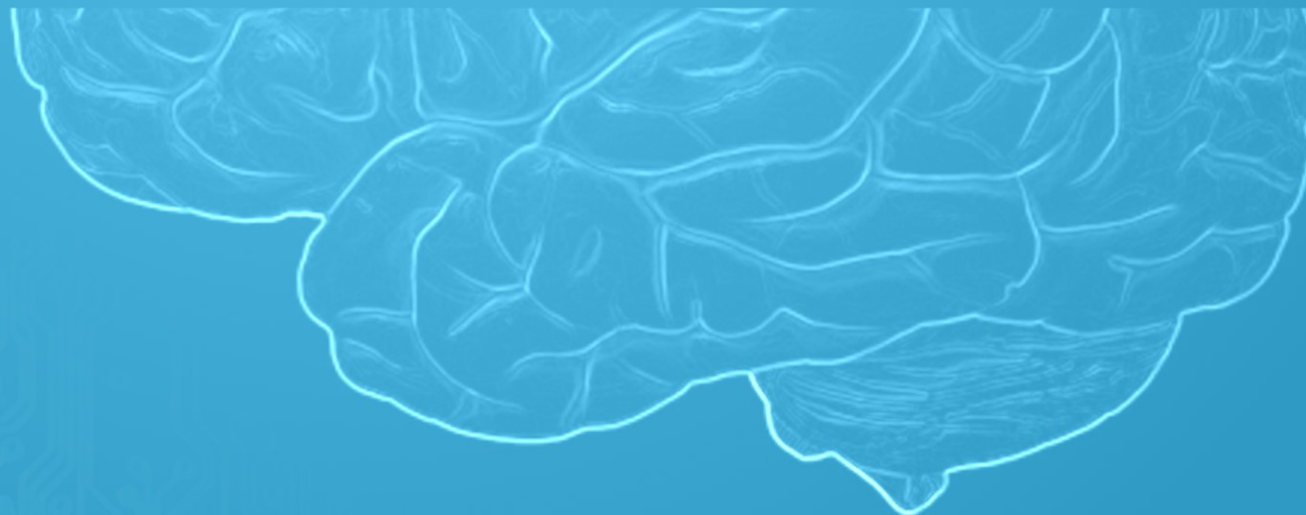




# Arhitecturi Paralele Soluții Arborescente

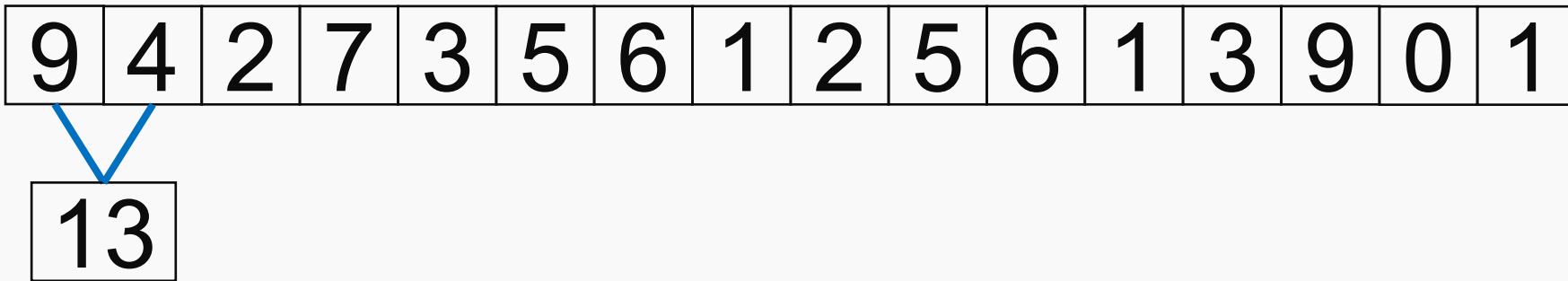
Lect. Dr. Ing. Cristian Chilipirea – [cristian.chilipirea@mta.ro](mailto:cristian.chilipirea@mta.ro)





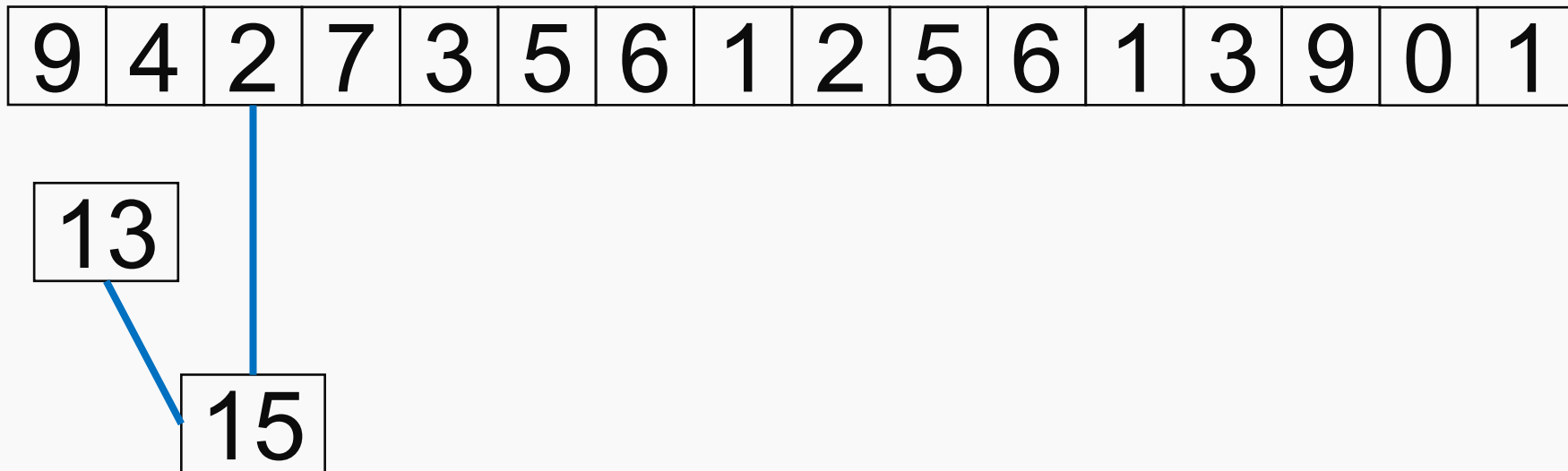


# Sumă - clasic



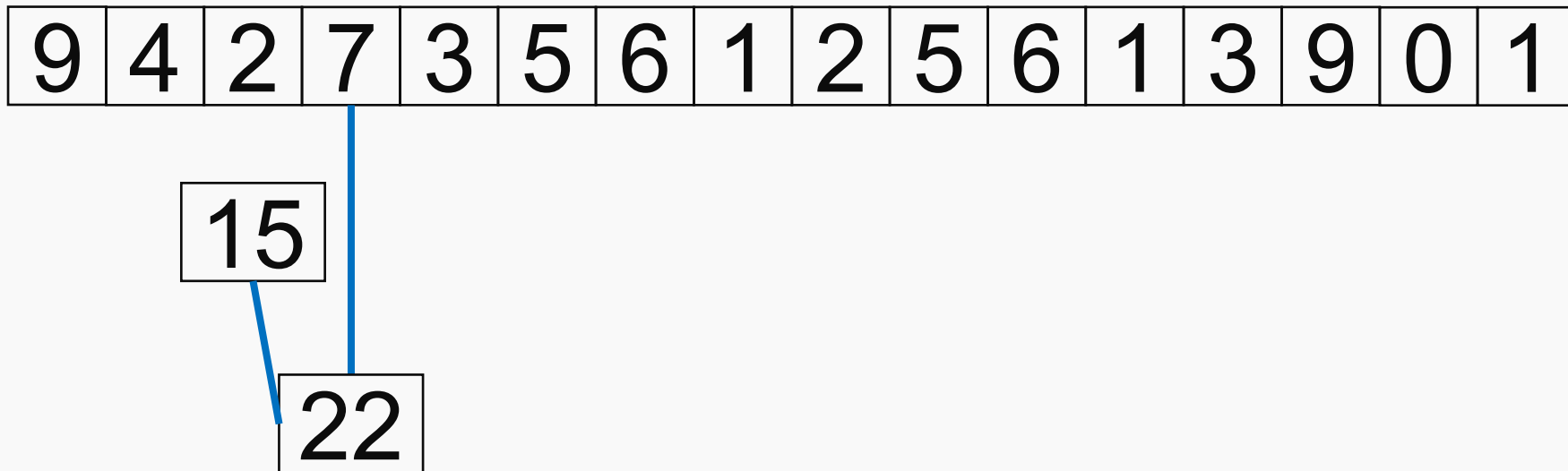


# Sumă - clasic



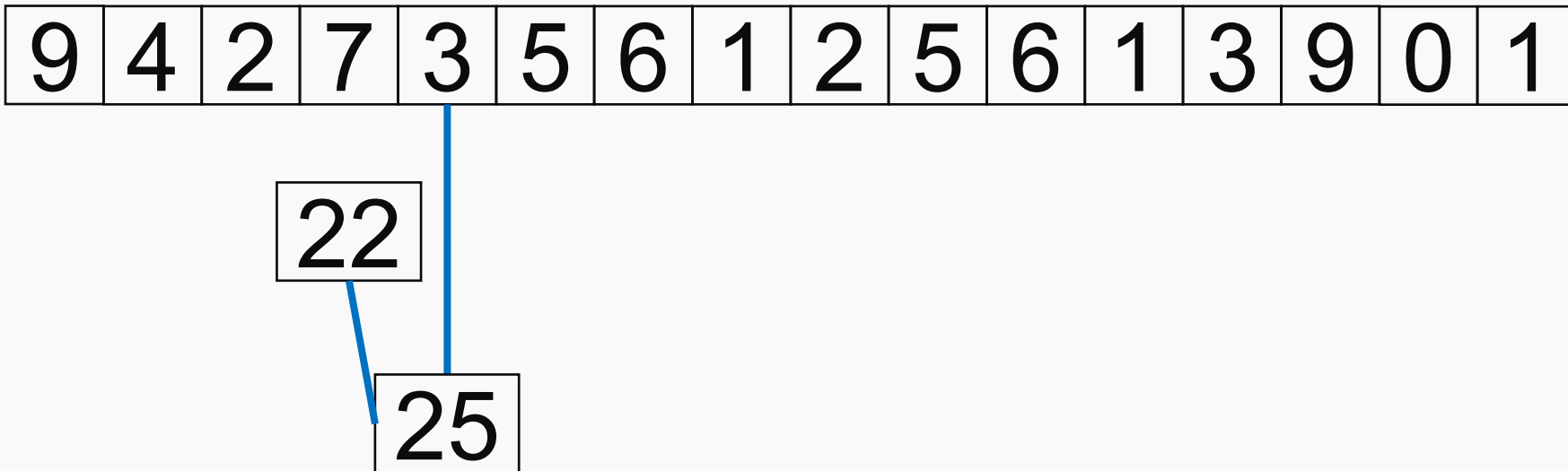


# Sumă - clasic



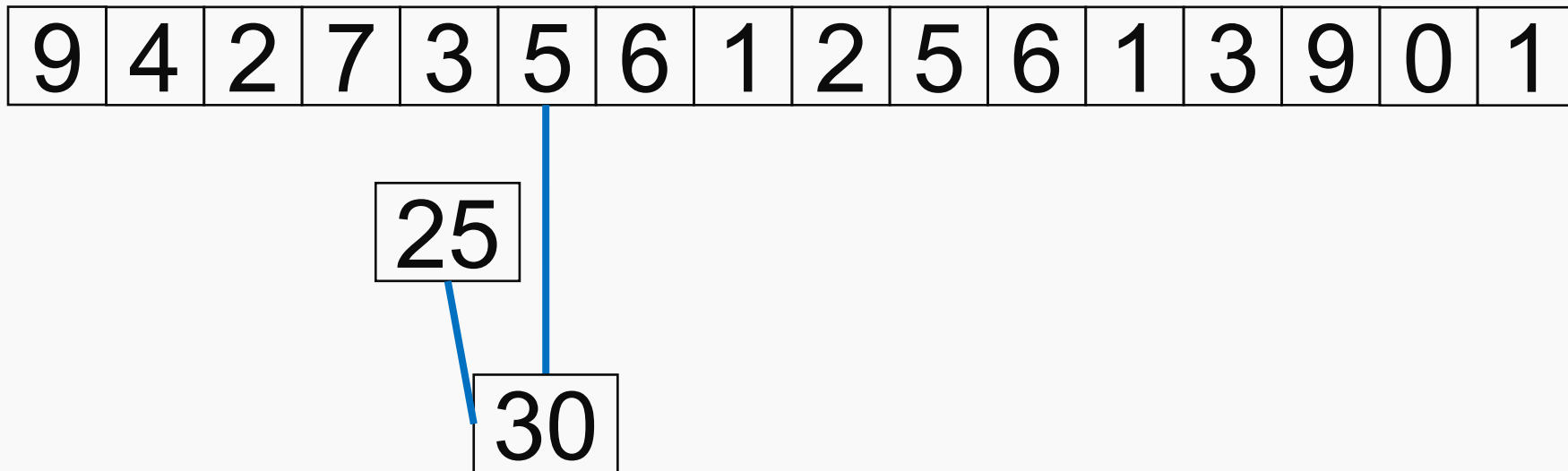


# Sumă - clasic



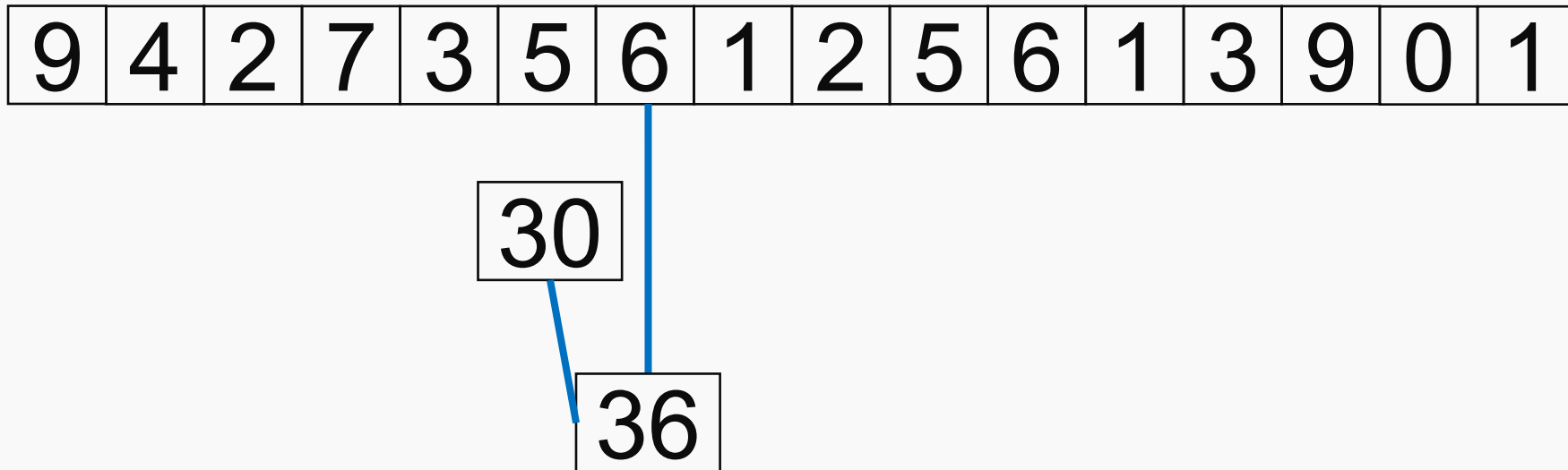


# Sumă - clasic





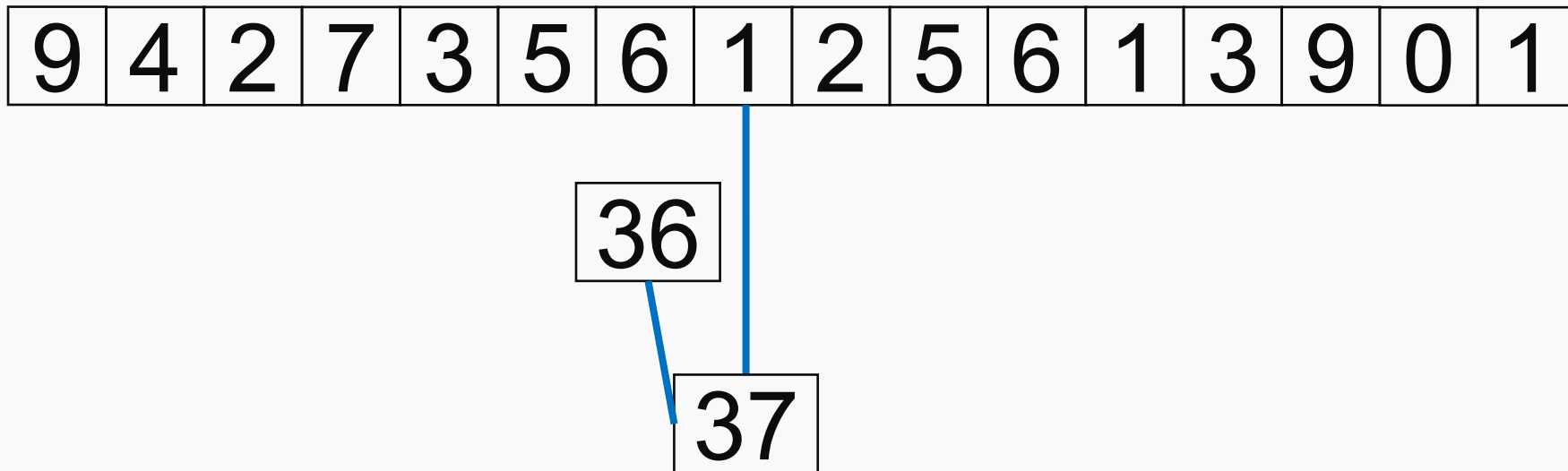
# Sumă - clasic





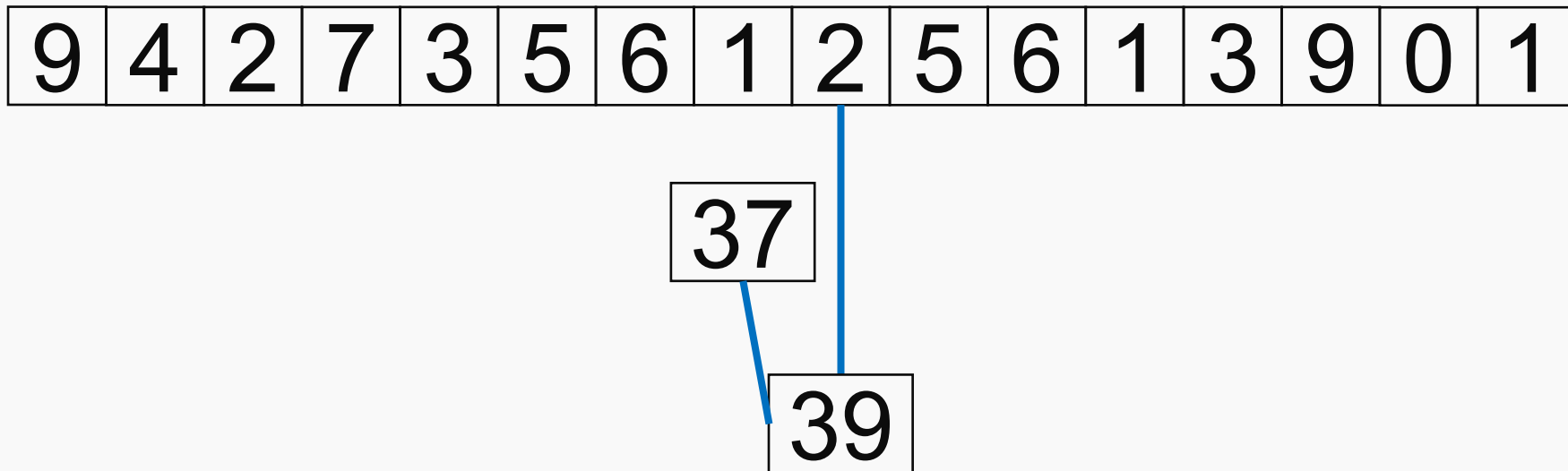


# Sumă - clasic



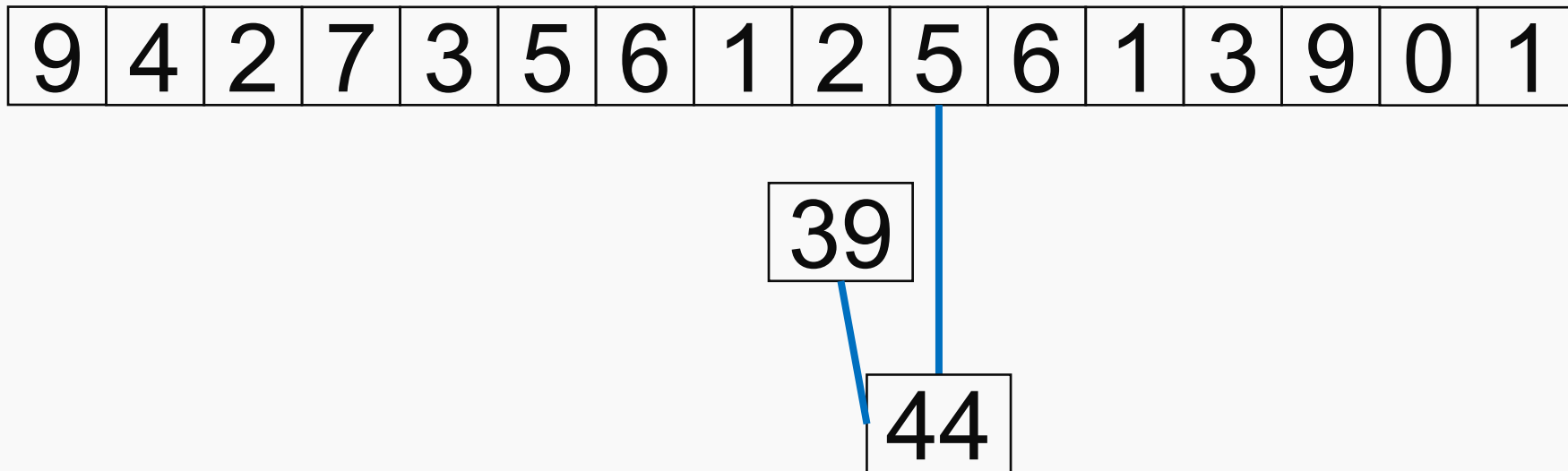


# Sumă - clasic





# Sumă - clasic





# Sumă - clasic

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

44

50



# Sumă - clasic

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

50

51



# Sumă - clasic

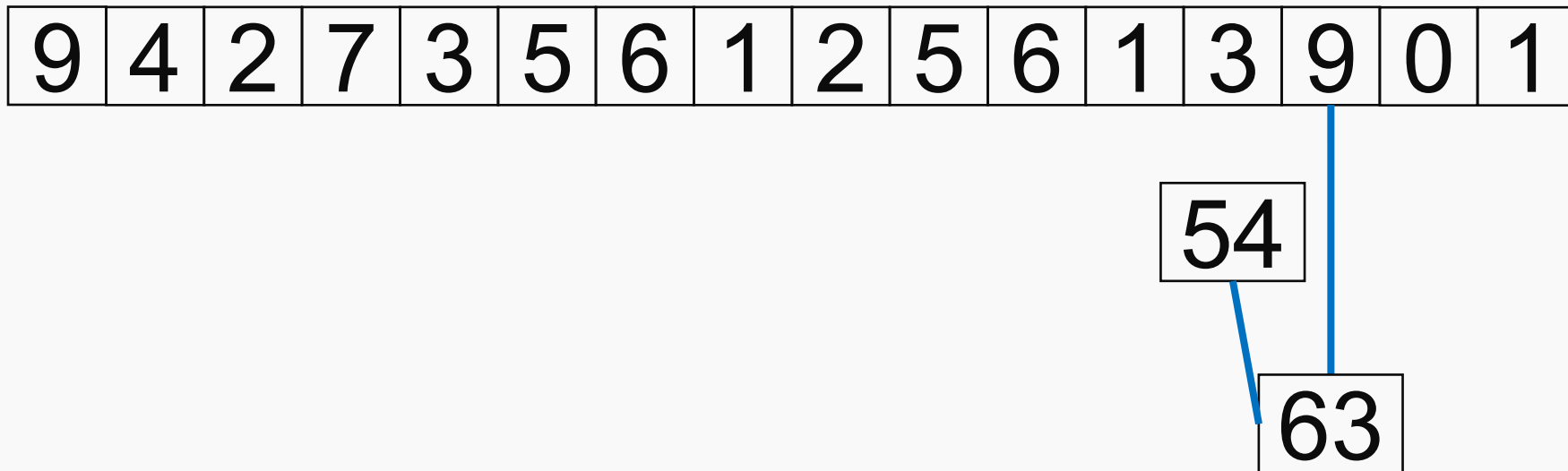
9	4	2	7	3	5	6	1	2	5	6	1	3	9	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

