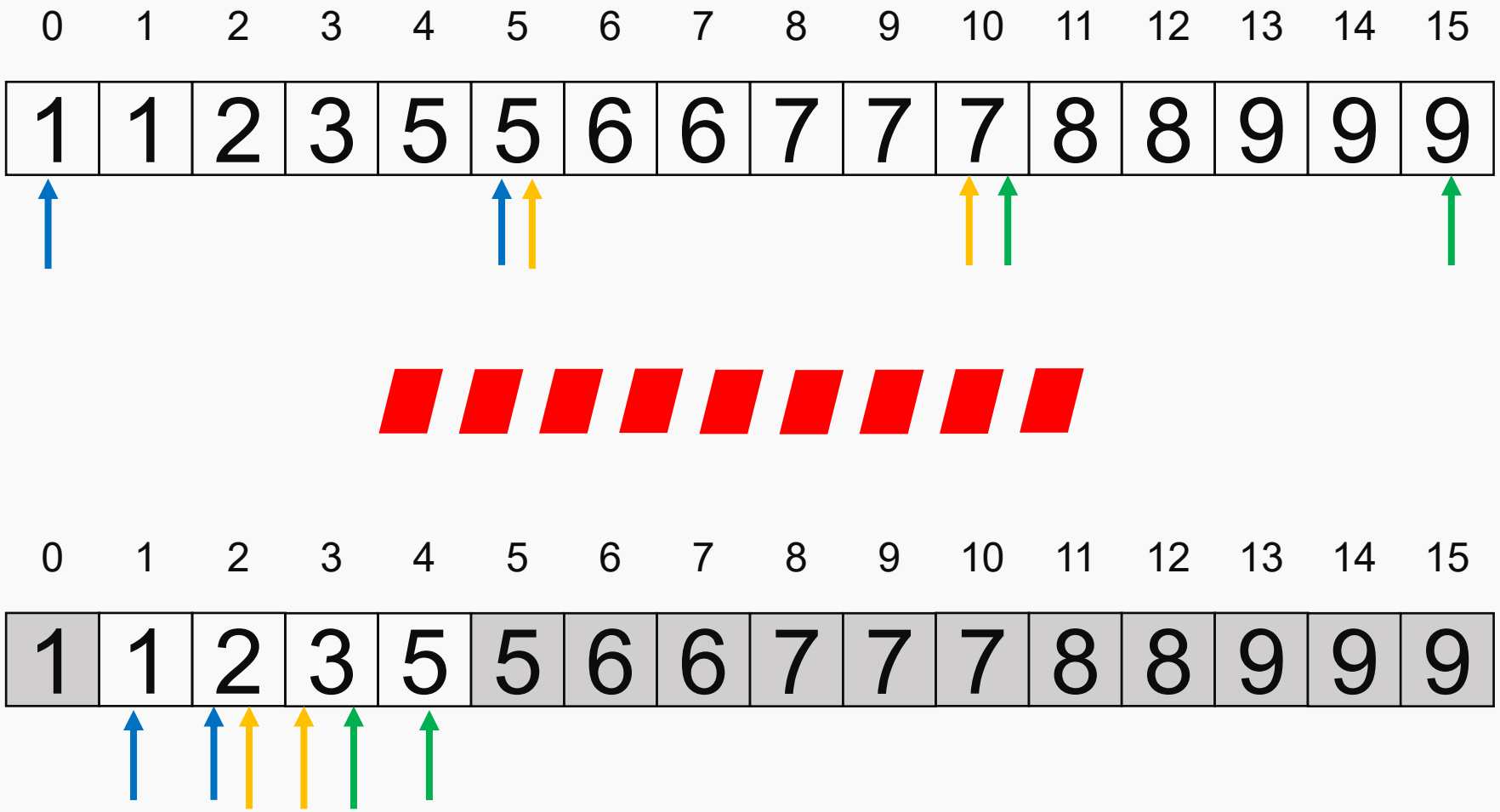
Căutare paralelă

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Operațiile aceste **NU** pot executa paralel

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

Căutare paralelă



Căutare paralelă – soluția 2

Mai greu de implementat
Mai puține thread-uri

Căutare paralelă – soluția 2

Căutăm 3

Între pozițiile 0 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 < 5 **3 < 7** **3 < 8**

Căutare paralelă – soluția 2

Căutăm 3

Între pozițiile 0 3

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 > 1

3 > 2

3 = 3 end

Complexitate: O($log_p(n)$)

Căutare paralelă – soluția 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 < 5 **3 < 7** **3 < 8**

Operațiile aceste **NU** pot executa paralel

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 > 1 **3 = 3** **end**

3 > 2

Căutare paralelă – soluția 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 < 5 **3 < 7** **3 < 8**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	5	5	6	6	7	7	7	8	8	9	9	9

3 > 1 **3 = 3**

3 > 2

Căutare paralelă - Complexitate

$$O(log_p(N))$$

Speedup?

