Arhitecturi Paralele
Abordarea algoritmilor în mod paralel

Lect. Dr. Ing. Cristian Chilipirea
cristian.chilipirea@mta.ro

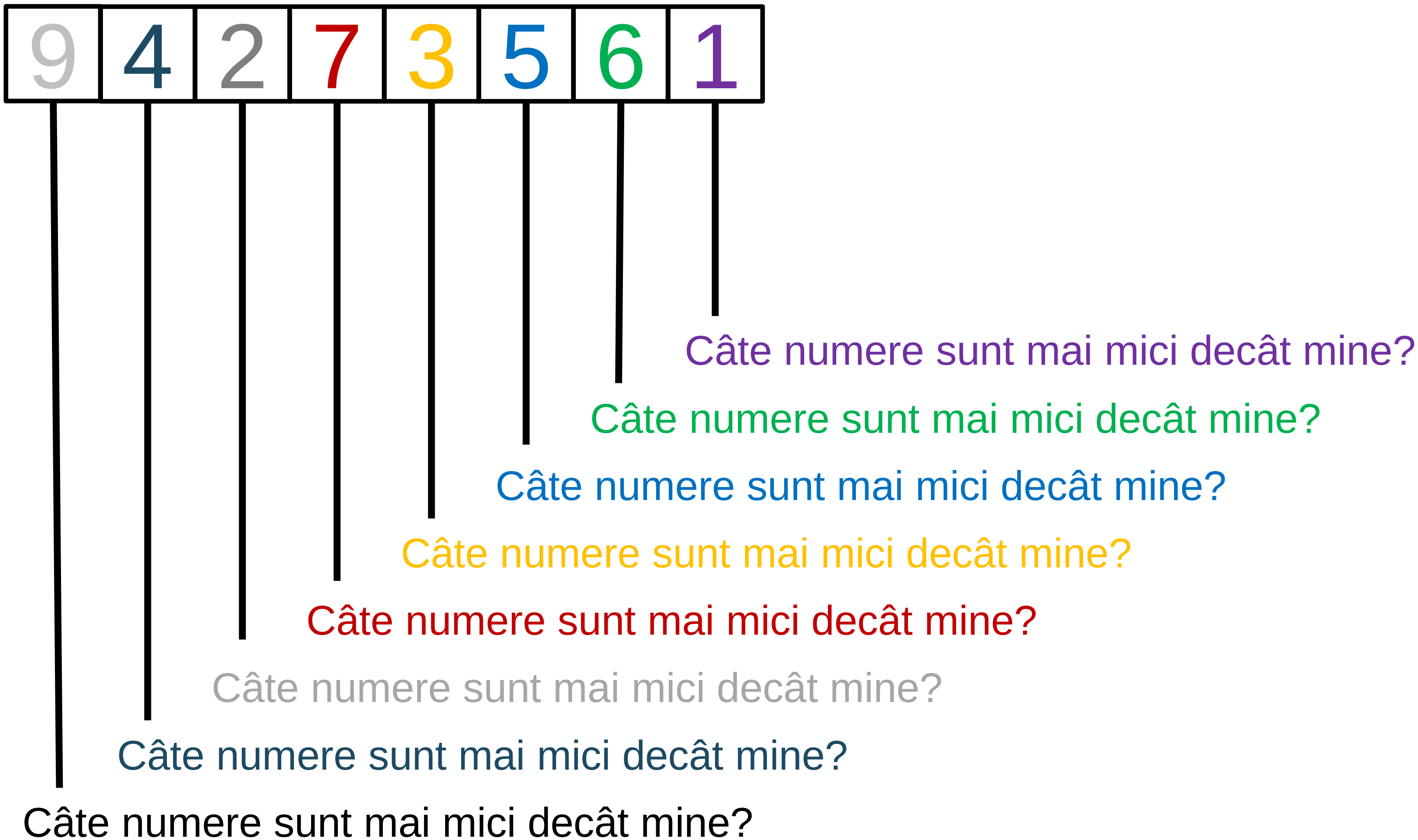
Curs susținut în parteneriat cu Prof. Florin Pop

Rank Sort

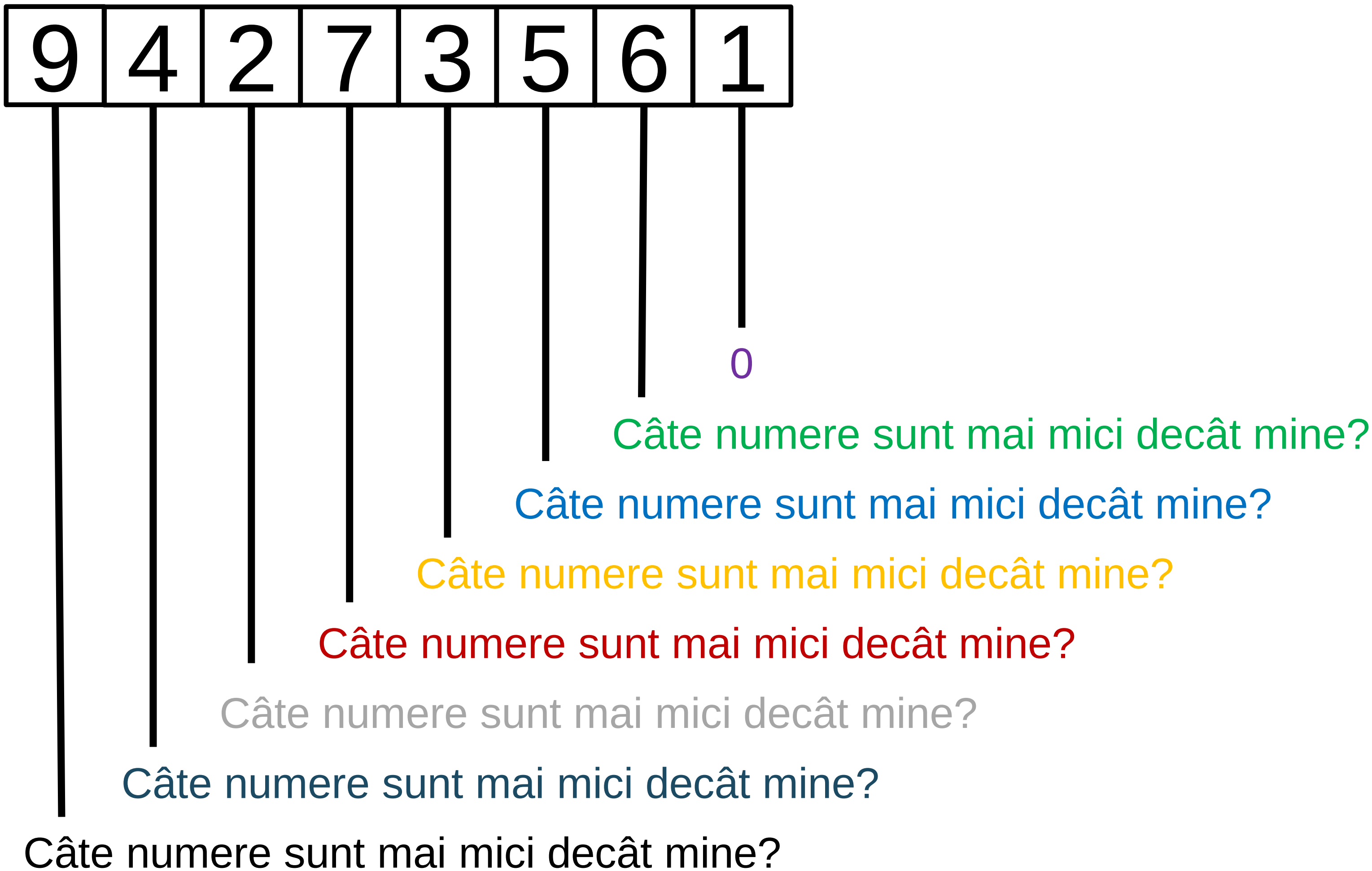
9	4	2	7	6	5	6	1

1	2	4	5	6	6	7	9

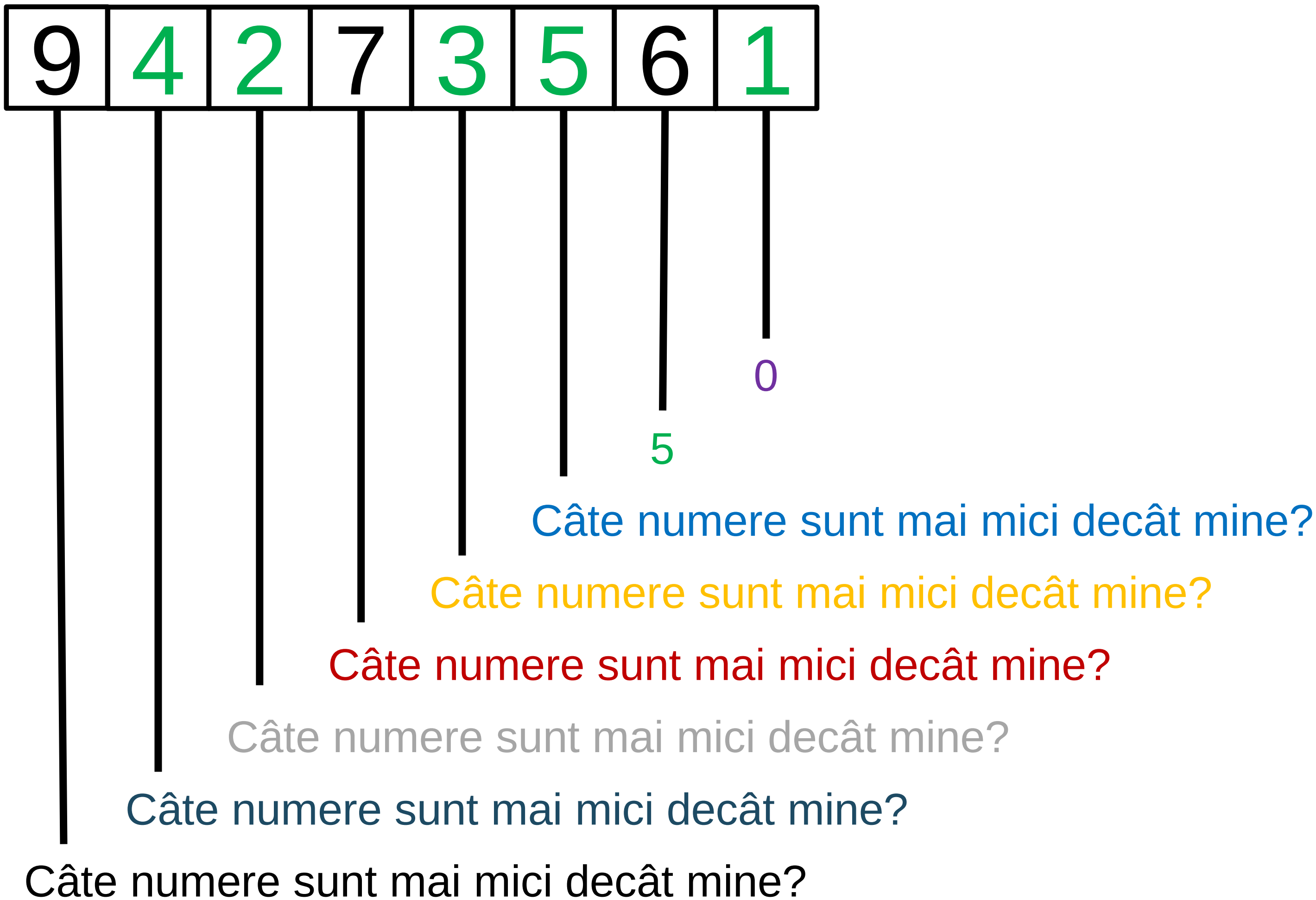
Rank Sort


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

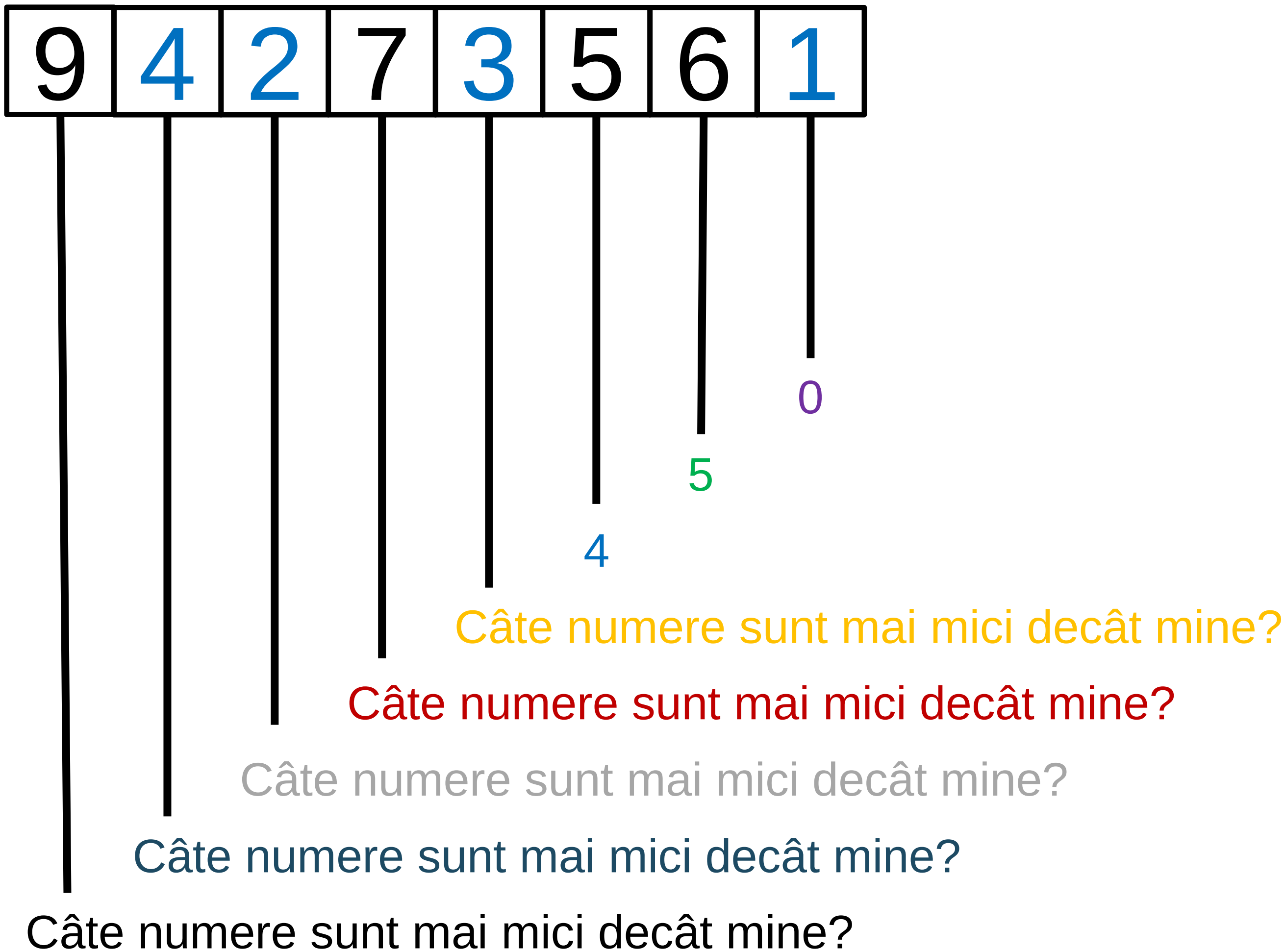

9 4 2 7 3 5 6 1
0
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

Rank Sort

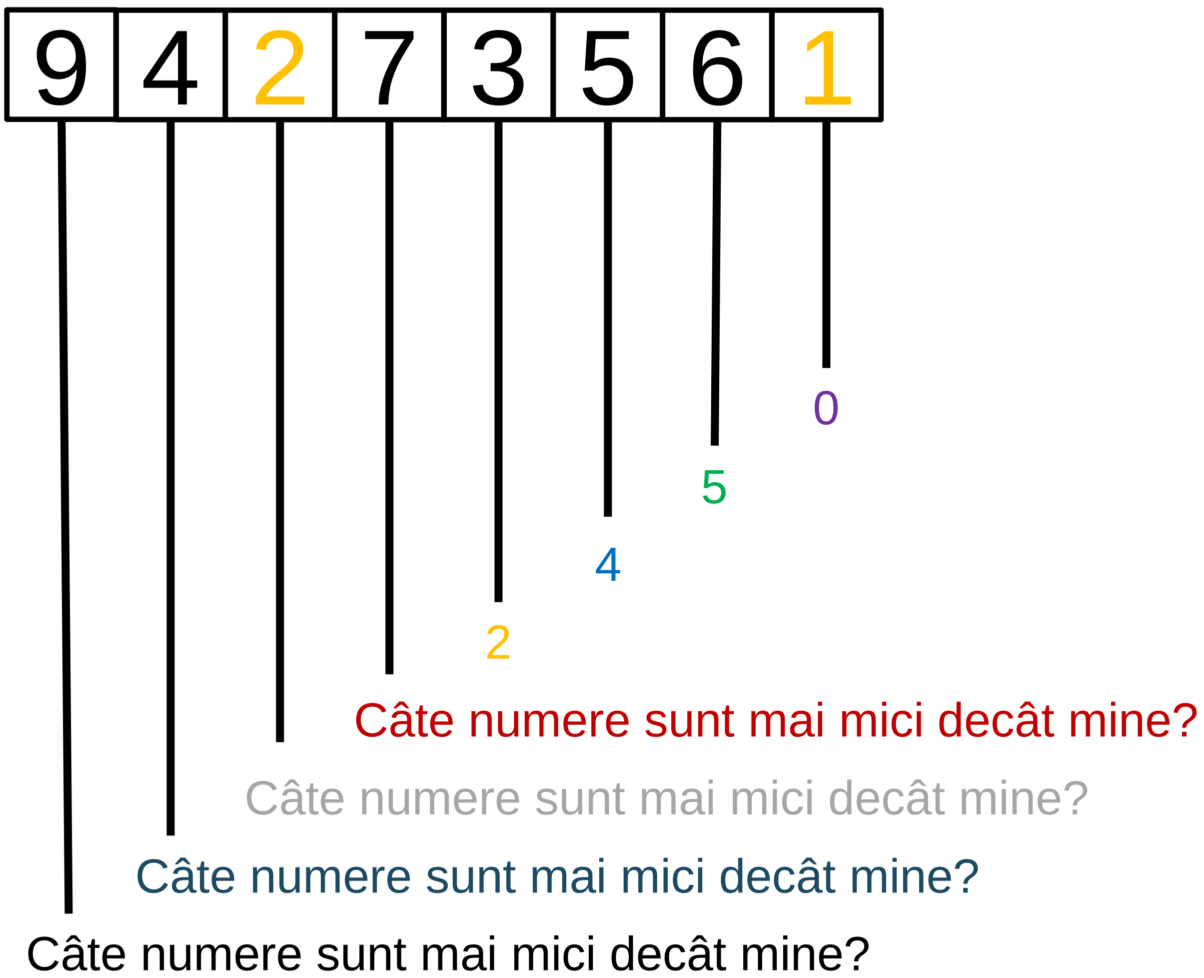


9 4 2 7 3 5 6 1
0
5
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?
Câte numere sunt mai mici decât mine?

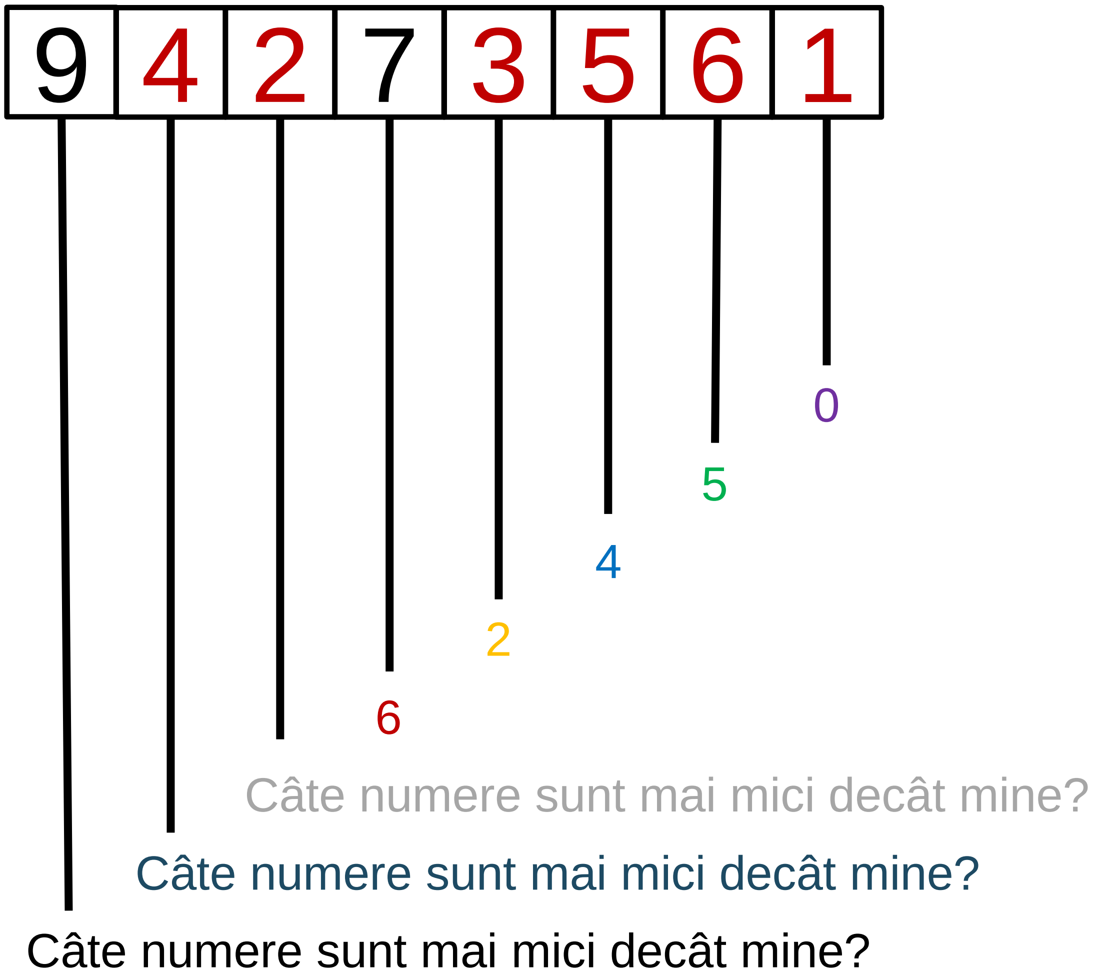
Rank Sort


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?

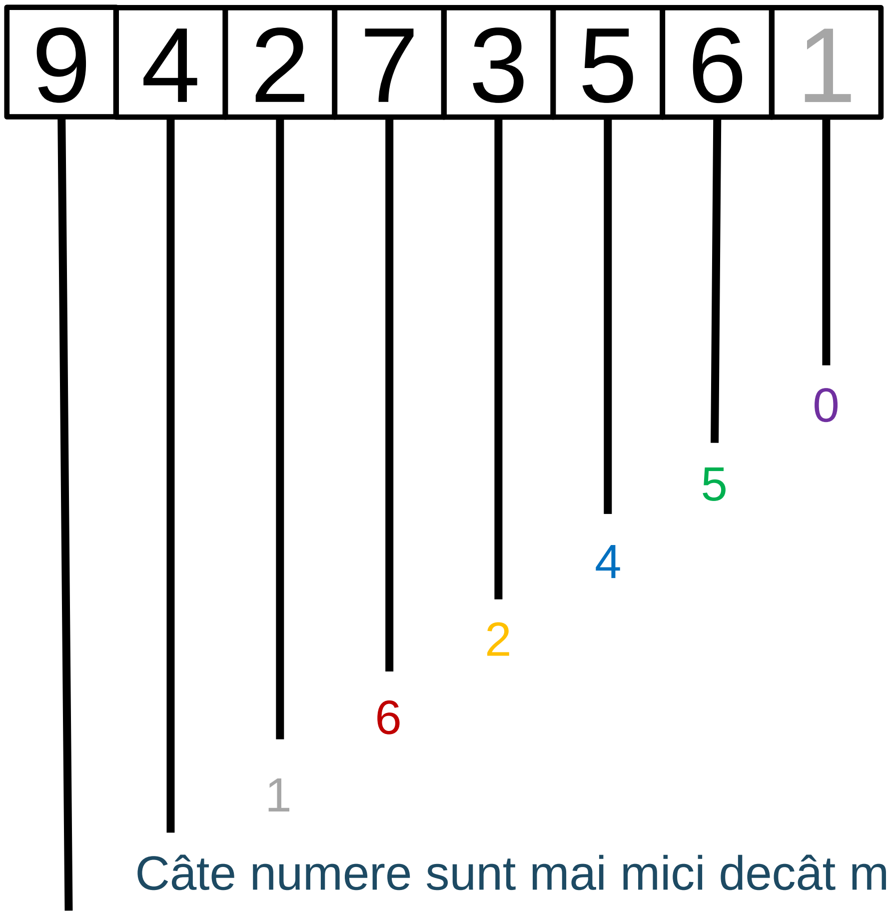
Rank Sort


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?

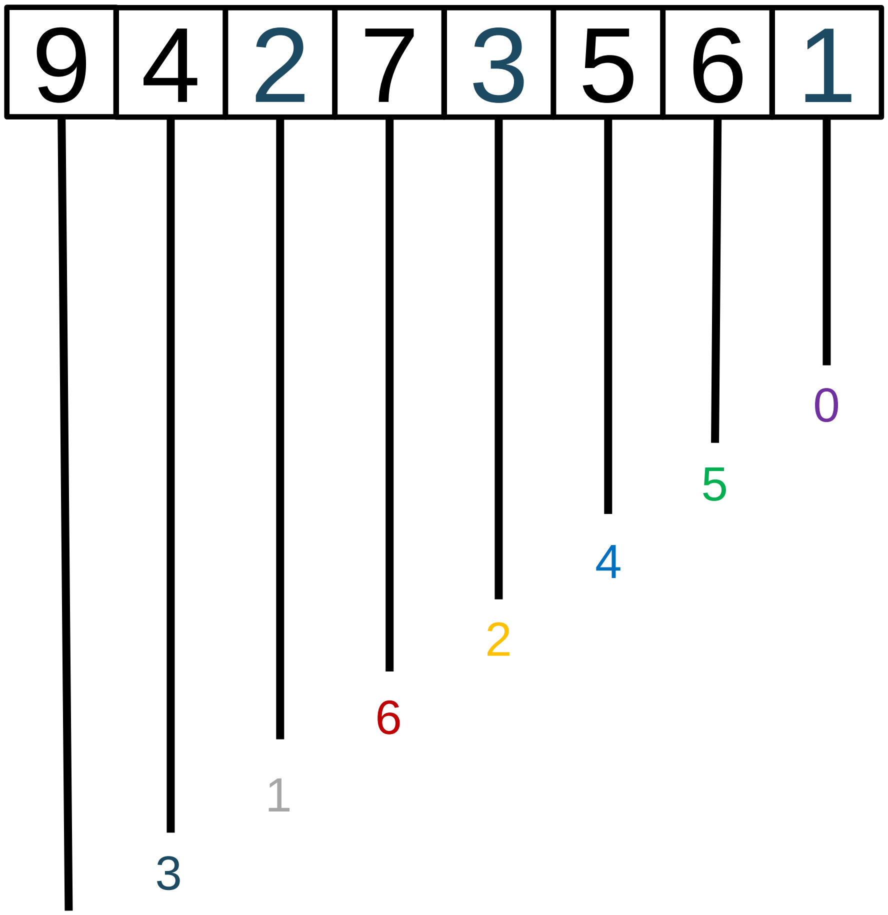
Rank Sort



Câte numere sunt mai mici decât mine?

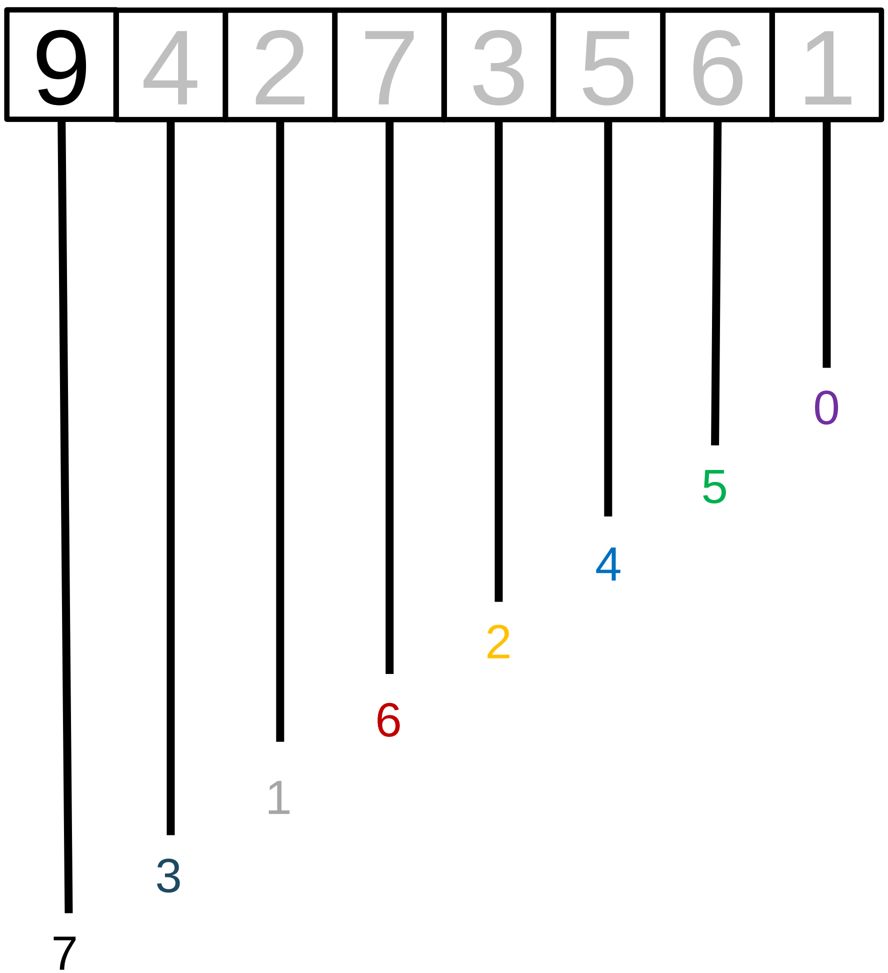
Câte numere sunt mai mici decât mine?