51

54



# Sumă - clasic





# Sumă - clasic

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

63

63





# Sumă - clasic

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

63

64



## Sumă - clasic

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

Complexitate  $O(N)$

64
----



# Sumă - paralelizare

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

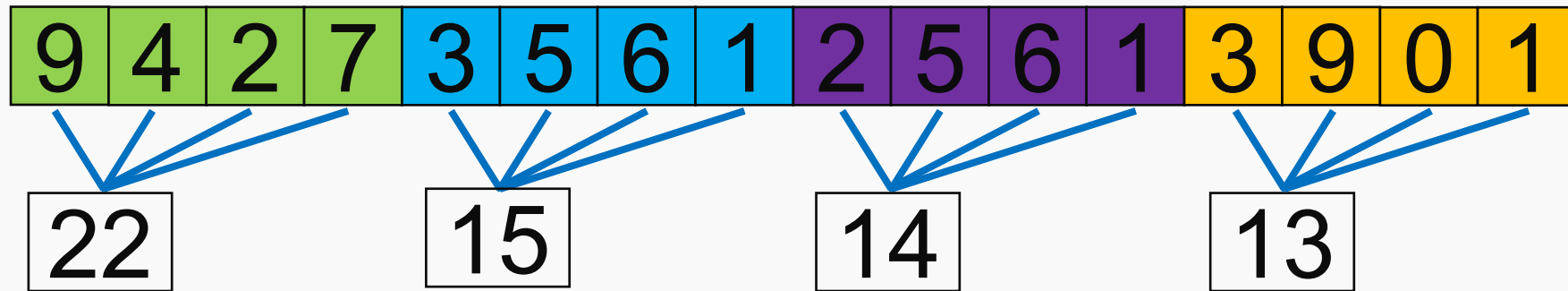


# Sumă - paralelizare

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

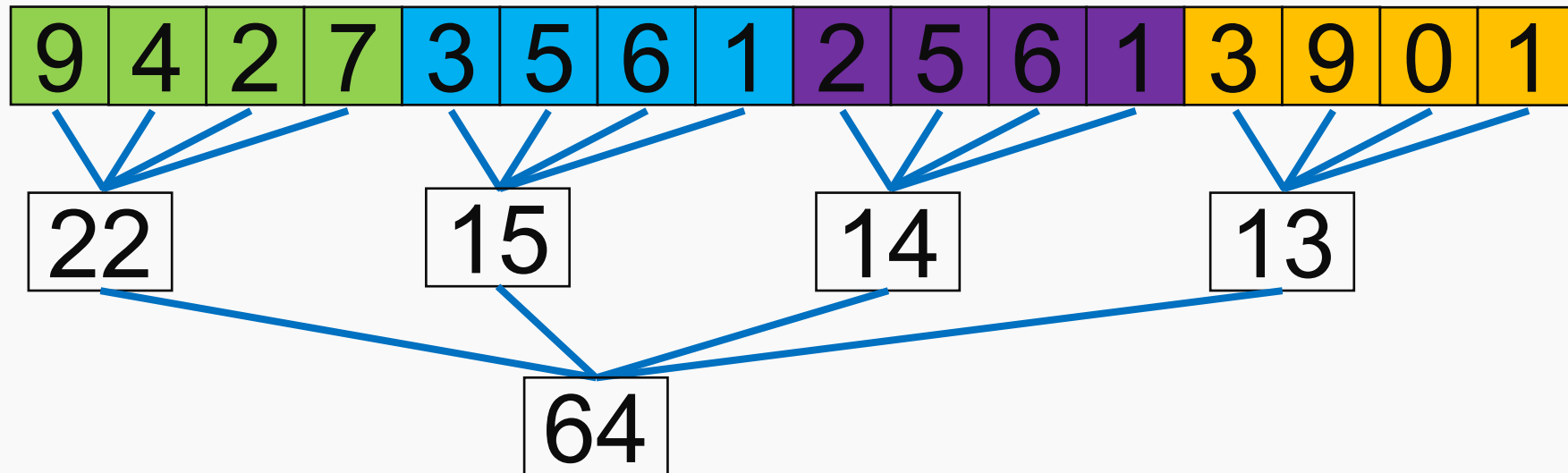


# Sumă - paralelizare





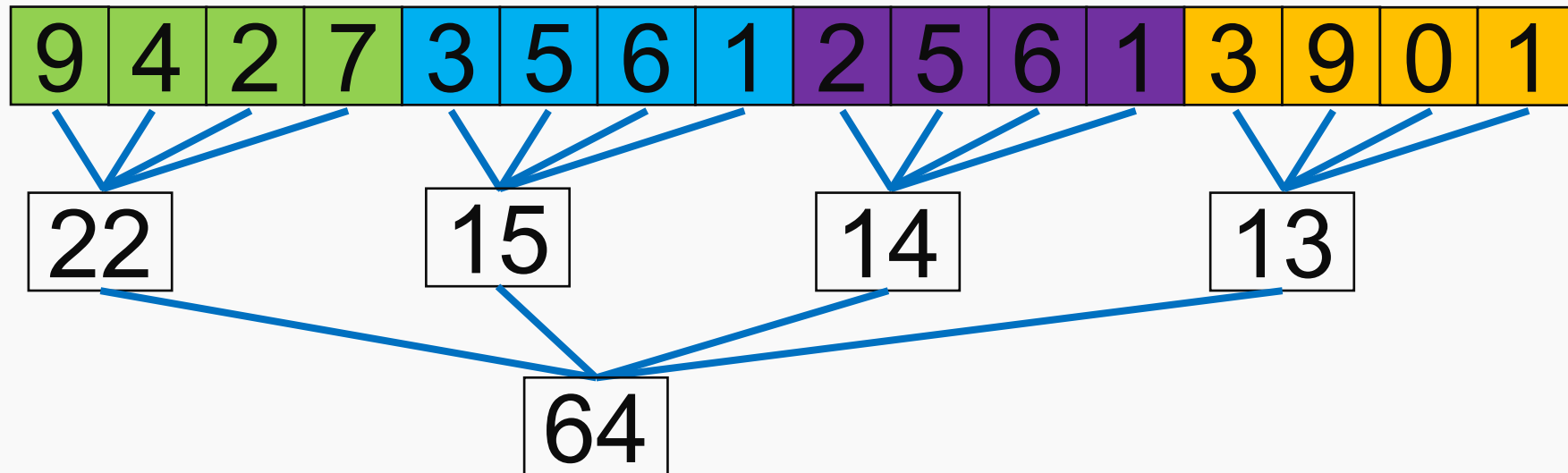
# Sumă - paralelizare



**Complexitate?**



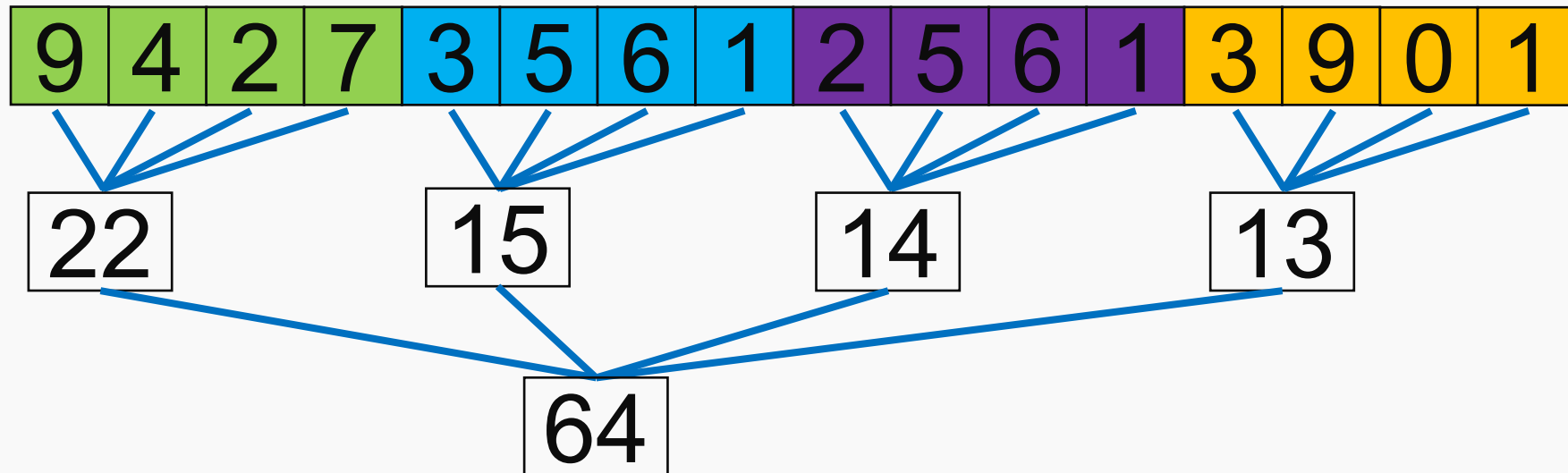
# Sumă - paralelizare



**Complexitate:**  $O(\frac{N}{P} + P)$



# Sumă - paralelizare



**Complexitate:**  $O(\frac{N}{P} + P)$

**Dar dacă P foarte mare?**





# Reduce

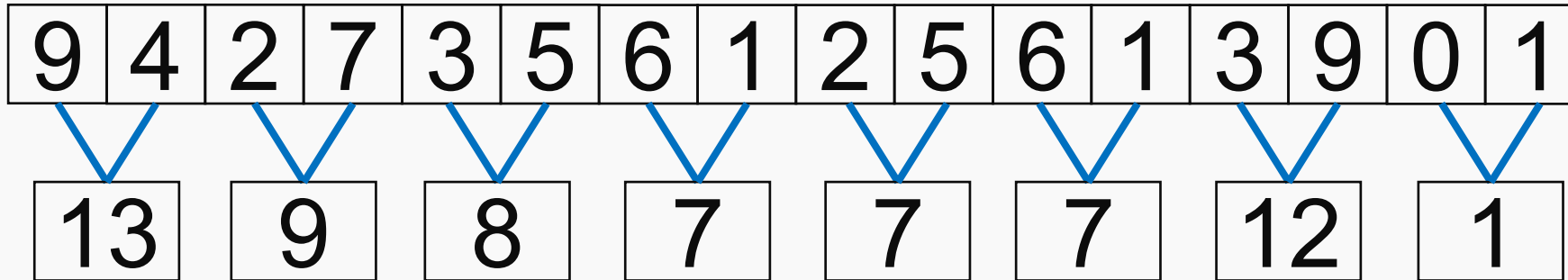
**Soluție paralelă pentru o problemă în care se aplică aceeași operație pe toate elementele unui vector.**