Căutare paralelă - Complexitate

$$O(log_p(N))$$

Speedup?

$$S = \frac{log_2(N)}{log_p(N)}$$

Căutare paralelă - Complexitate

$$O(log_p(N))$$

Speedup?

$$S = \frac{log_2(N)}{log_p(N)} = \frac{log(P)}{log(2)} = log_2(P)$$

Merge sort paralel - idee

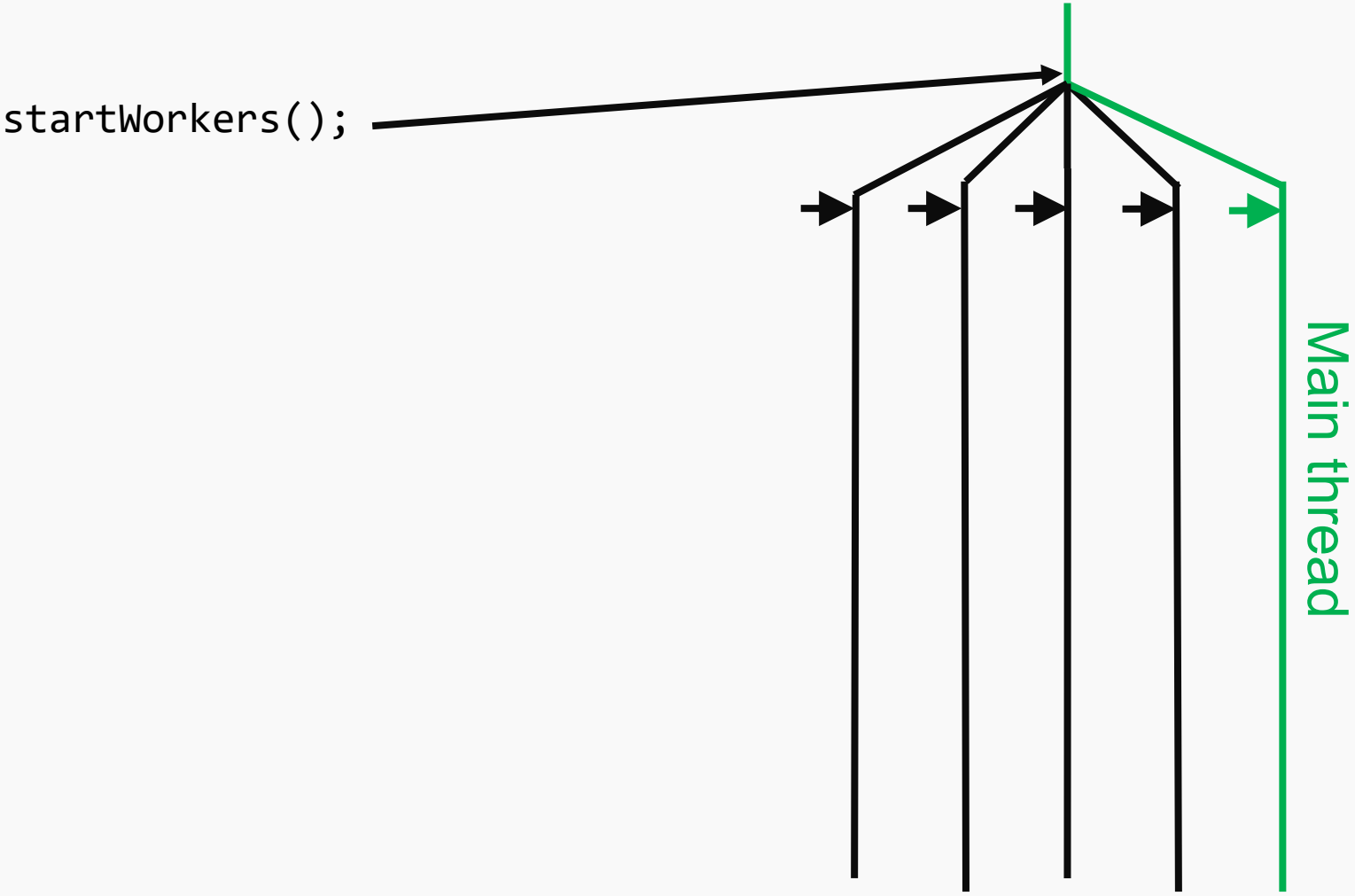
Operația de merge poate și ea fi paralelizată.

Pentru a o paraleliza ne bazăm pe cătare binară (sau chiar paralelă) și pe rank sort.