Rank Sort

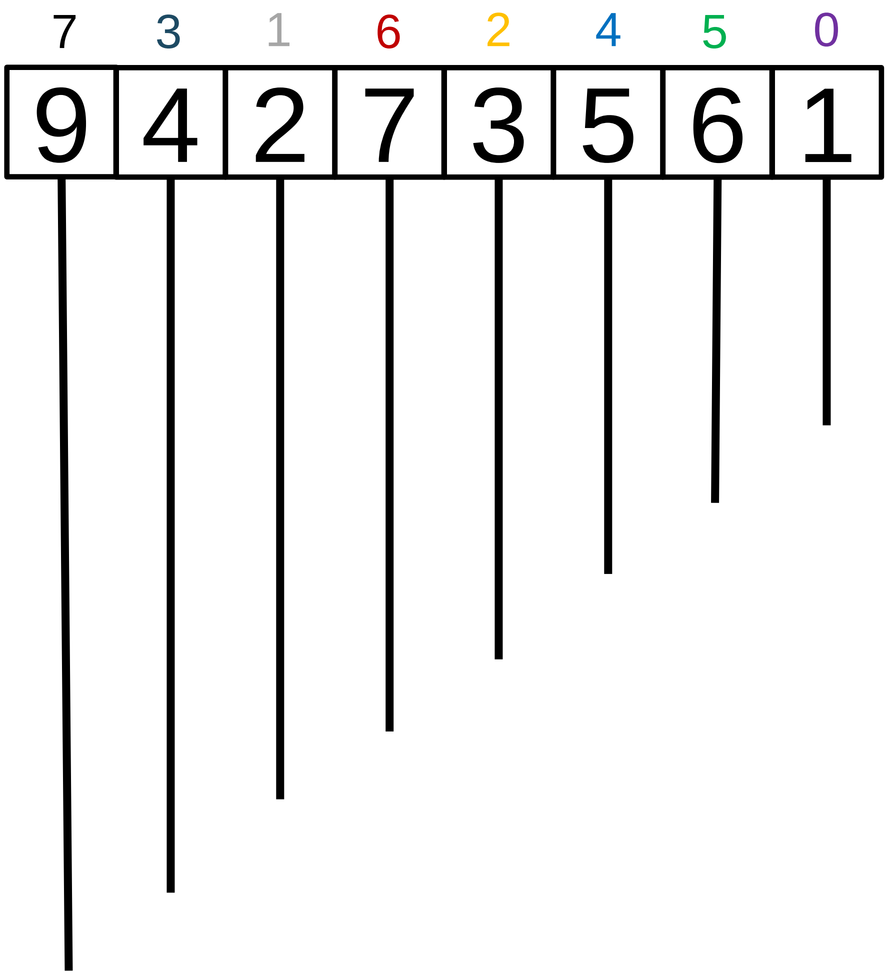


Câte numere sunt mai mici decât mine?

Rank Sort

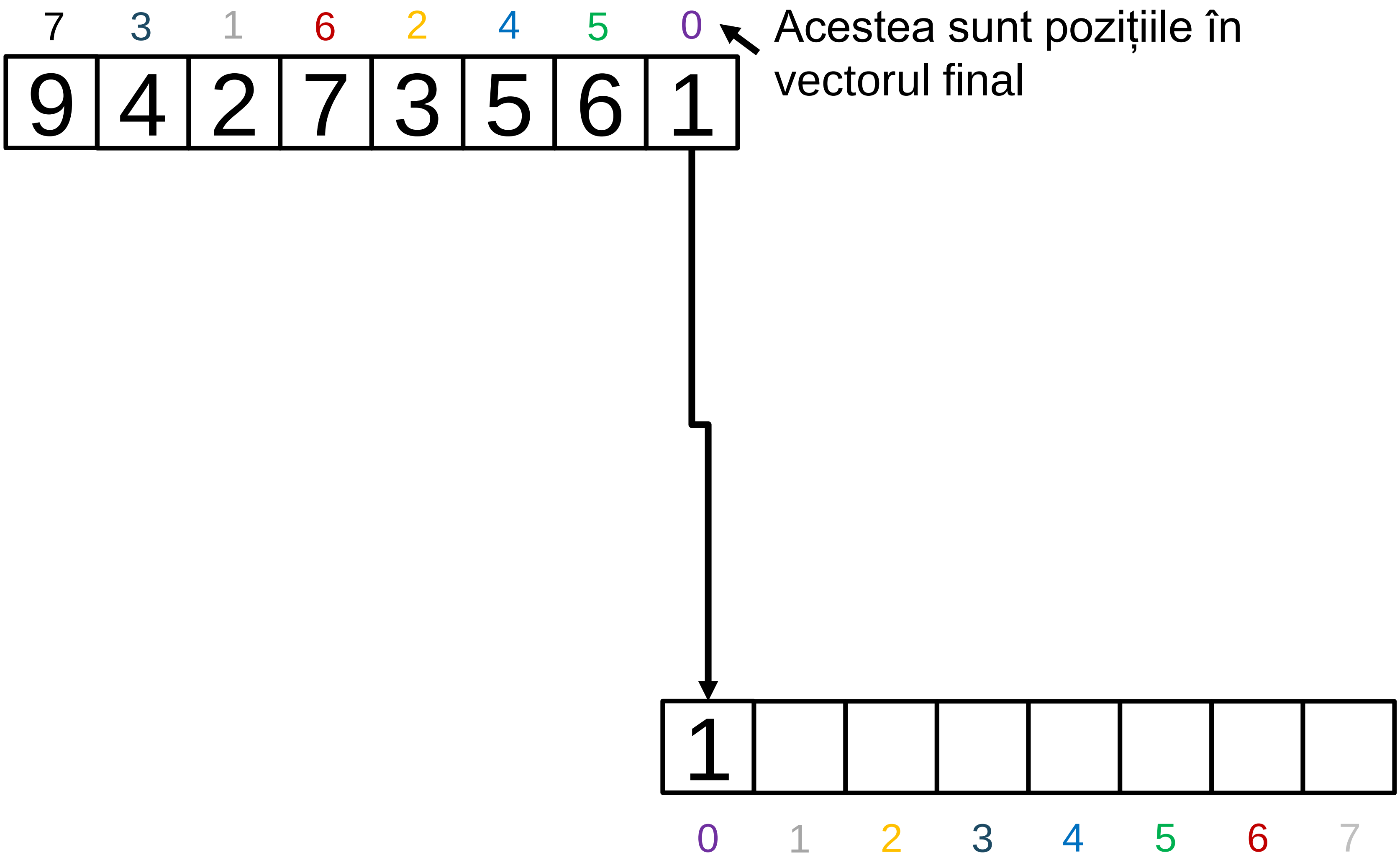


Rank Sort

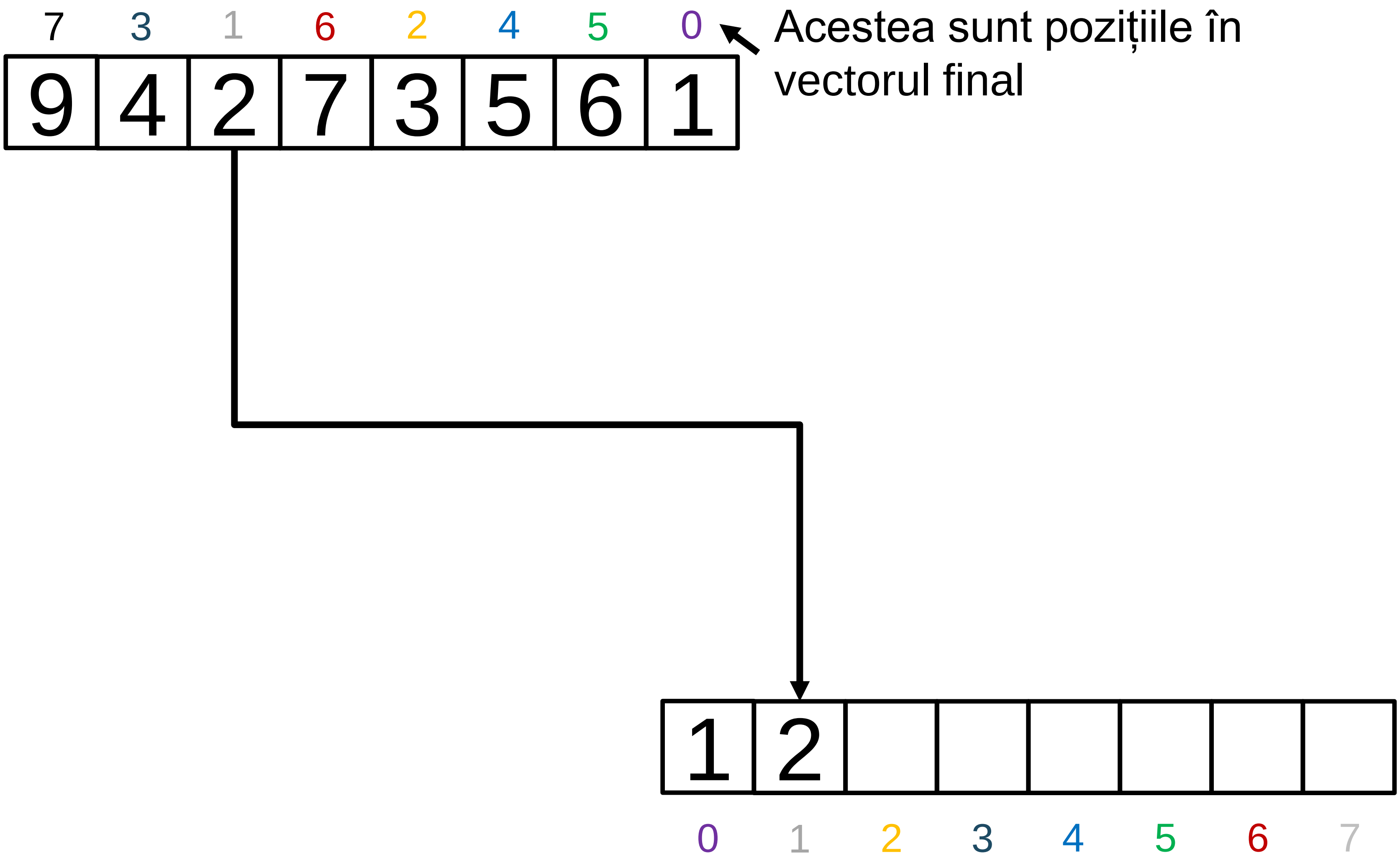


Acestea sunt pozițiile în vectorul final

Rank Sort

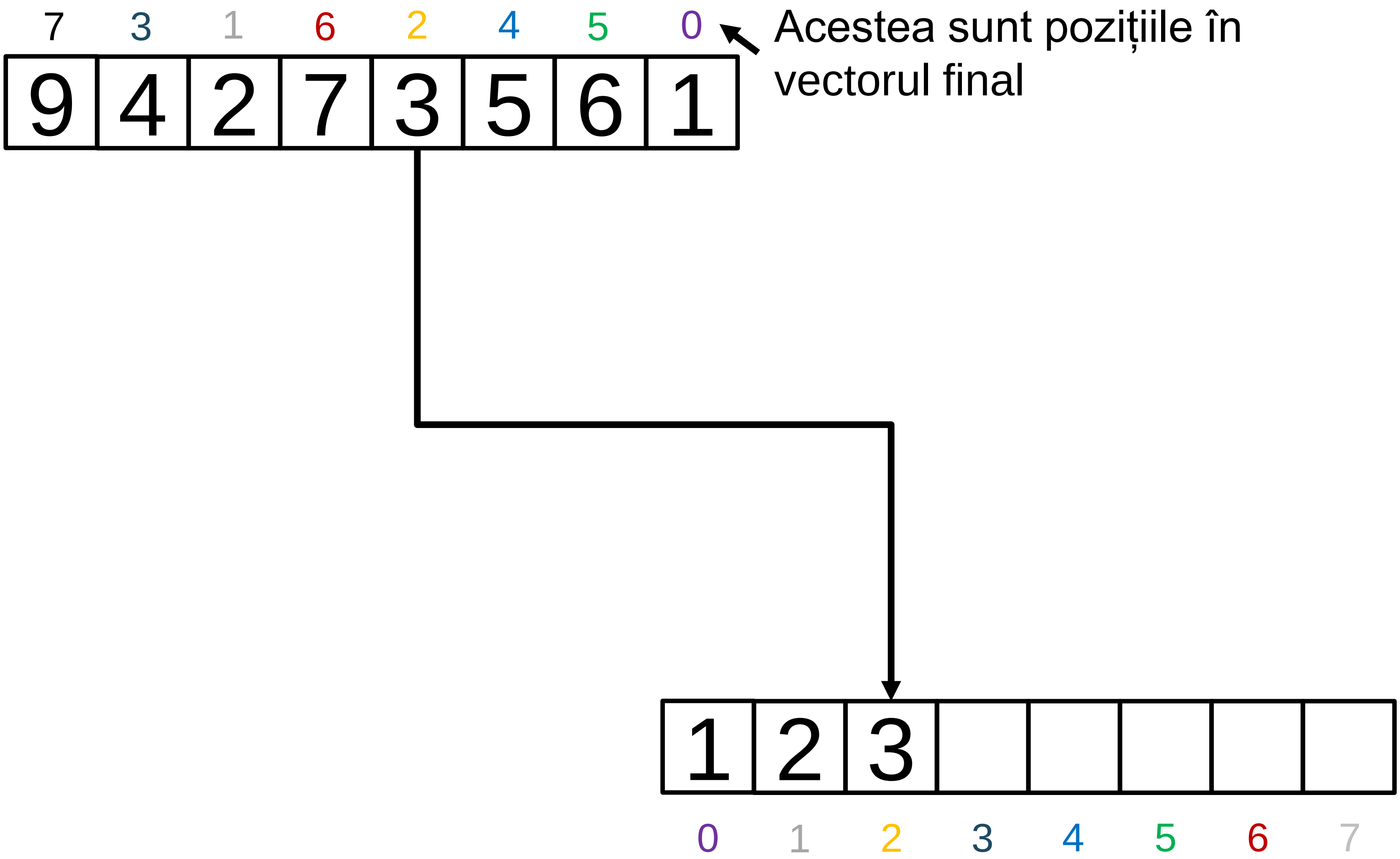

7 3 1 6 2 4 5 0
Acestea sunt pozițiile în vectorul final
9 4 2 7 3 5 6 1
1
0 1 2 3 4 5 6 7

Rank Sort

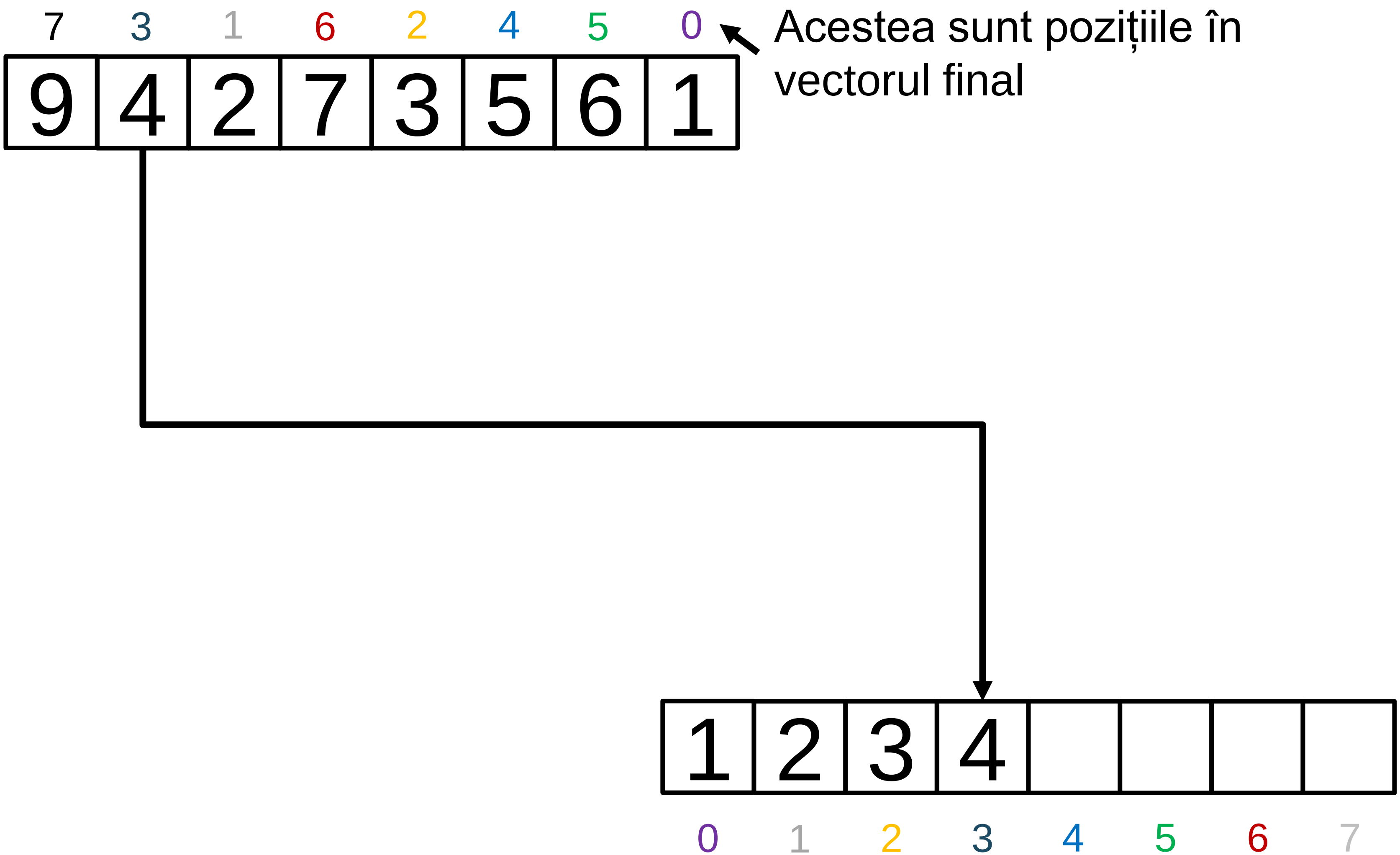


7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2
0 1 2 3 4 5 6 7

Rank Sort

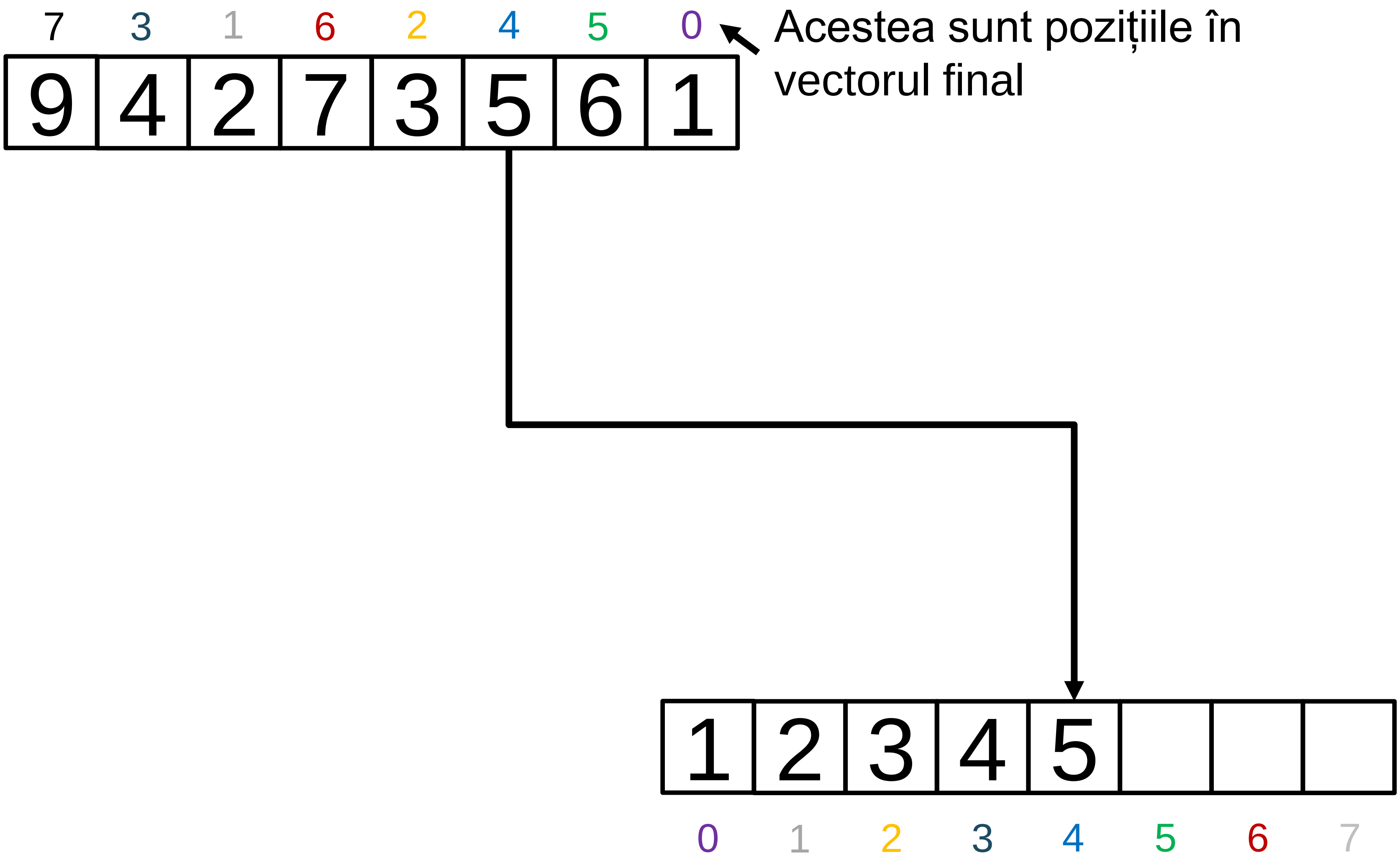

7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2 3
0 1 2 3 4 5 6 7

Rank Sort

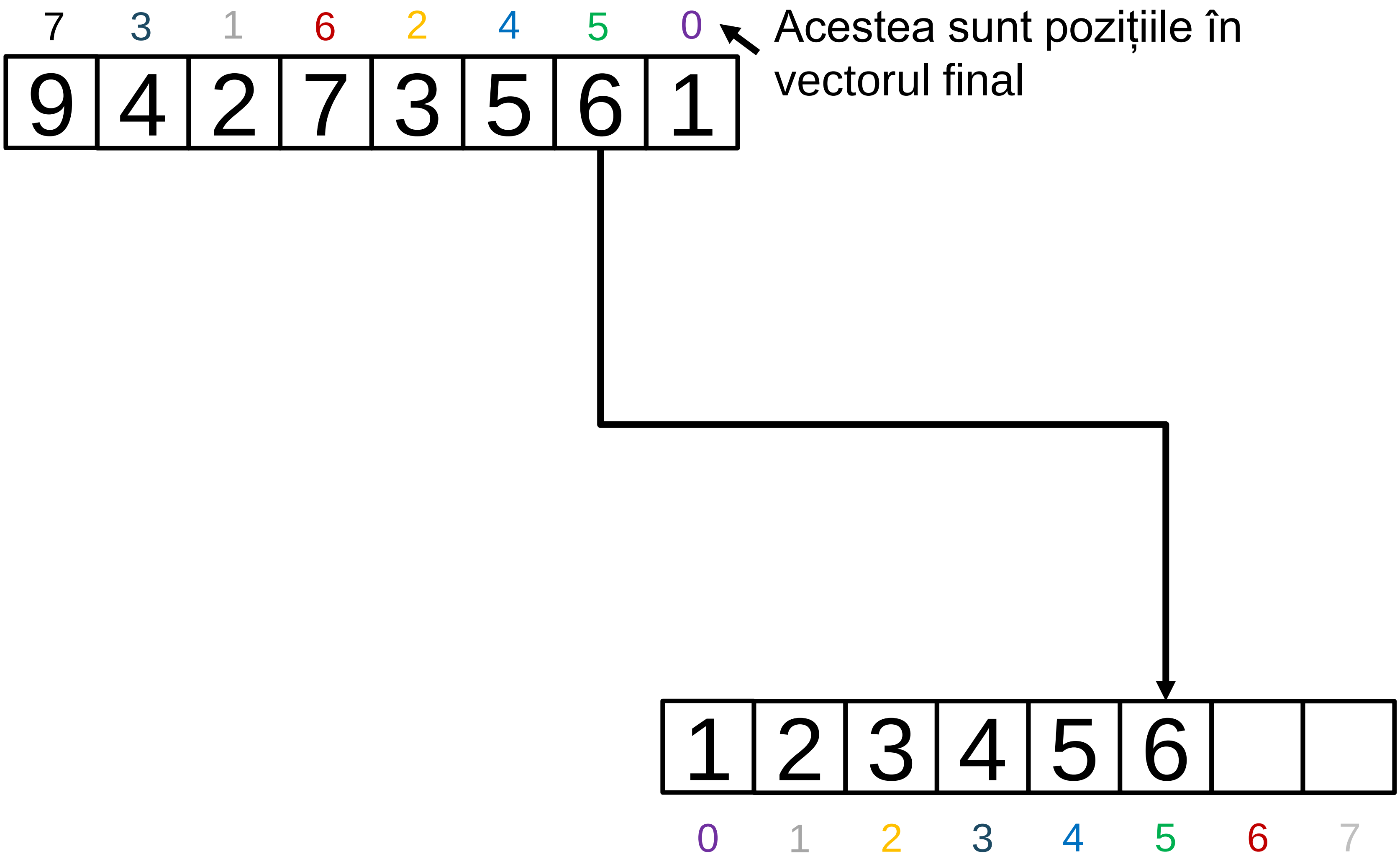


7 3 1 6 2 4 5 0
9 4 2 7 3 5 6 1
Acestea sunt pozițiile în vectorul final
1 2 3 4
0 1 2 3 4 5 6 7

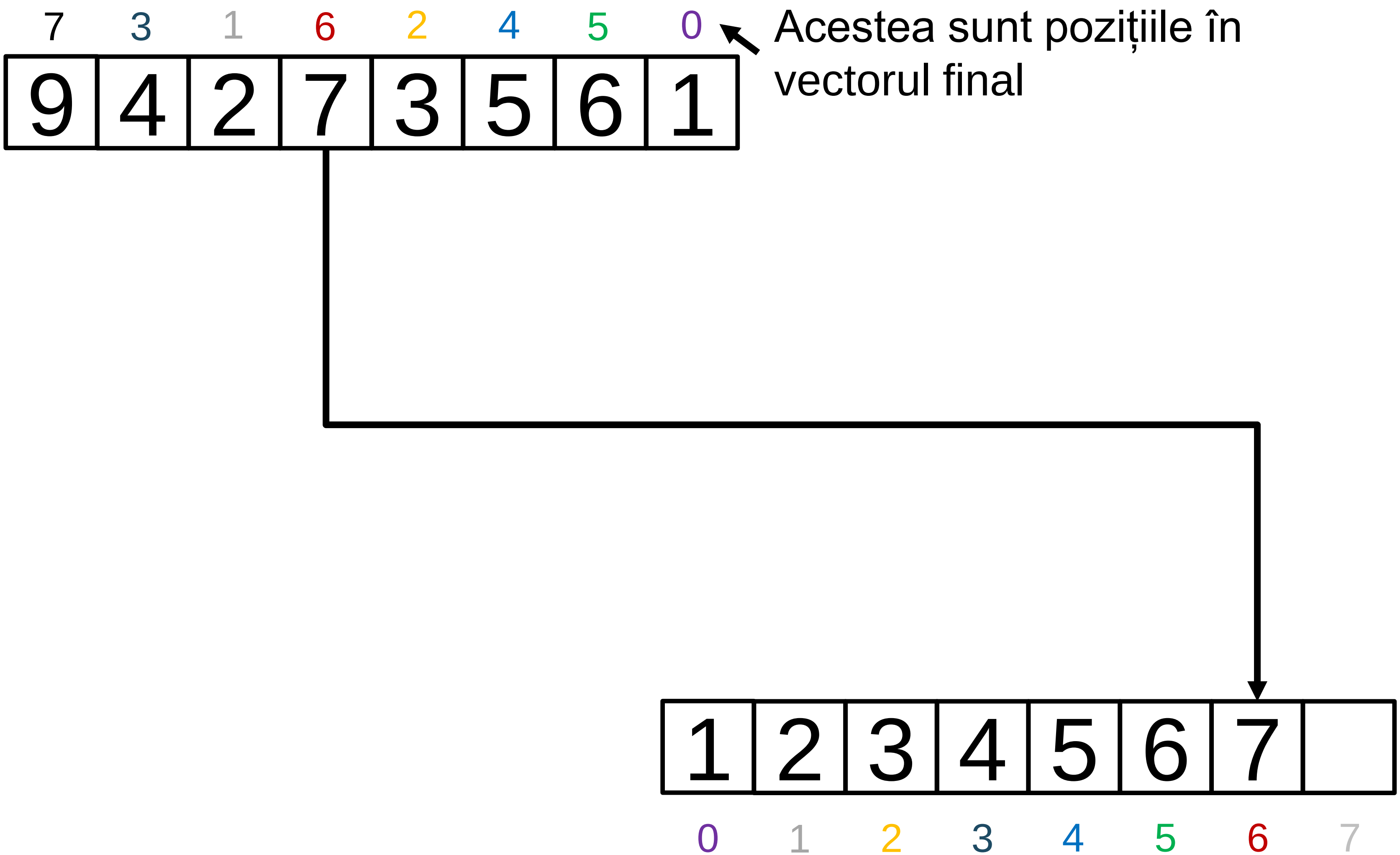
Rank Sort


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

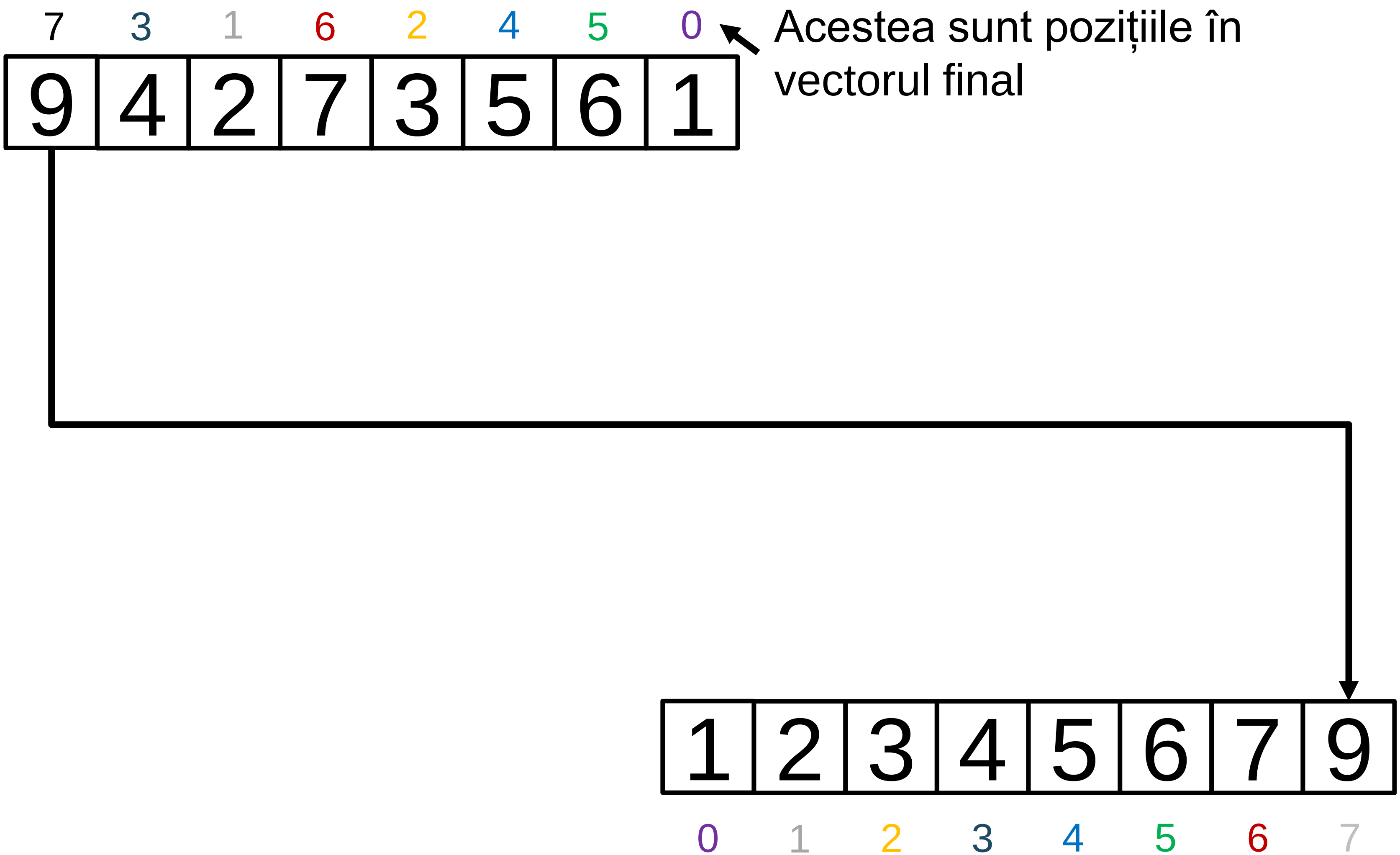

7 3 1 6 2 4 5 0
Acestea sunt pozițiile în vectorul final
9 4 2 7 3 5 6 1
1 2 3 4 5 6
0 1 2 3 4 5 6 7