**Se poate executa în  $O(\log(N))$  pe  $N$  procesoare organizând calculele într-o formă arborescentă.**

**Operația trebuie să fie comutativă de exemplu:  
+, \*, min, max, and, ...**



## Reduce cu sumă

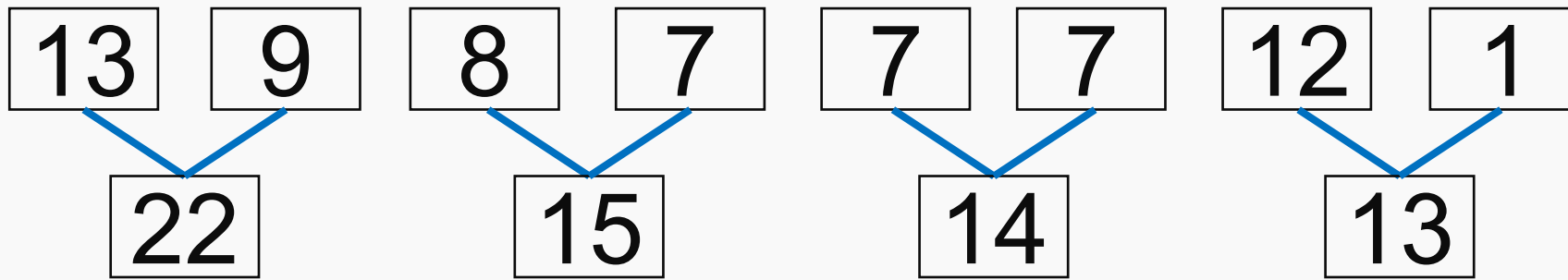


Pot fi executate în paralel





# Reduce cu sumă

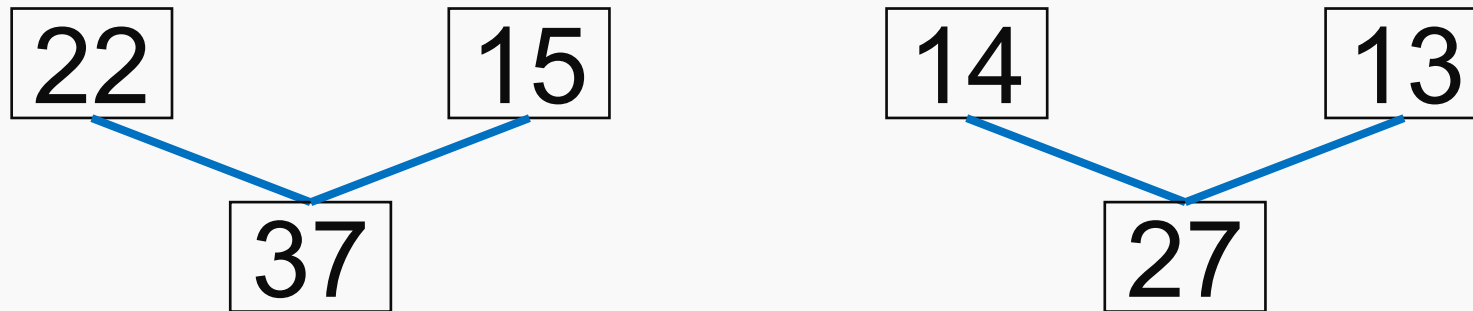


Pot fi executate în paralel





# Reduce cu sumă

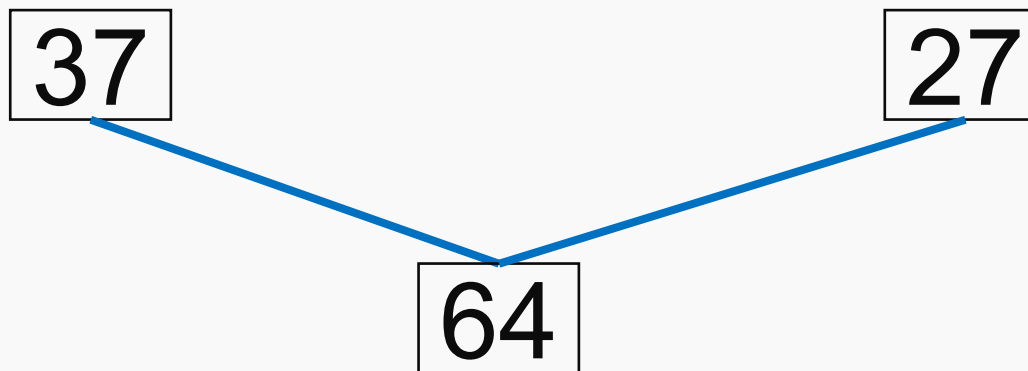


Pot fi executate în paralel



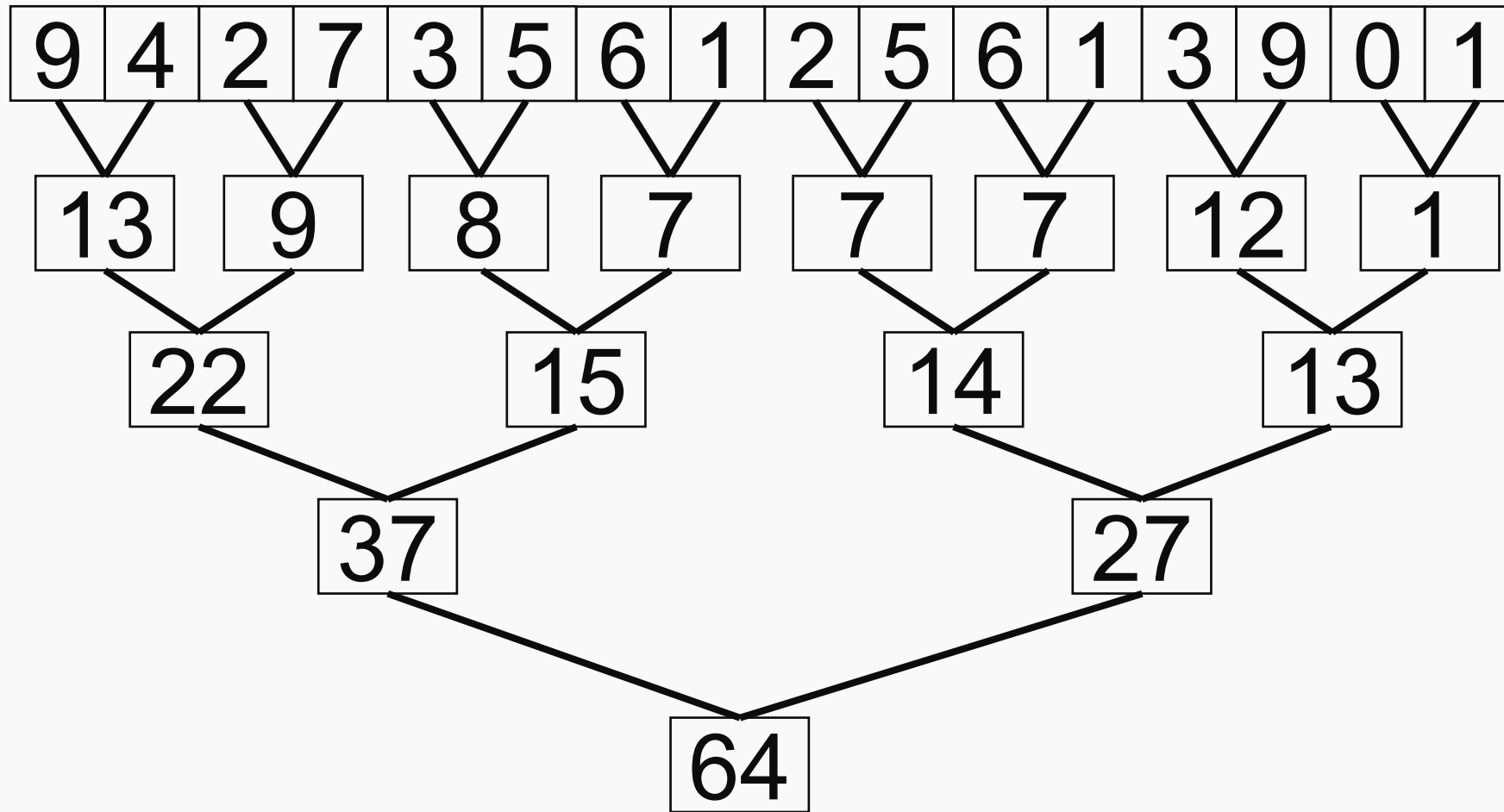


# Reduce cu sumă



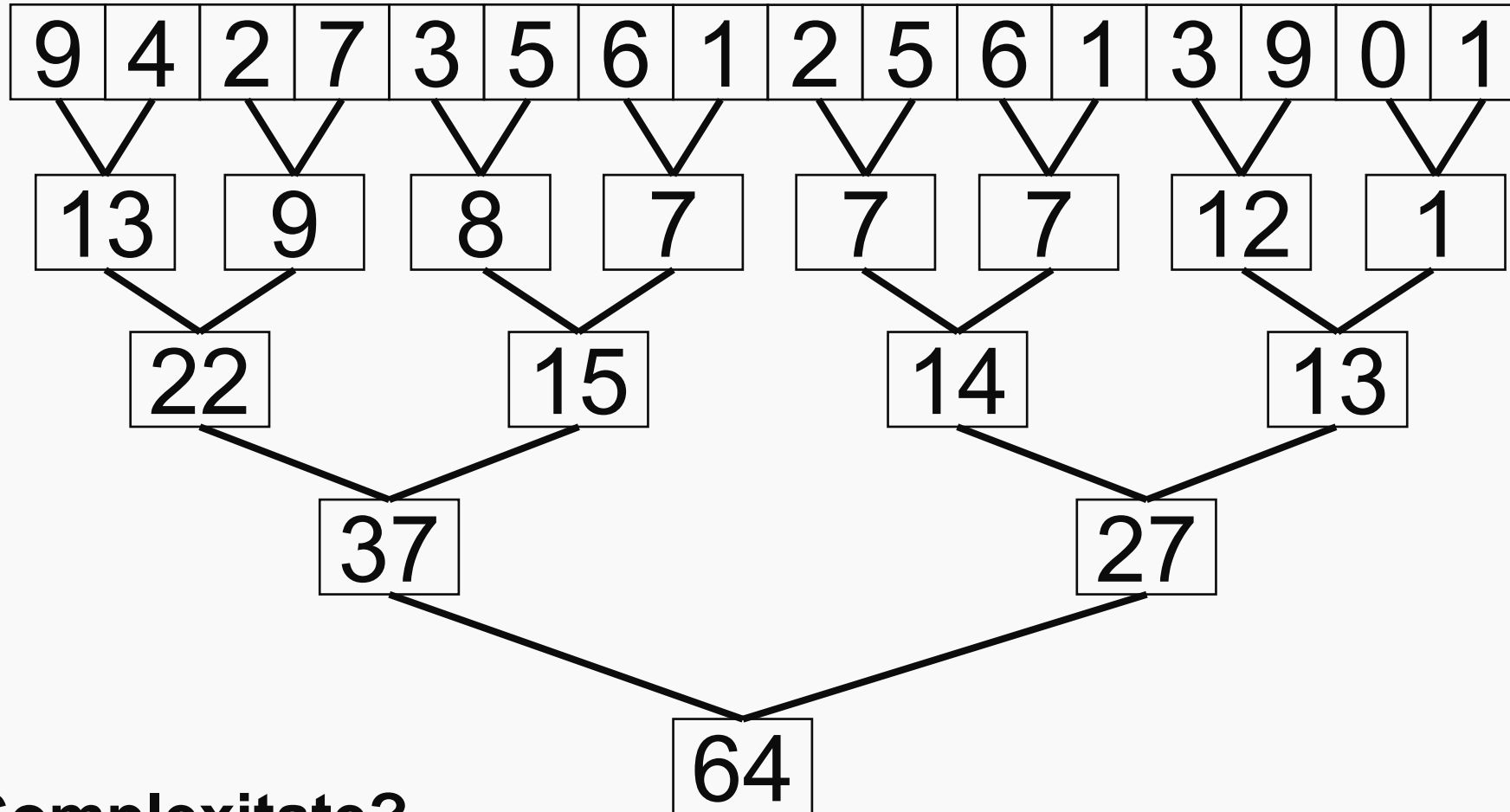


## Reduce cu sumă





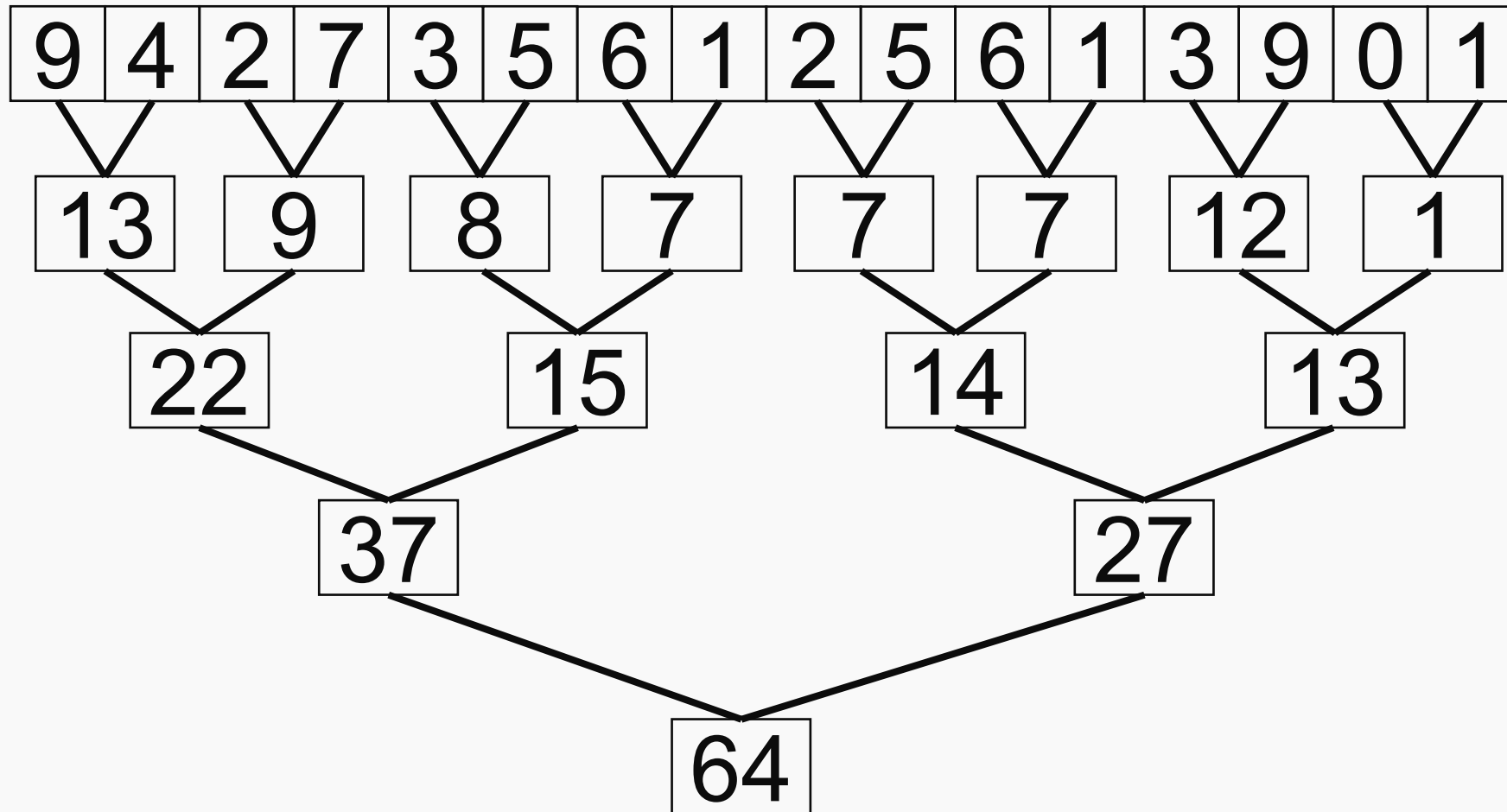
## Reduce cu sumă



**Complexitate?**



## Reduce cu sumă

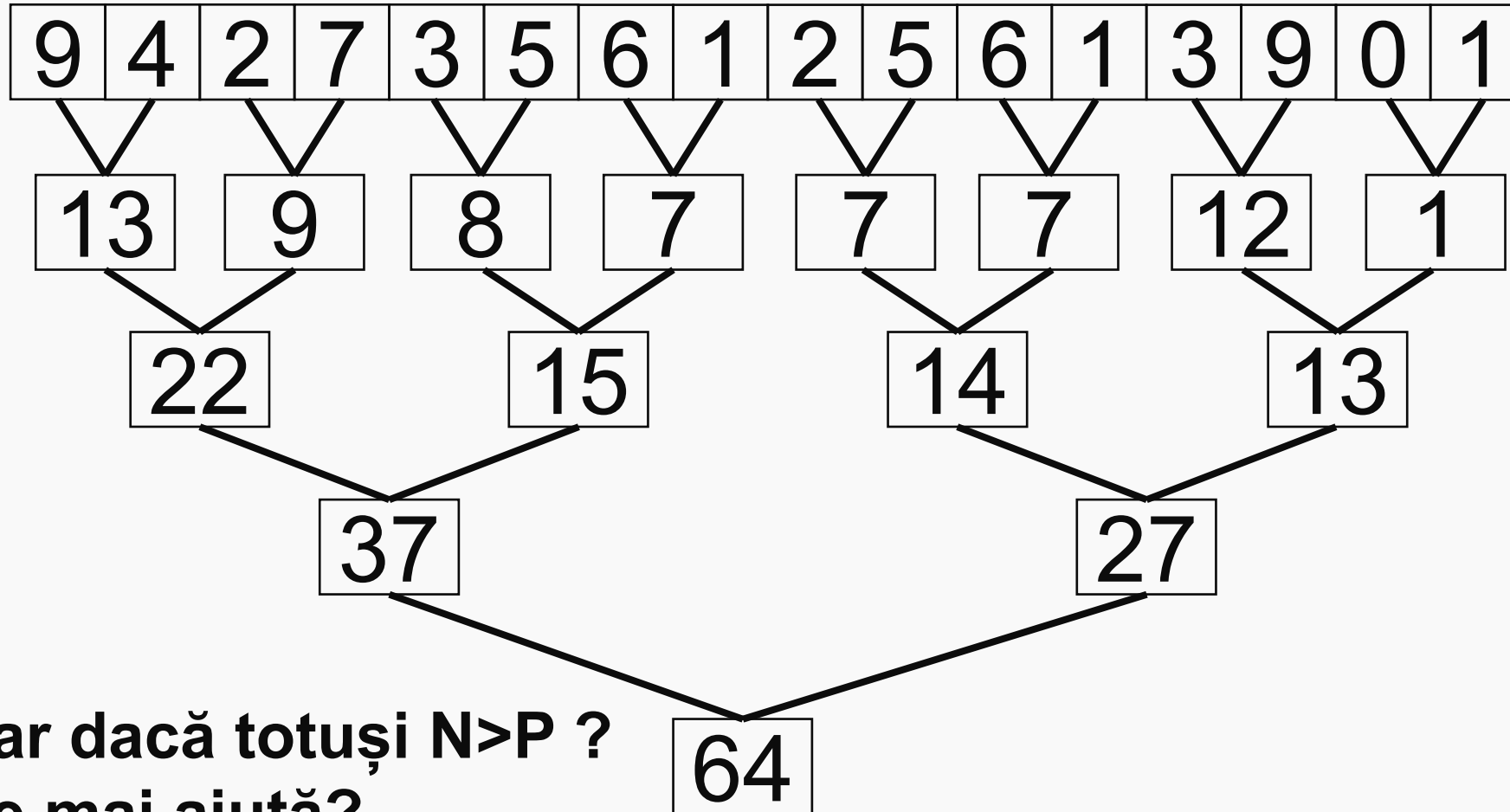


**Complexitate:  $O(\log(N))$  dacă  $P=N$**





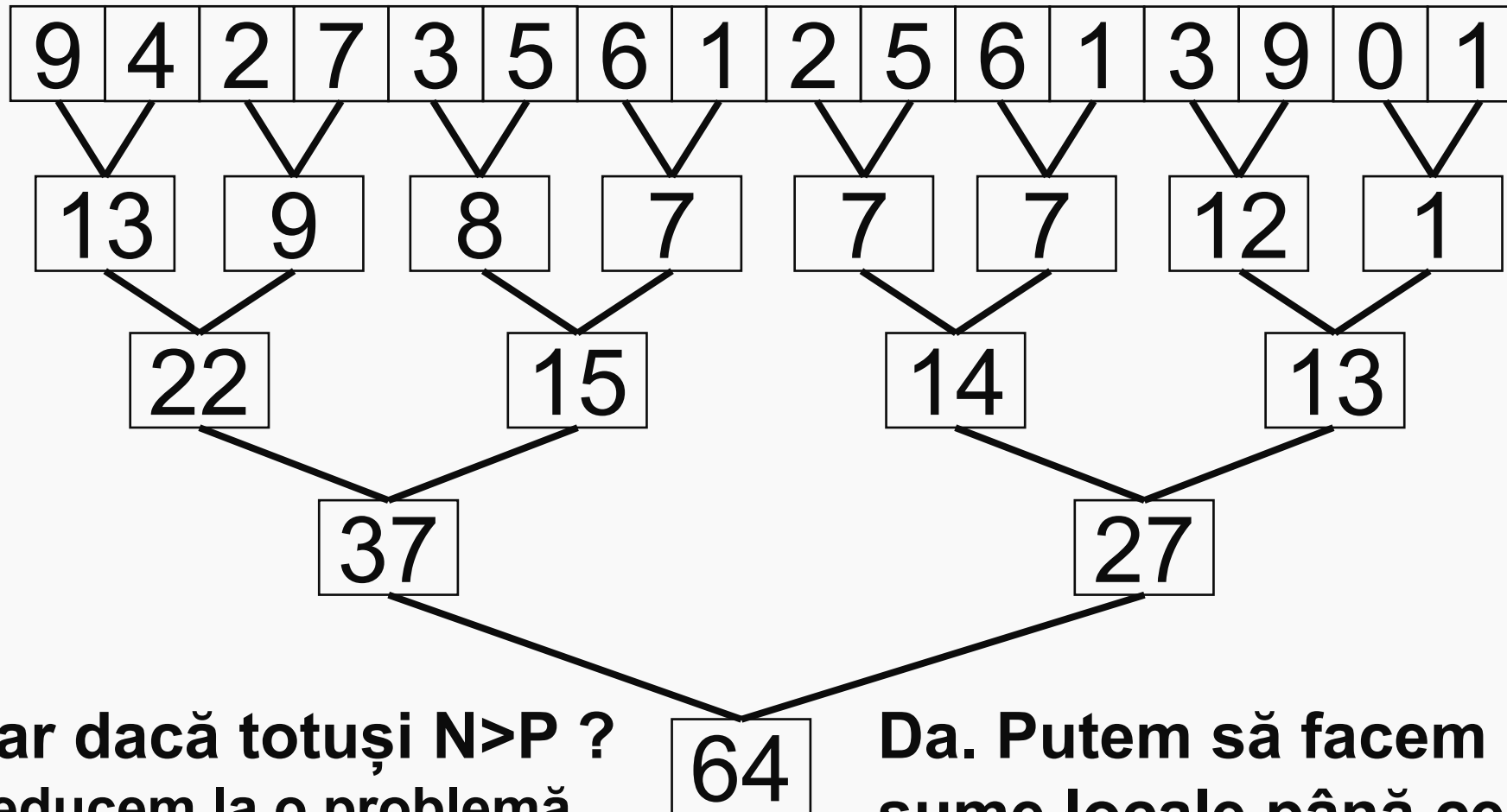
## Reduce cu sumă



**Dar dacă totuși  $N > P$  ?  
Ne mai ajută?**

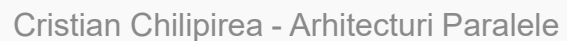


## Reduce cu sumă



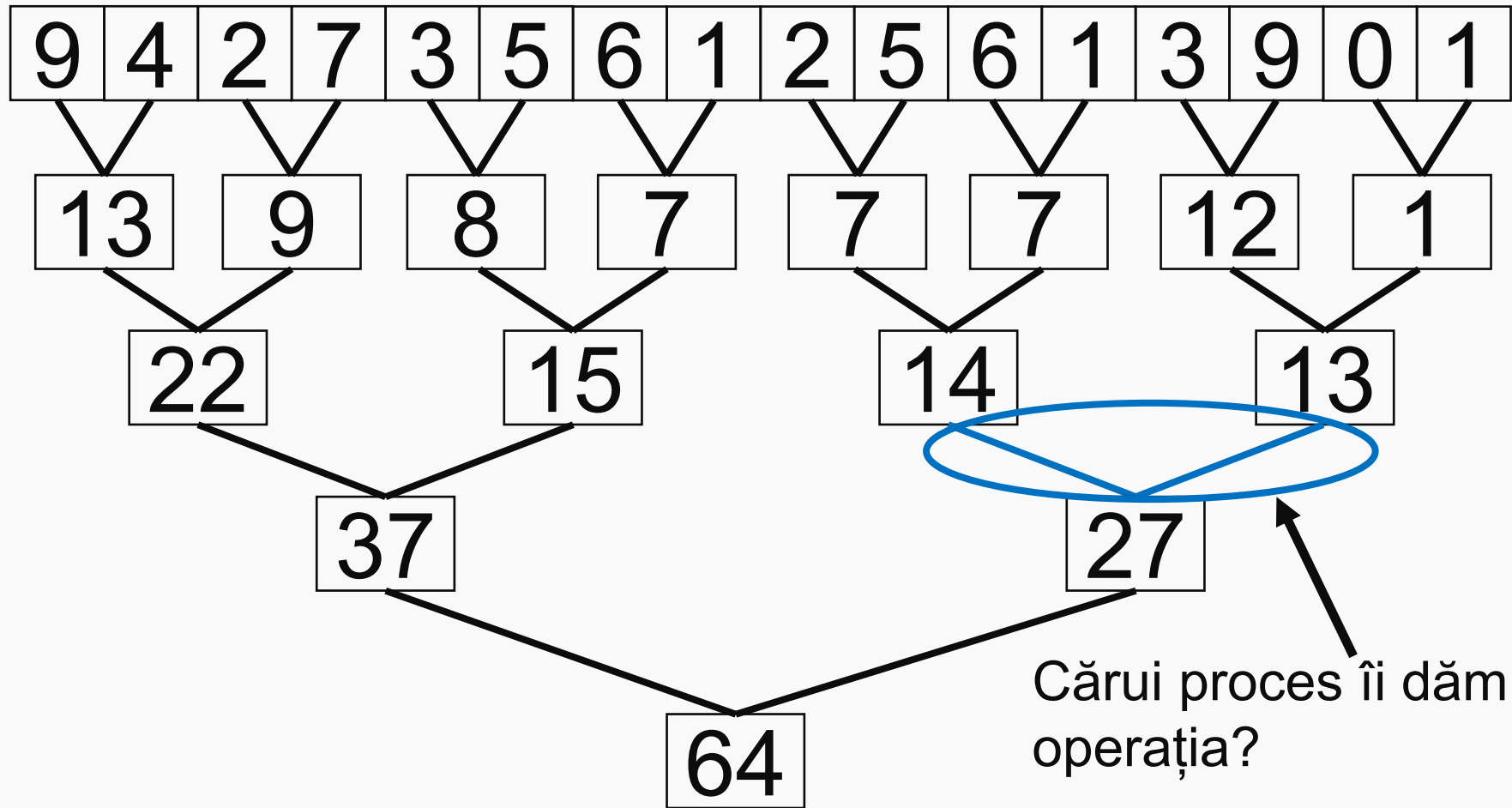
**Dar dacă totuși  $N > P$  ?**  
Reducem la o problemă  
deja rezolvată