Executor Service
sau
Replicated Workers
sau
Thread Pool

Abordare de probleme recursive în paralel

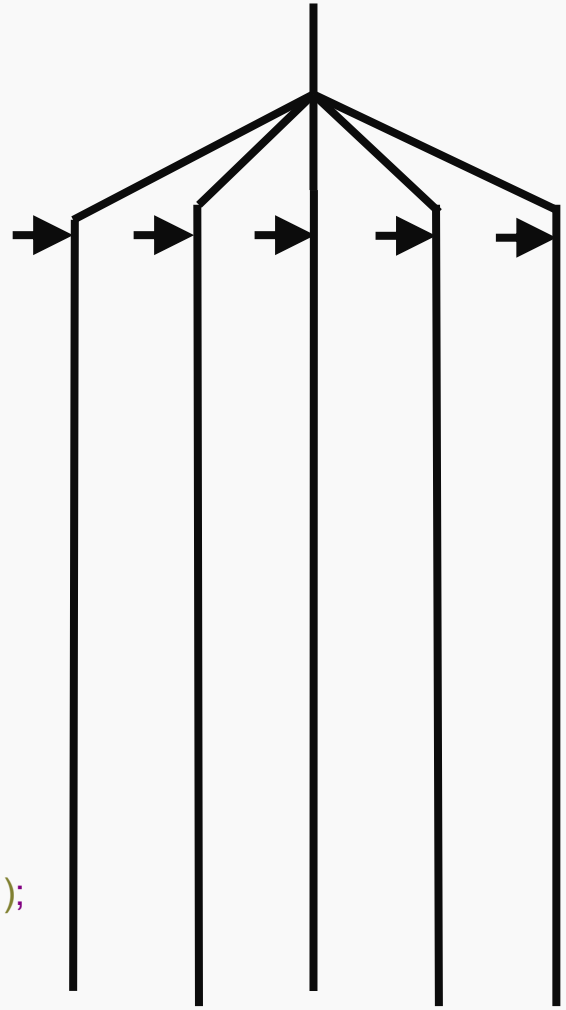
Replicated Workers



Replicated Workers

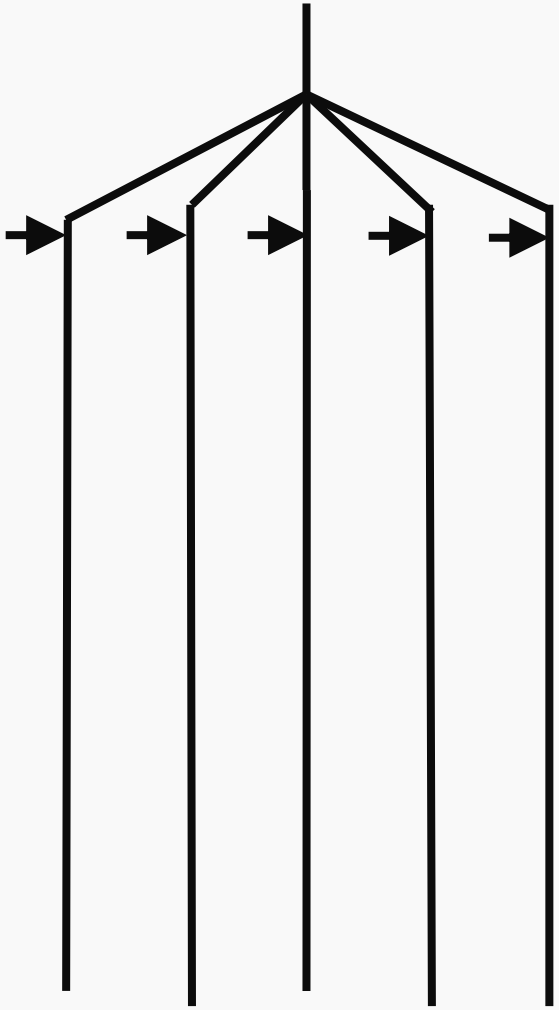
```
Task makeTask(int i)
{
        Task task;
        int * newData = (int*)malloc(sizeof(int));
        newData[0]=i;
        task.data=newData;
        task.runTask = printSomething;
        return task;
}

void printSomething (void * data, int thread_id)
{
        int task_id = *(int*)data;
        if(task_id>N) {
                forceShutDownWorkers();
                return;
        }
        printf("Something %i from thread %i\n", task_id, thread_id);
        putTask(makeTask(task_id+1));
}
```