Rank Sort



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

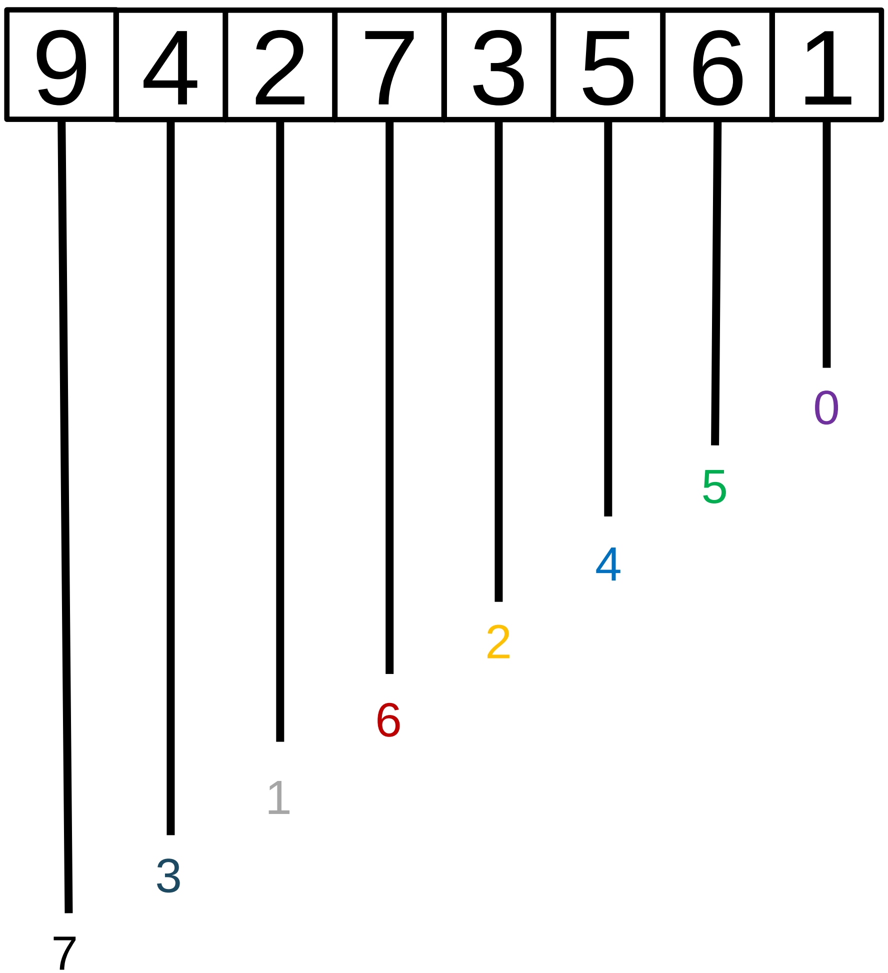
Rank Sort Paralel

9	4	2	7	3	5	6	1

Răspunsul la toate întrebările poate fi determinat în paralel

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?

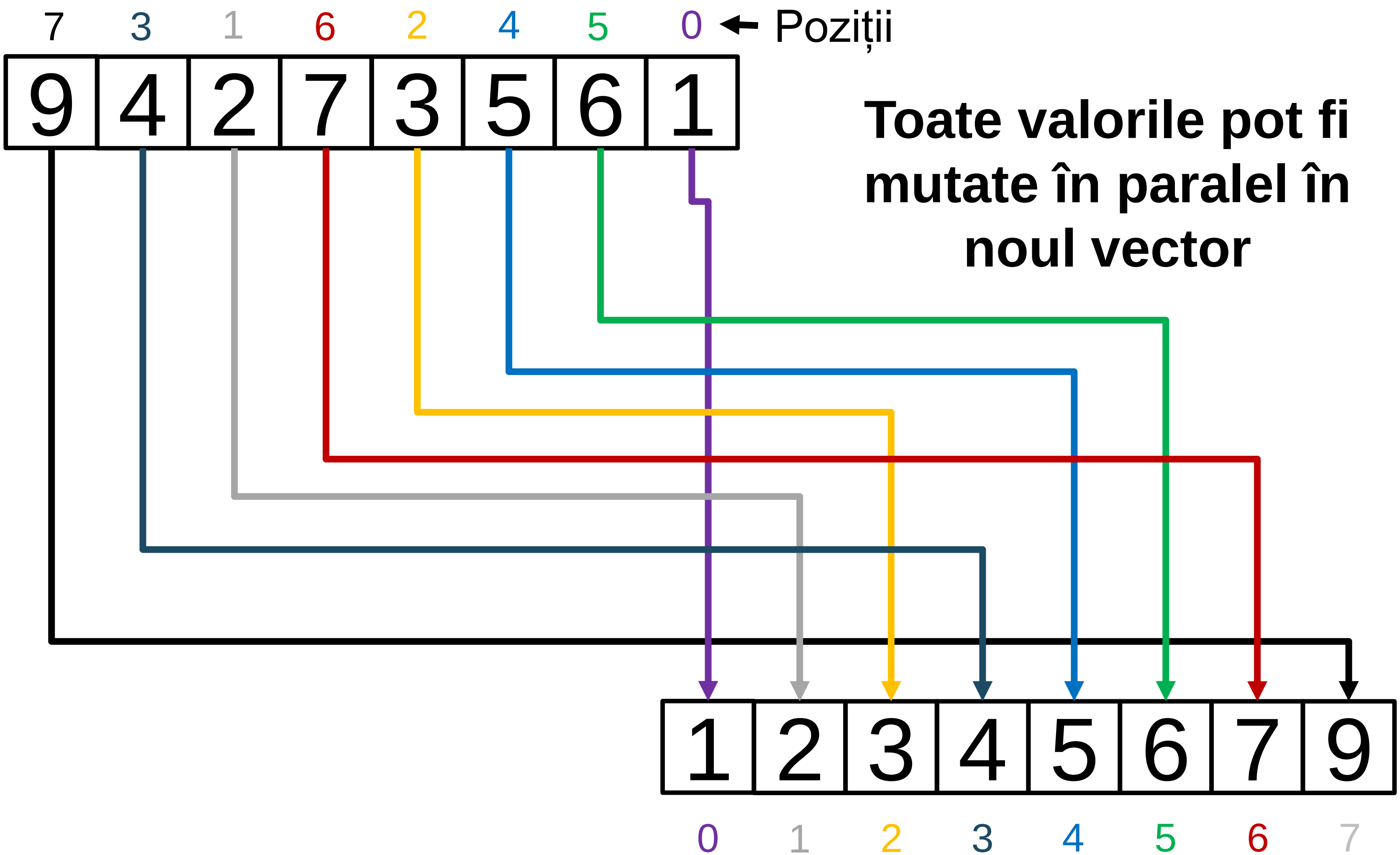
Rank Sort Paralel



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

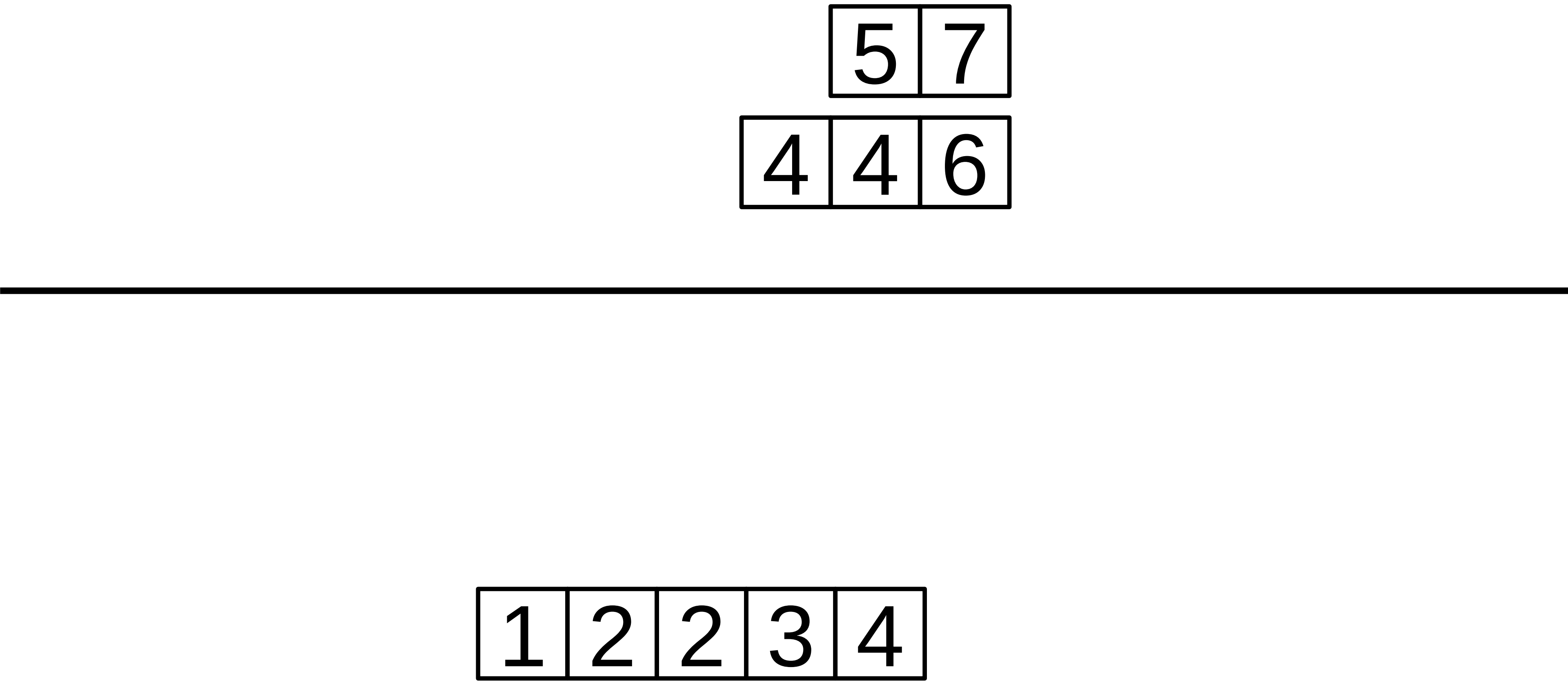
Algoritm merge

4	5	7

4	4	6

1	2	2	3

Algoritm merge



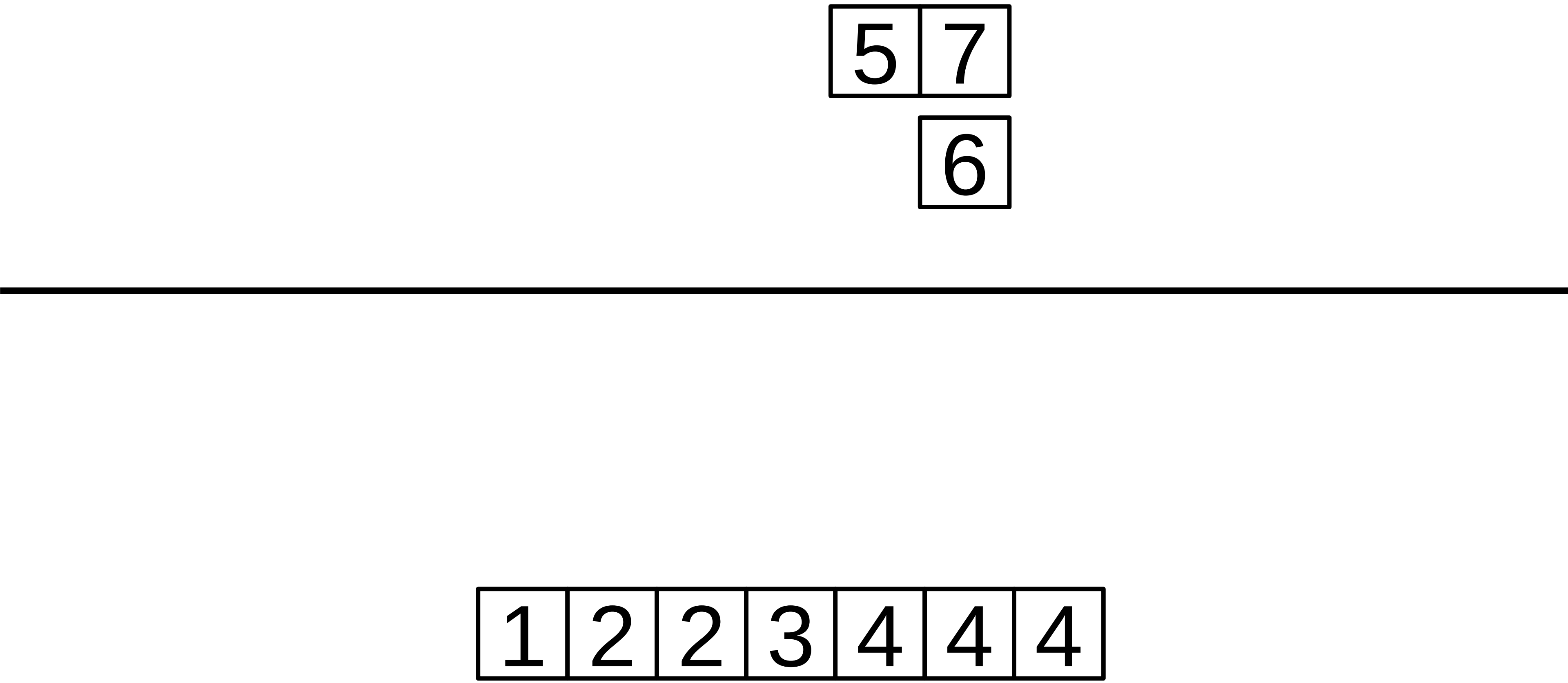
Algoritm merge

5	7

4	6

1	2	2	3	4	4

Algoritm merge



Algoritm merge

7
6

1	2	2	3	4	4	4	5

Algoritm merge

7

1	2	2	3	4	4	4	5	6

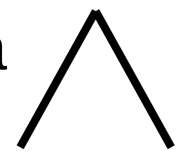
Algoritm merge

1	2	2	3	4	4	4	5	6	7

Algoritm merge

1	2	2	3	4	4	4	5	6	7

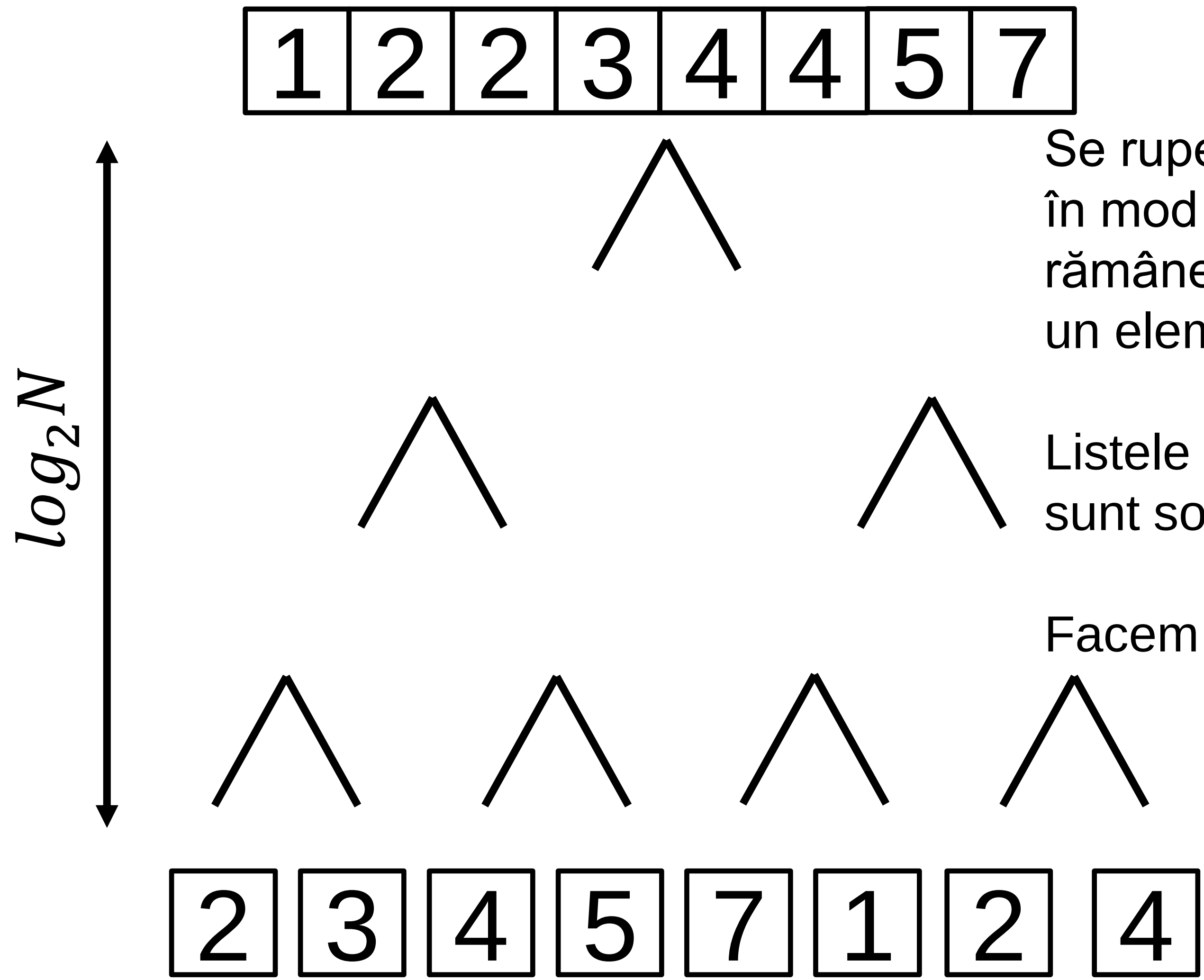
Folosim acest semn pentru a reprezenta operația **MERGE**