**Da. Putem să facem**  
sume locale până ce  
avem  $P$  valori.



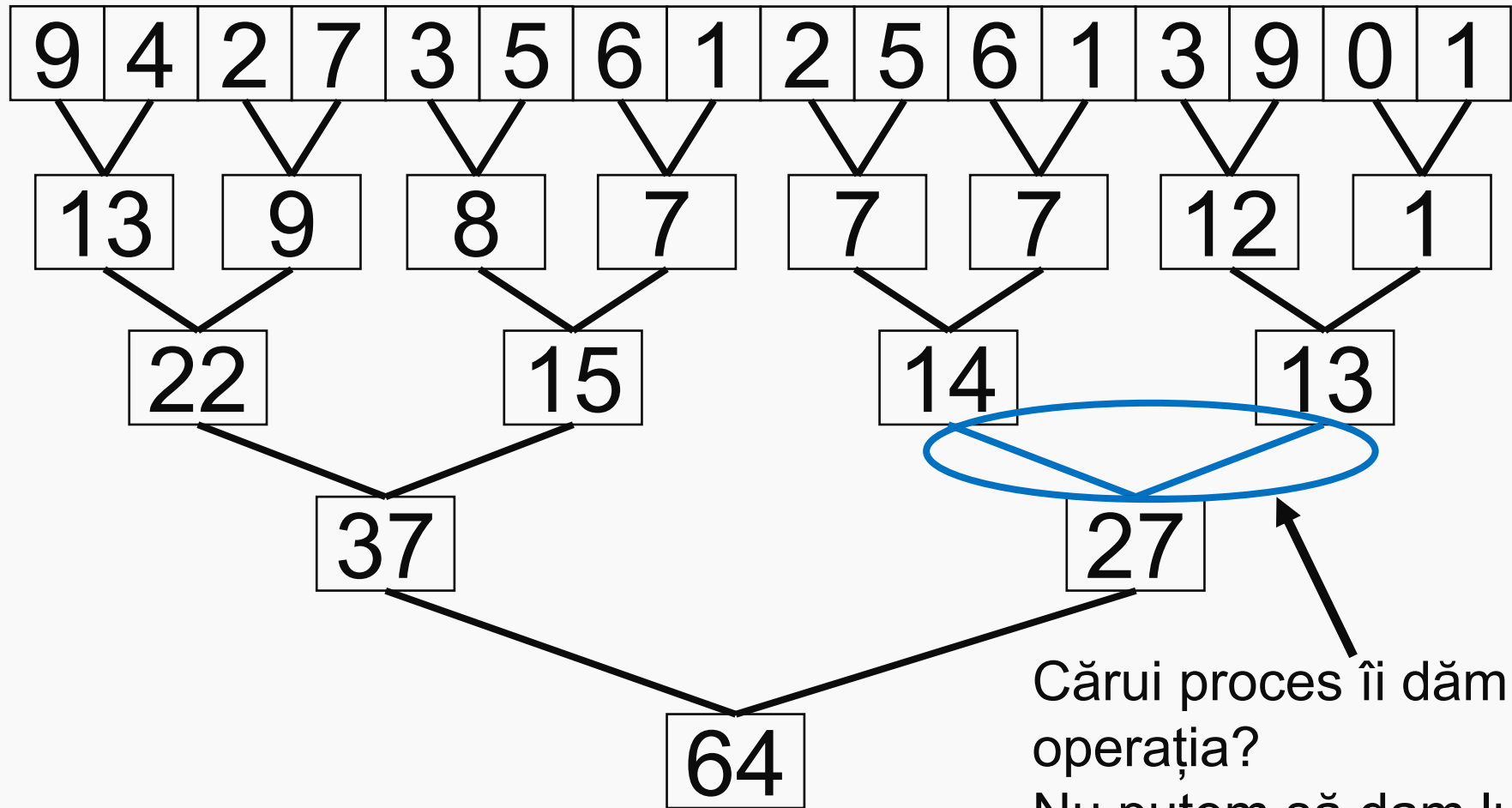


# Reduce - implementare





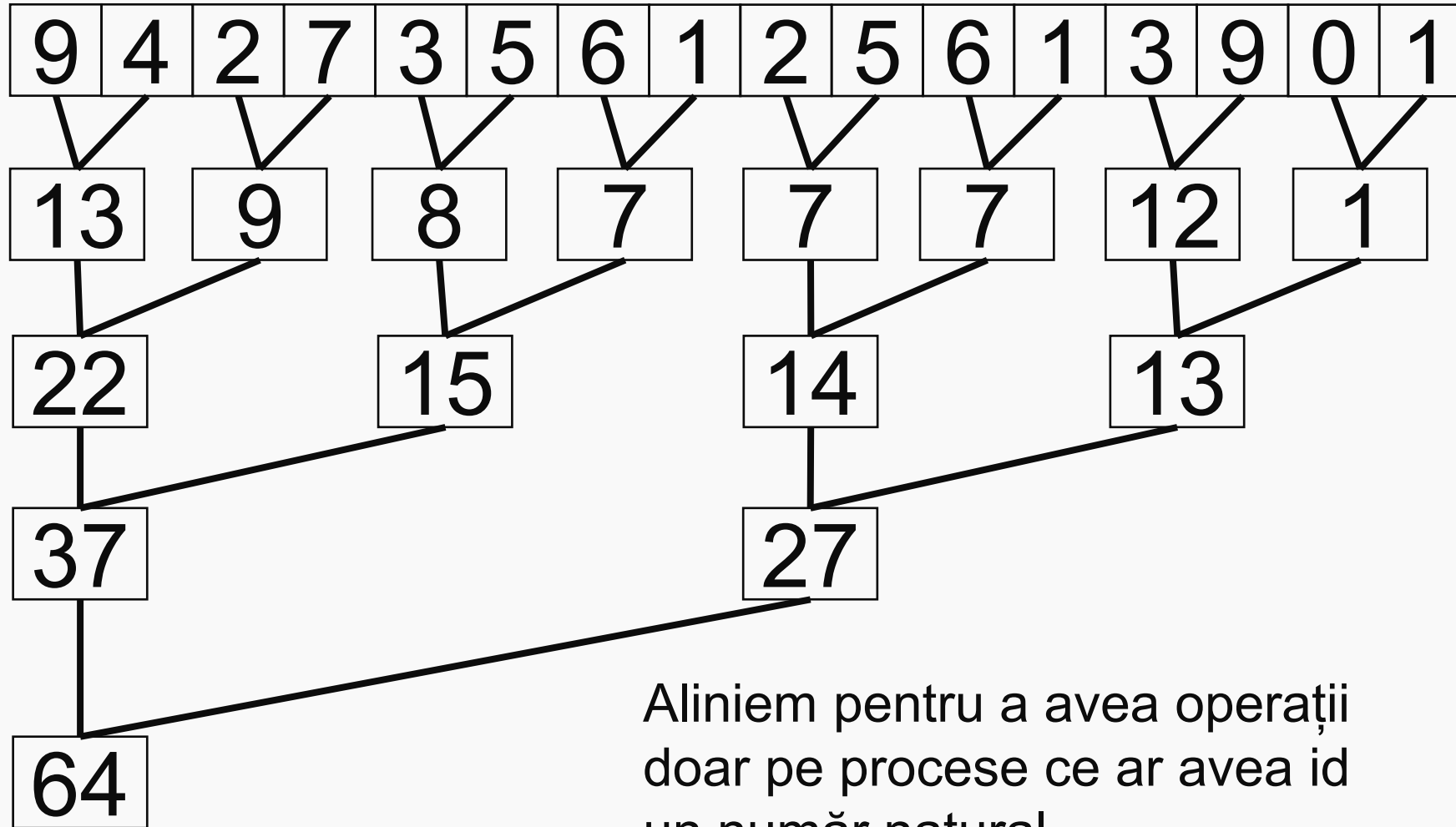
# Reduce - implementare



Căru proces îi dăm  
operația?  
Nu putem să dam lui  
12.5



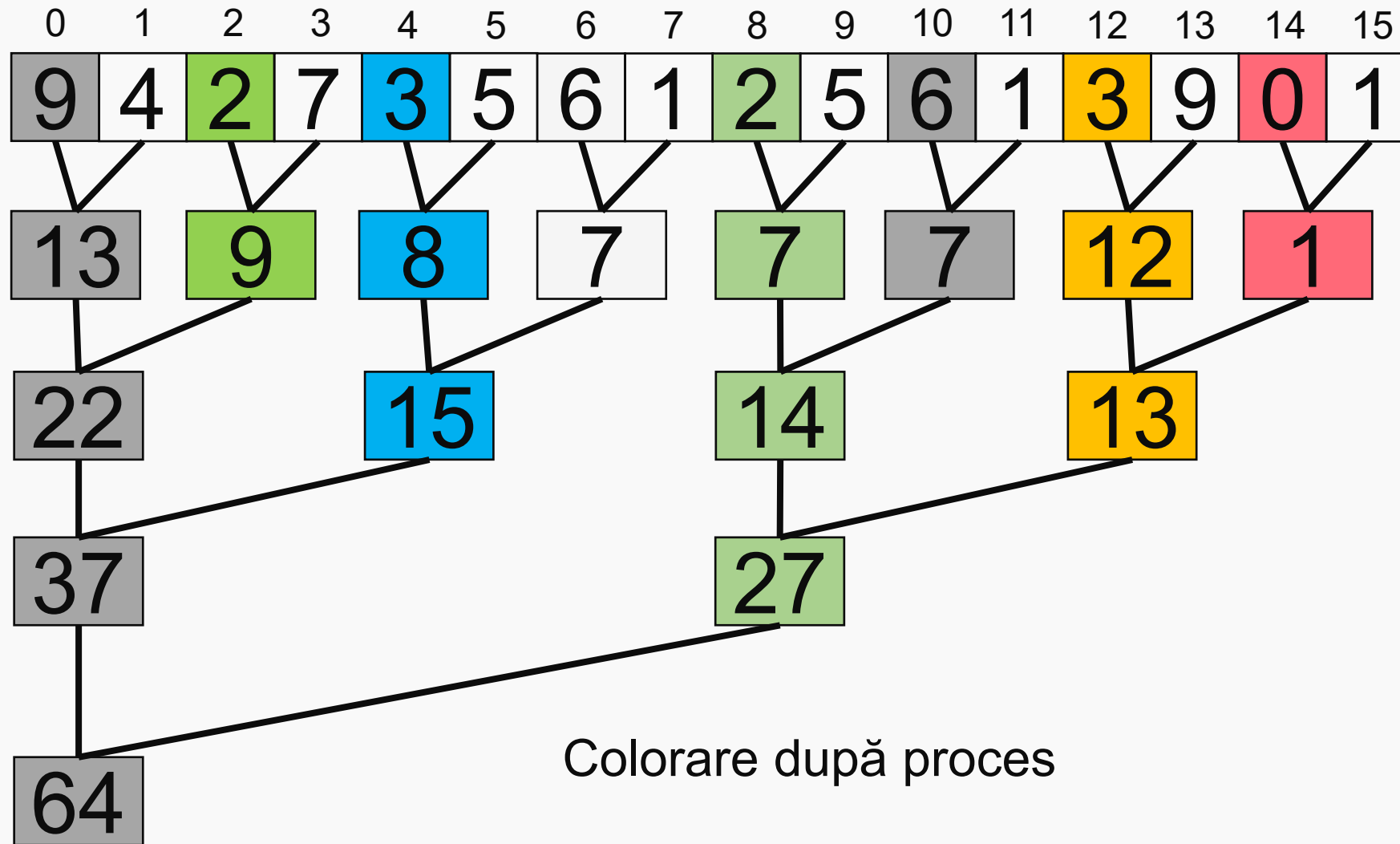
## Reduce - implementare



Aliniem pentru a avea operații  
doar pe procese ce ar avea id  
un număr natural

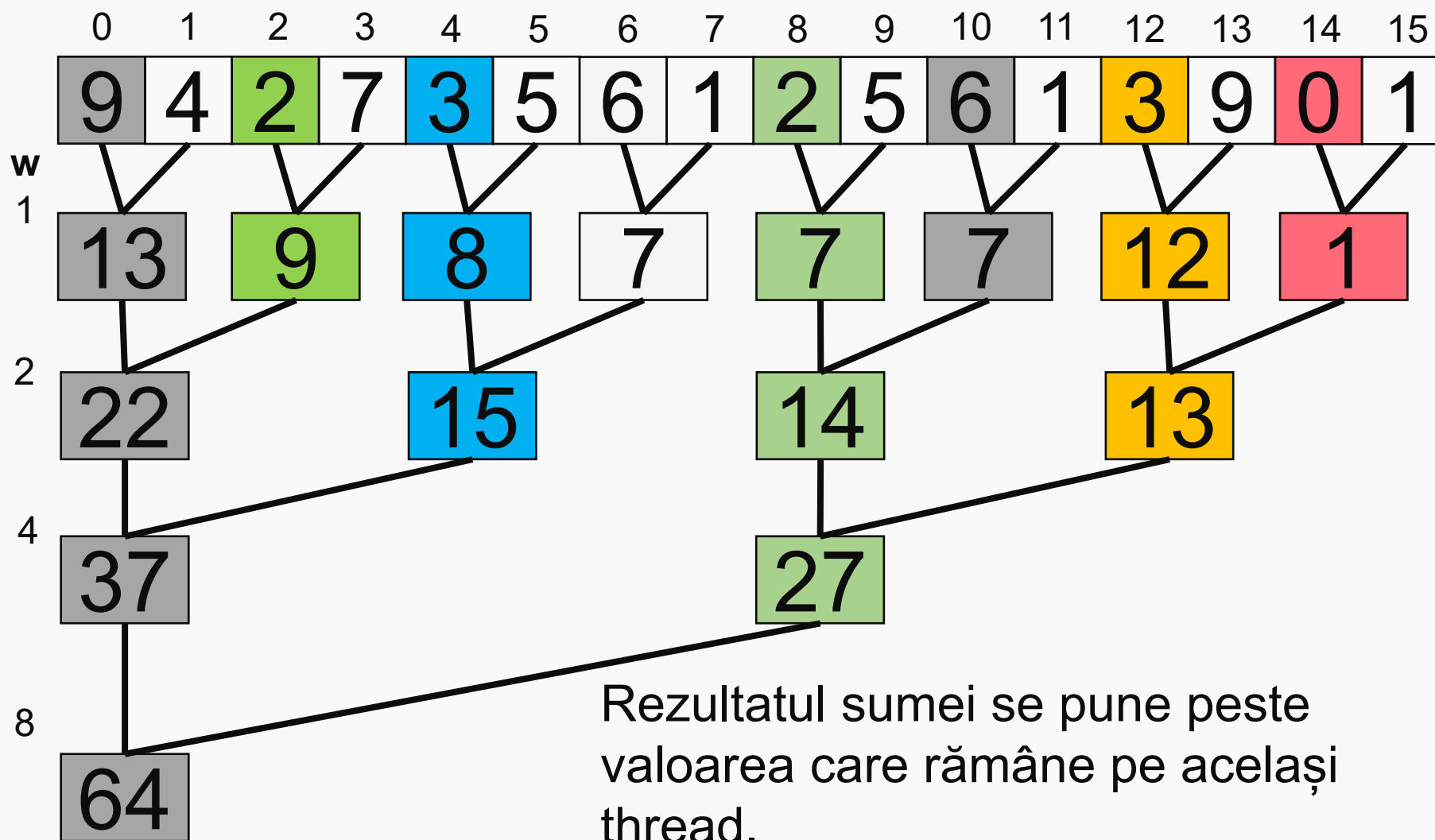


## Reduce - implementare





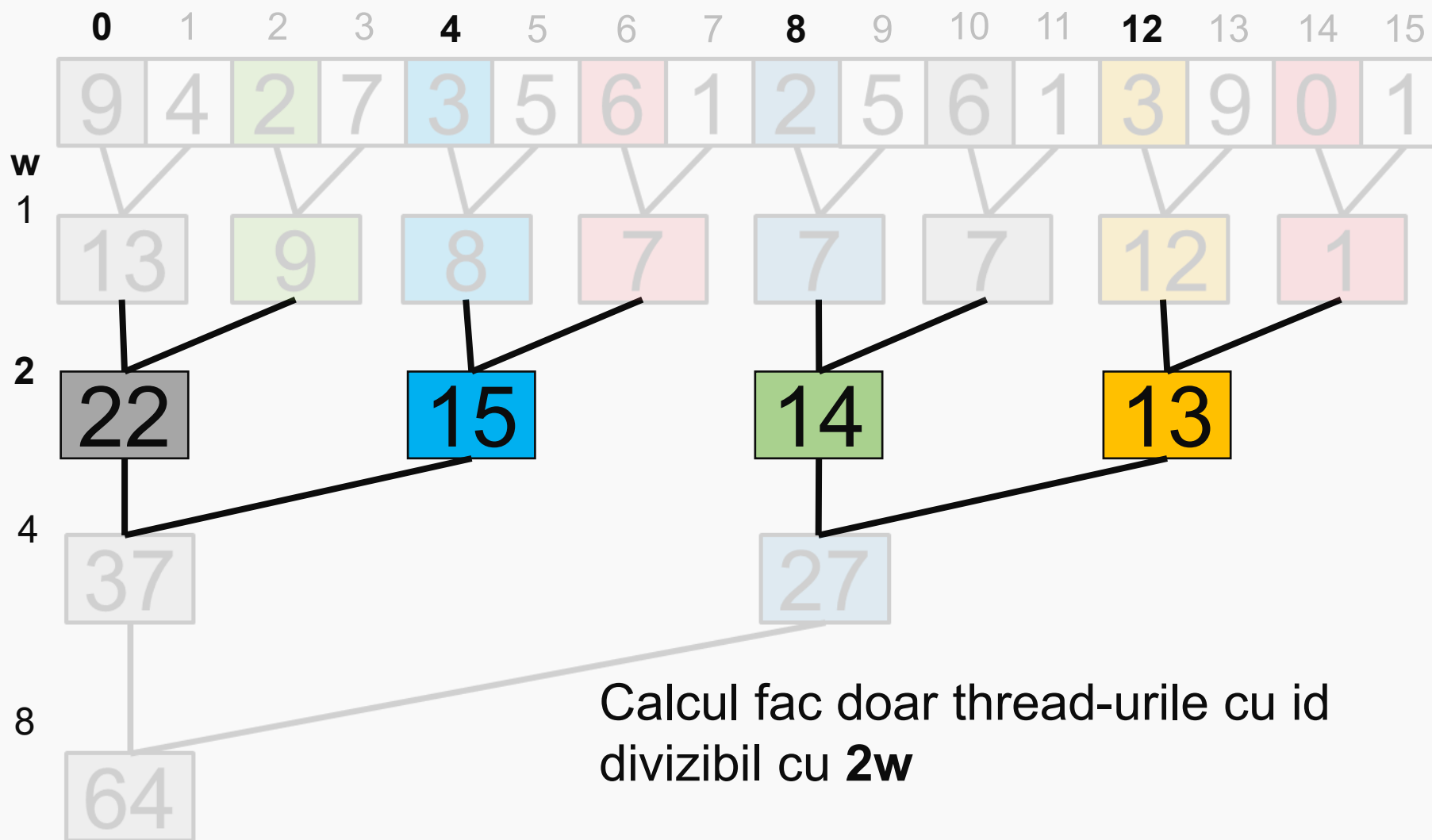
## Reduce – implementare – imagine globală





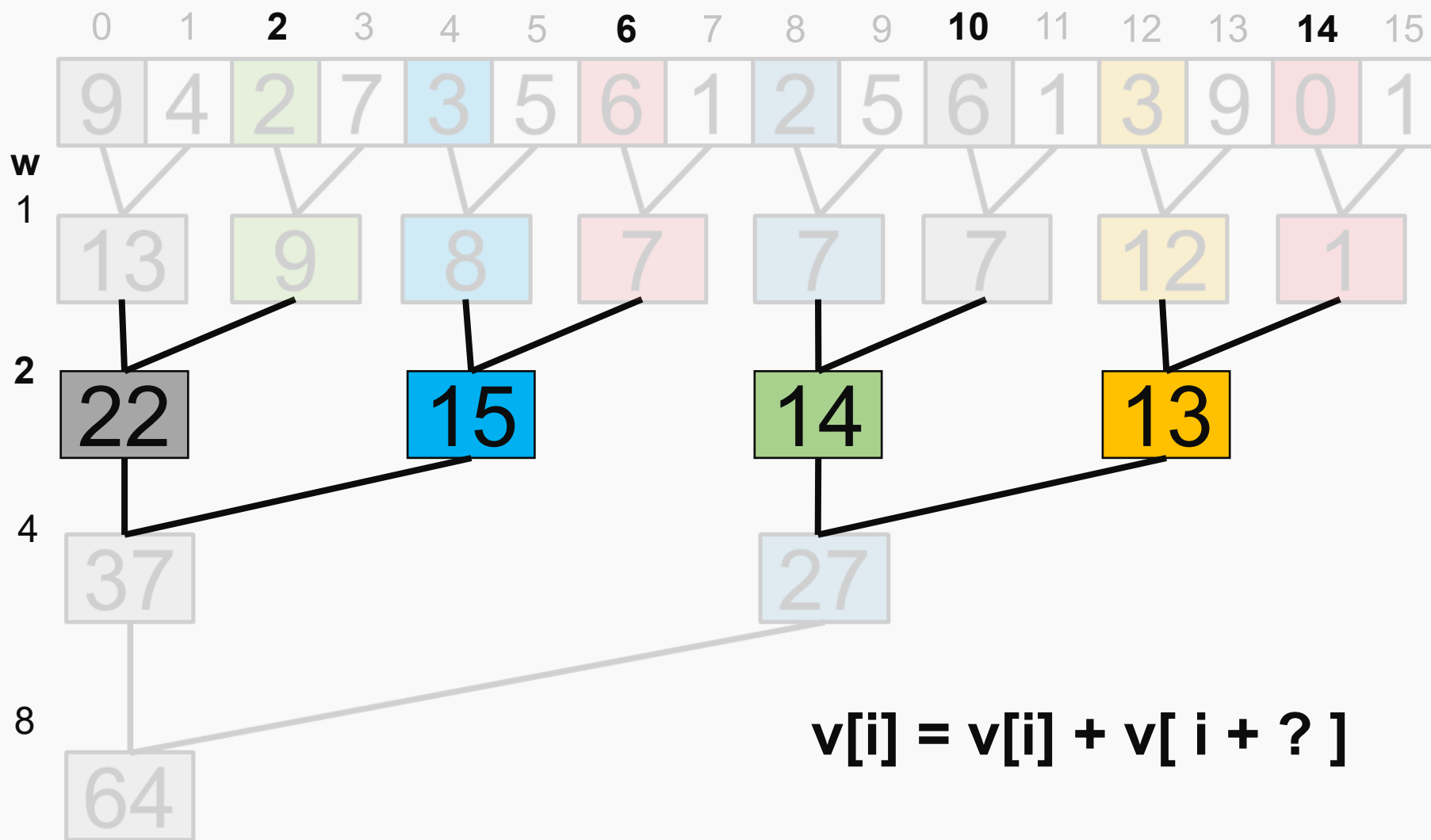


# Reduce – implementare – imagine globală





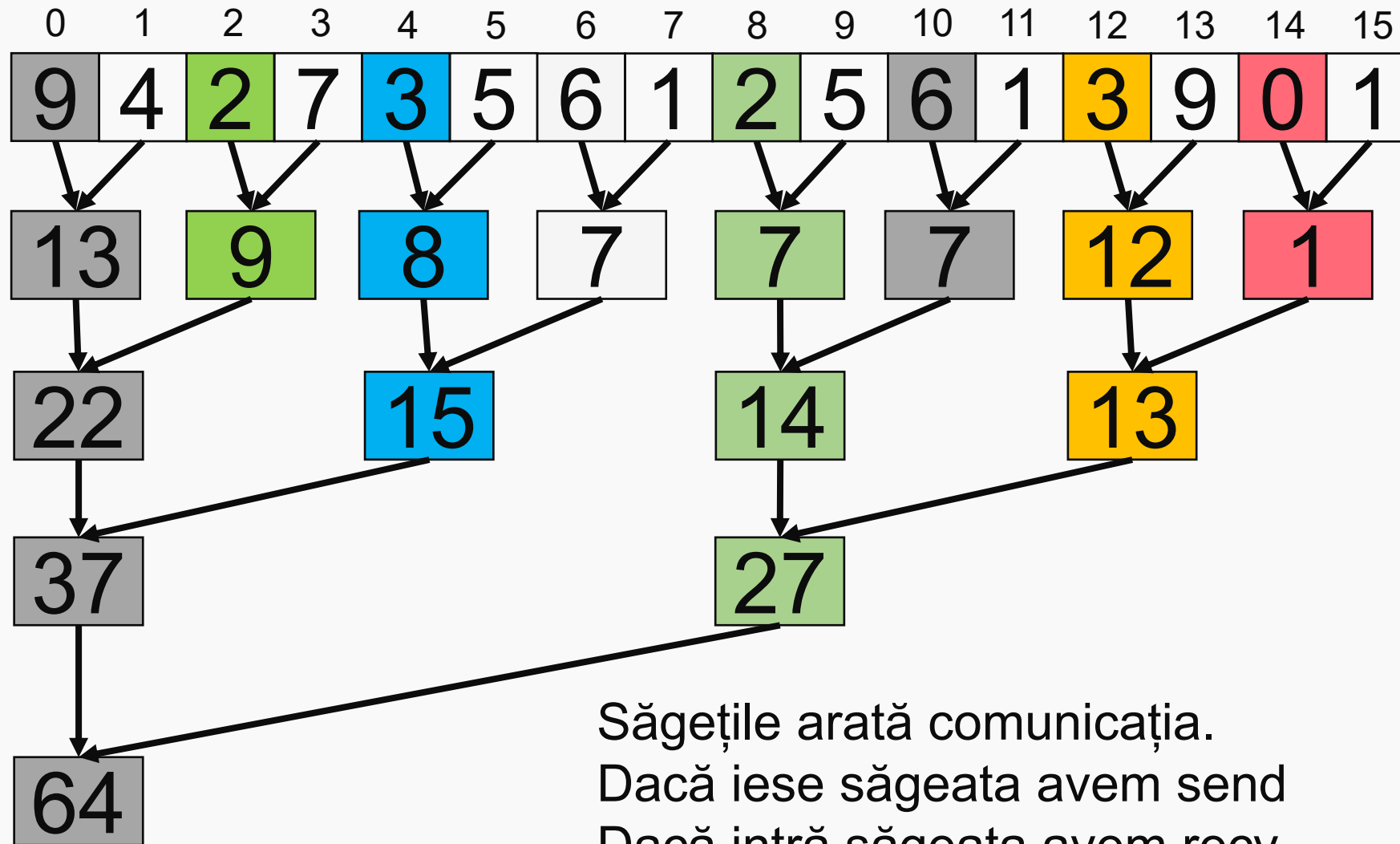
# Reduce – implementare – imagine globală