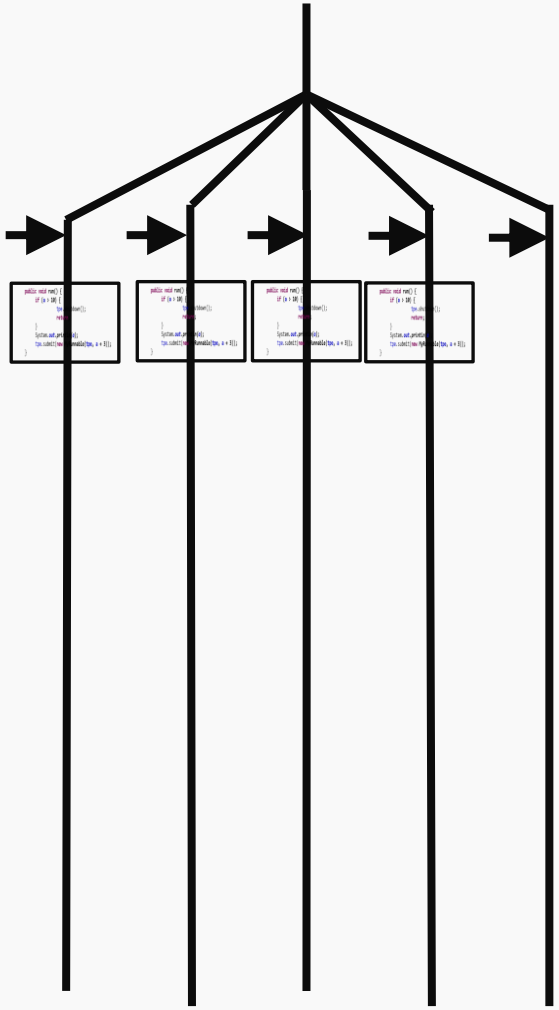


Replicated Workers

putTask(Task1);
putTask(Task2);
putTask(Task3);
.....