2	3	4	5	7

1	2	4	4	6

Merge sort

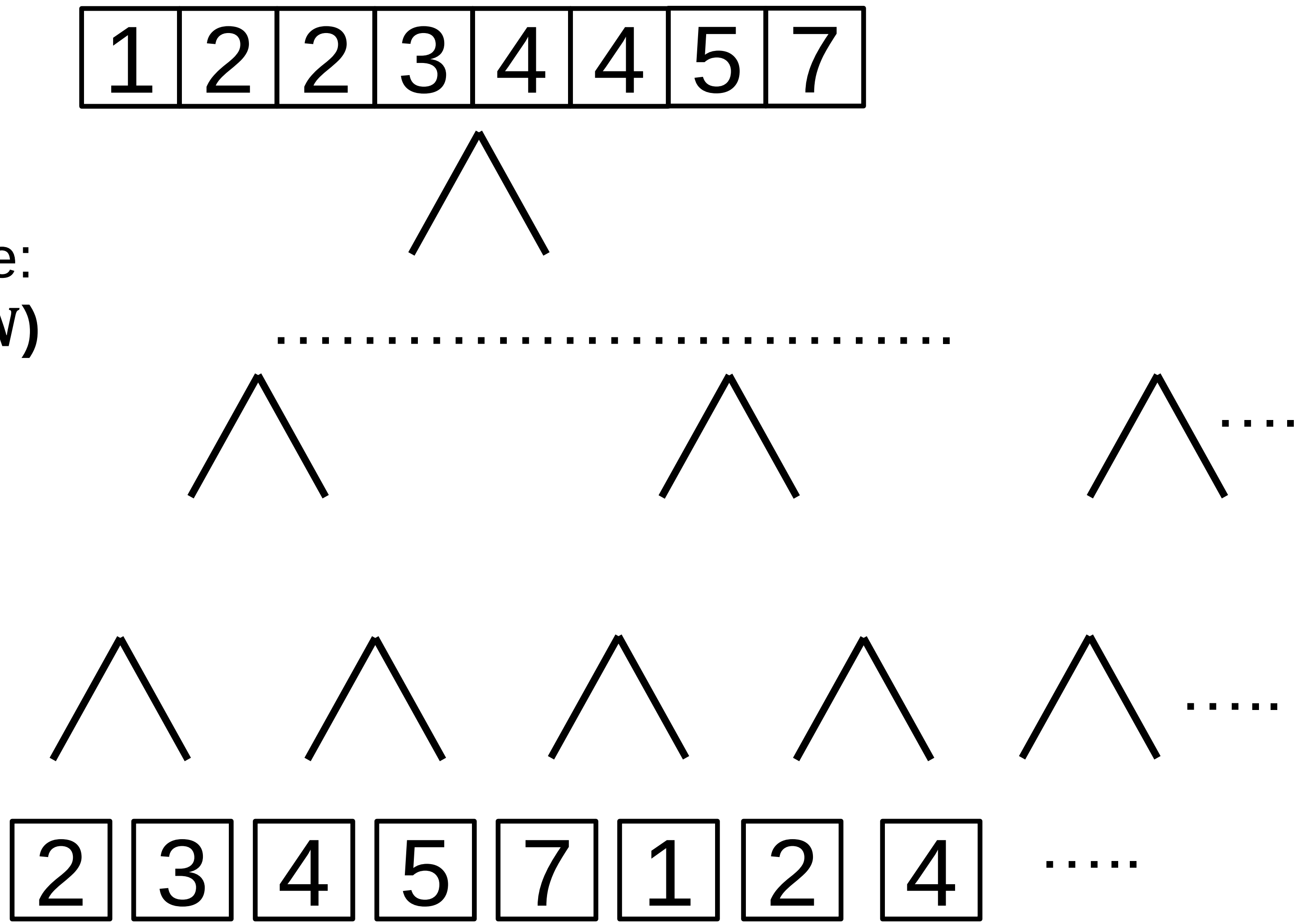


Se rupe lista în două în mod repetat până rămânem cu liste de un element

Listele de un element sunt sortate

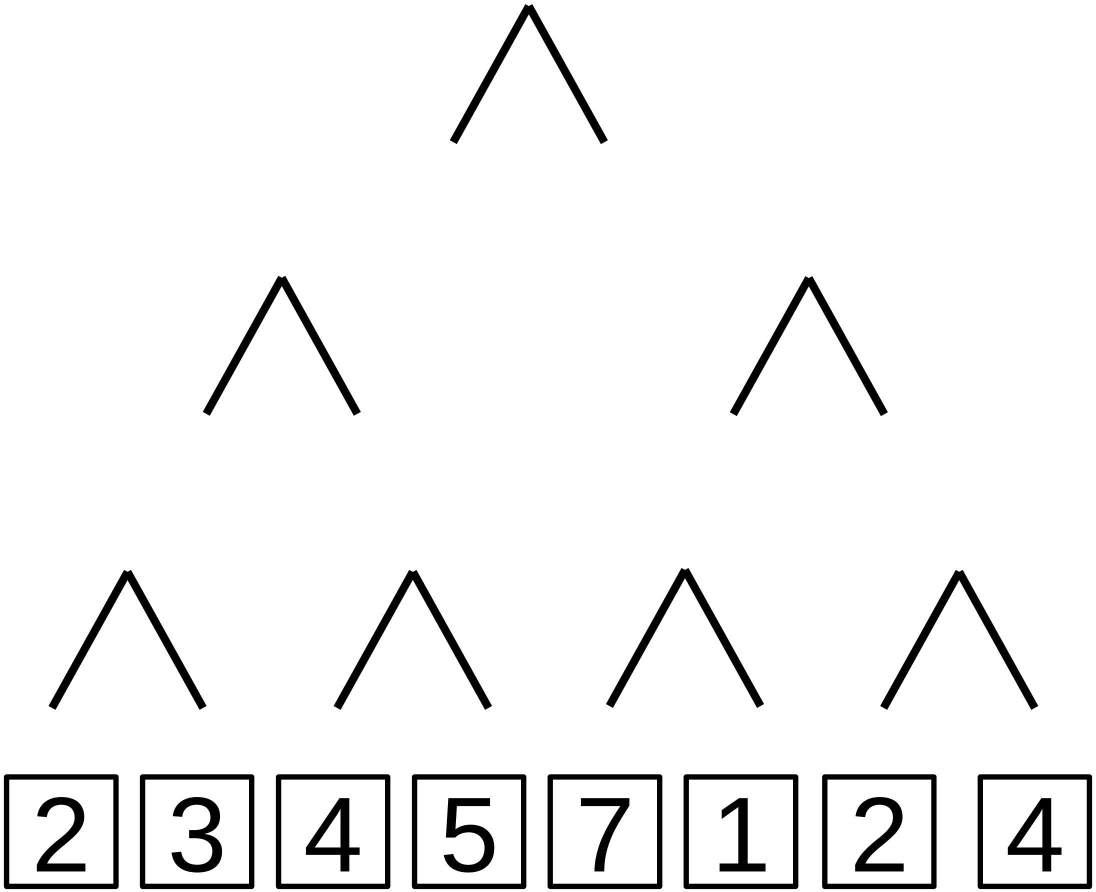
Facem **merge** listelor două câte două

Merge sort

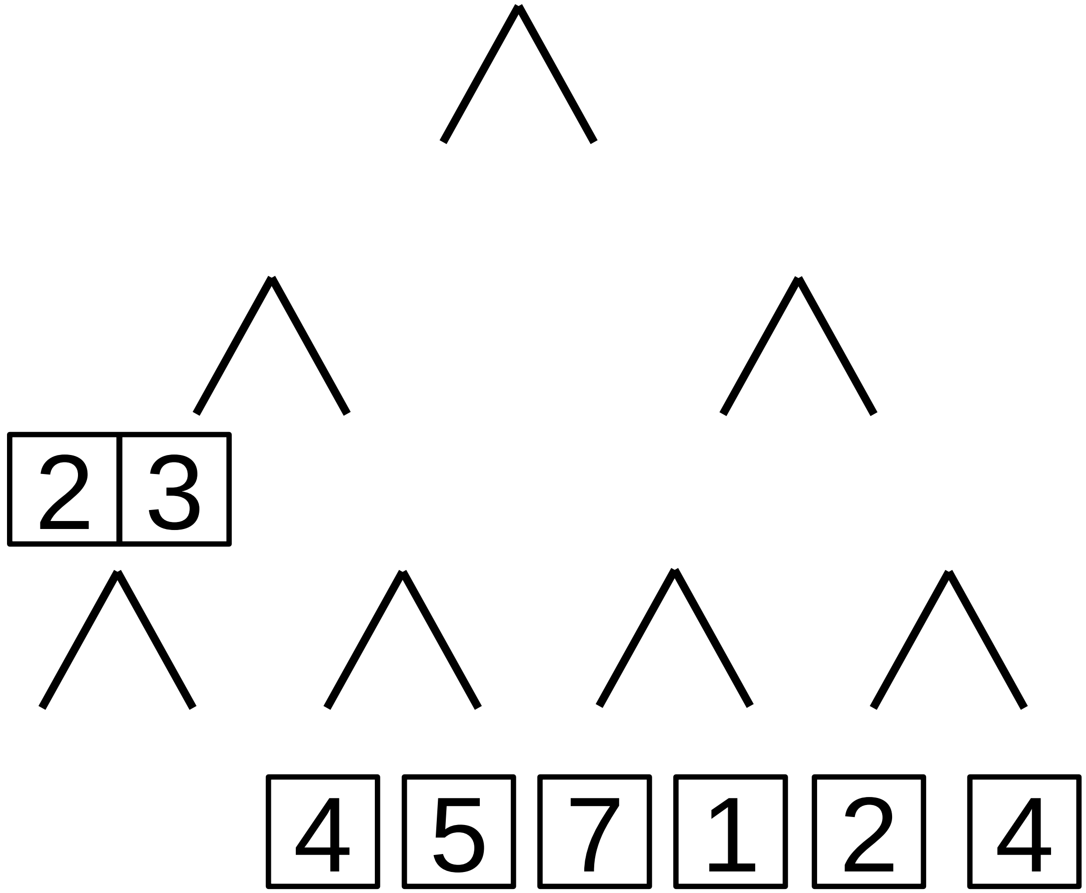


Complexitate:
$\mathbf{O(}\boldsymbol{N * log_2 N}\mathbf{)}$

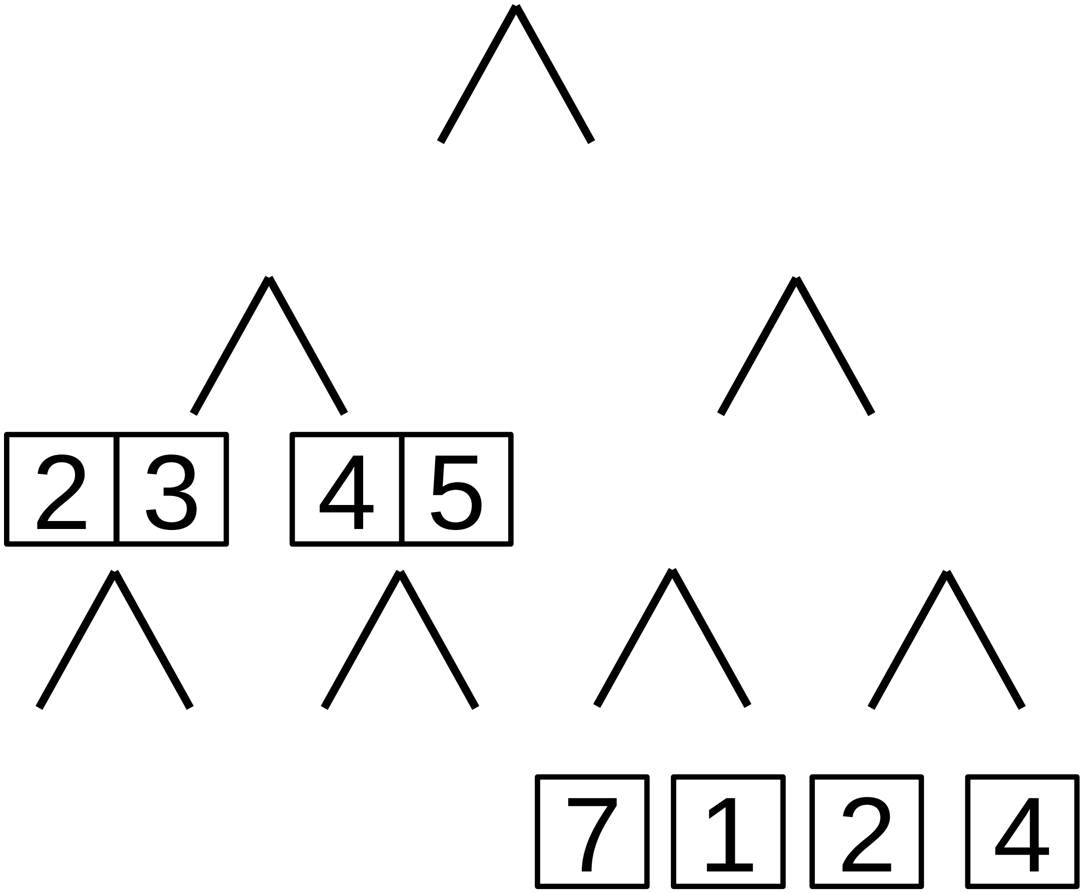
Merge sort



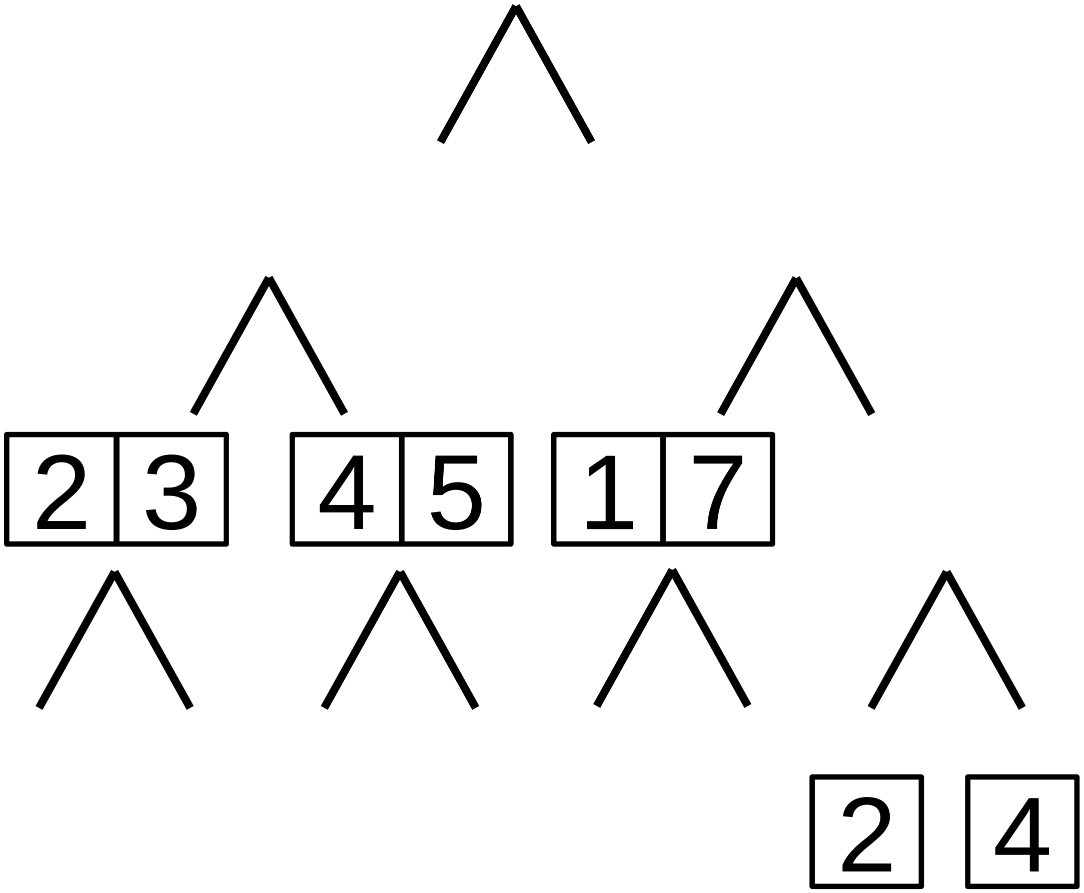
Merge sort



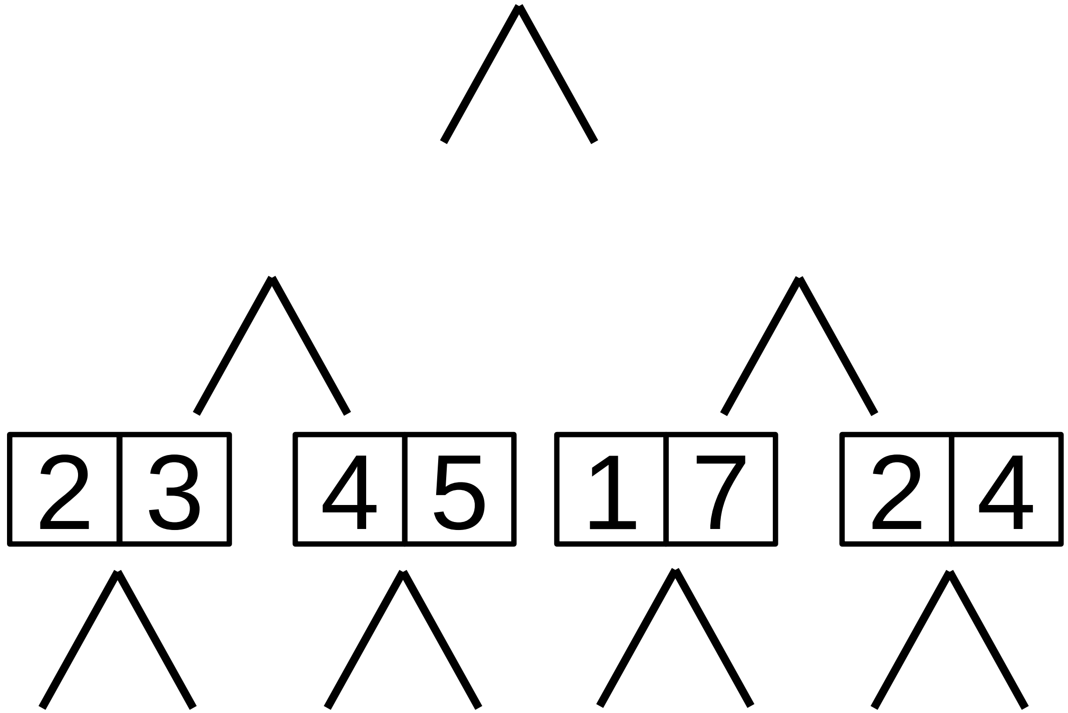
Merge sort



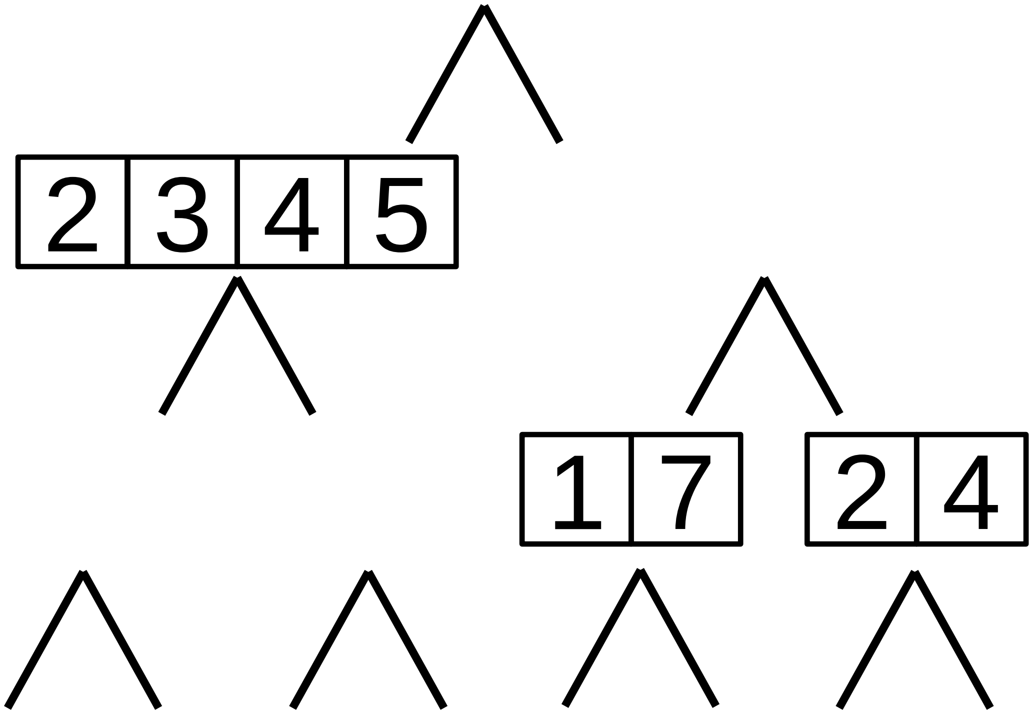
Merge sort



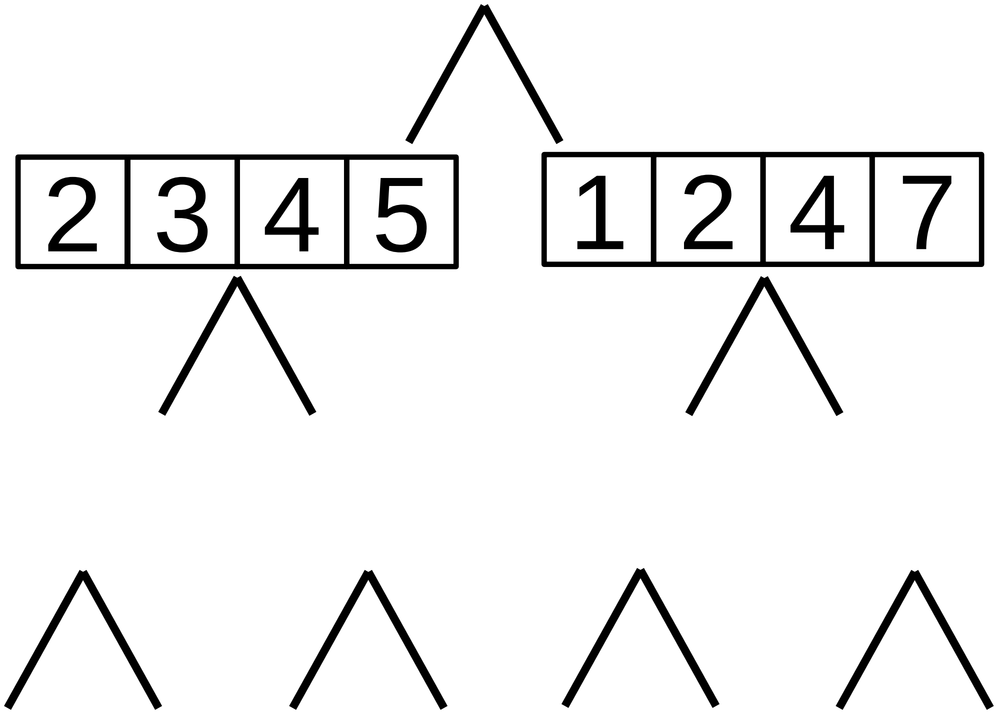
Merge sort



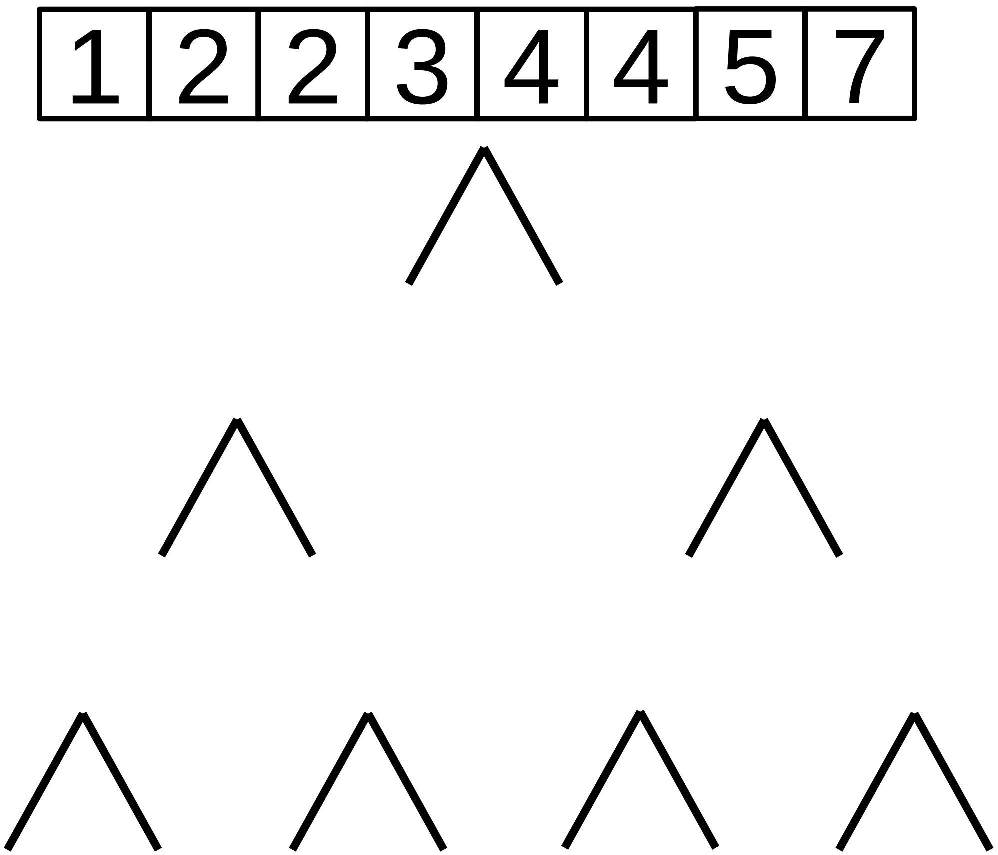
Merge sort



Merge sort



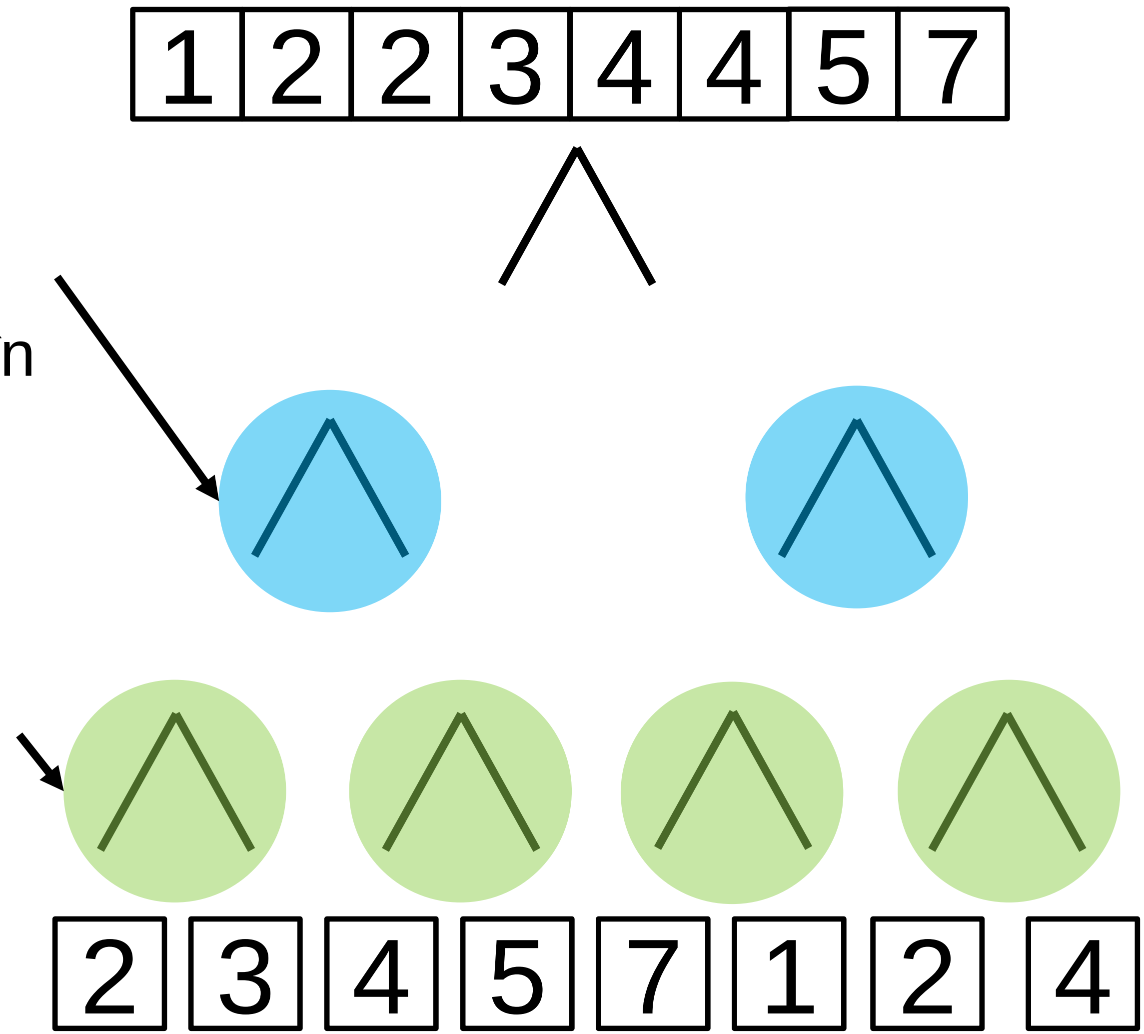
Merge sort



Merge sort paralel

Operațiile **albastre** pot fi executate în paralel

Operațiile **verzi** pot fi executate în paralel

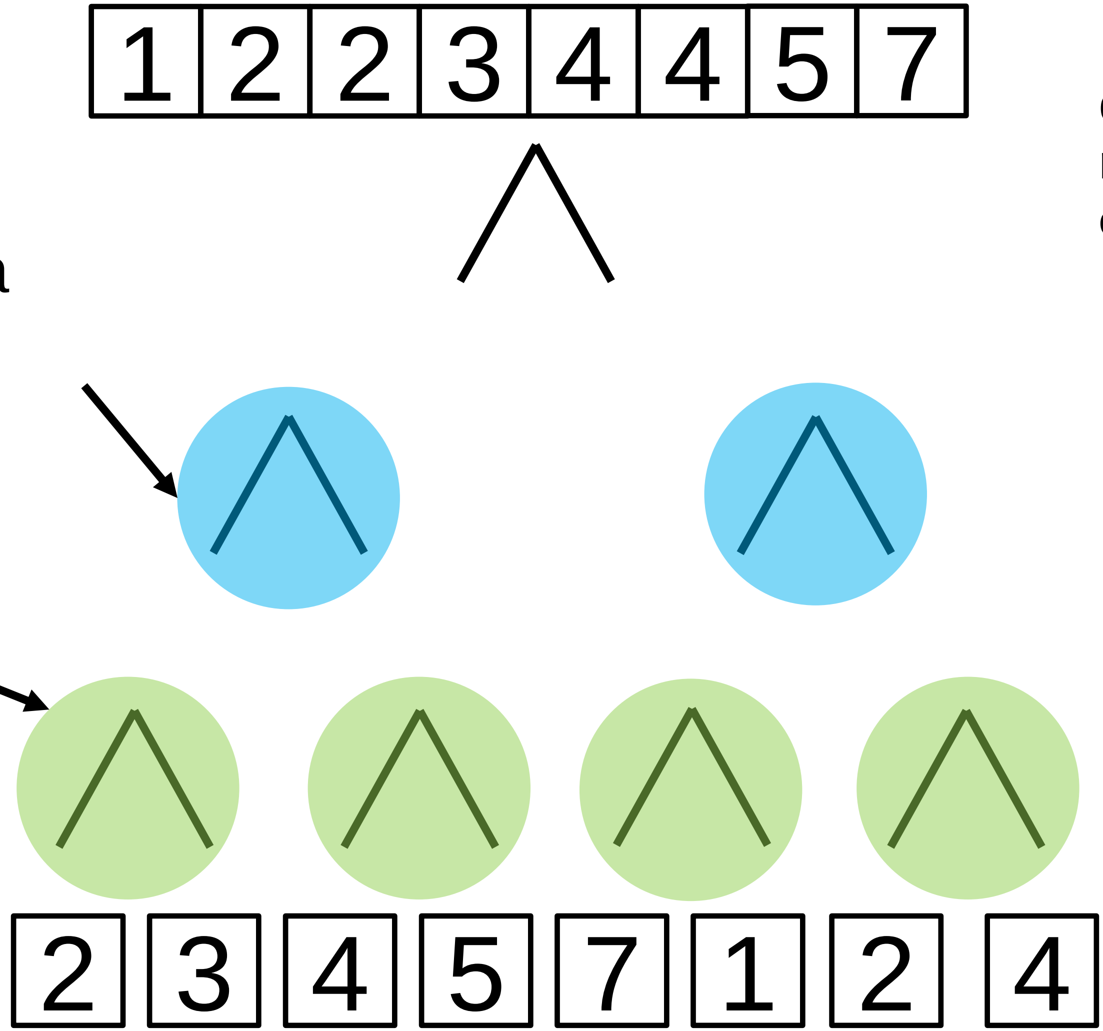


Merge sort paralel

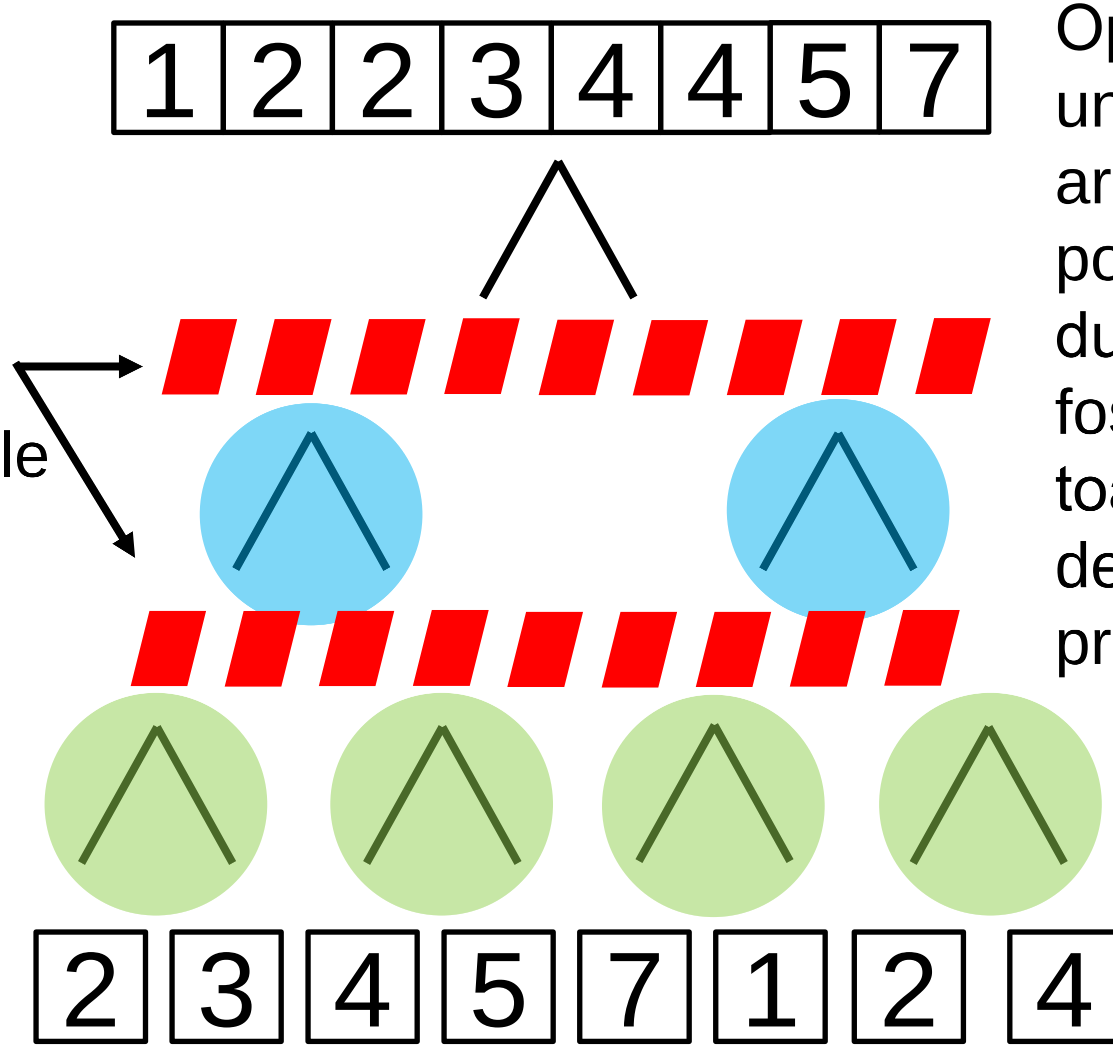
Operația

depinde de rezultatul operațiilor

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



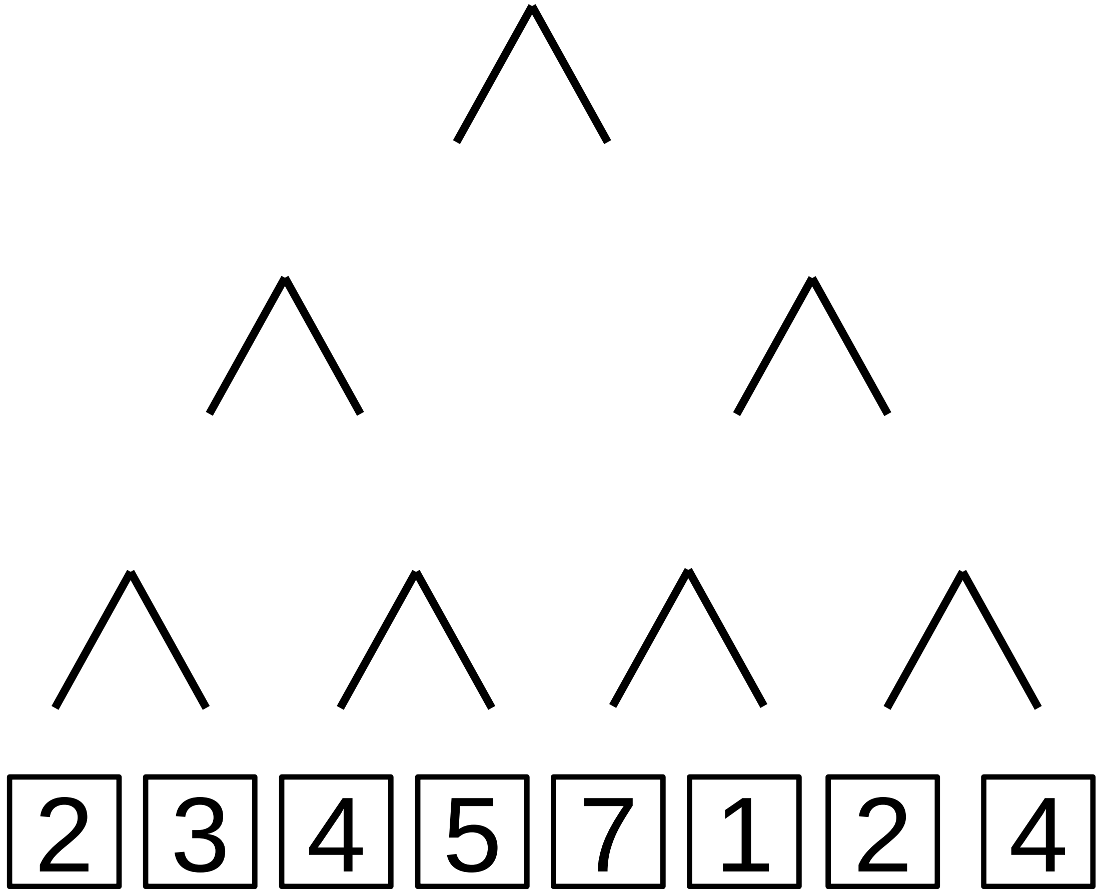
Merge sort paralel



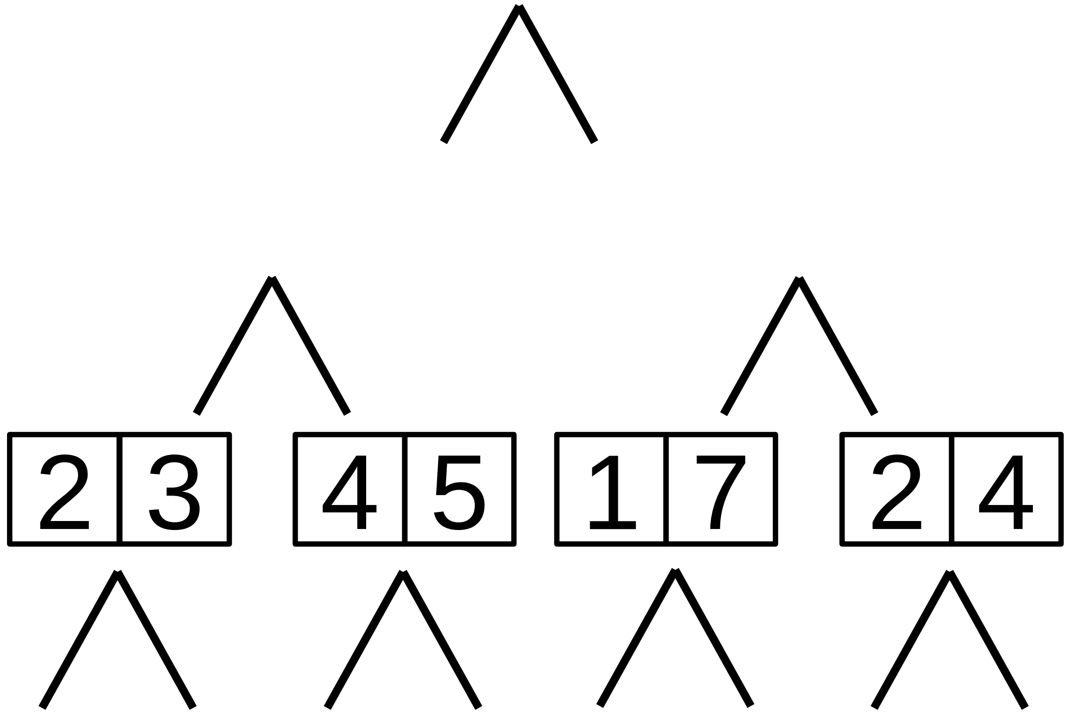
Soluție:
Barrier
Între nivelele arborelui