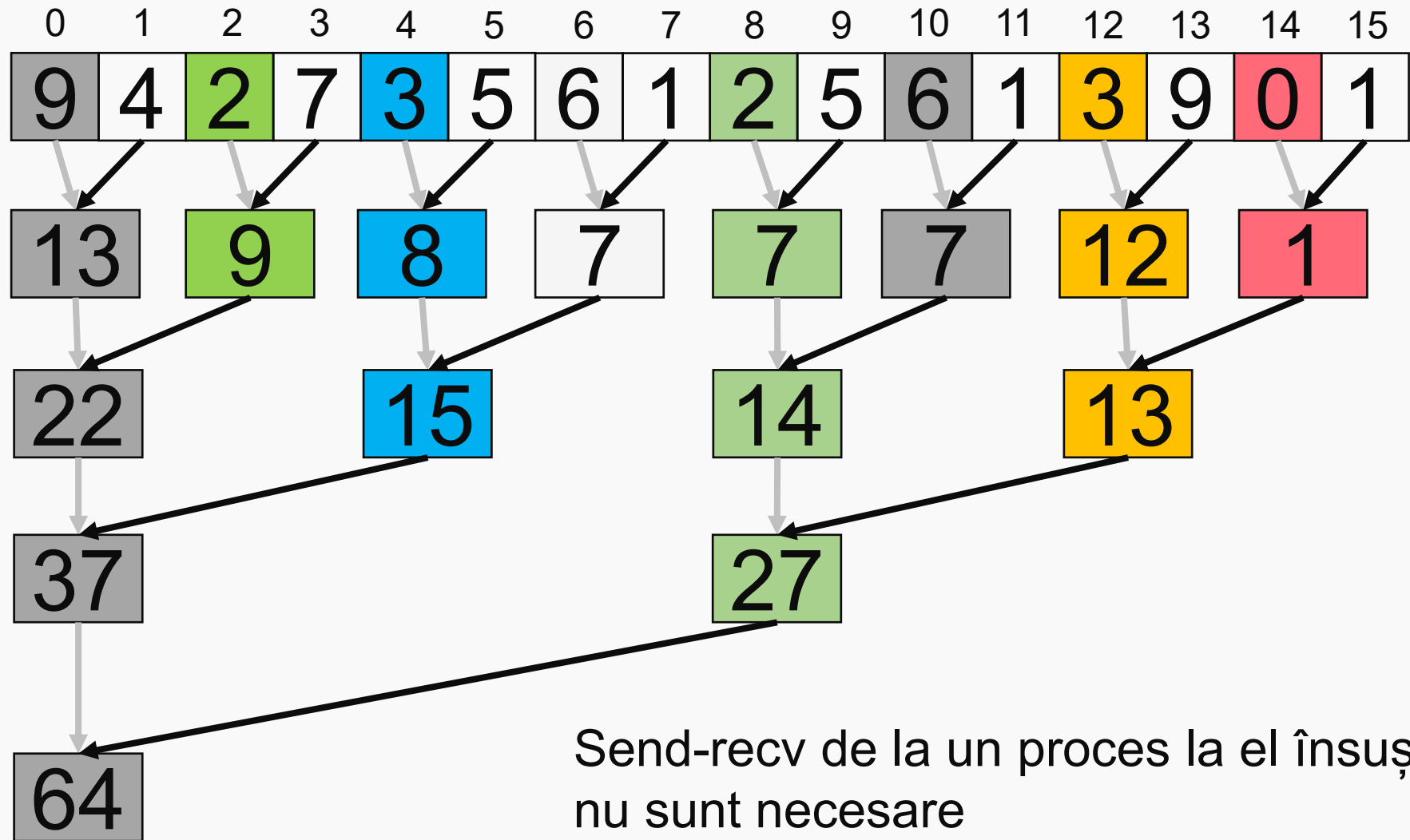


## Reduce – implementare – model comunicație



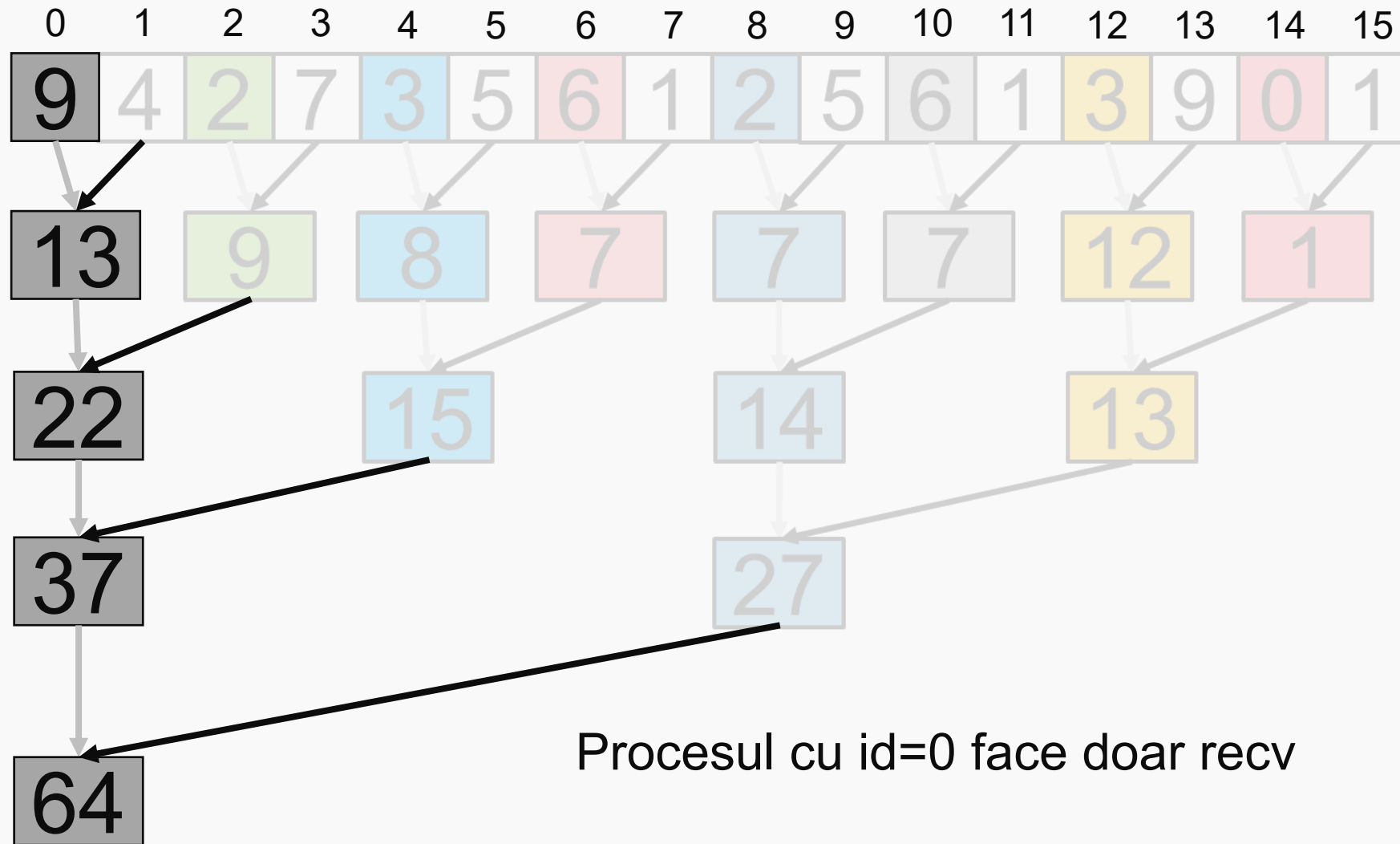


## Reduce – implementare – imagine locală



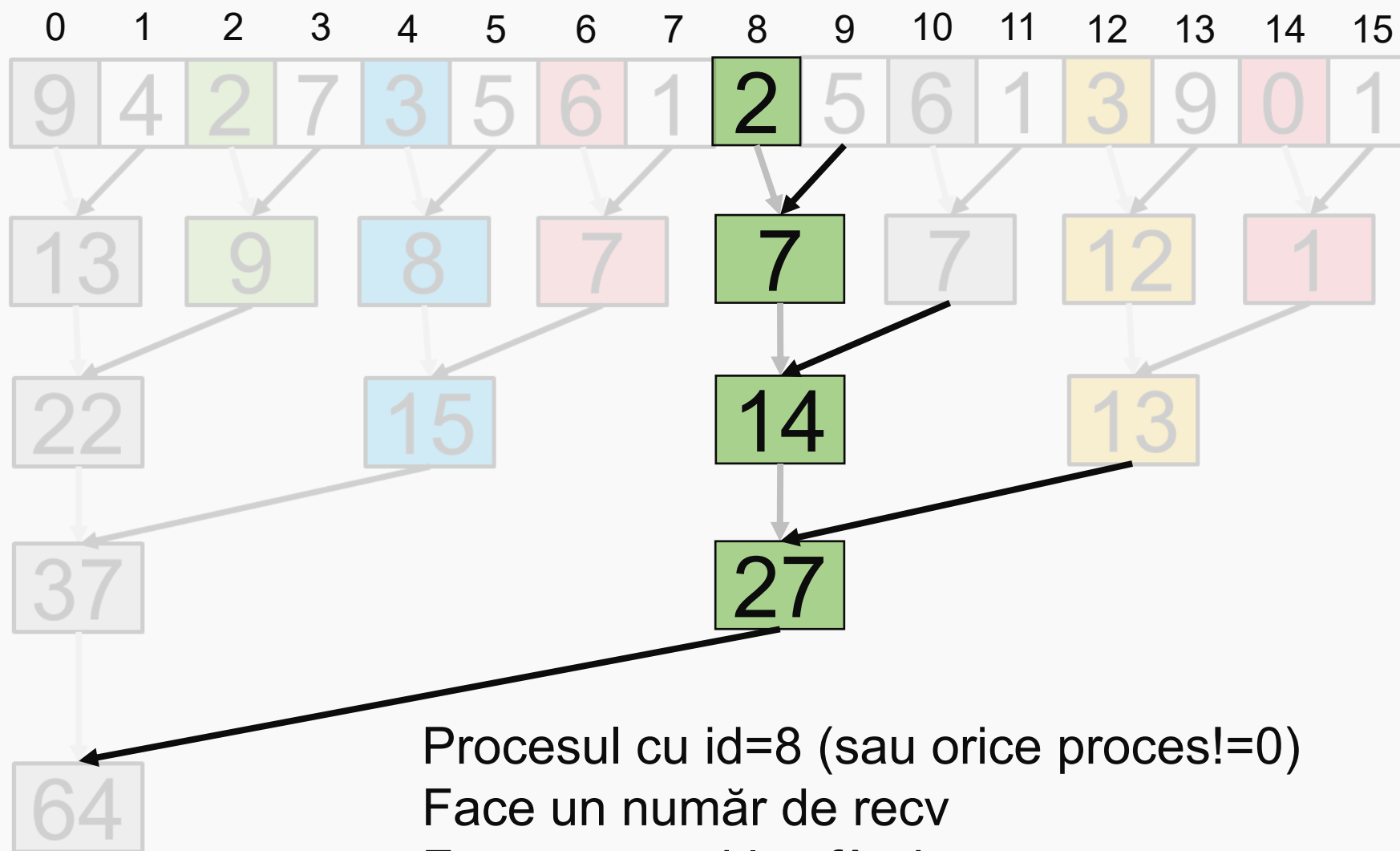


## Reduce – implementare – imagine locală





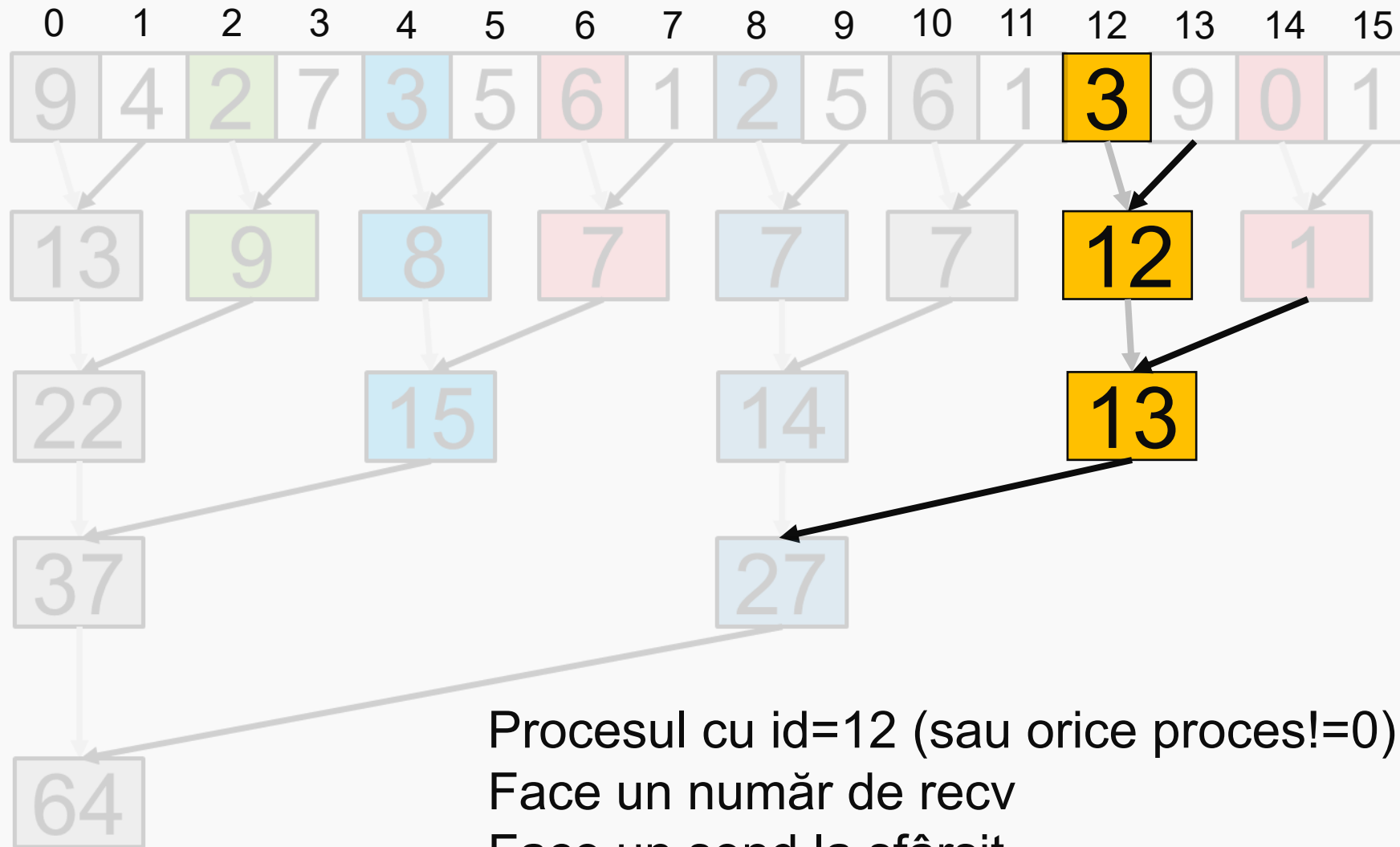
## Reduce – implementare – imagine locală



Procesul cu id=8 (sau orice proces!=0)  
Face un număr de recv  
Face un send la sfârșit



## Reduce – implementare – imagine locală

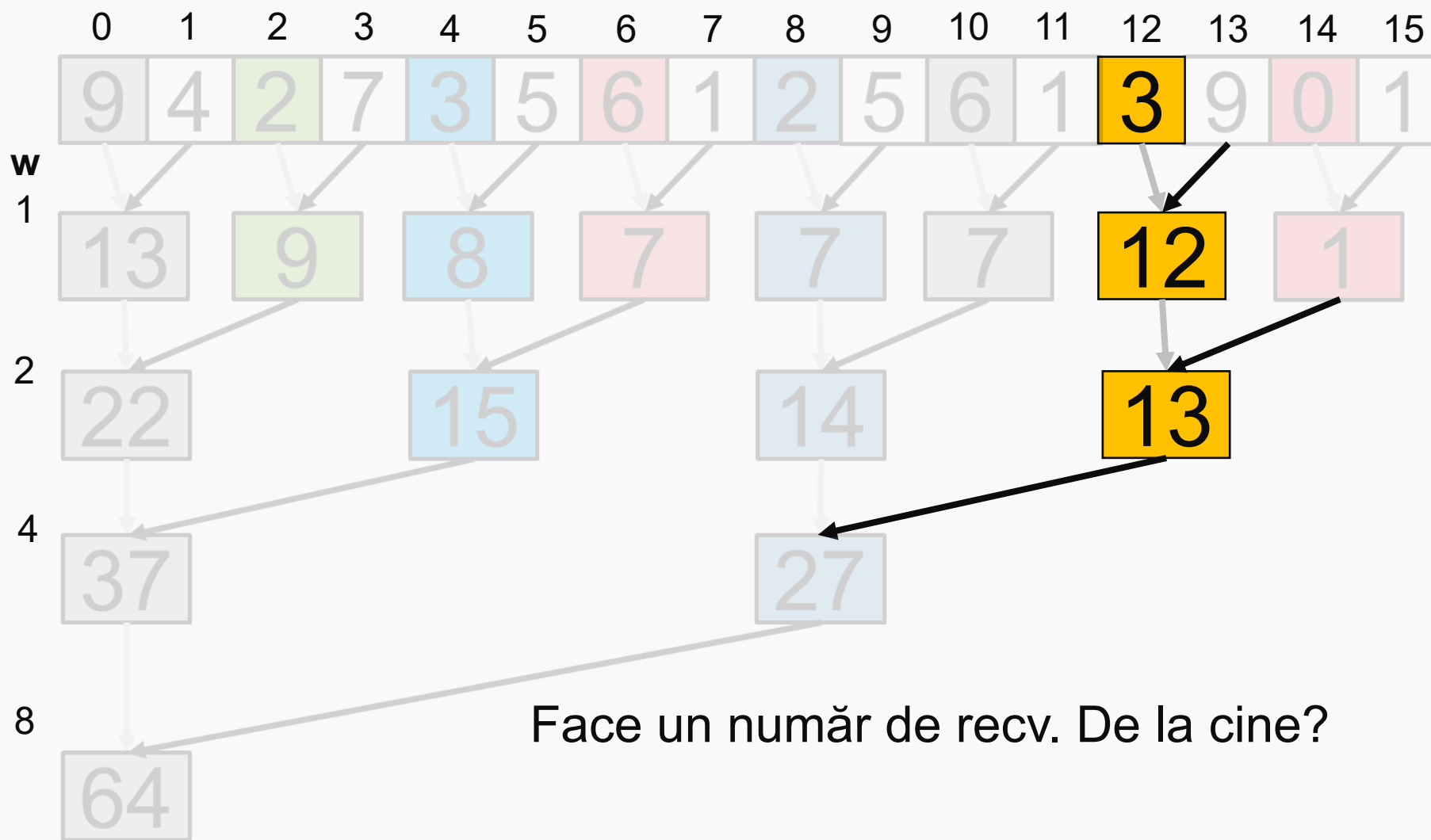


Procesul cu id=12 (sau orice proces!=0)  
Face un număr de recv  
Face un send la sfârșit





## Reduce – implementare – imagine locală





## Reduce – implementare – imagine locală



Face un număr de recv. De la cine?  
De la  $id + w$   
Cât timp?



## Reduce – implementare – imagine locală



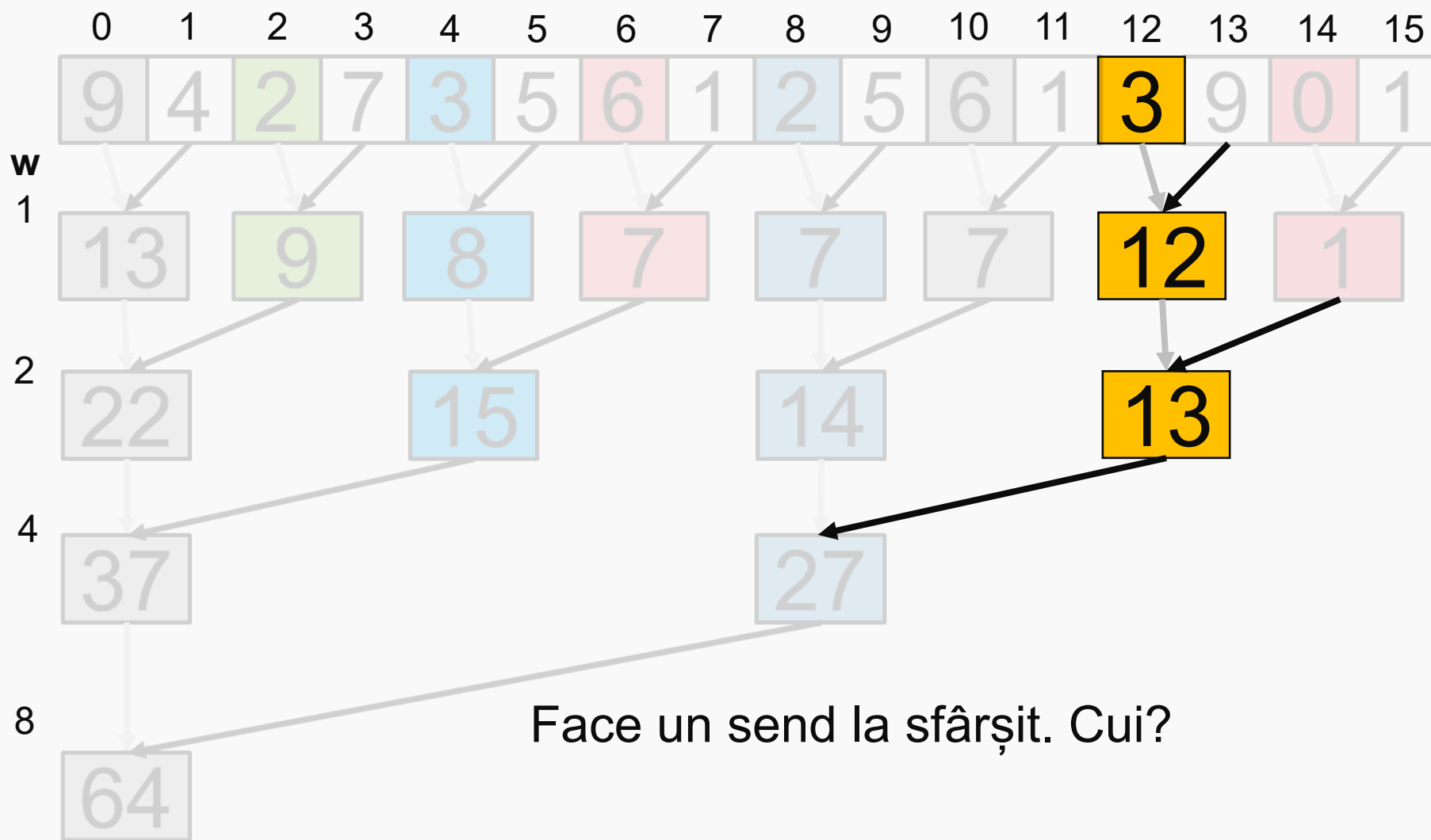
Face un număr de recv. De la cine?