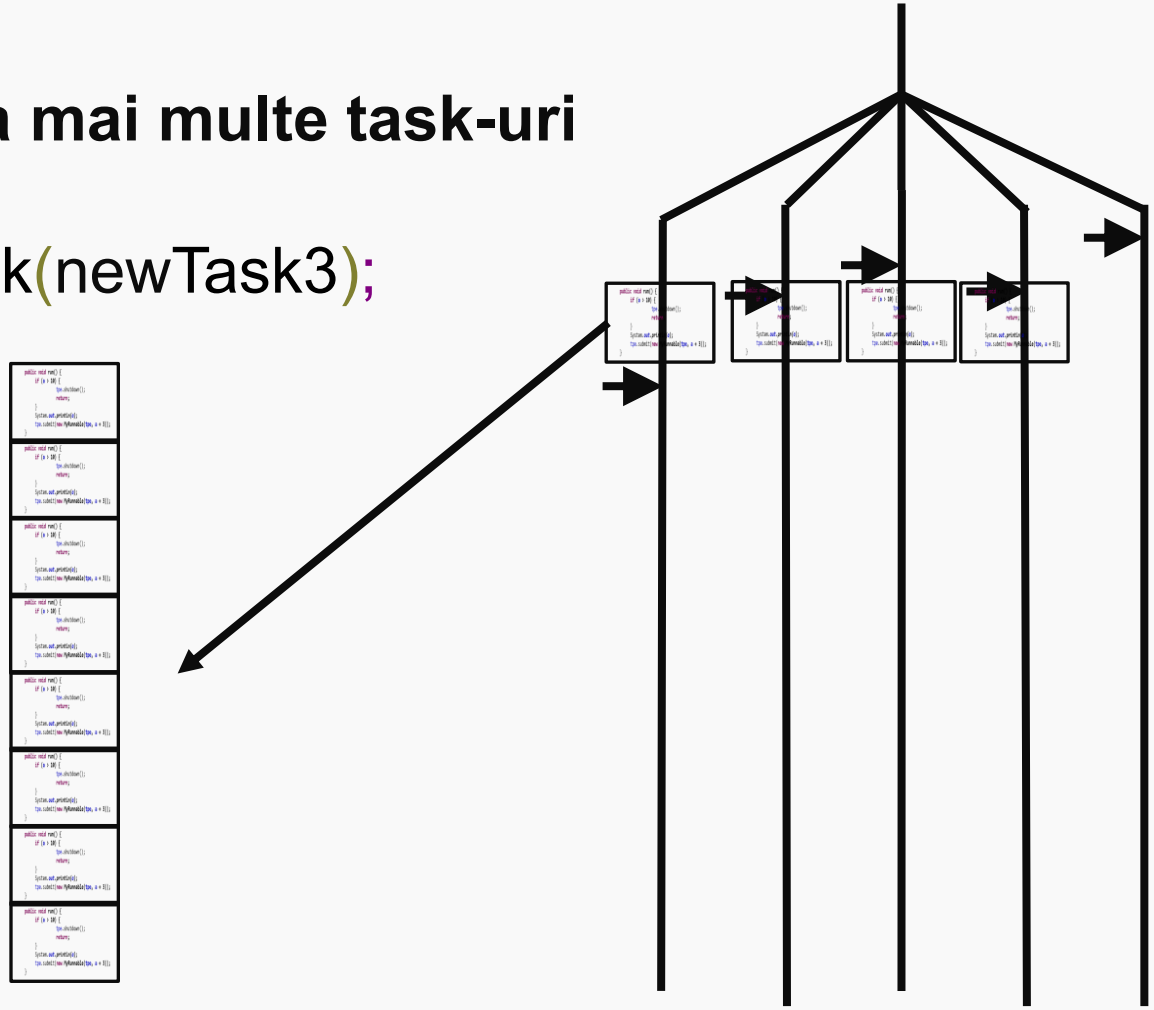
Replicated Workers



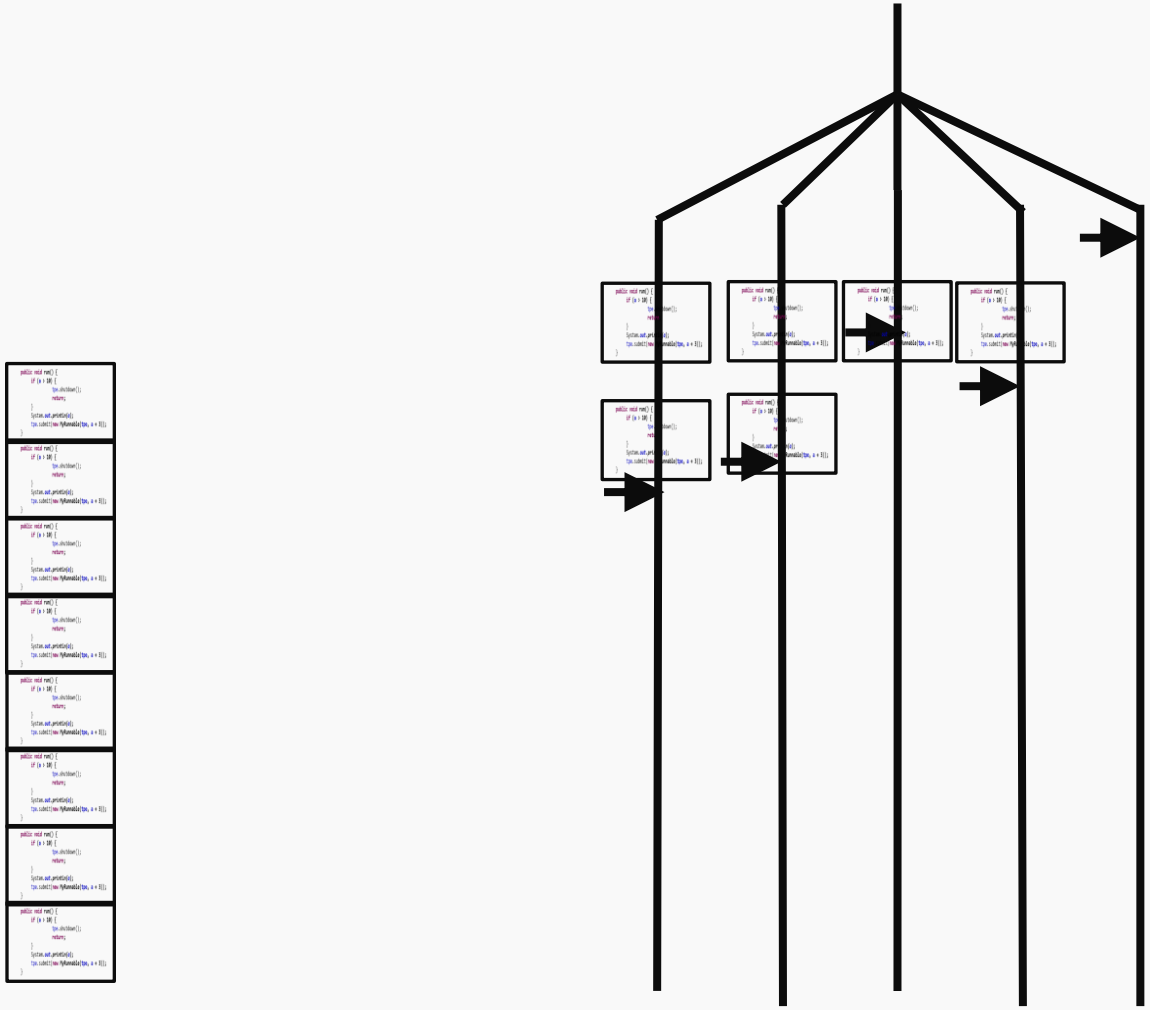