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

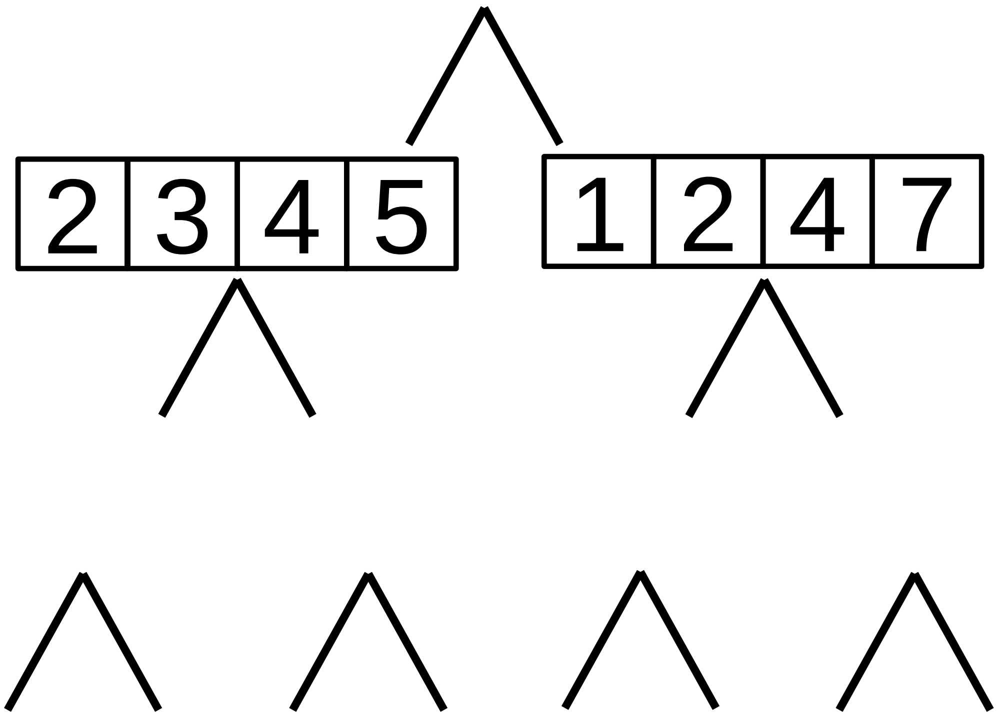
Merge sort paralel



Merge sort paralel

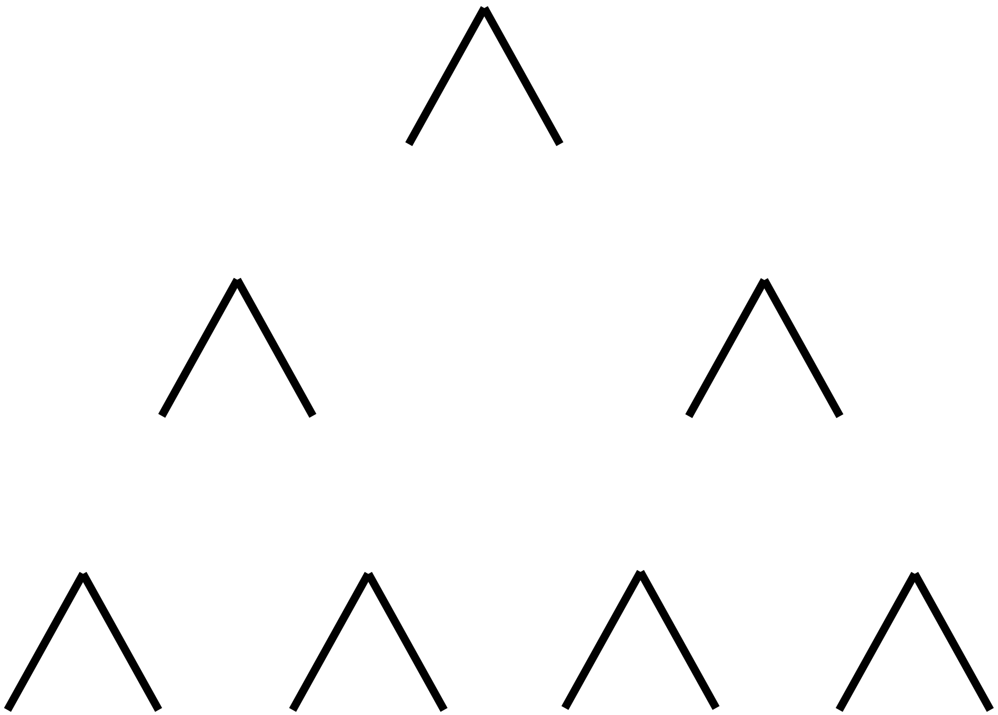


Merge sort paralel



Merge sort paralel

1	2	2	3	4	4	5	7

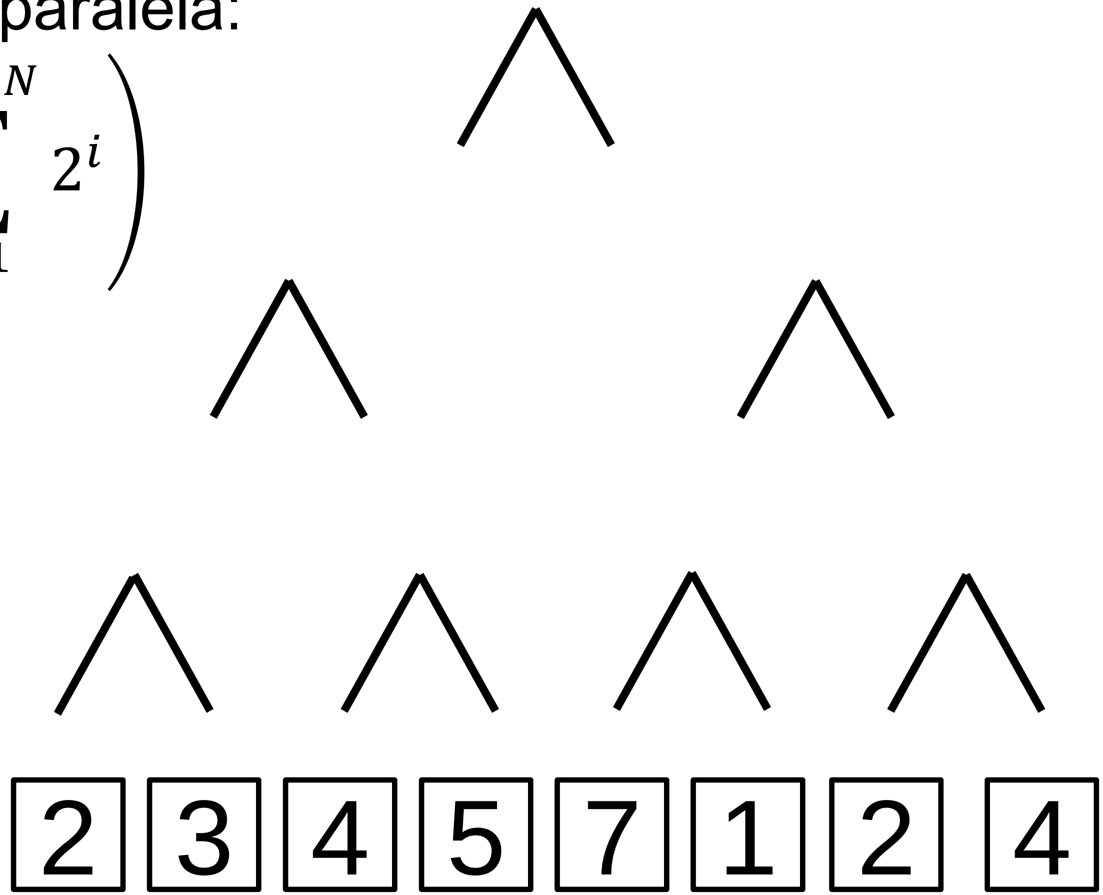


Merge sort paralel - complexitate

Complexitate paralelă:

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

Dacă P = N



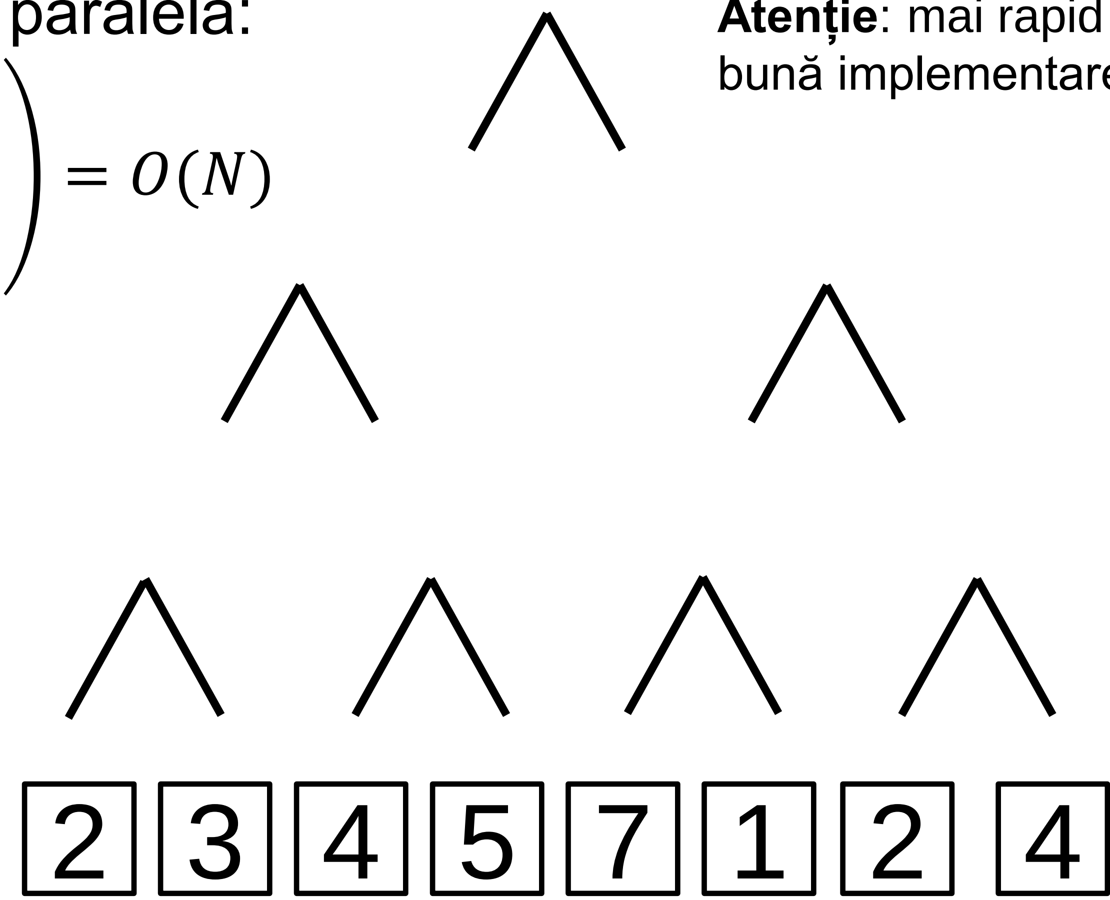
Merge sort paralel - complexitate

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 $\boxed{3}$

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 $\boxed{3}$

Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$

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 $\boxed{3}$

Între pozițiile $\boxed{3}\boxed{3}$

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 $\boxed{3}$

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 $\boxed{3}$ Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$ Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$

Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$

Între pozițiile $\boxed{1}\boxed{4}$

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 $\boxed{3}$ Între pozițiile $\boxed{1}\boxed{4}$

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 $\boxed{3}$ Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$ Între pozițiile $\boxed{0}\boxed{15}$

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 $\boxed{3}$

Între pozițiile $\boxed{1}\boxed{4}$

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

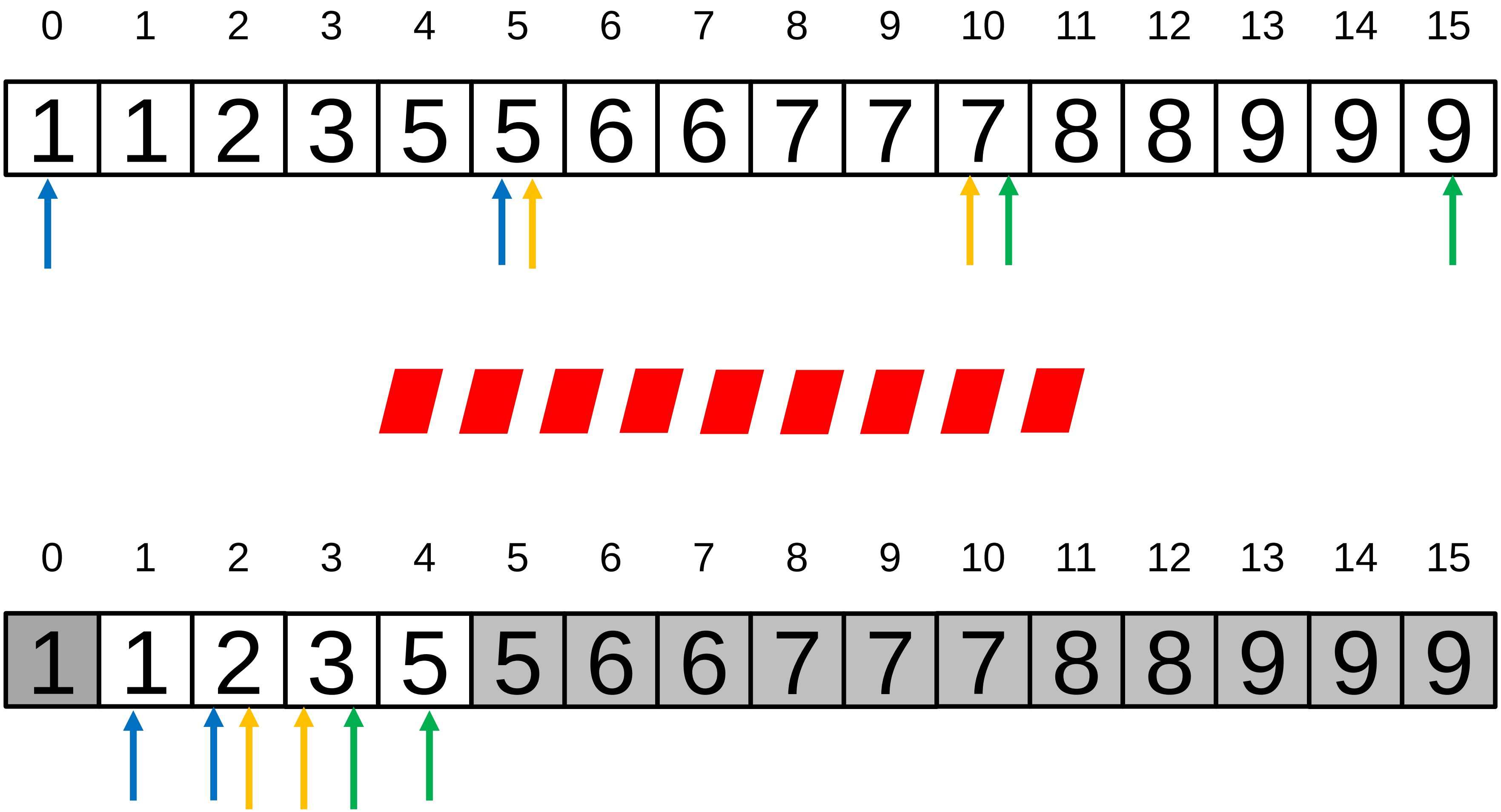
Căutare paralelă

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 $\boxed{3}$

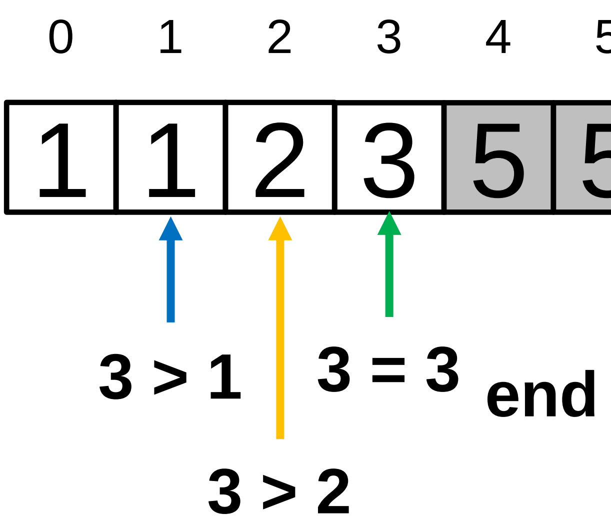
Între pozițiile $\boxed{0}\boxed{15}$

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**

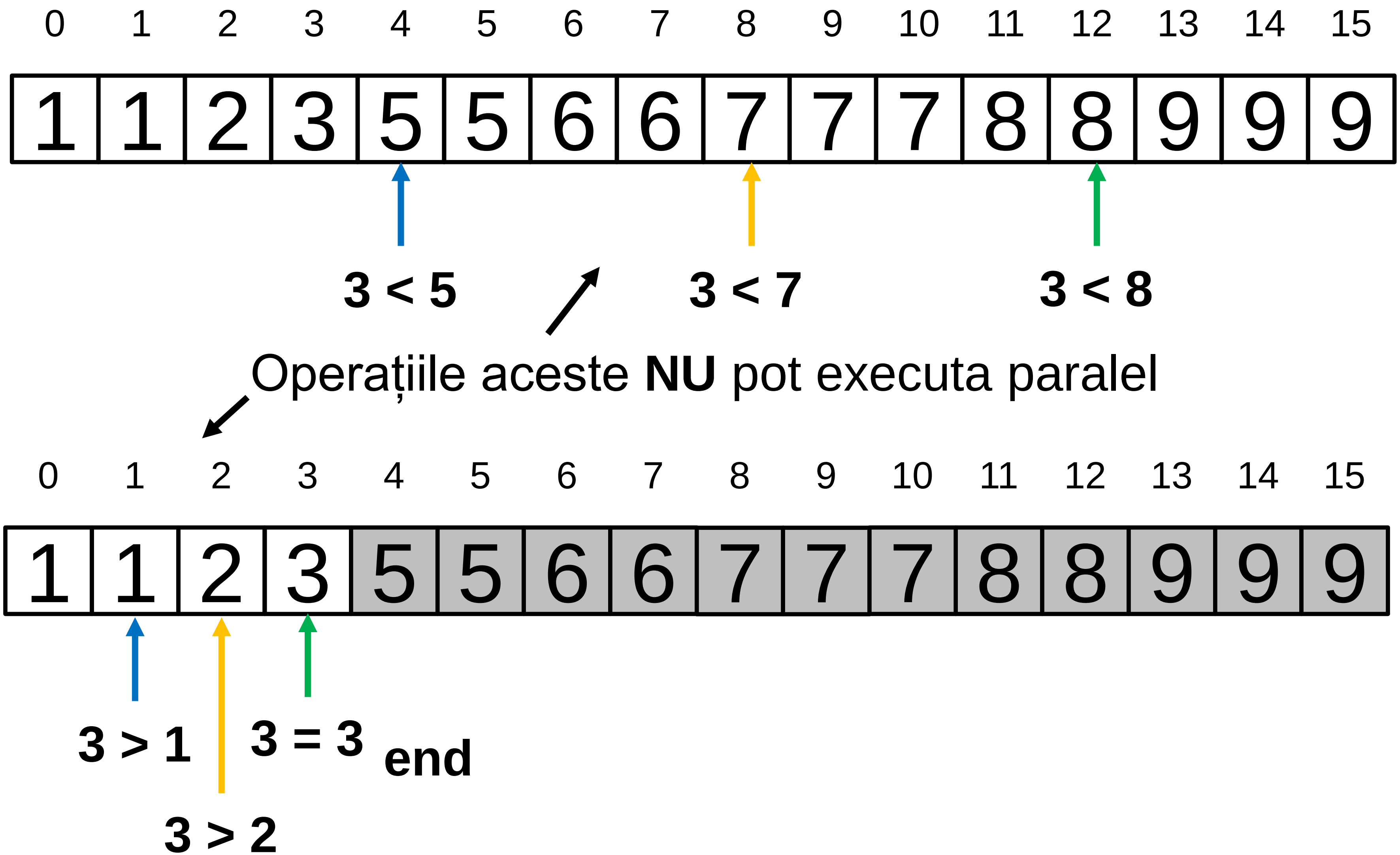
Căutare paralelă – soluția 2

Căutăm $\boxed{3}$ Între pozițiile $\boxed{0}\boxed{3}$



Complexitate: O($log_p(n)$)

Căutare paralelă – soluția 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

$$\boldsymbol{O(log_p(N))}$$

Speedup?

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

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 căutare binară (sau chiar paralelă) și pe rank sort.

Pipeline

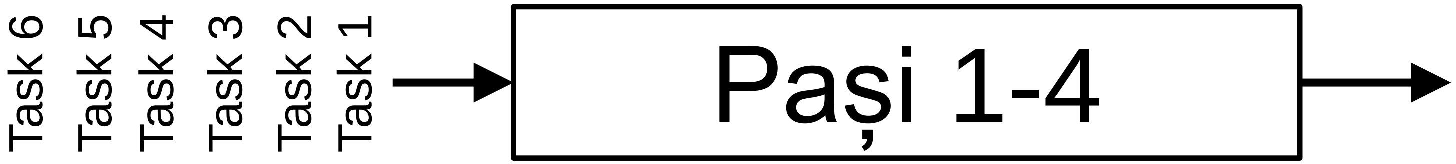
- Pipeline de instrucțiuni CPU
- Pipeline grafic (randare, antialiasting)
- Diferiți algoritmi

→ Pas1 → Pas2 → Pas3 → Pas4 →

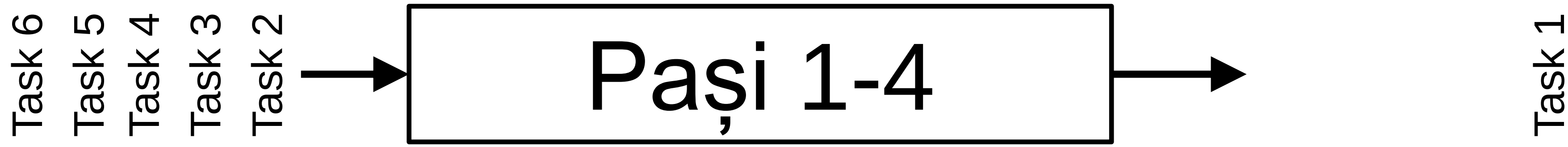
Un **pas** poate fi un:

- thread
- proces
- element hardware

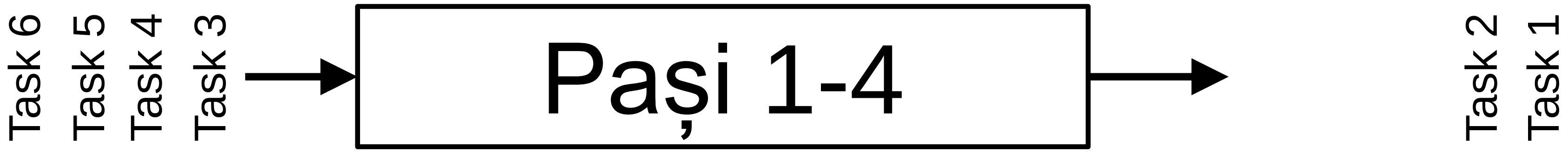
Fără Pipeline



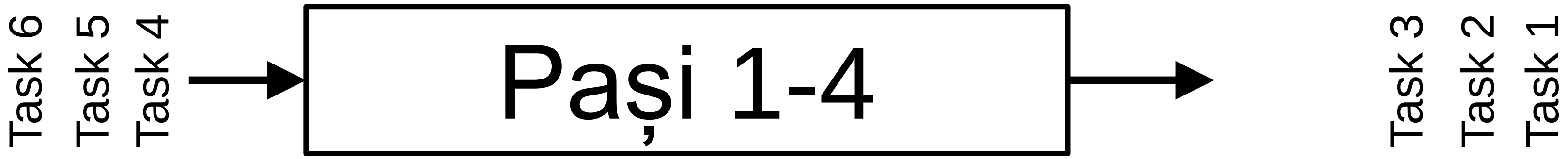
Fără Pipeline



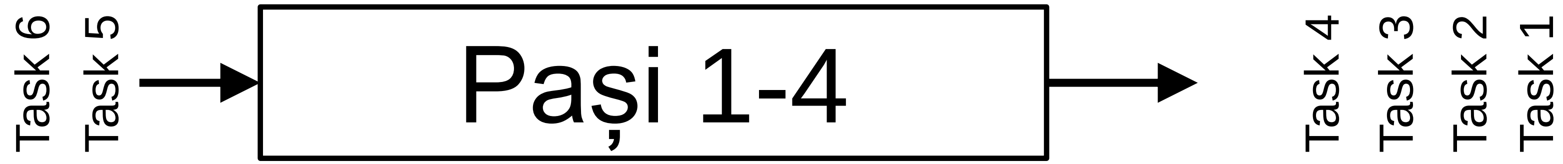
Fără Pipeline



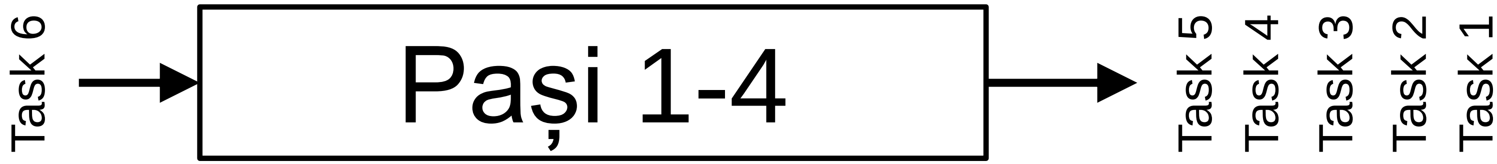
Fără Pipeline



Fără Pipeline

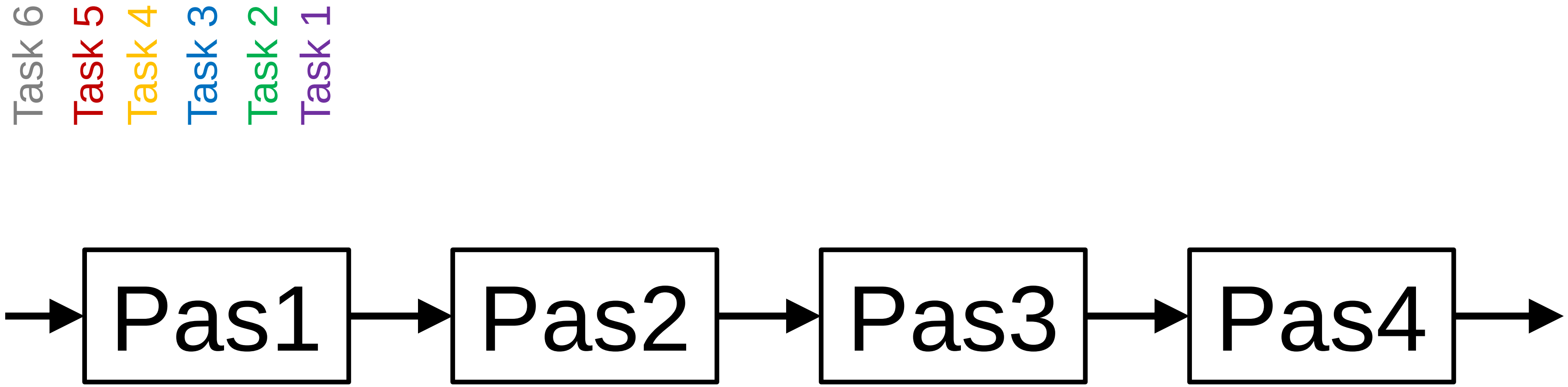


Fără Pipeline

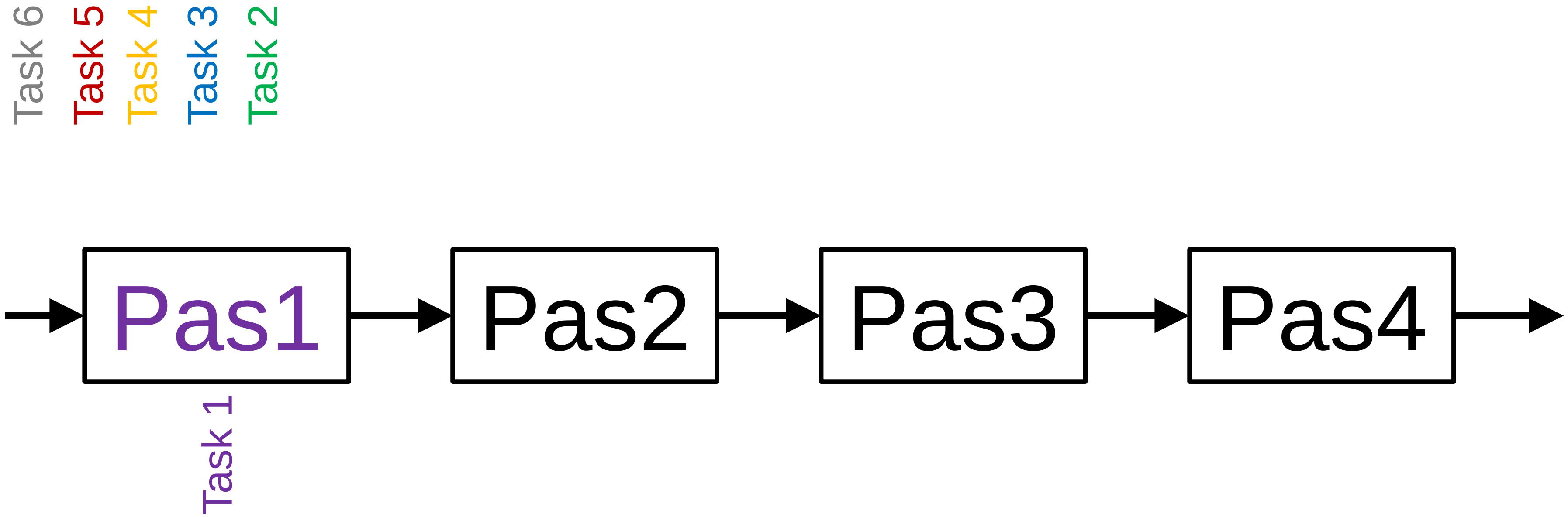


$$\boldsymbol{total\_execution\_time = task\_execution\_time * number\_of\_tasks}$$