De la **id + w**

Cât timp? Până ce id nu divizibil cu **2w**

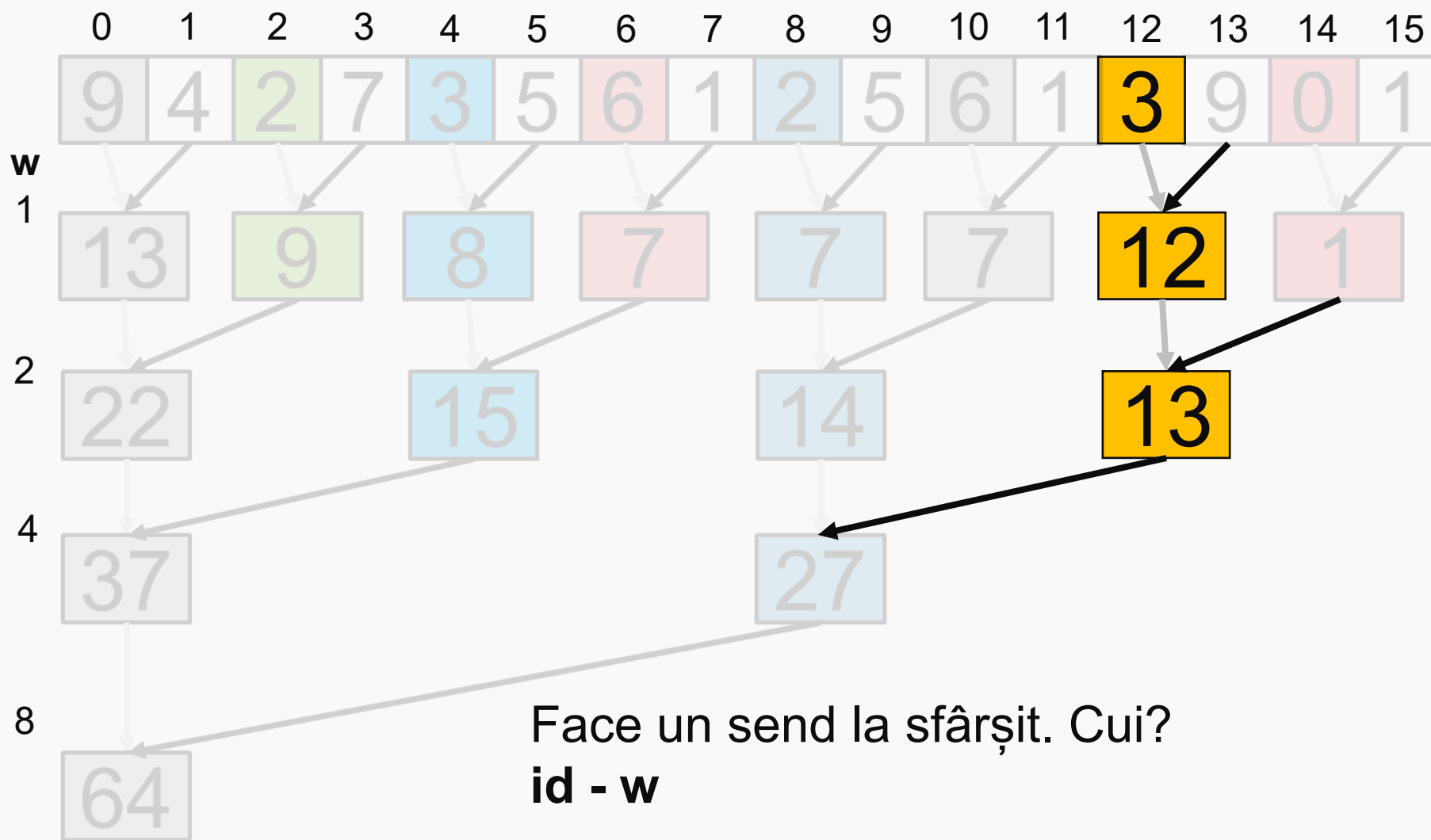


## Reduce – implementare – imagine locală





## Reduce – implementare – imagine locală







# Scan

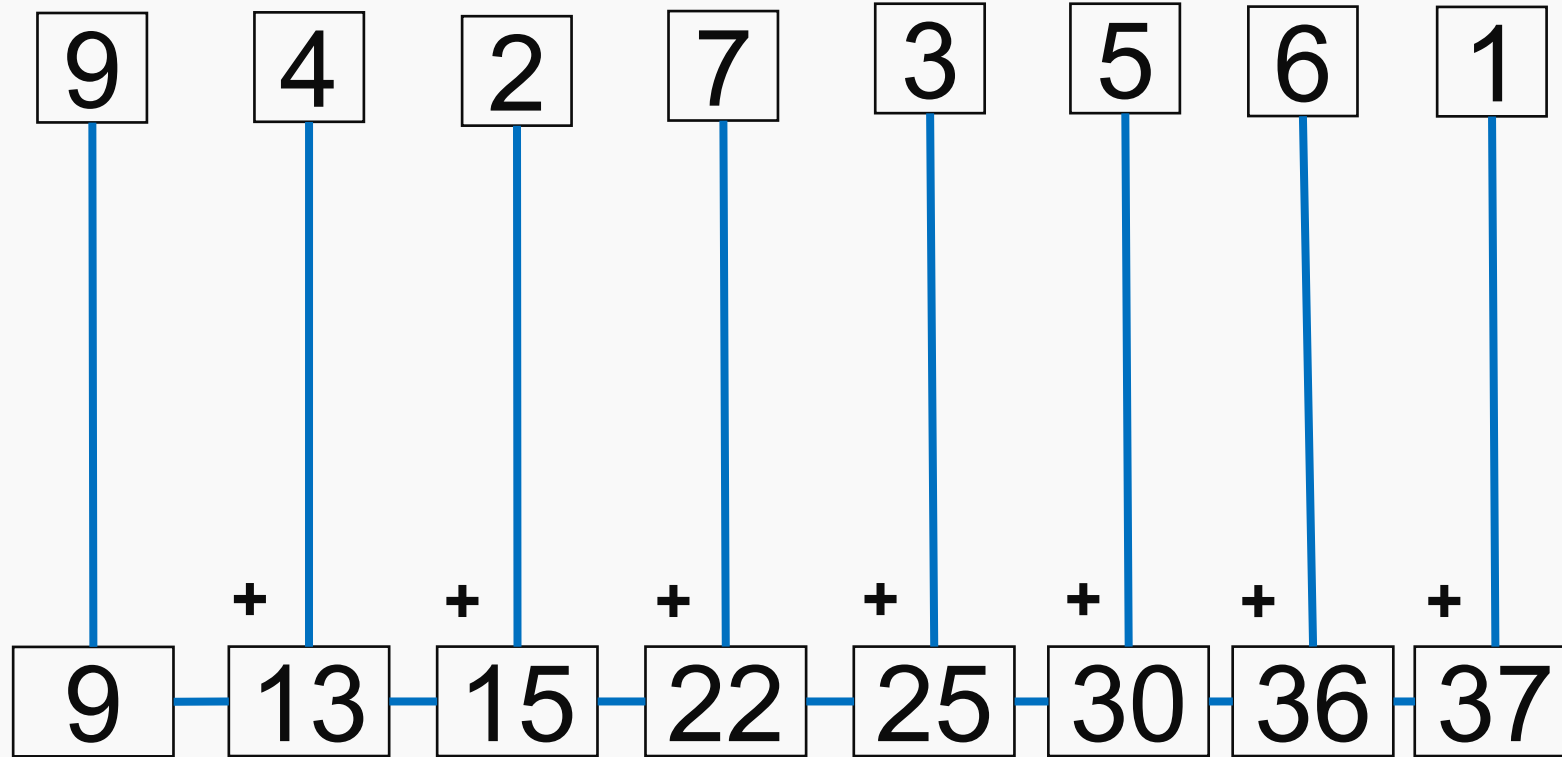
**Similar cu reduce dar se păstrează toate rezultatele intermediare.**