Replicated Workers

Task-urile pot crea mai multe task-uri

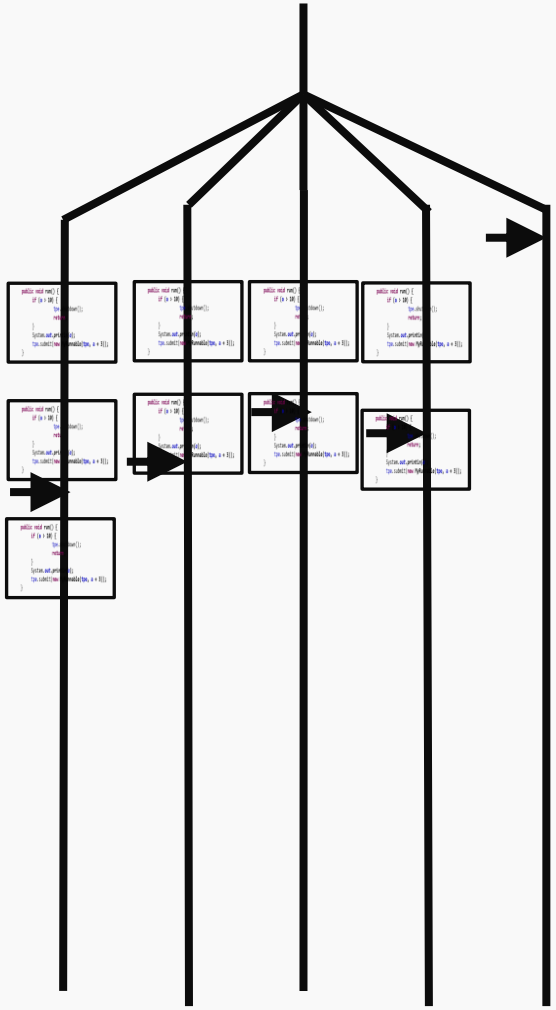
putTask(newTask3);