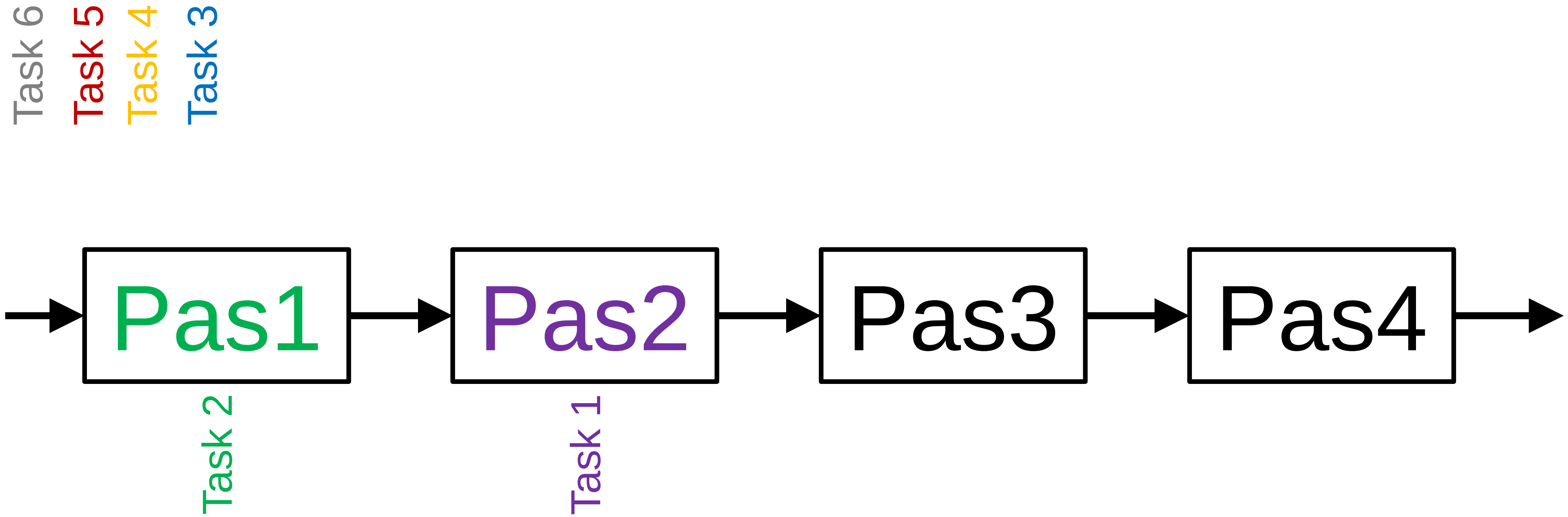
Pipeline



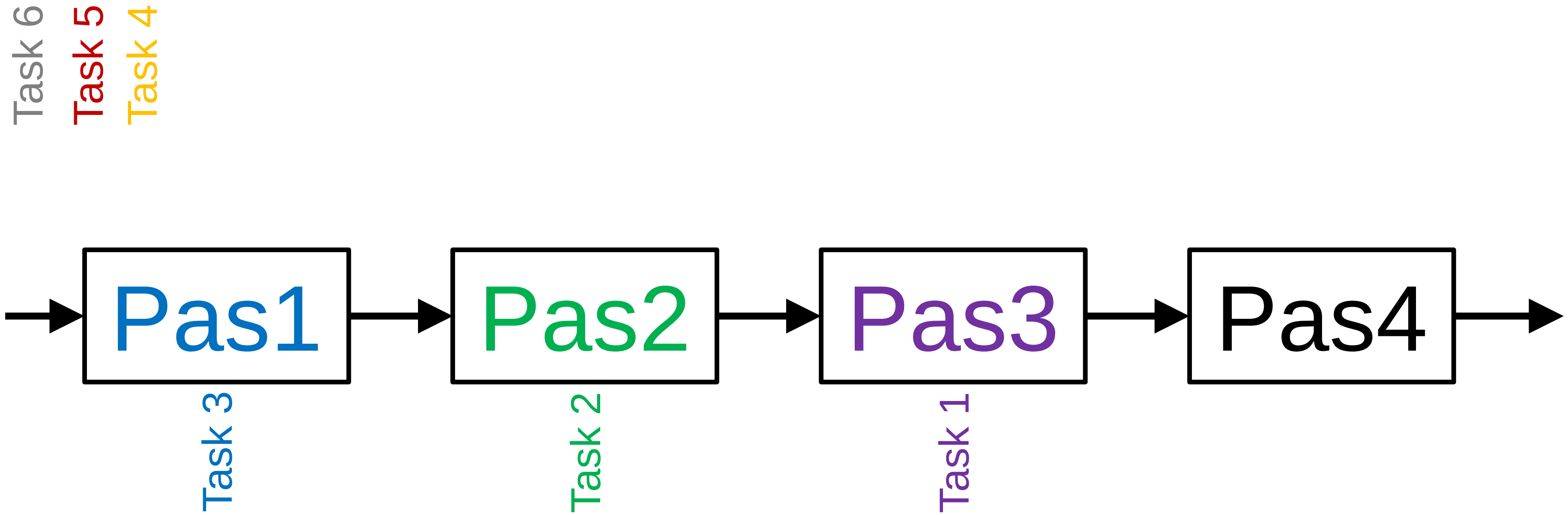
Pipeline



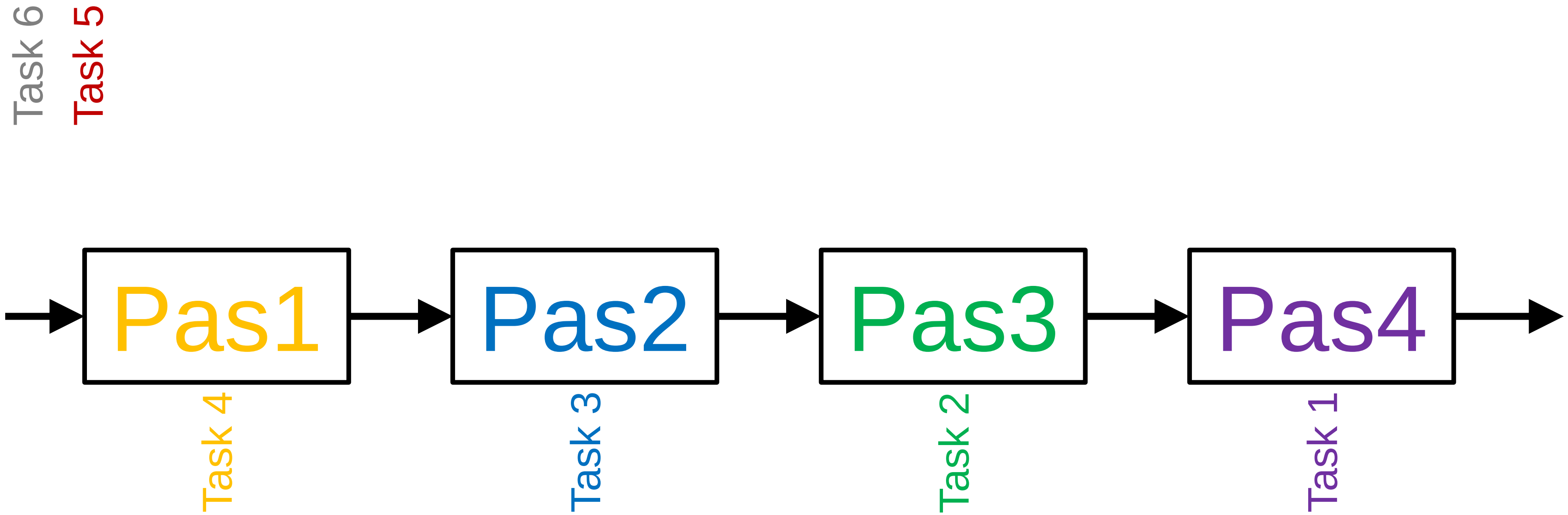
Pipeline



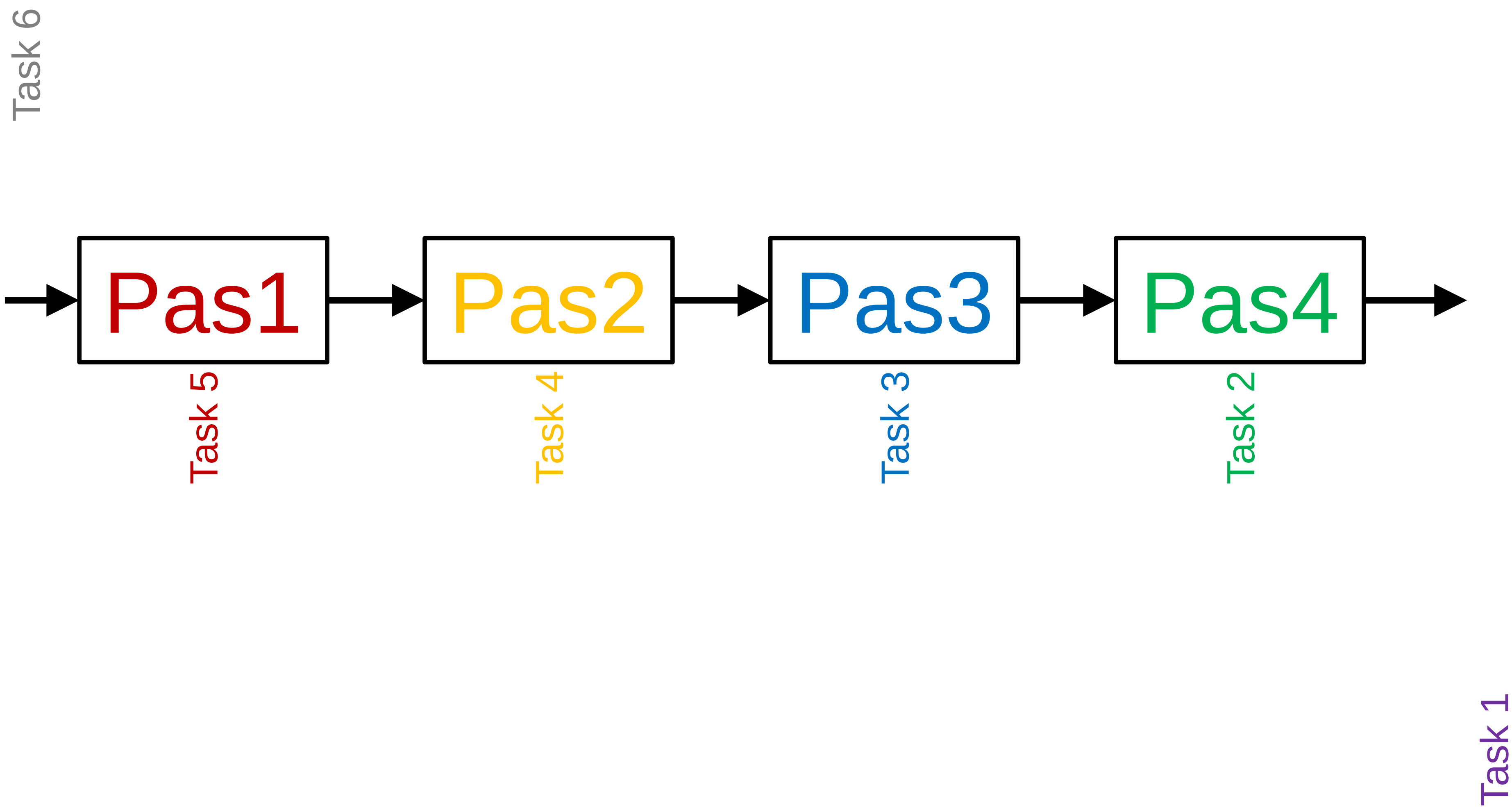
Pipeline



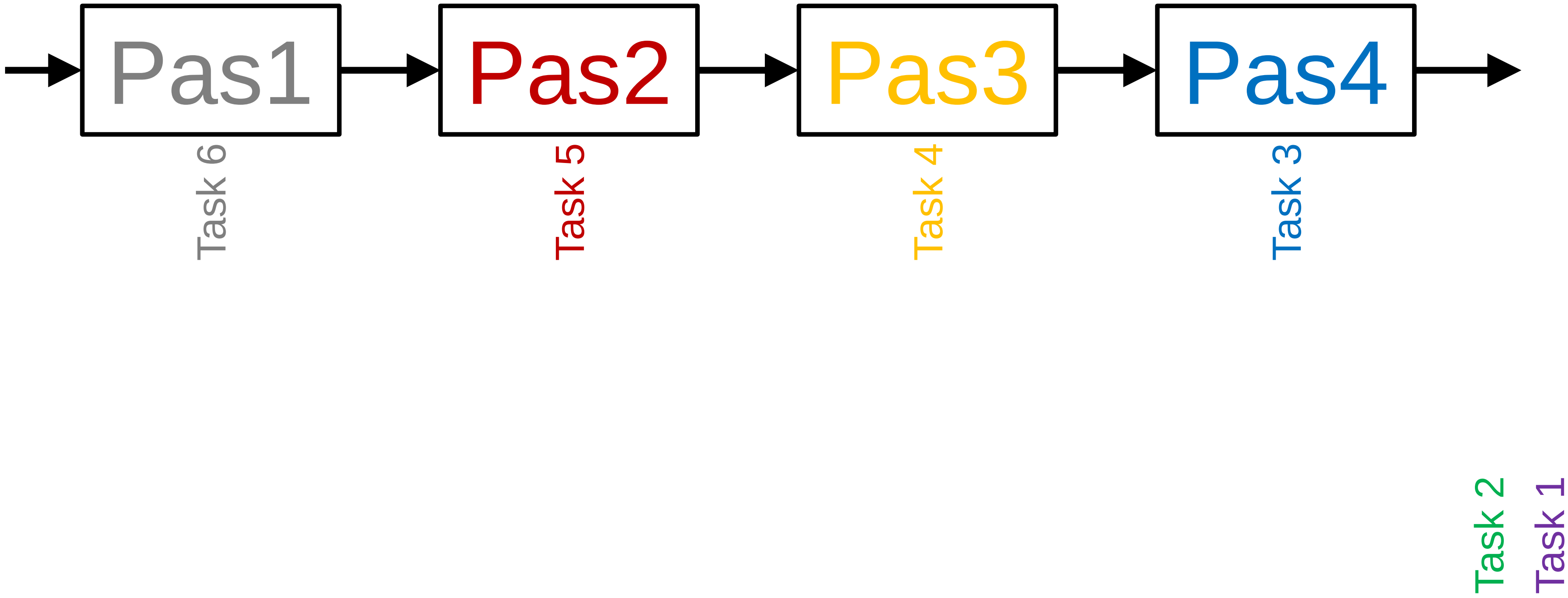
Pipeline



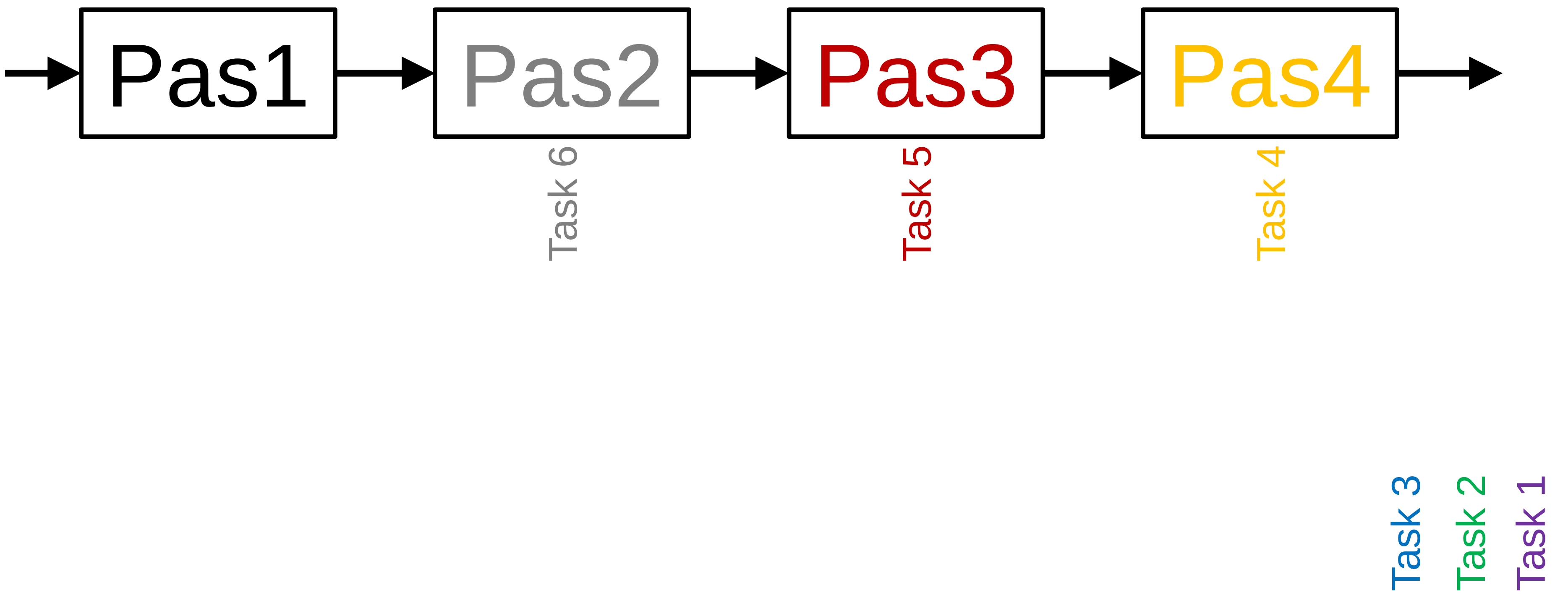
Pipeline



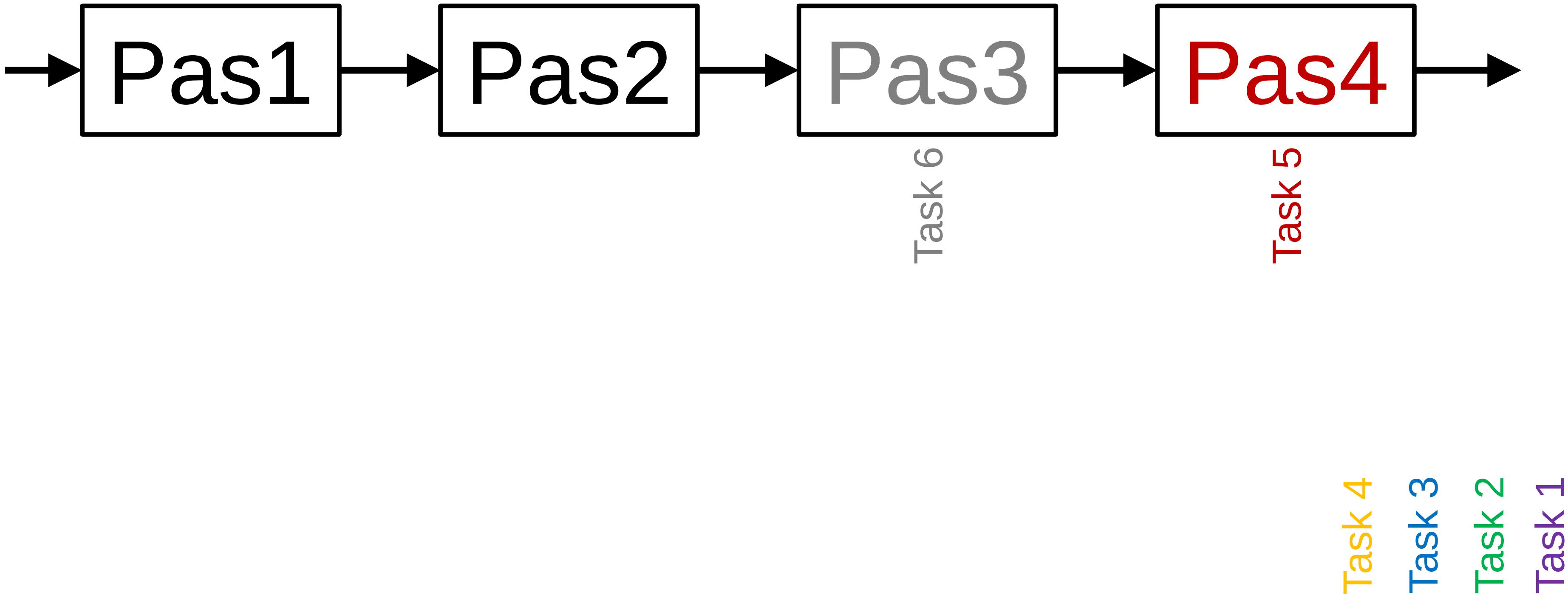
Pipeline



Pipeline

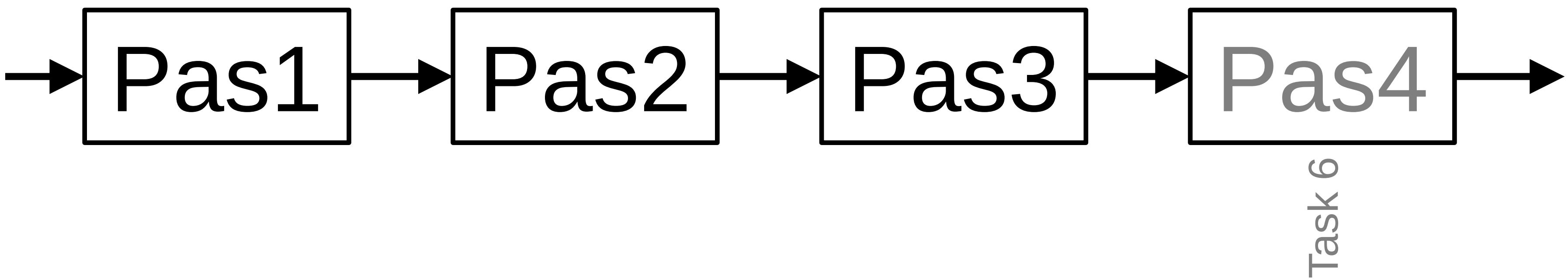


Pipeline



Pipeline

Ideal: $$step\_execution\_time = \frac{task\_execution\_time}{number\_of\_steps}$$

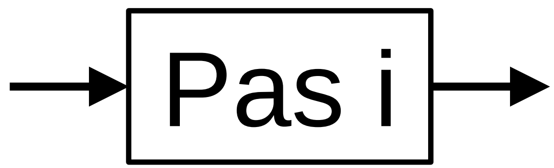


După avem mai mult decât $number\_of\_steps$ tasks timpul devine:

$$total\_execution\_time = number\_of\_tasks * step\_execution\_time$$

Un task se termină la fiecare “step tick”

Pipeline – ghid programare




```
Inițializare
for(un număr de pași) {
        primește date de la Pas(i-1)
        procesează
        trimite date la Pas(i+1)
}
Finalizare
```