**Se poate executa în  $O(\log(N))$  pe  $N$  procesoare organizând calculele într-o formă arborescentă.**

**Operația trebuie să fie comutativă de exemplu:  
+, \*, min, max, and, ...**



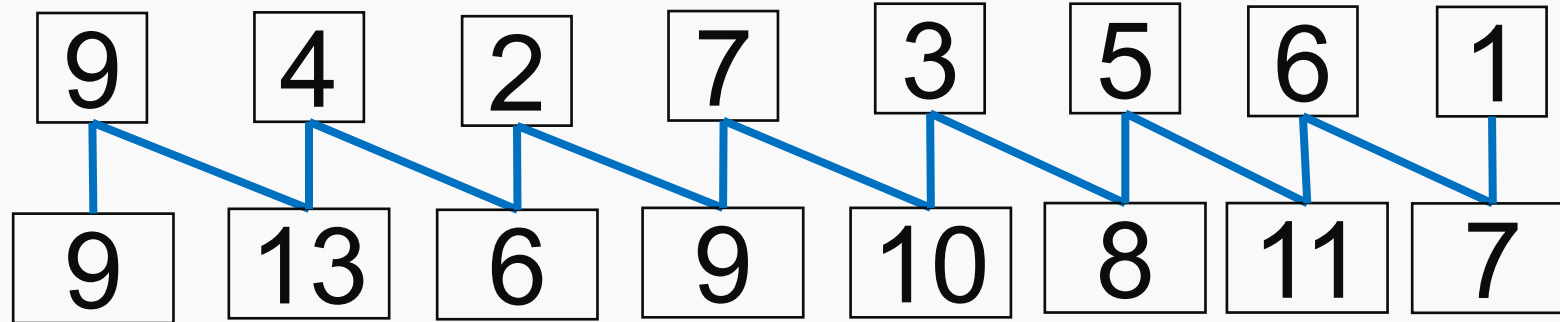
## Scan cu sumă







## Scan cu sumă

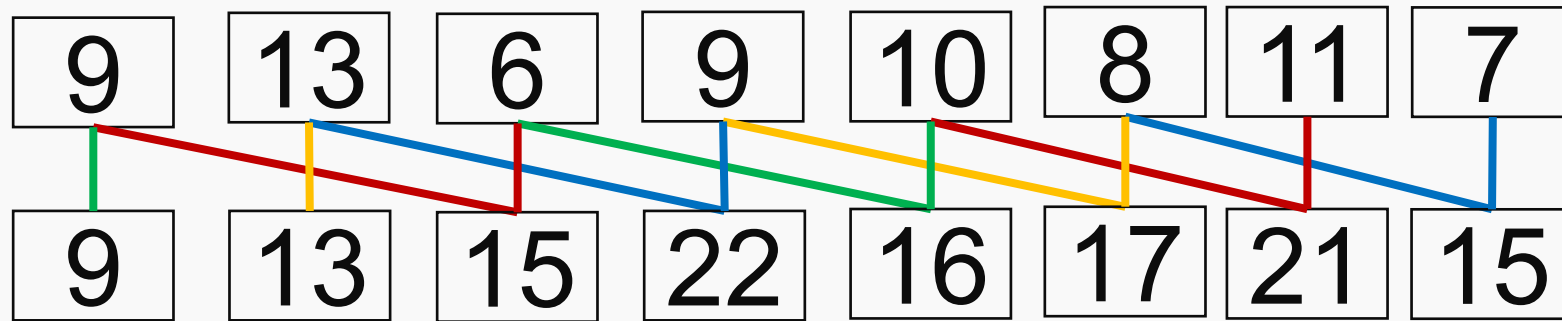


Pot fi executate în paralel





# Scan cu sumă

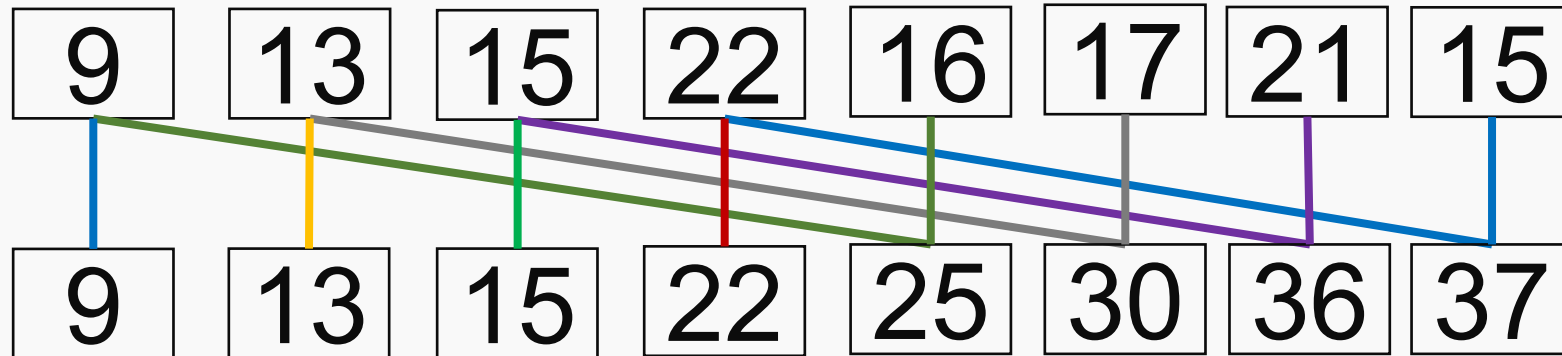


Pot fi executate în paralel





# Scan cu sumă

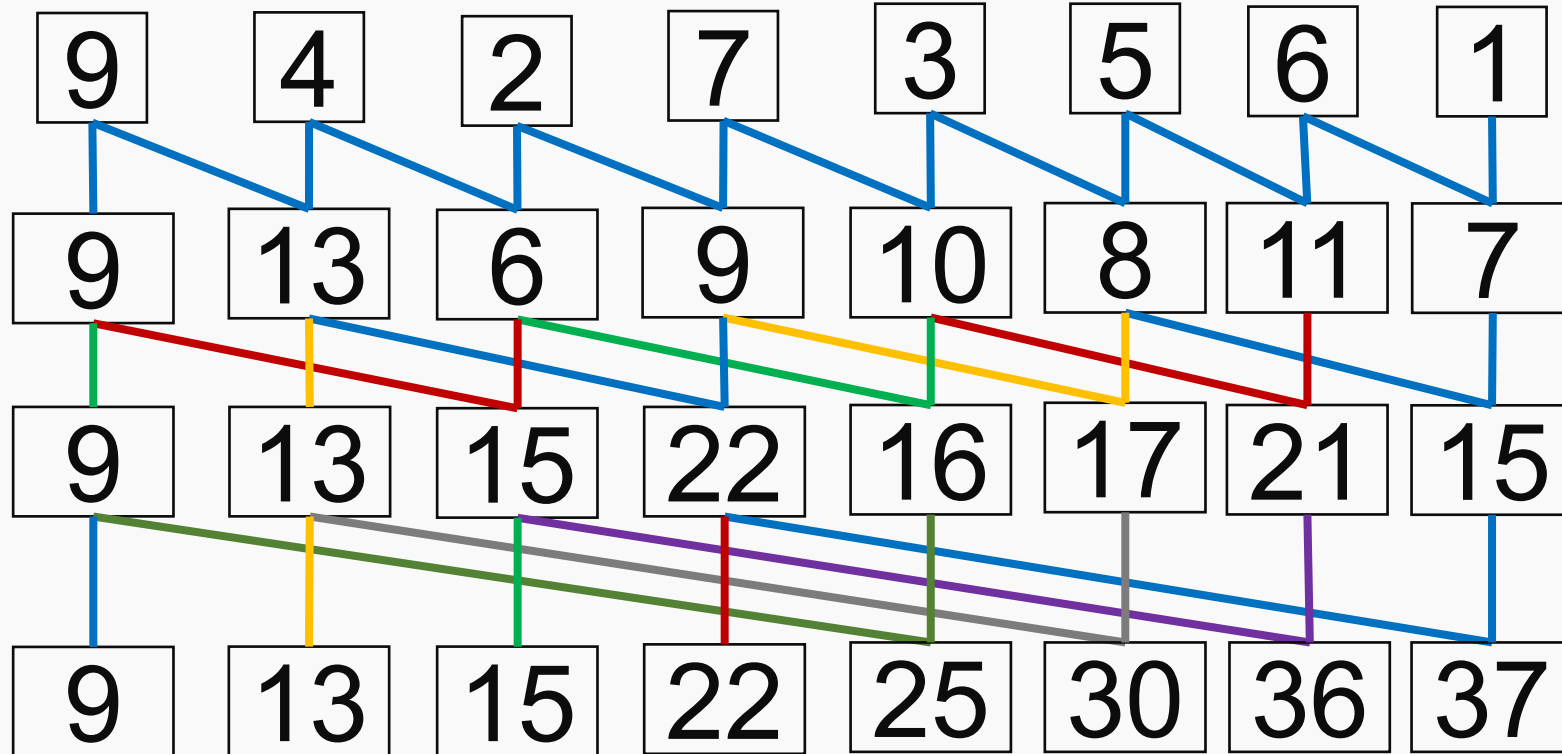


Pot fi executate în paralel



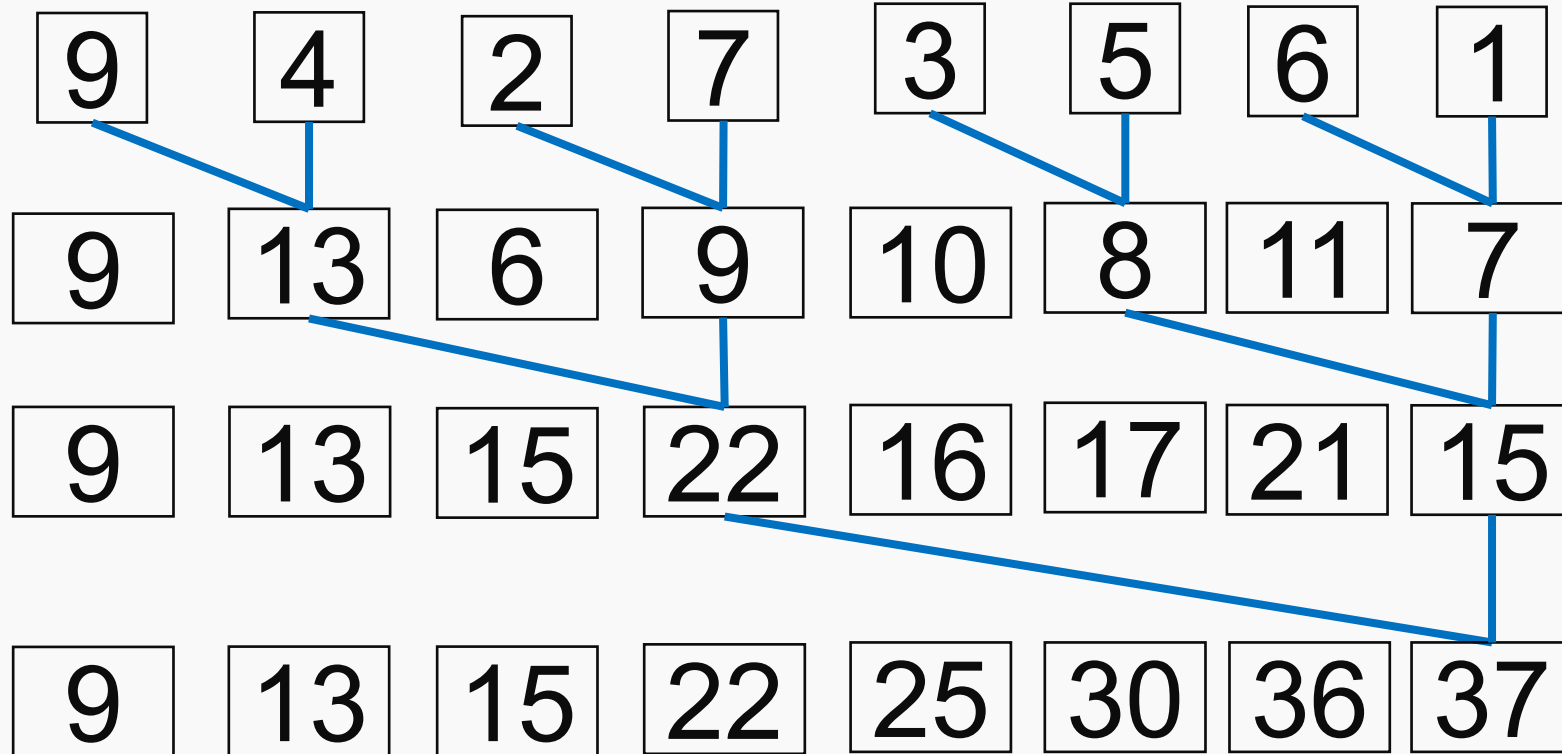


# Scan – Cum funcționează?



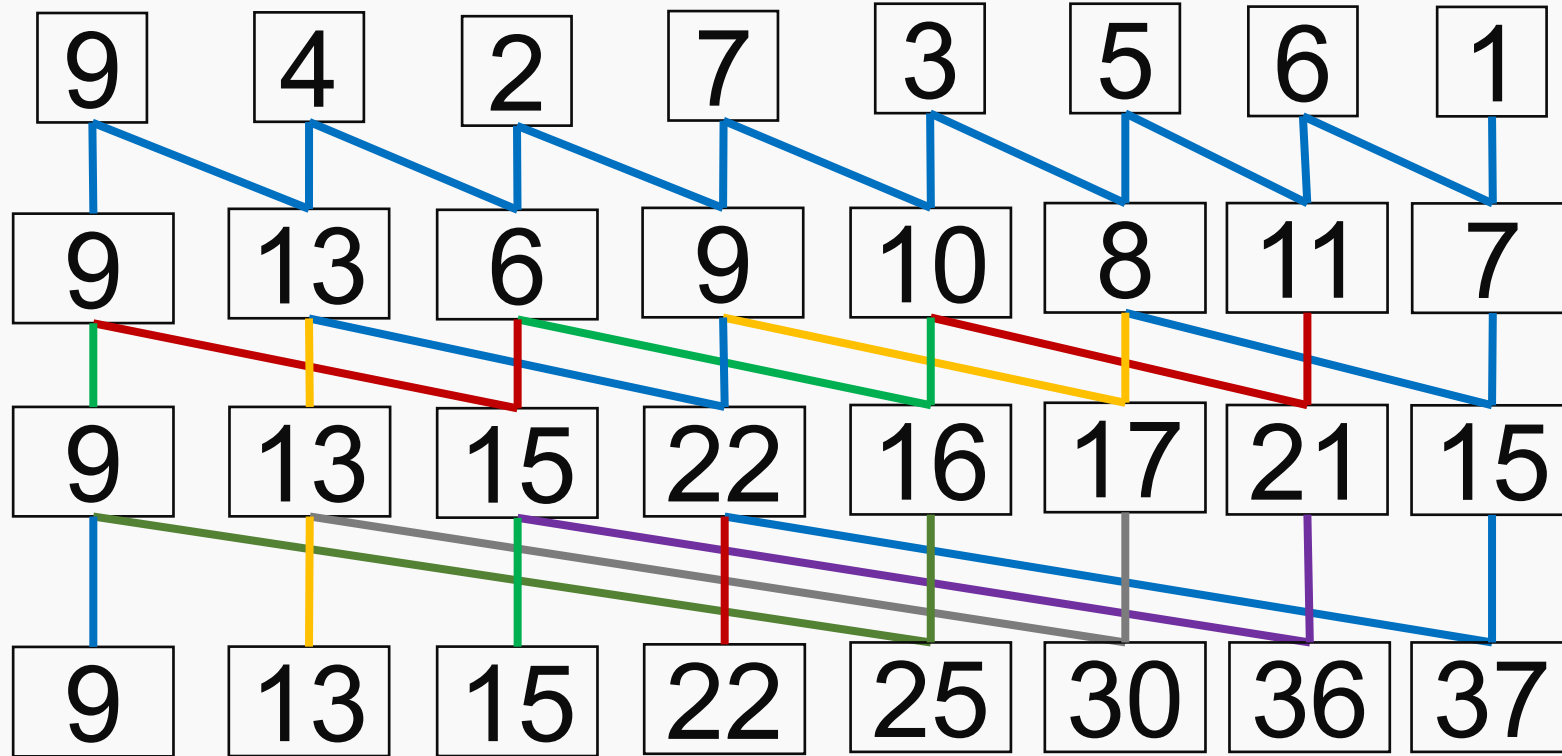


## Scan – Cum funcționează?



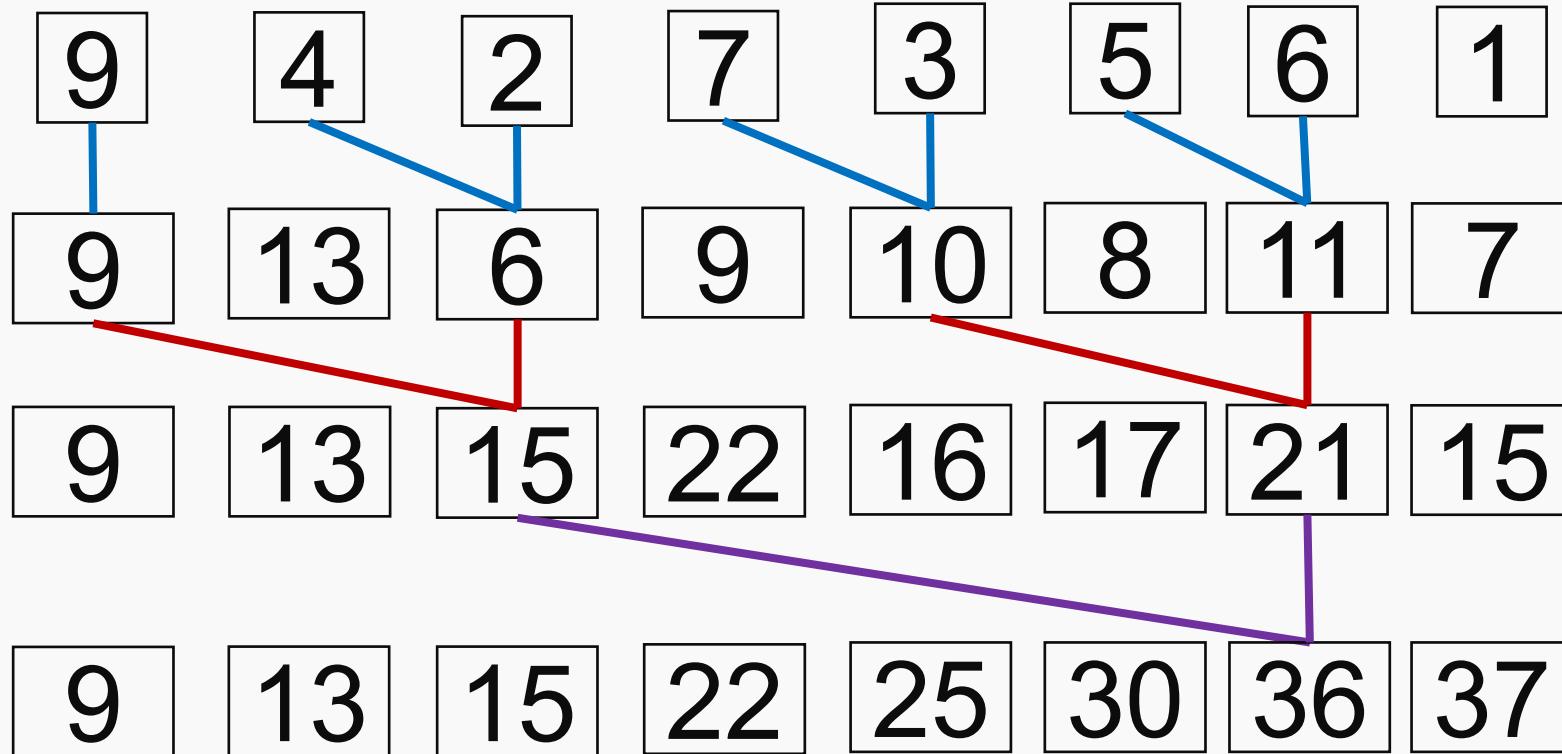


# Scan – Cum funcționează?



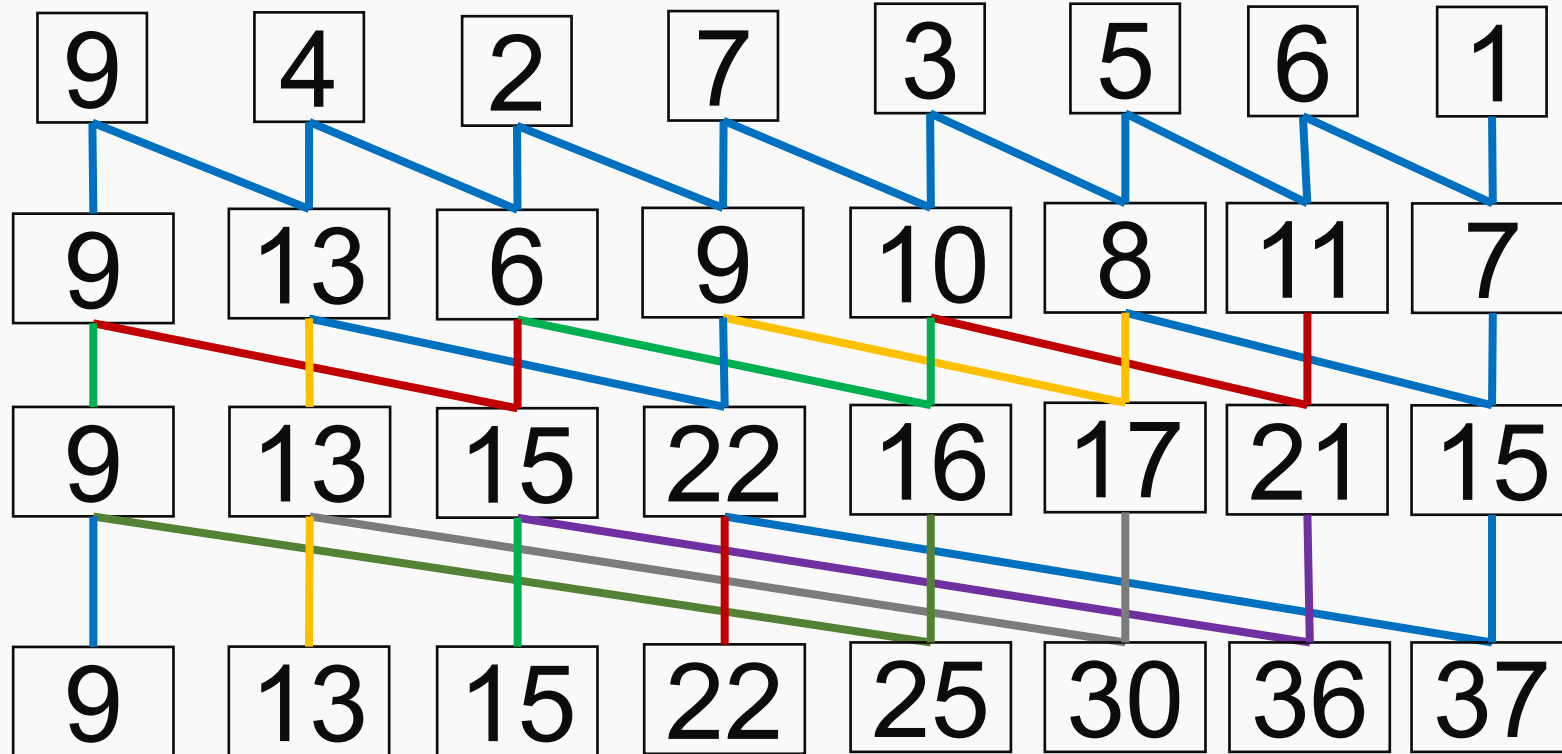


## Scan – Cum funcționează?





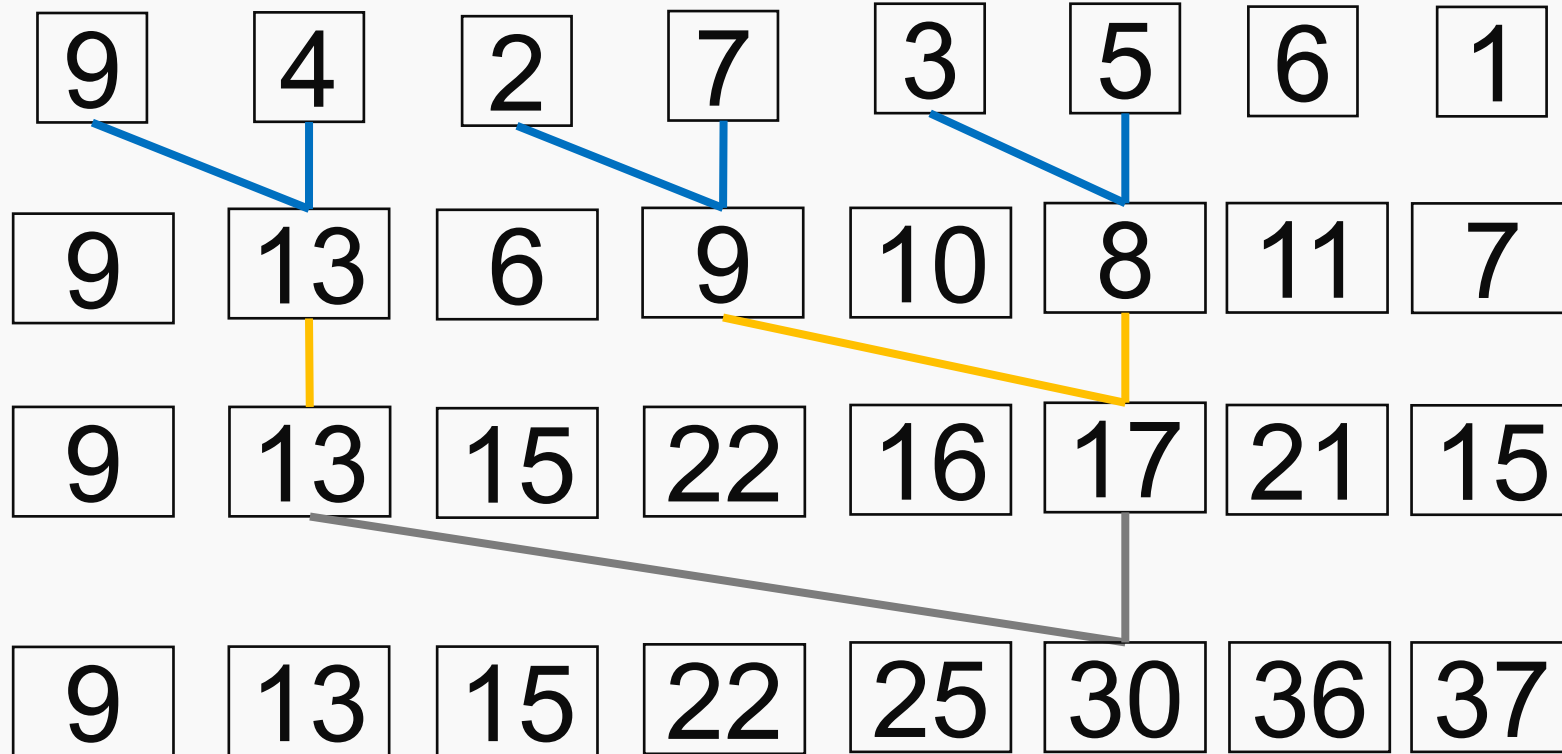
# Scan – Cum funcționează?