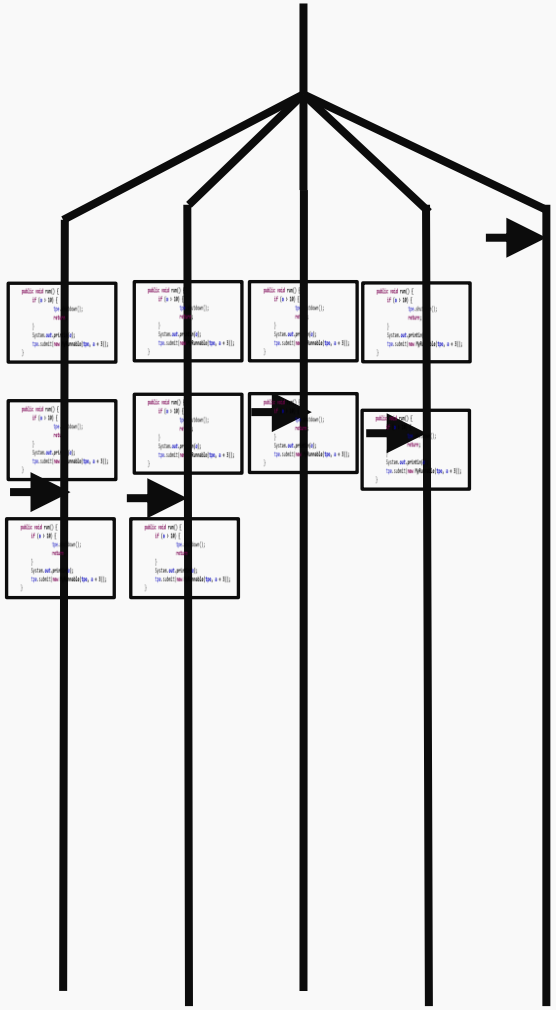
Replicated Workers



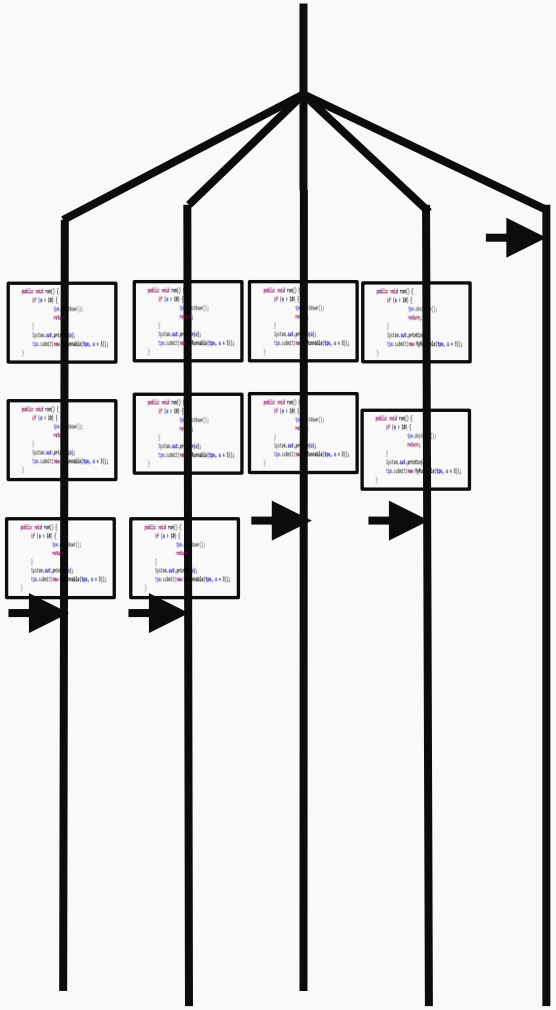
Replicated Workers



Replicated Workers

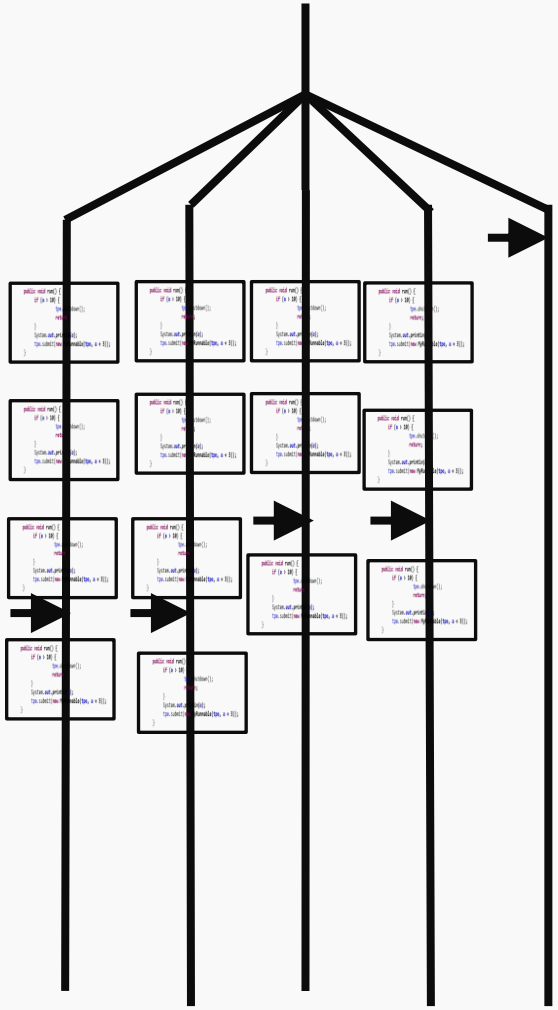


Replicated Workers



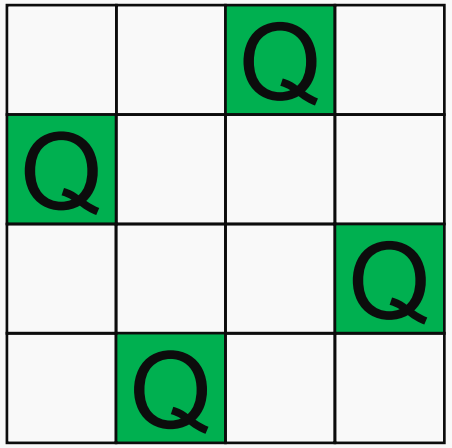
Replicated Workers

Când oprim thread-urile?
Depinde de problemă.
Dacă e suficient putem să oprim imediat după găsirea unei soluții.
Trebuie ținut cont că unele probleme nu au soluție.



```
joinWorkerThreads();
```