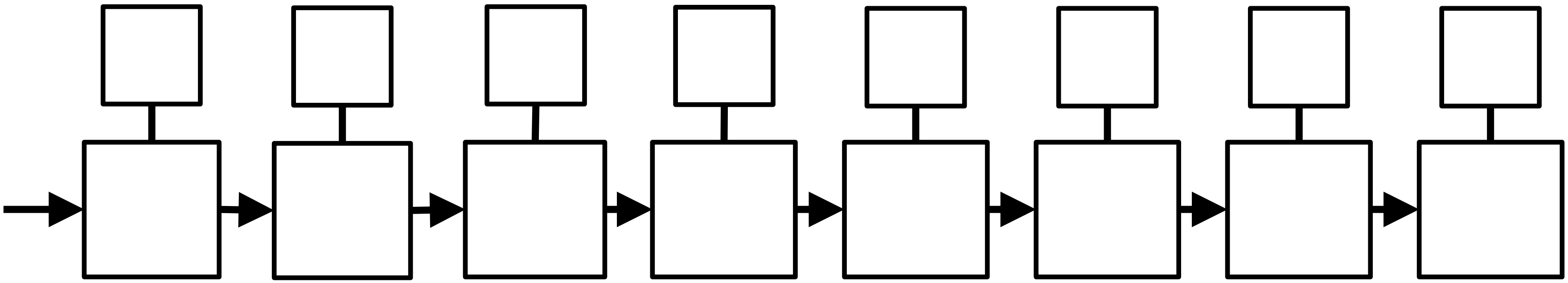
Sortare cu pipeline

9	4	2	7	6	5	6	1

1	2	4	5	6	6	7	9

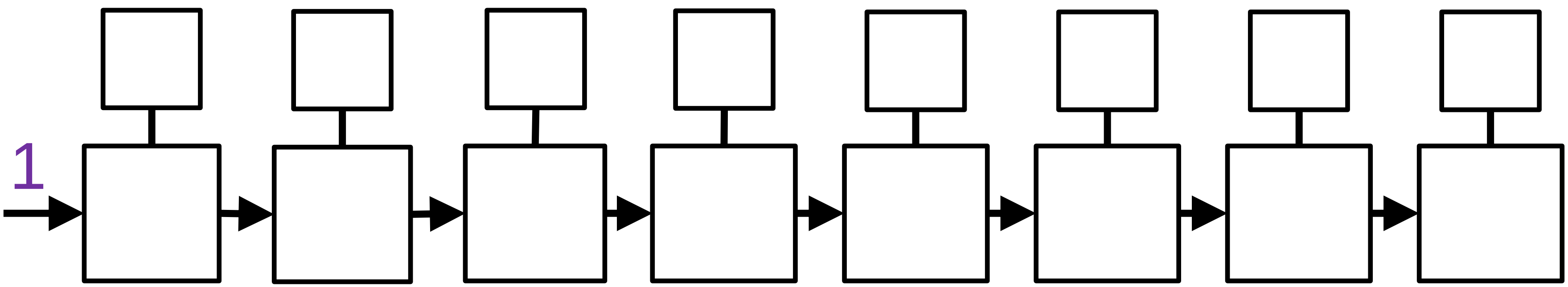
Sortare cu pipeline

9	4	2	7	6	5	6	1

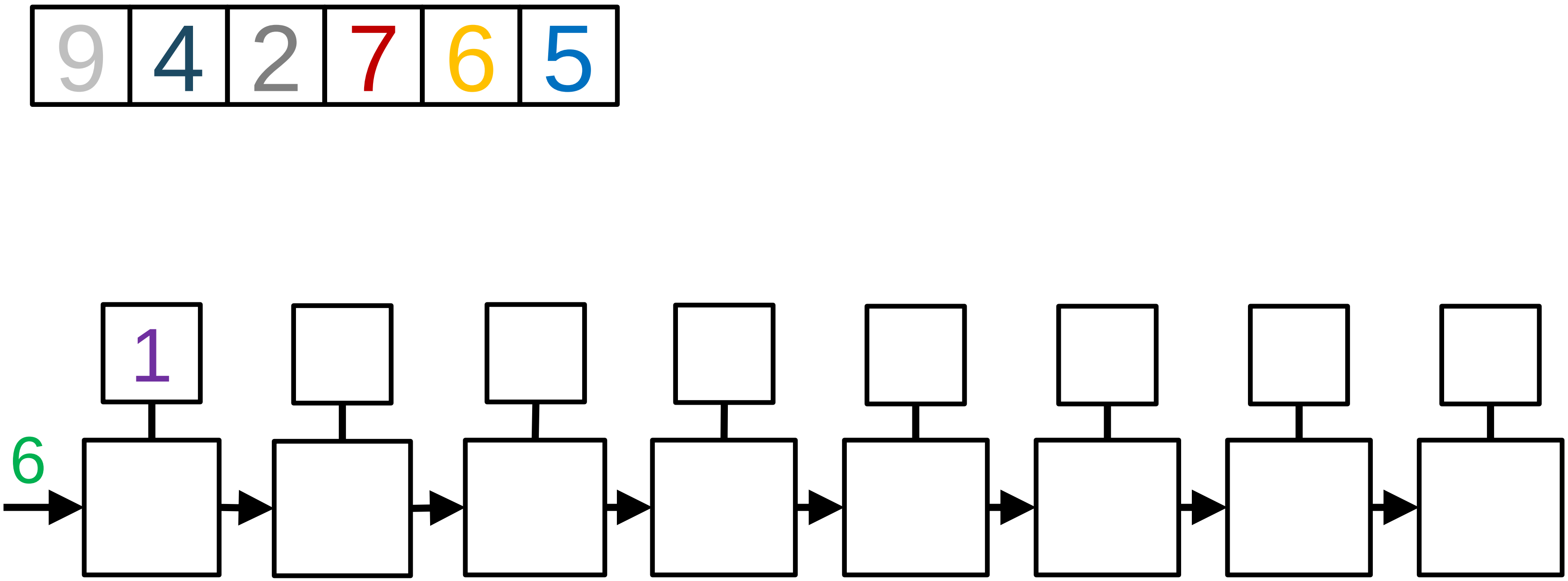


Sortare cu pipeline

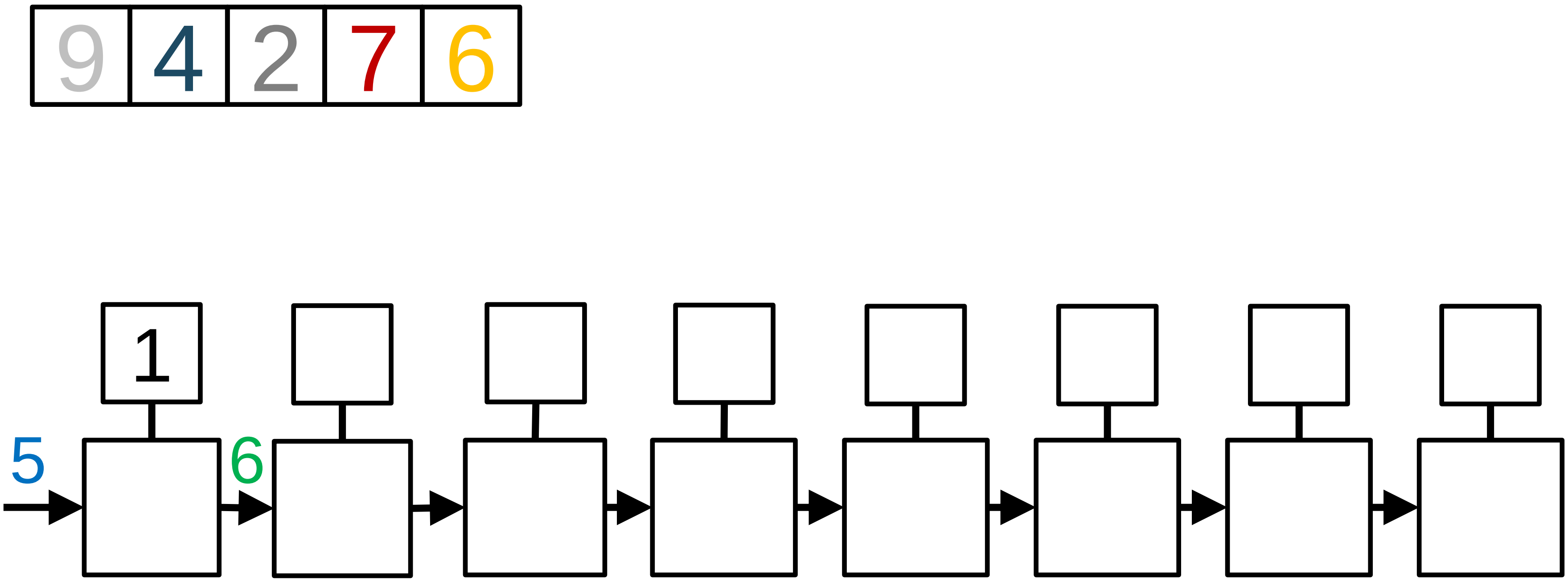
9	4	2	7	6	5	6



Sorting with pipeline

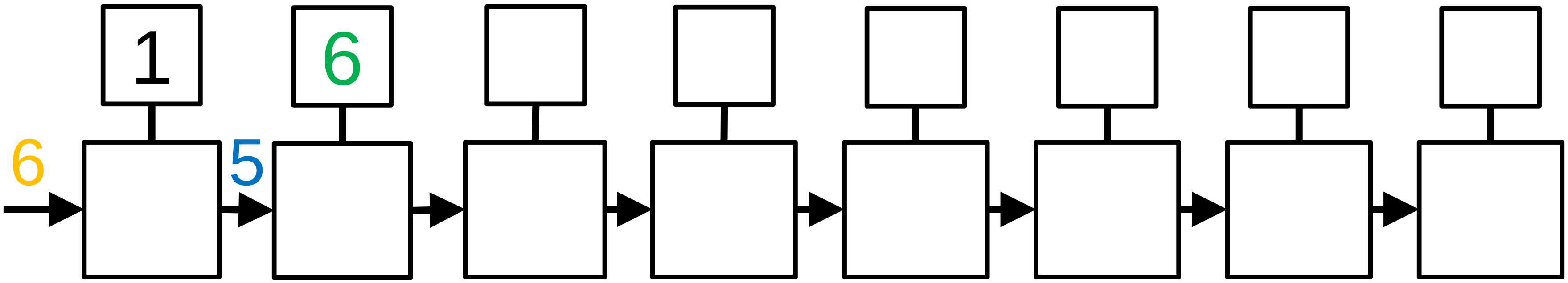


Sortare cu pipeline

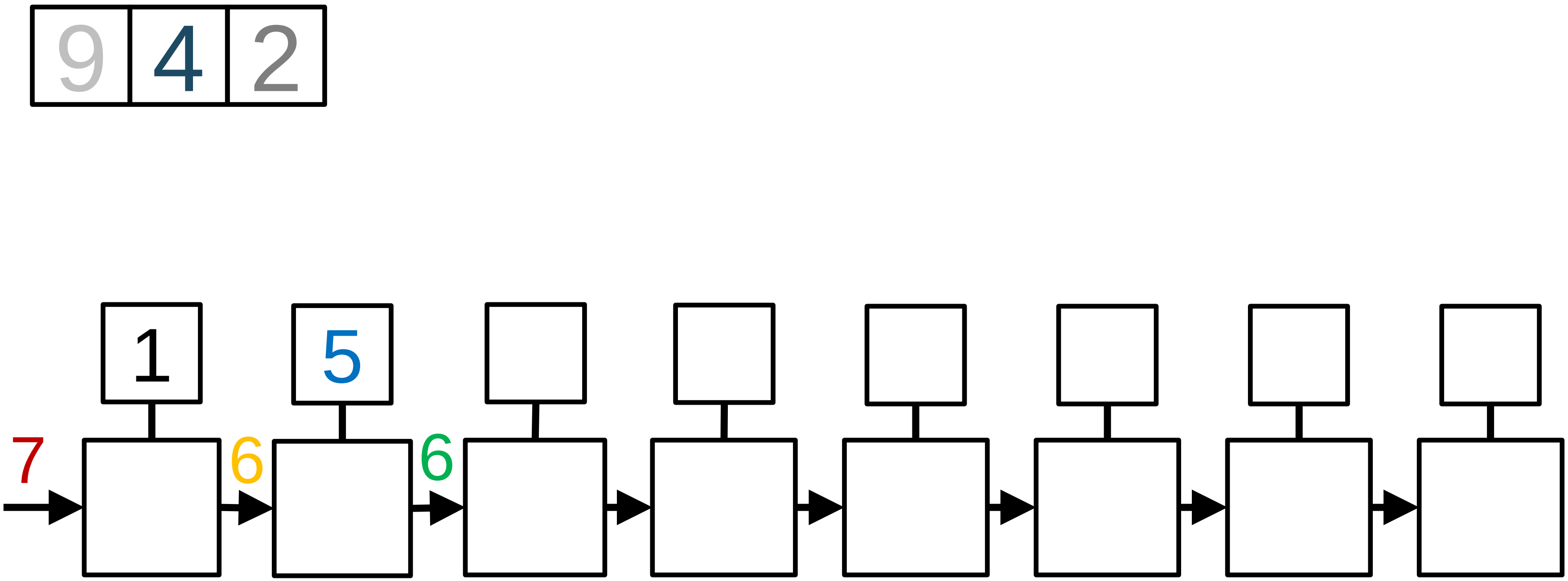


Sortare cu pipeline

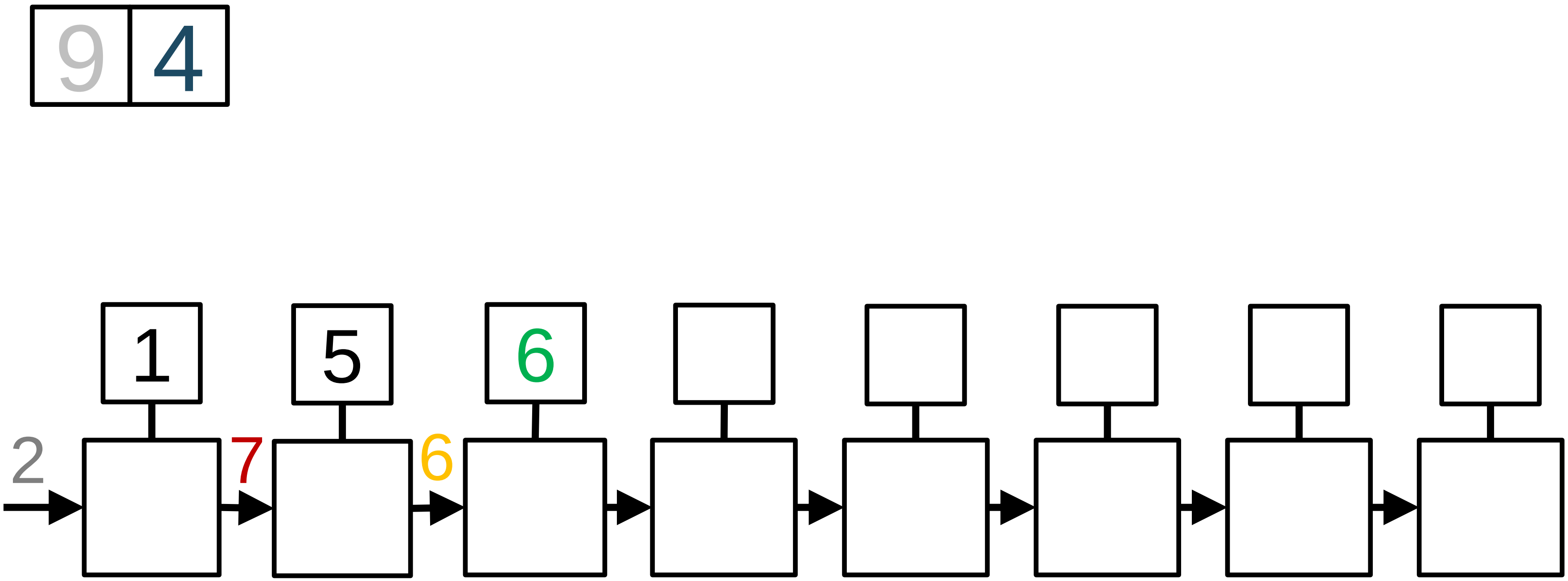
9	4	2	7



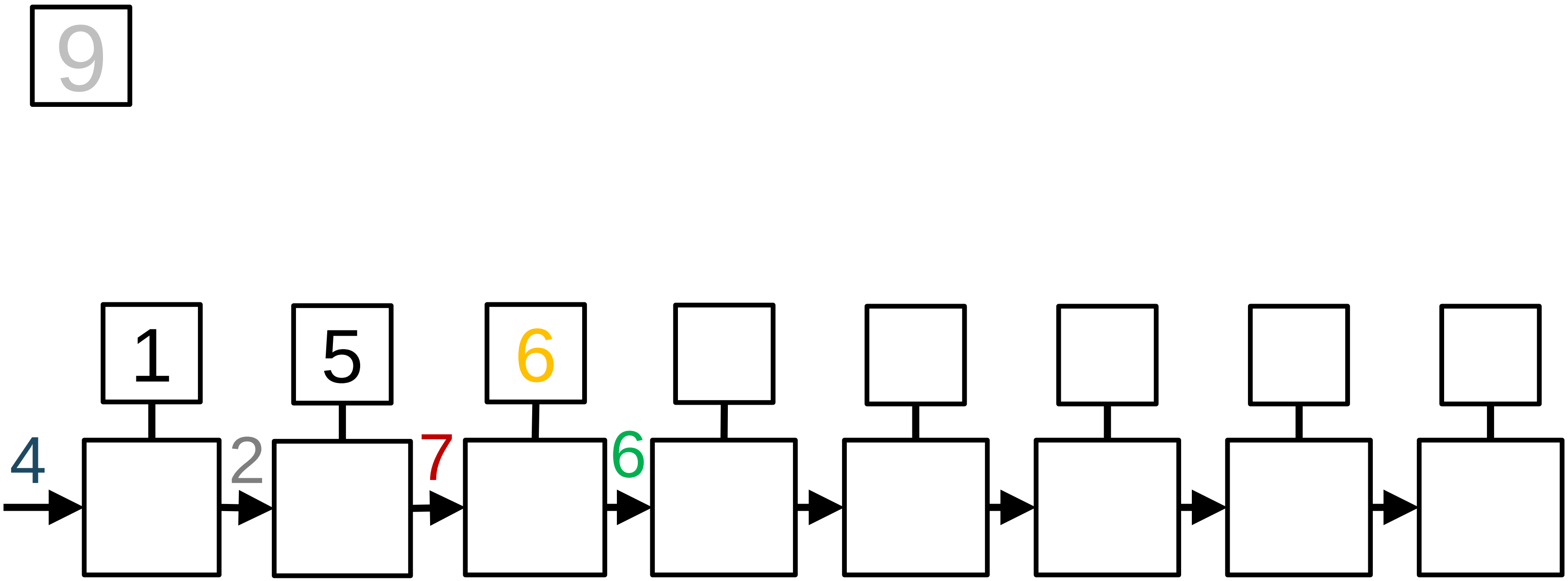
Sortare cu pipeline



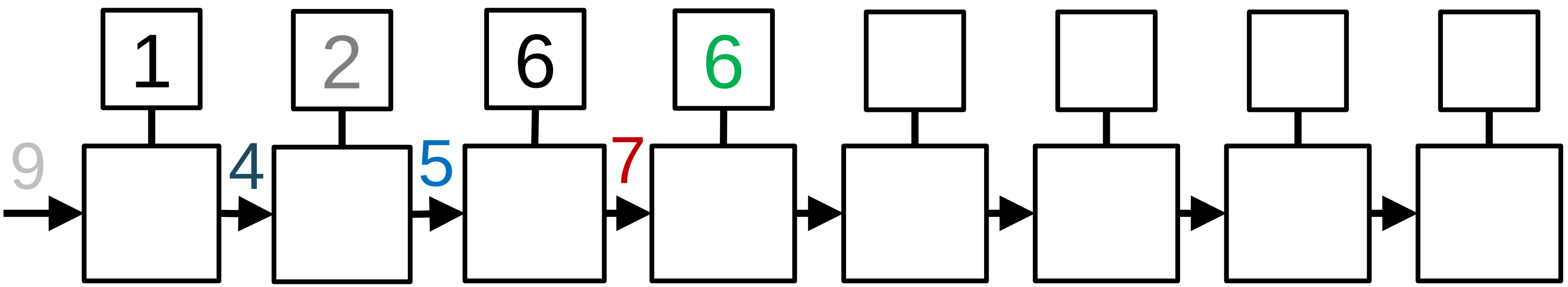
Sortare cu pipeline



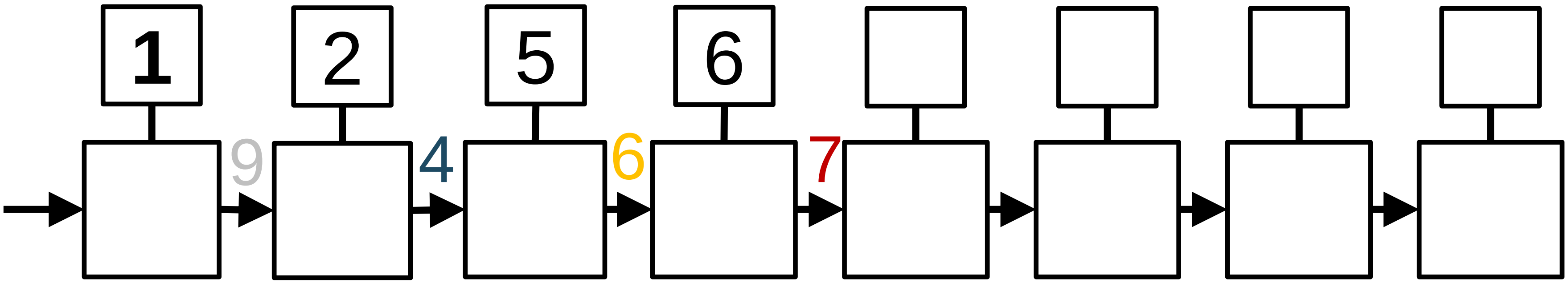
Sortare cu pipeline



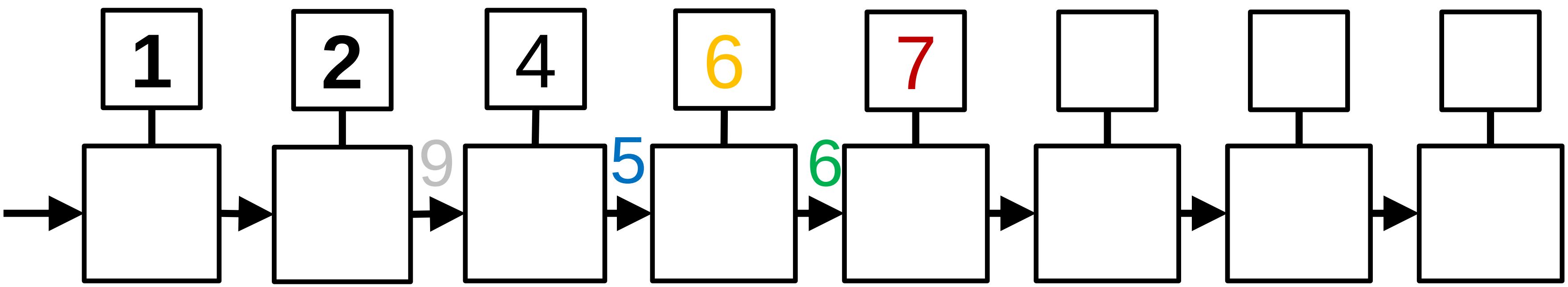
Sortare cu pipeline



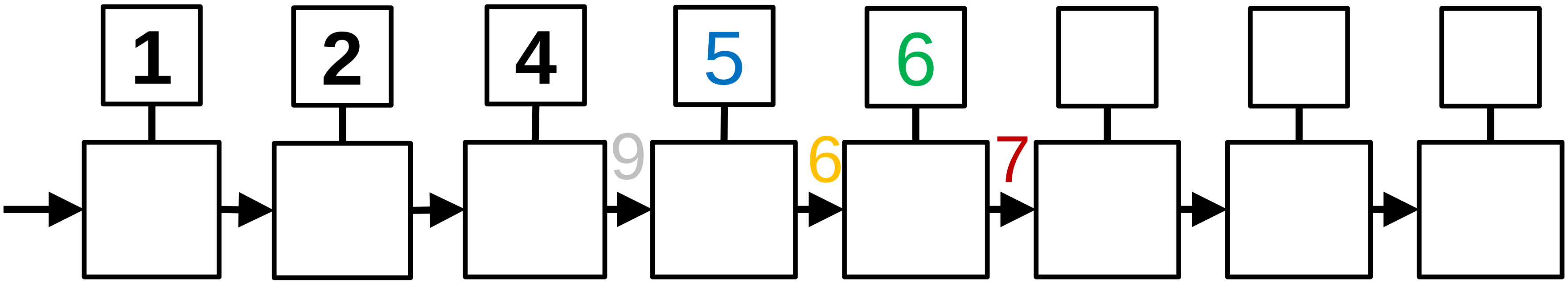
Sortare cu pipeline



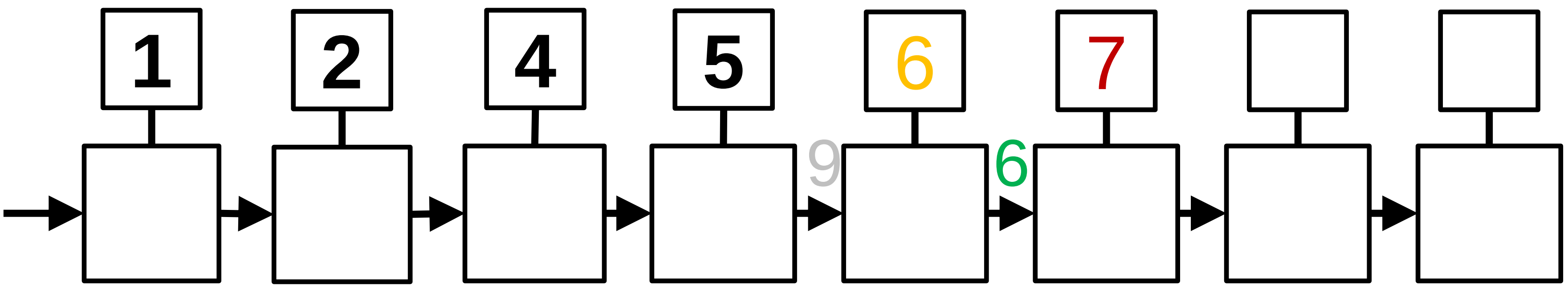
Sortare cu pipeline



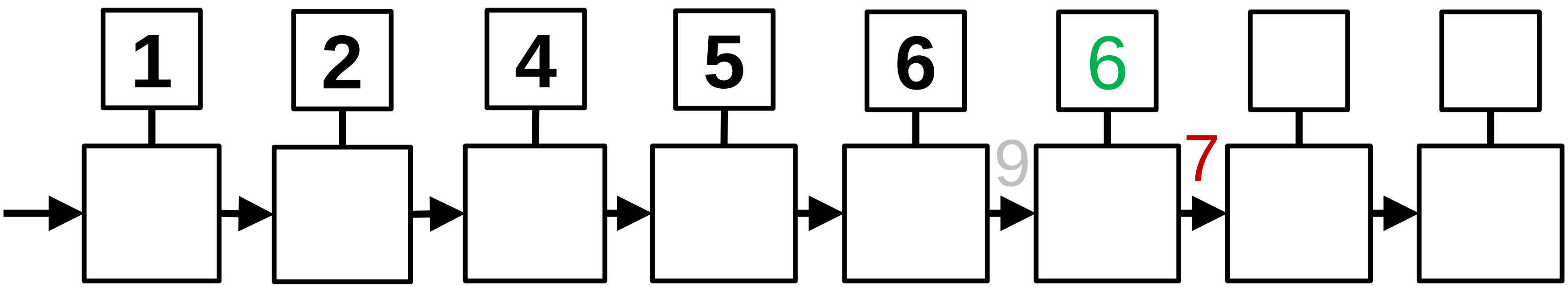
Sortare cu pipeline



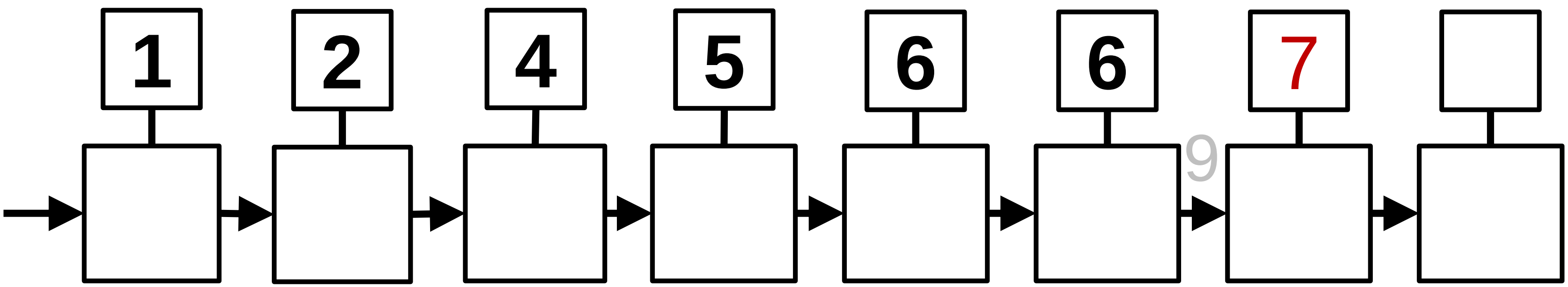
Sortare cu pipeline



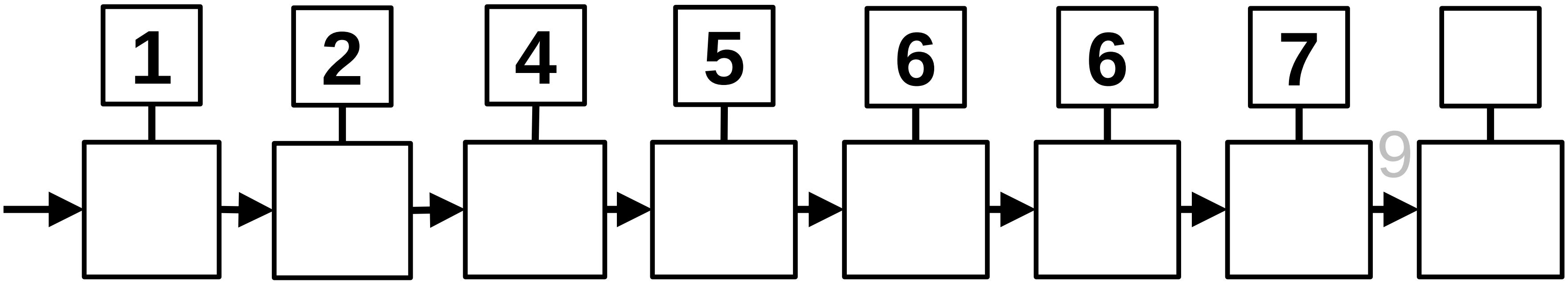
Sortare cu pipeline



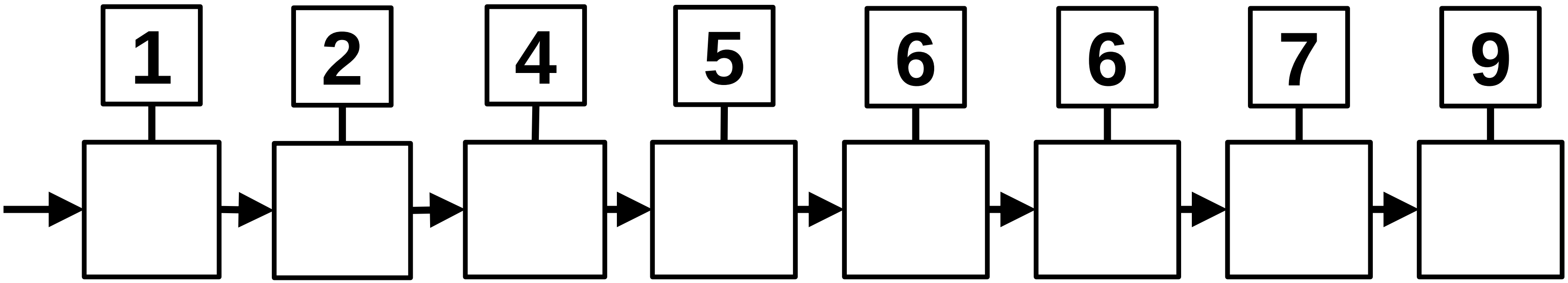
Sortare cu pipeline



Sortare cu pipeline



Sortare cu pipeline



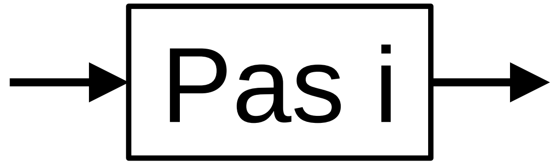
Sortare cu pipeline - complexitate

$$O(N)$$

pentru P=N

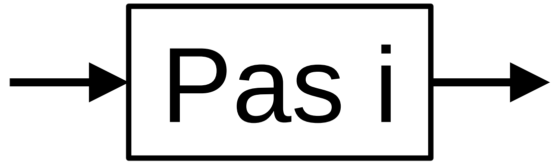
Dar comunicația e foarte lentă

Pipeline – ghid programare




```
Inițializare
for(un număr de pași) {
        primește date de la Pas(i-1)
        procesează
        trimite date la Pas(i+1)
}
Finalizare
```

Sortare cu Pipeline – ghid programare



noop;

for(fiecare face cu o operație mai puțin decât pasul precedent) {

    primește număr de la Pas(i-1)

    ține local numărul minim între cel primit sau cel avut

    trimite numărul mai mare la Pas(i+1)

}

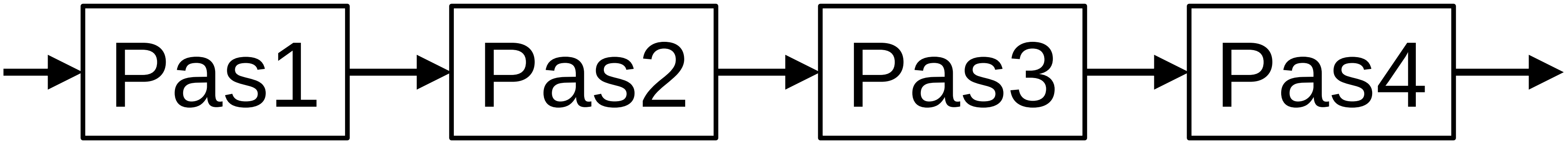
Scrie numărul la poziția i

Calcul unui polinom cu pipeline

$$P(x) = \sum_{i=0}^{n} a_i x^i = a_0 x^0 + a_1 x^1 + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n, n \geq 0$$

Se dorește să calculăm valoarea lui P pentru diverși x.
Use case: desenarea graficului aferent polinomului.

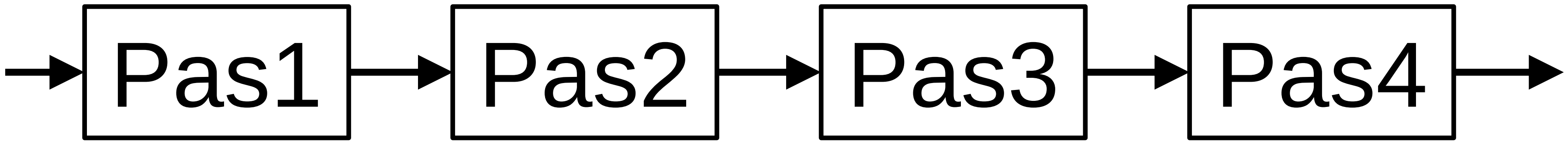
Pipeline



Calcul unui polinom cu pipeline

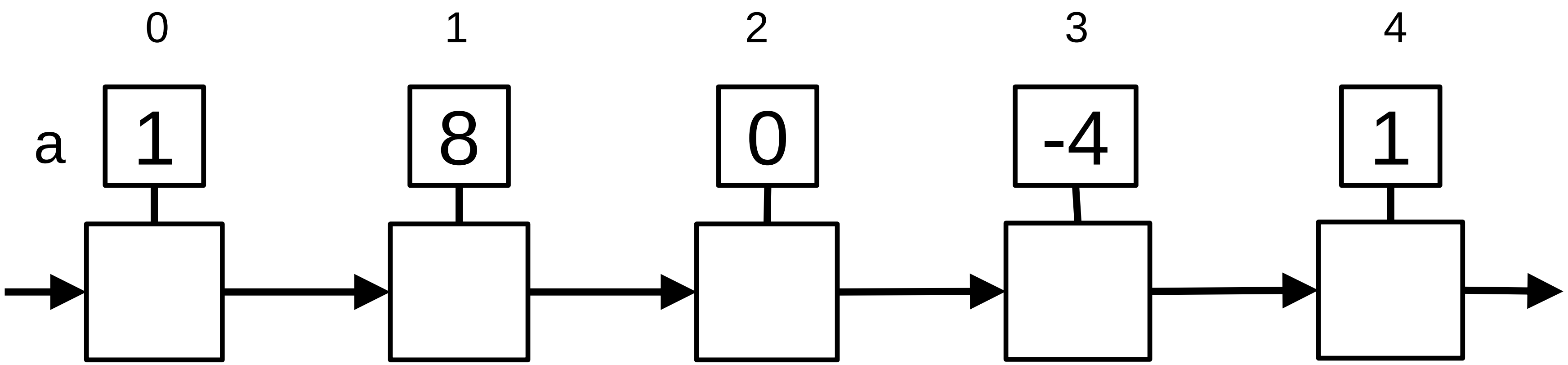
$$P(x) = 1 + 8x + (-4)x^3 + x^4$$

	0	1	2	3	4
a	1	8	0	-4	1



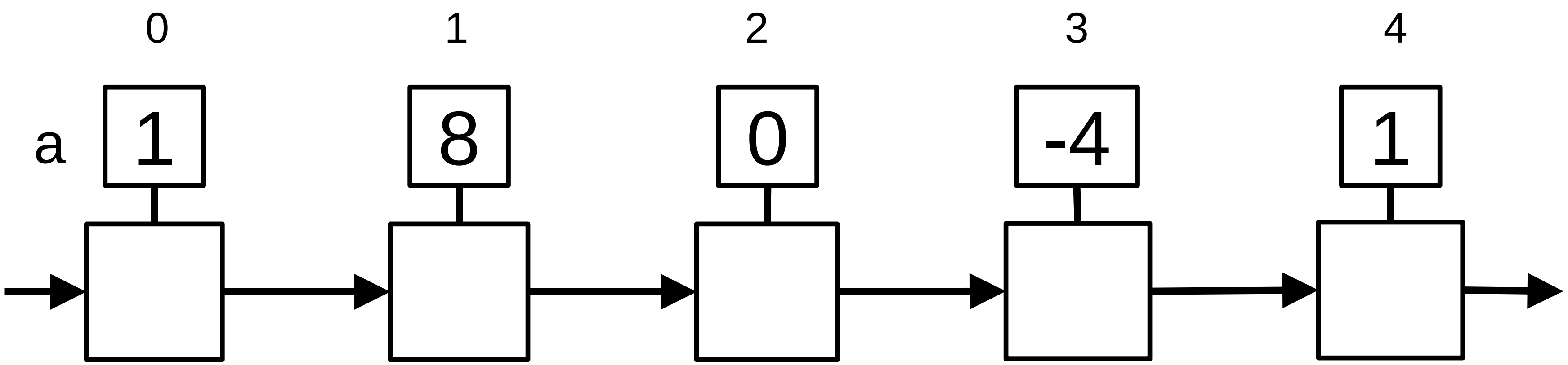
Calcul unui polinom cu pipeline

$$P(x) = 1 + 8x + (-4)x^3 + x^4$$



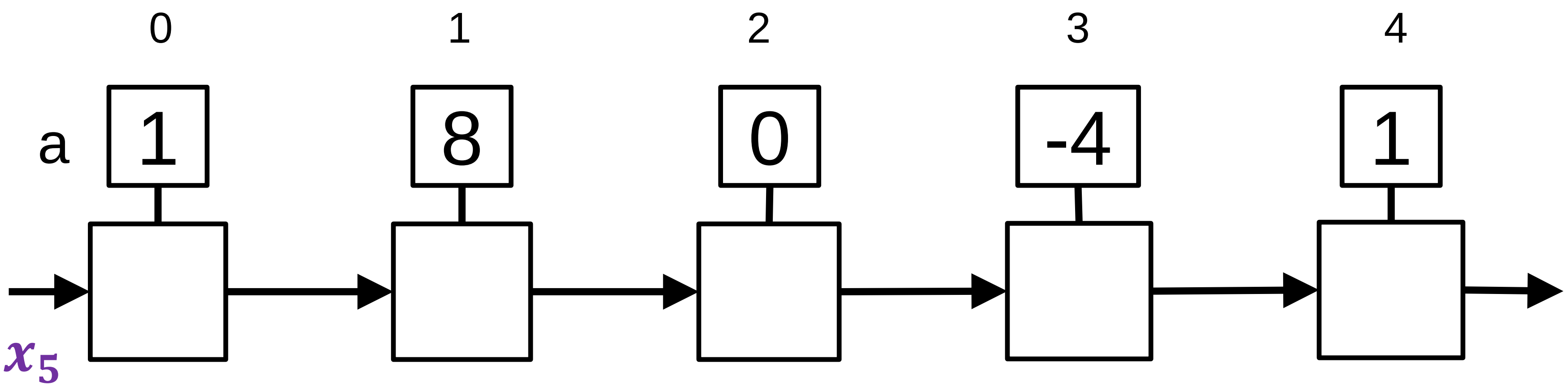
Calcul unui polinom cu pipeline

x_1 x_2 x_3 x_4 x_5



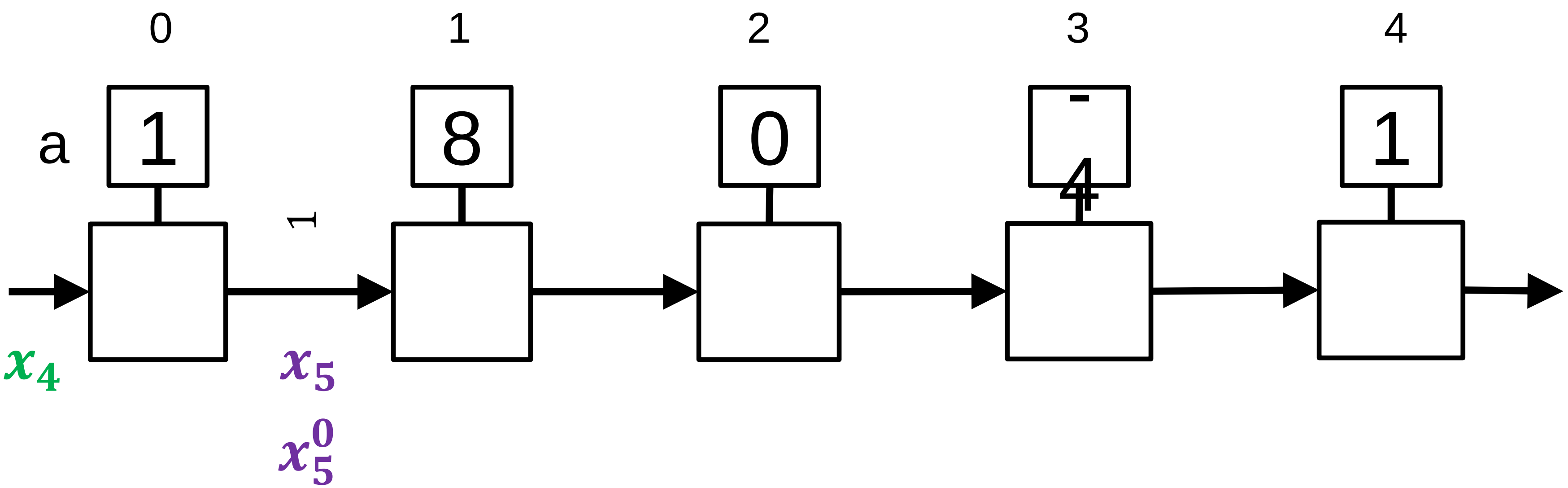
Calcul unui polinom cu pipeline

x_1 x_2 x_3 x_4

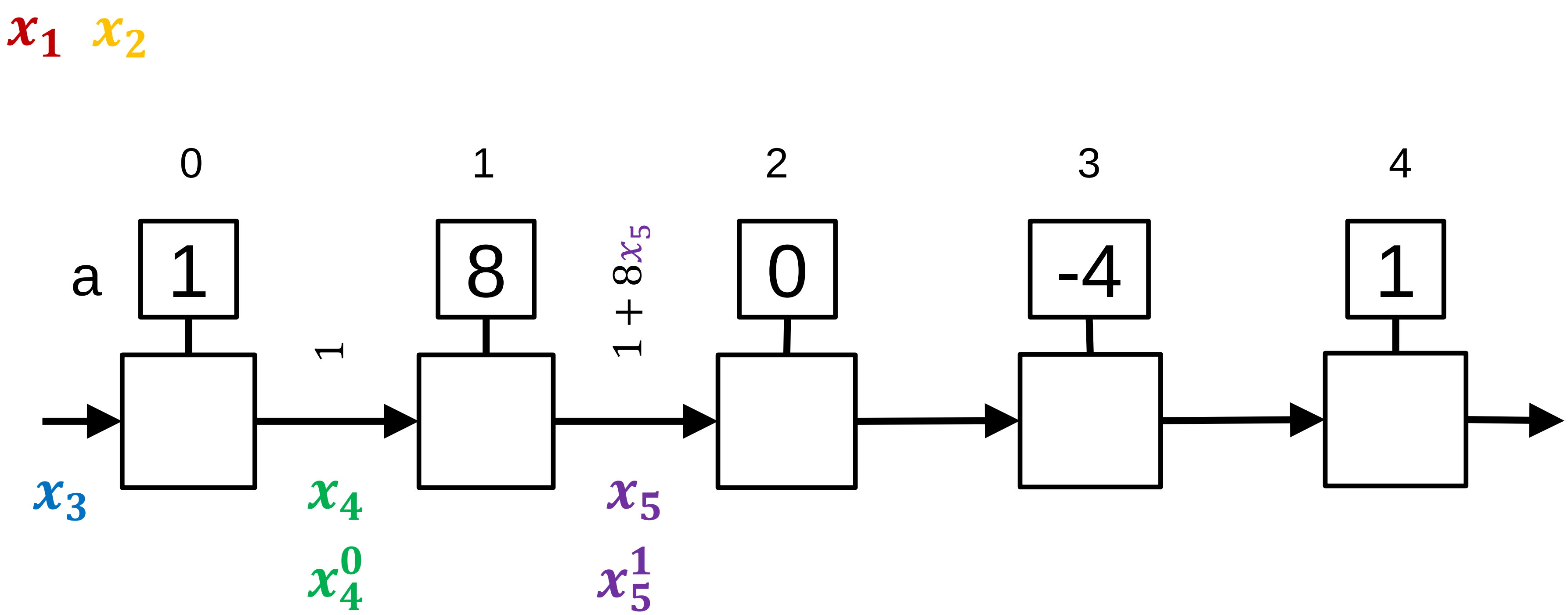


Calcul unui polinom cu pipeline

x_1 x_2 x_3

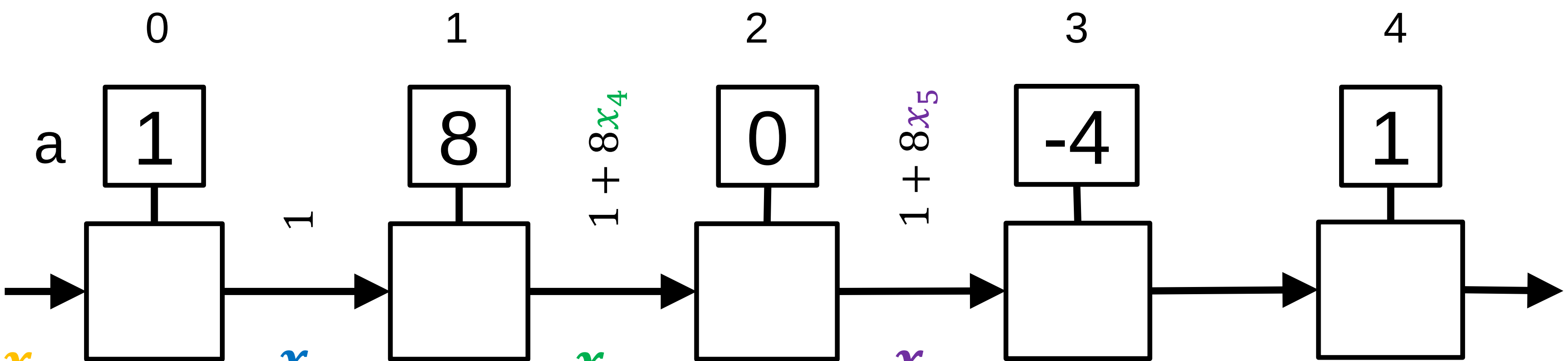


Calcul unui polinom cu pipeline



Calcul unui polinom cu pipeline

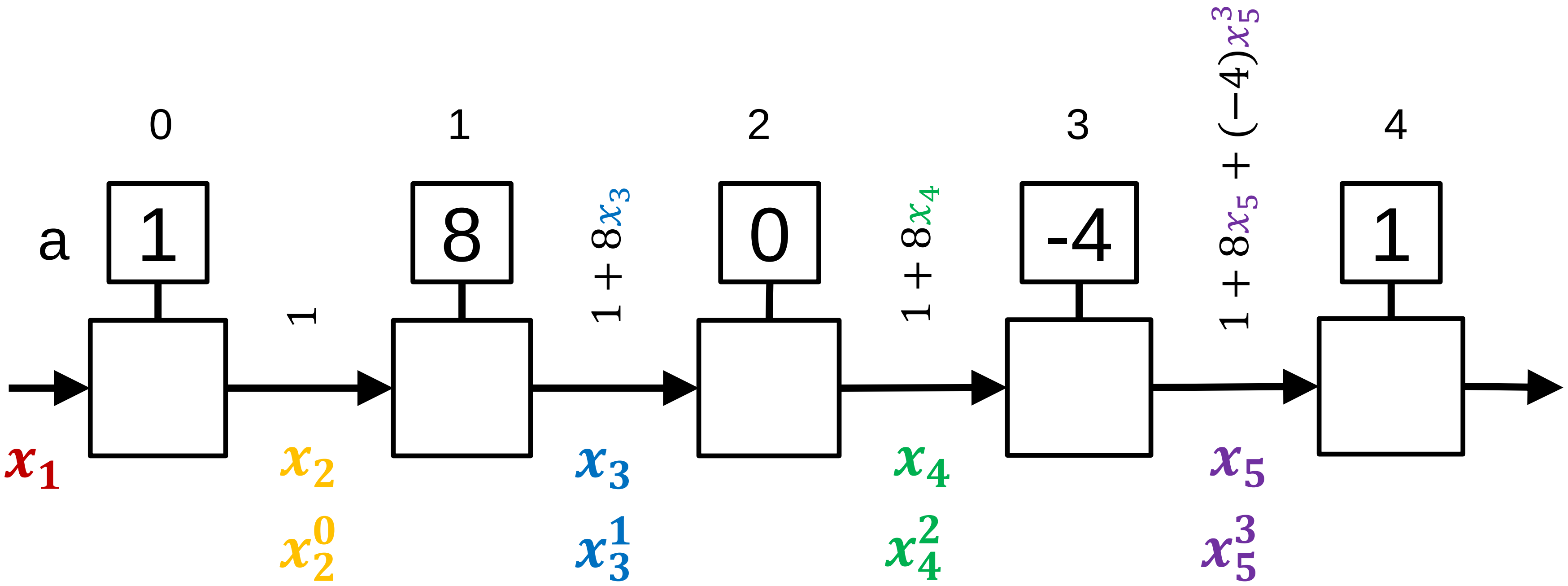
x_1



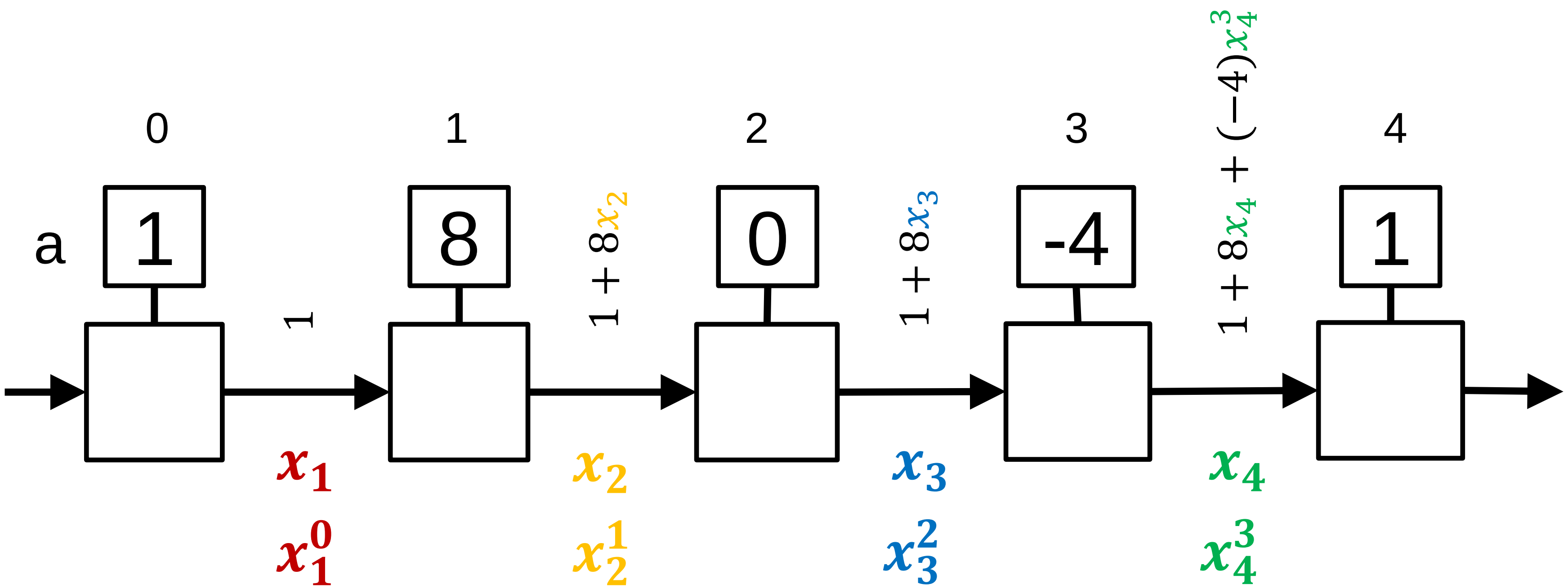
x_2 x_3 x_4 x_5

x_3^0 x_4^1 x_5^2

Calcul unui polinom cu pipeline

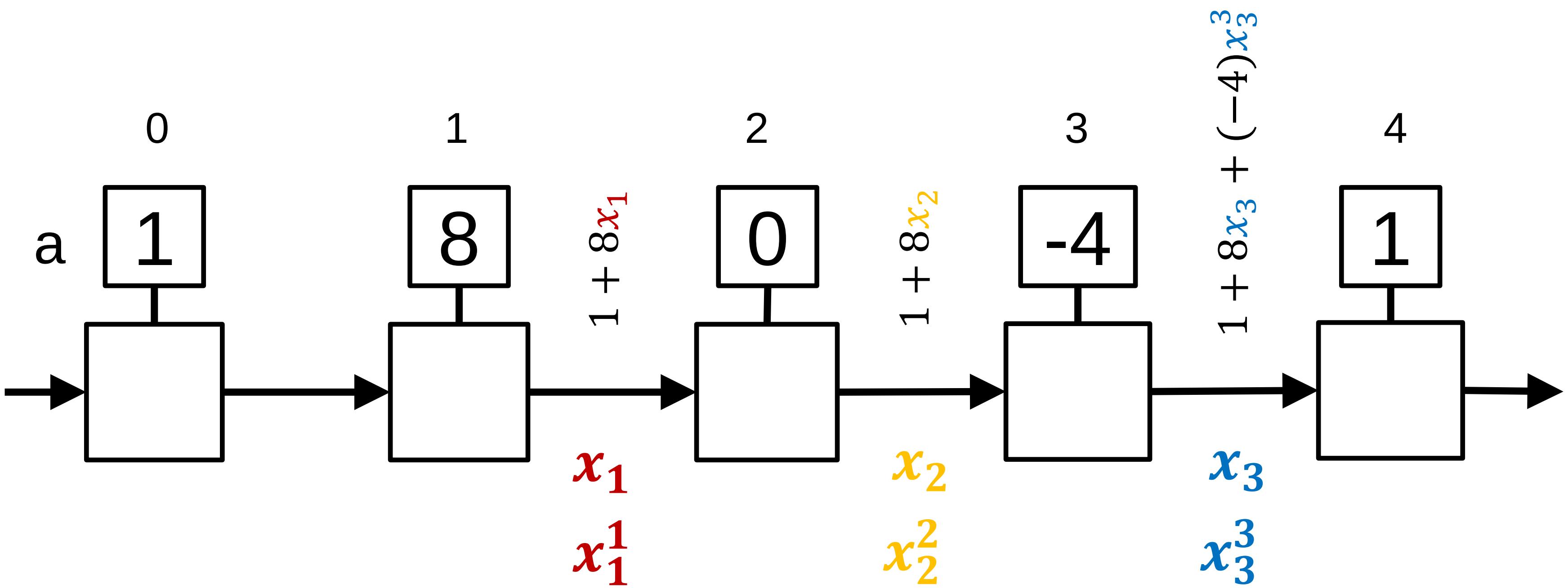


Calcul unui polinom cu pipeline



$$\mathbf{1 + 8}x_5 + (\mathbf{-4})x_5^3 + x_5^4$$

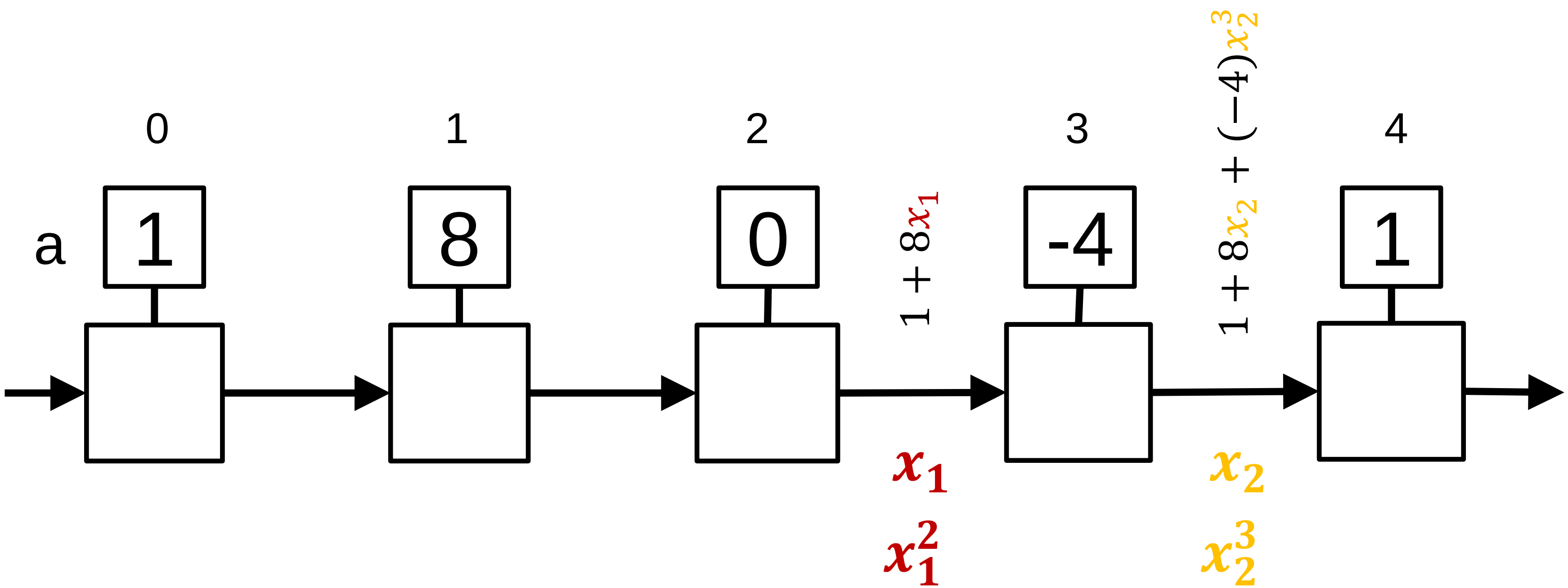
Calcul unui polinom cu pipeline



$$\mathbf{1 + 8x_4 + (-4)x_4^3 + x_4^4}$$

$$\mathbf{1 + 8x_5 + (-4)x_5^3 + x_5^4}$$

Calcul unui polinom cu pipeline

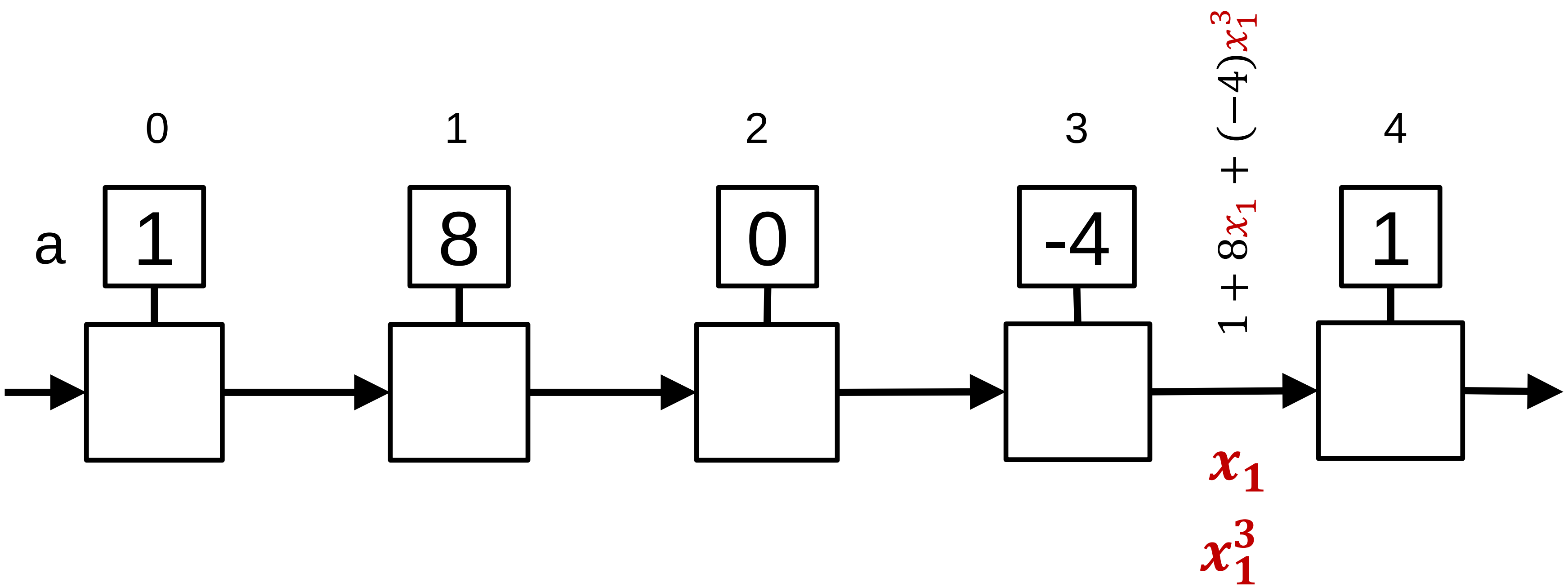


$$\mathbf{1} + \mathbf{8}x_3 + (-\mathbf{4})x_3^3 + x_3^4$$

$$\mathbf{1} + \mathbf{8}x_4 + (-\mathbf{4})x_4^3 + x_4^4$$

$$\mathbf{1} + \mathbf{8}x_5 + (-\mathbf{4})x_5^3 + x_5^4$$

Calcul unui polinom cu pipeline

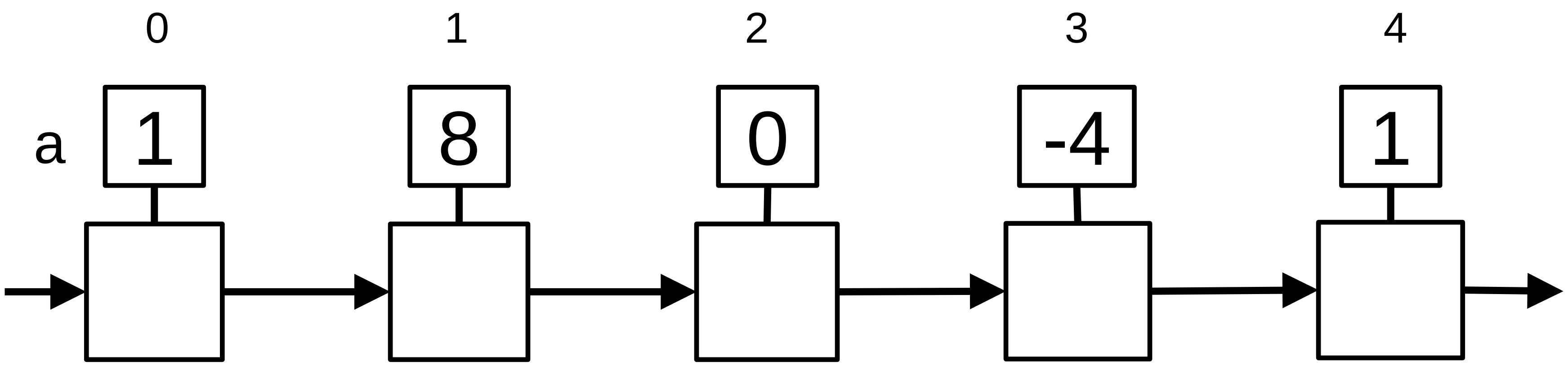


$$\mathbf{1} + \mathbf{8}x_3 + (-\mathbf{4})x_3^3 + x_3^4$$

$$\mathbf{1} + \mathbf{8}x_4 + (-\mathbf{4})x_4^3 + x_4^4$$

$$\mathbf{1} + \mathbf{8}x_2 + (-\mathbf{4})x_2^3 + x_2^4 \qquad \mathbf{1} + \mathbf{8}x_5 + (-\mathbf{4})x_5^3 + x_5^4$$

Calcul unui polinom cu pipeline

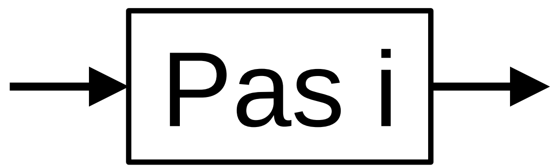


$$\mathbf{1} + \mathbf{8}x_3 + (-\mathbf{4})x_3^3 + x_3^4$$

$$\mathbf{1} + \mathbf{8}x_1 + (-\mathbf{4})x_1^3 + x_1^4 \qquad \mathbf{1} + \mathbf{8}x_4 + (-\mathbf{4})x_4^3 + x_4^4$$

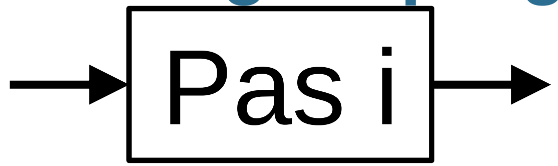
$$\mathbf{1} + \mathbf{8}x_2 + (-\mathbf{4})x_2^3 + x_2^4 \qquad \mathbf{1} + \mathbf{8}x_5 + (-\mathbf{4})x_5^3 + x_5^4$$

Pipeline – ghid programare




```
Inițializare
for(un număr de pași) {
        primește date de la Pas(i-1)
        procesează
        trimite date la Pas(i+1)
}
Finalizare
```

Pipeline – ghid programare



Primește coeficientul potrivit pasului

for(un număr de pași egal cu numărul de valori) {

primește de la Pas(i-1): polinom parțial calculat

valoarea originală

valoare originală ^ (i-1)

Calculează (valoarea originală ^ i) și adaugă la polinom parțial calculat produsul dintre coeficient și aceasta.

trimite spre Pas(i+1): noul polinom parțial calculat, valoarea originală și valoare originală ^ i)

}