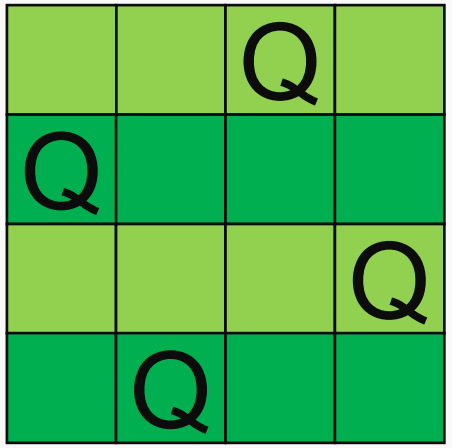
N Queens Problem



Se cere aranjare a N regine pe o tablă NxN în așa fel încât să nu se atace

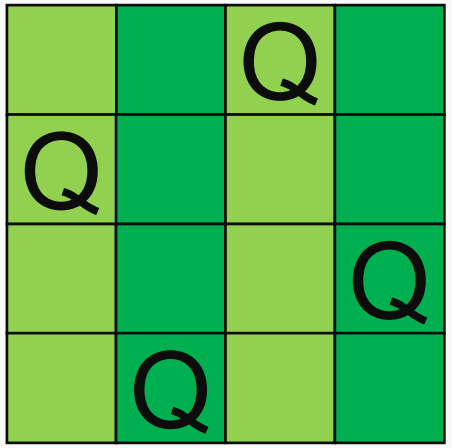
N Queens Problem

Nu avem voie mai mult de o regină pe linie



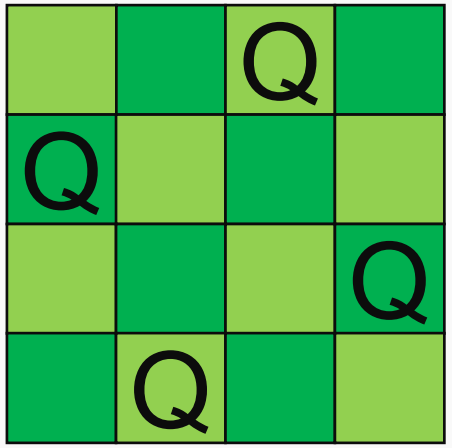
N Queens Problem

Nu avem voie mai mult de o regină pe coloană

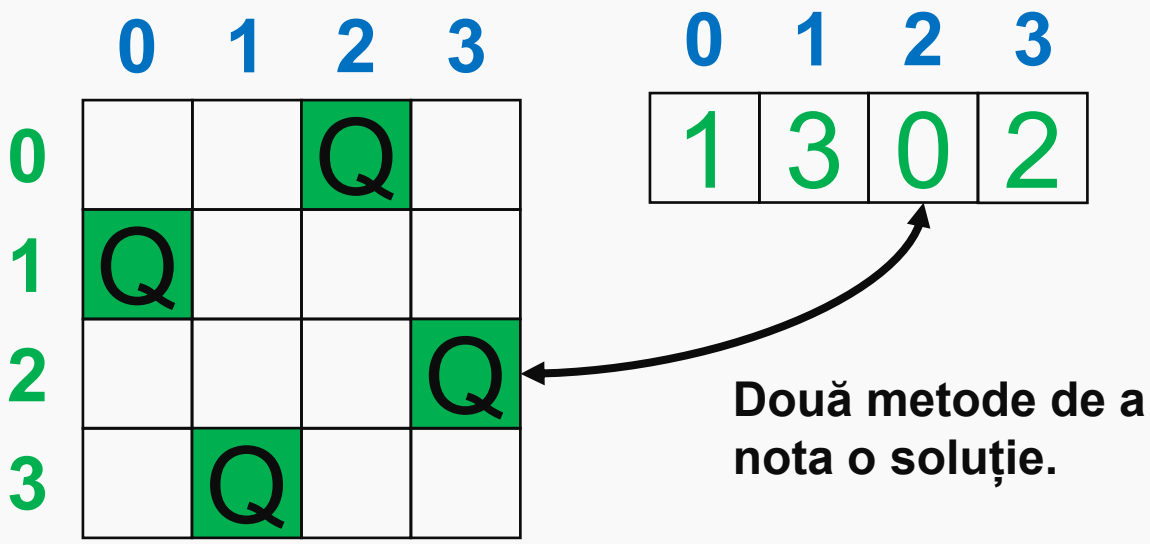


N Queens Problem

Nu avem voie mai mult de o regină pe diagonală



N Queens Problem



Poziția din vector reprezintă coloana pe care este așezată regina.

Valoarea din vector reprezintă linia pe care este așezată regina.

N Queens Problem – Soluție

	0	1	2	3
0	Q			
1				
2				
3				

0	1	2	3
0			

N Queens Problem – Soluție

	0	1	2	3
0	Q	Q		
1				
2				
3				

0	1	2	3
0	0		

Conflict linie

N Queens Problem – Soluție

	0	1	2	3
0	Q			
1		Q		
2				
3				

0	1	2	3
0	1		

Conflict diagonală

N Queens Problem – Soluție

	0	1	2	3
0	Q			
1				
2		Q		
3				

0	1	2	3
0	2		

OK.

Și tot așa până punem toate reginele

N Queens Problem – Soluție paralelă

	0	1	2	3
0				
1				
2				
3				

N Queens Problem – Soluție paralelă

0	1	2	3	
0	0			x
0	1			x
0	2			
0	3			

0	1	2	3	
1	0			x
1	1			x
1	2			x
1	3			

Și tot așa…

0	1	2	3	
2	0			
2	1			x
2	2			x
2	3			x

0	1	2	3	
3	0			
3	1			
3	2			x
3	3			x