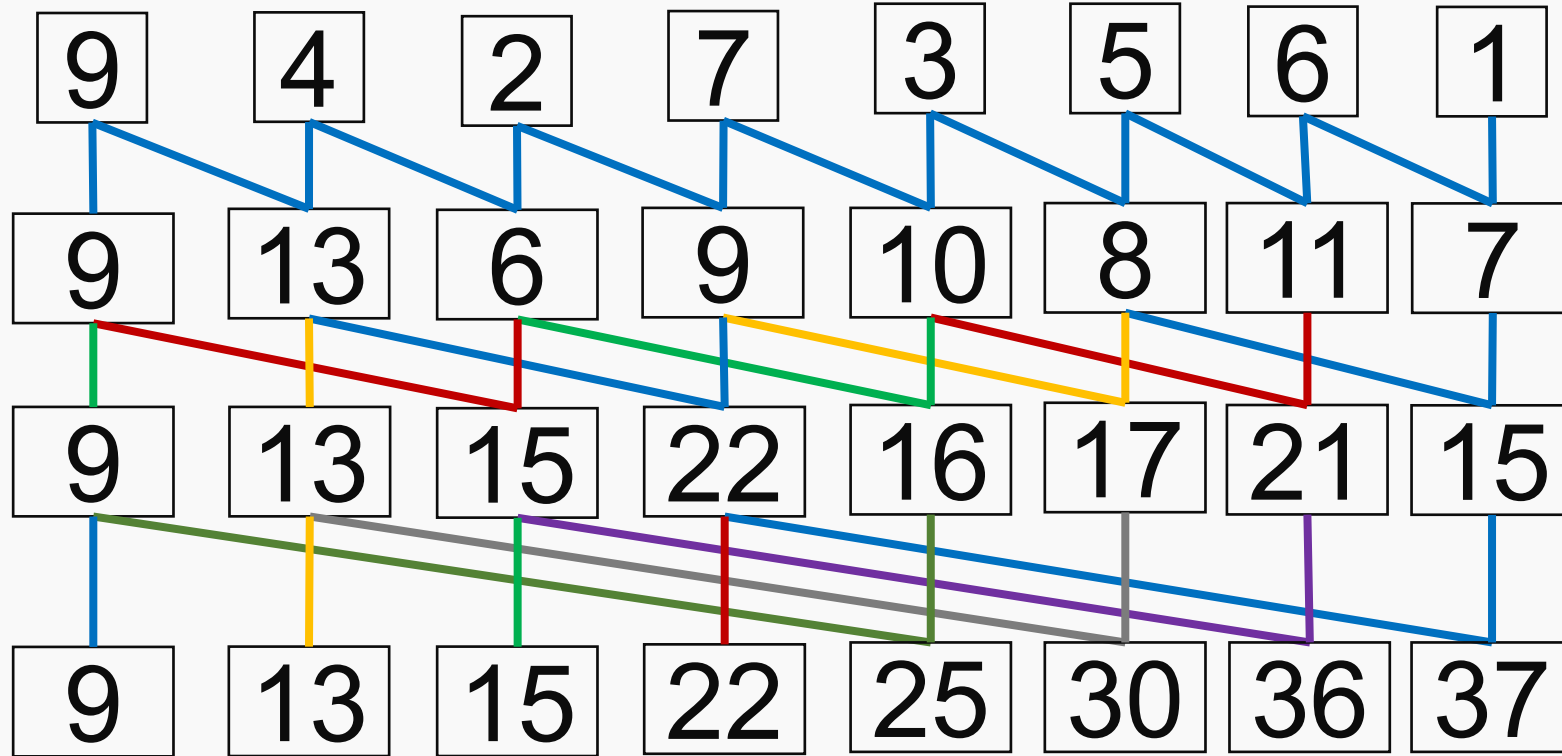


## Scan – Cum funcționează?



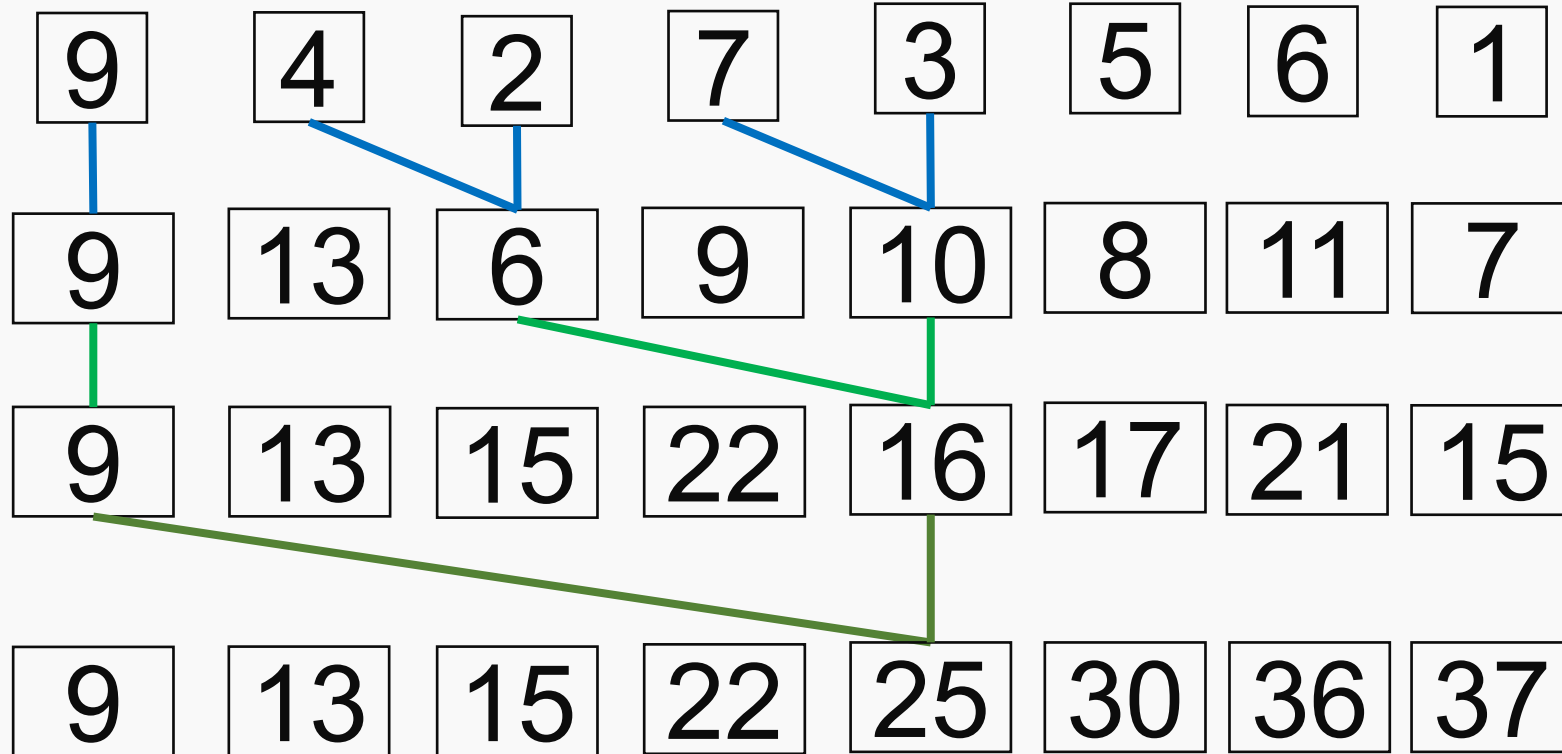


# Scan – Cum funcționează?



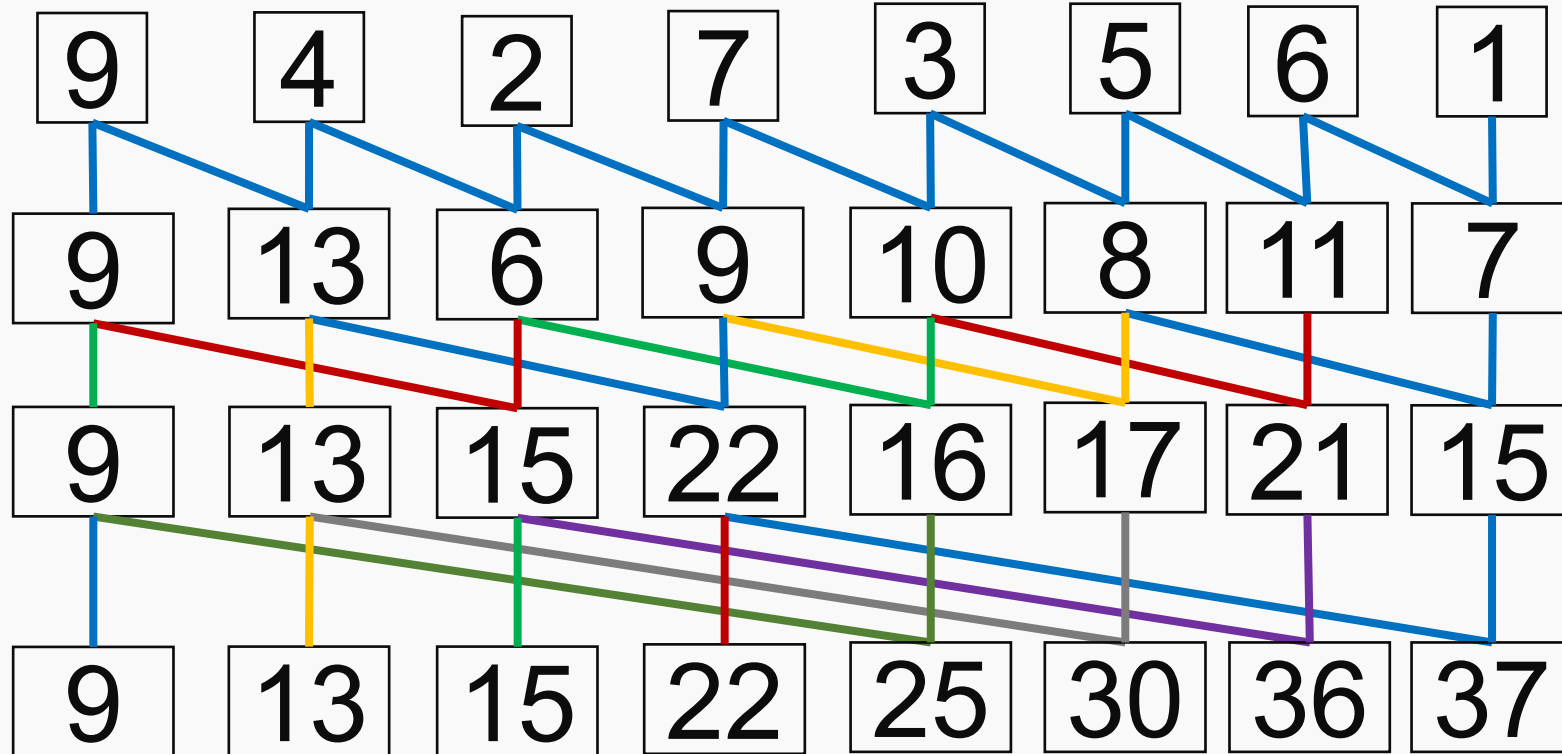


## Scan – Cum funcționează?



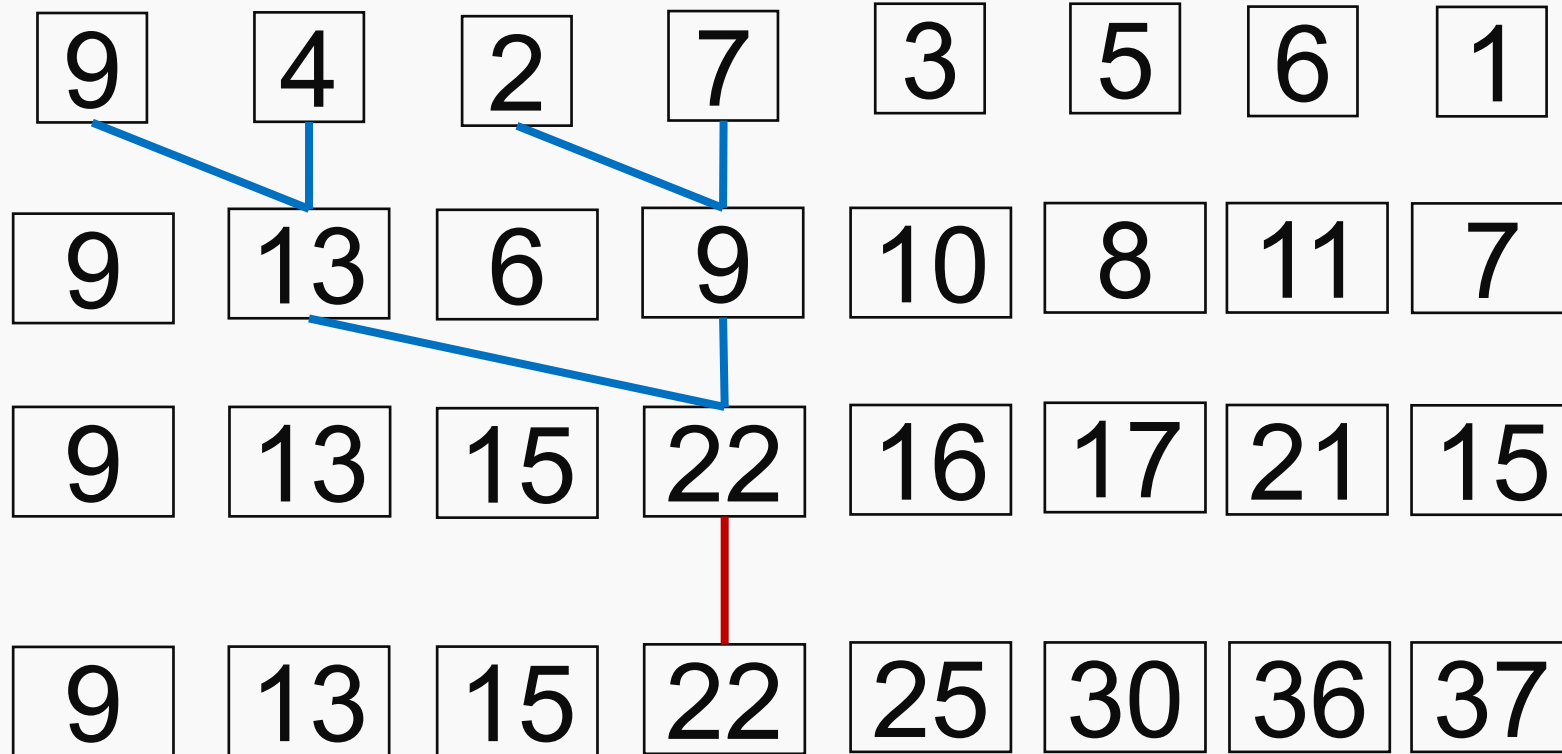


# Scan – Cum funcționează?



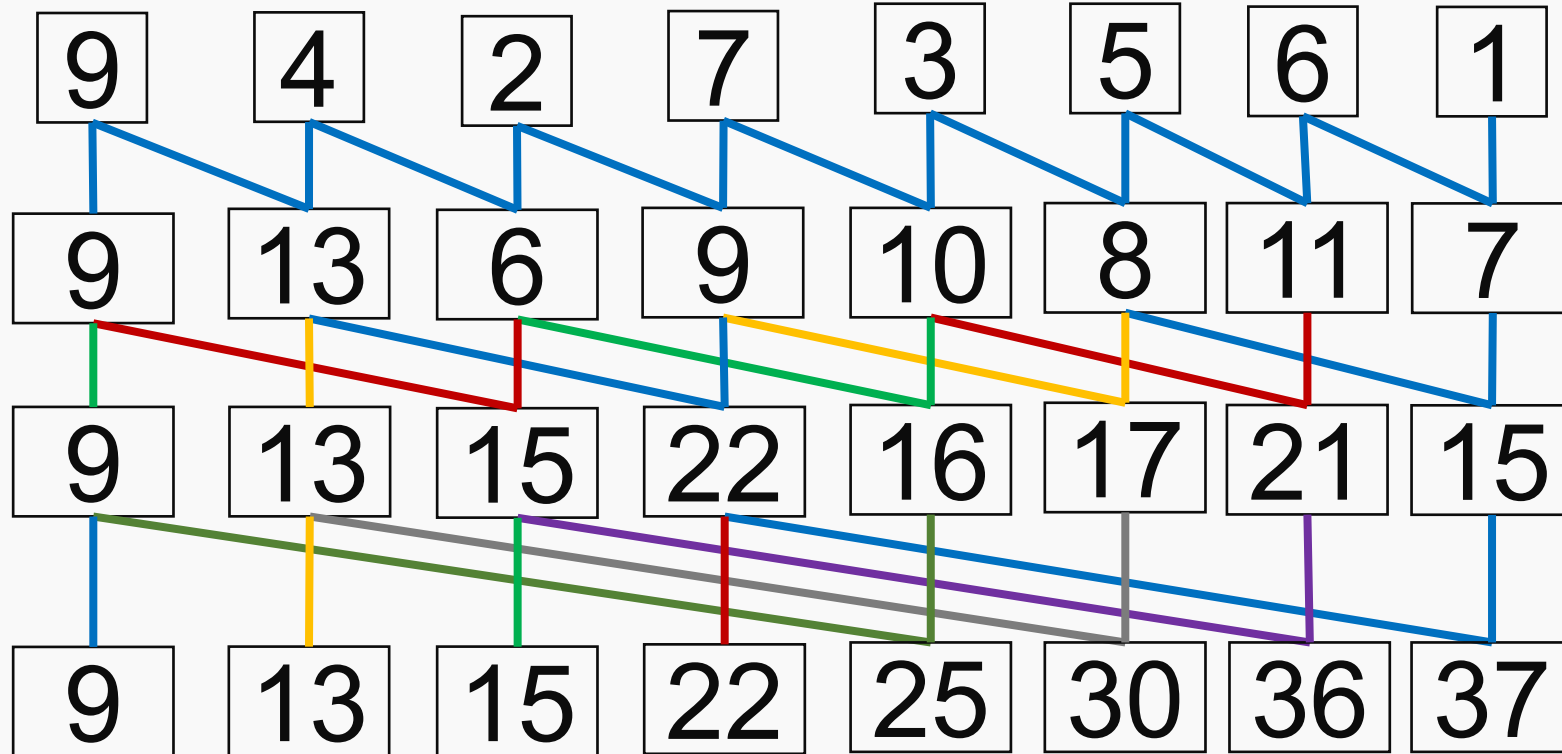


## Scan – Cum funcționează?



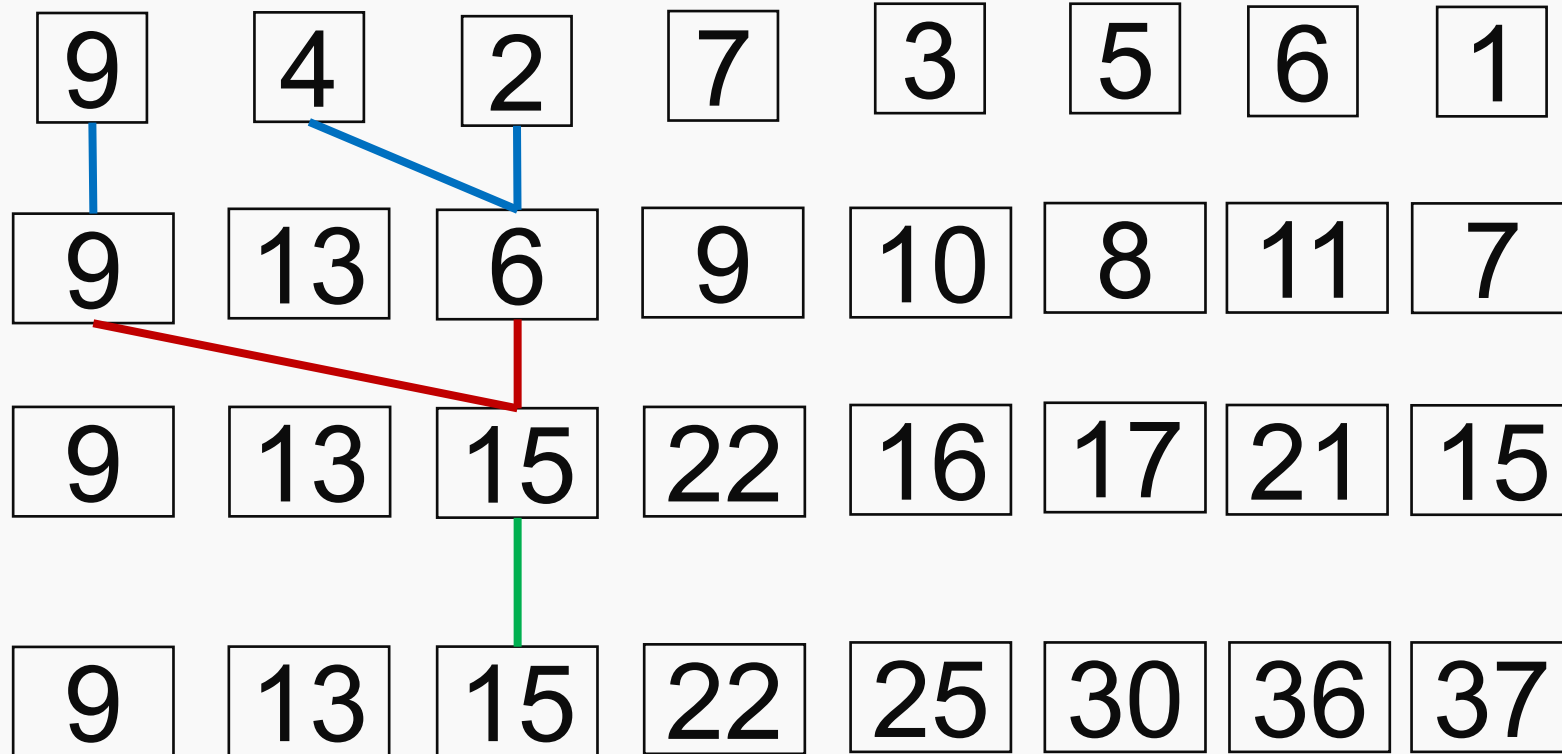


# Scan – Cum funcționează?



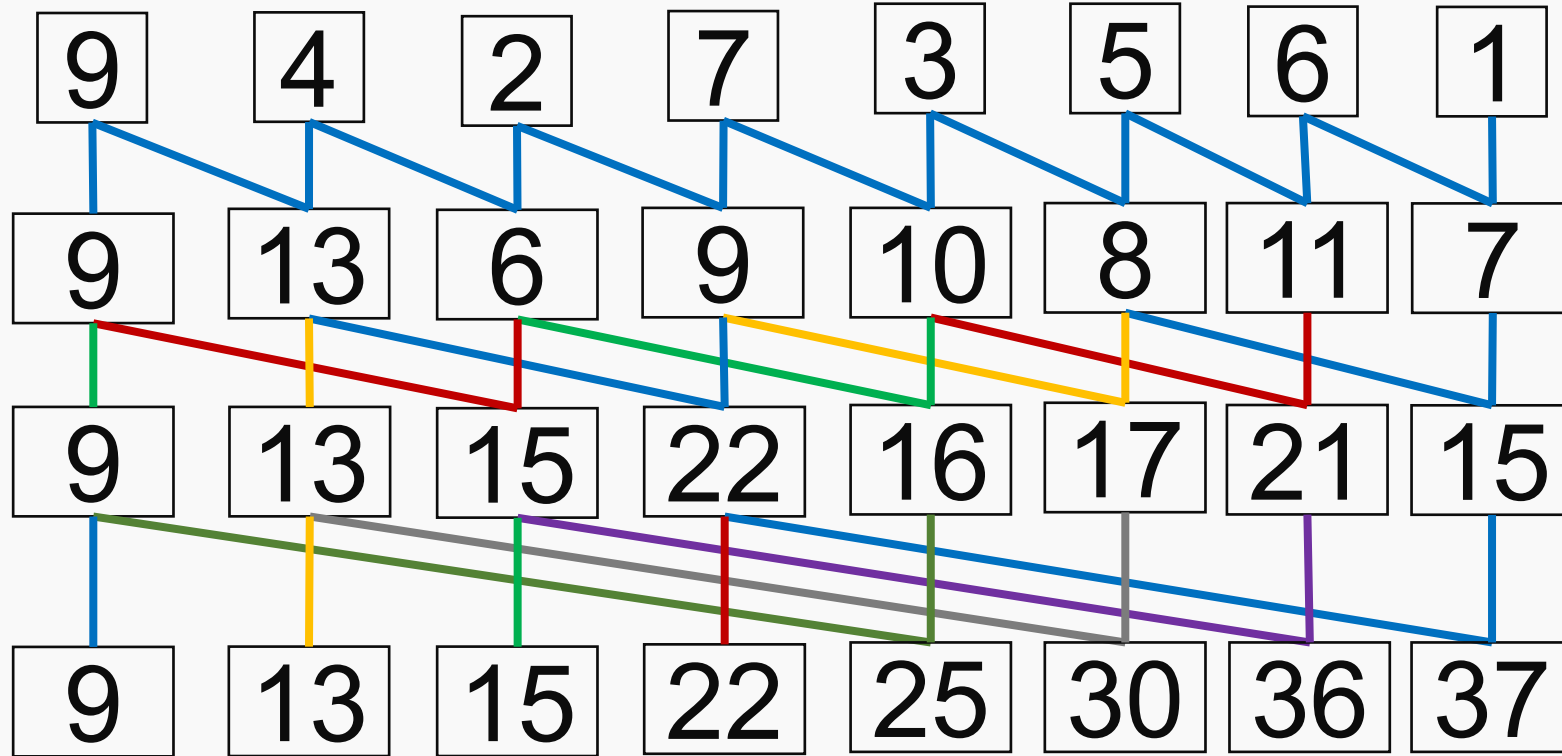


## Scan – Cum funcționează?





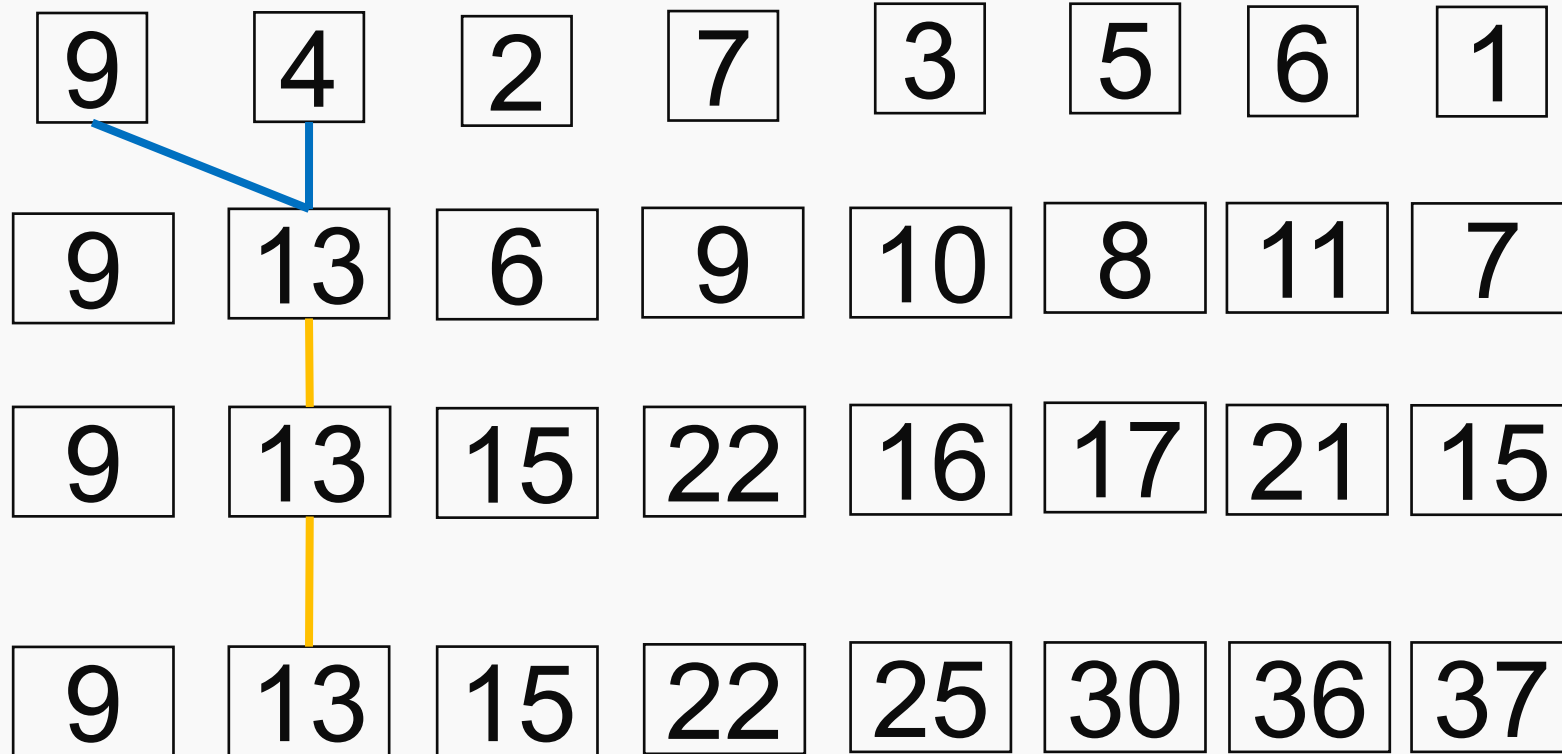
# Scan – Cum funcționează?





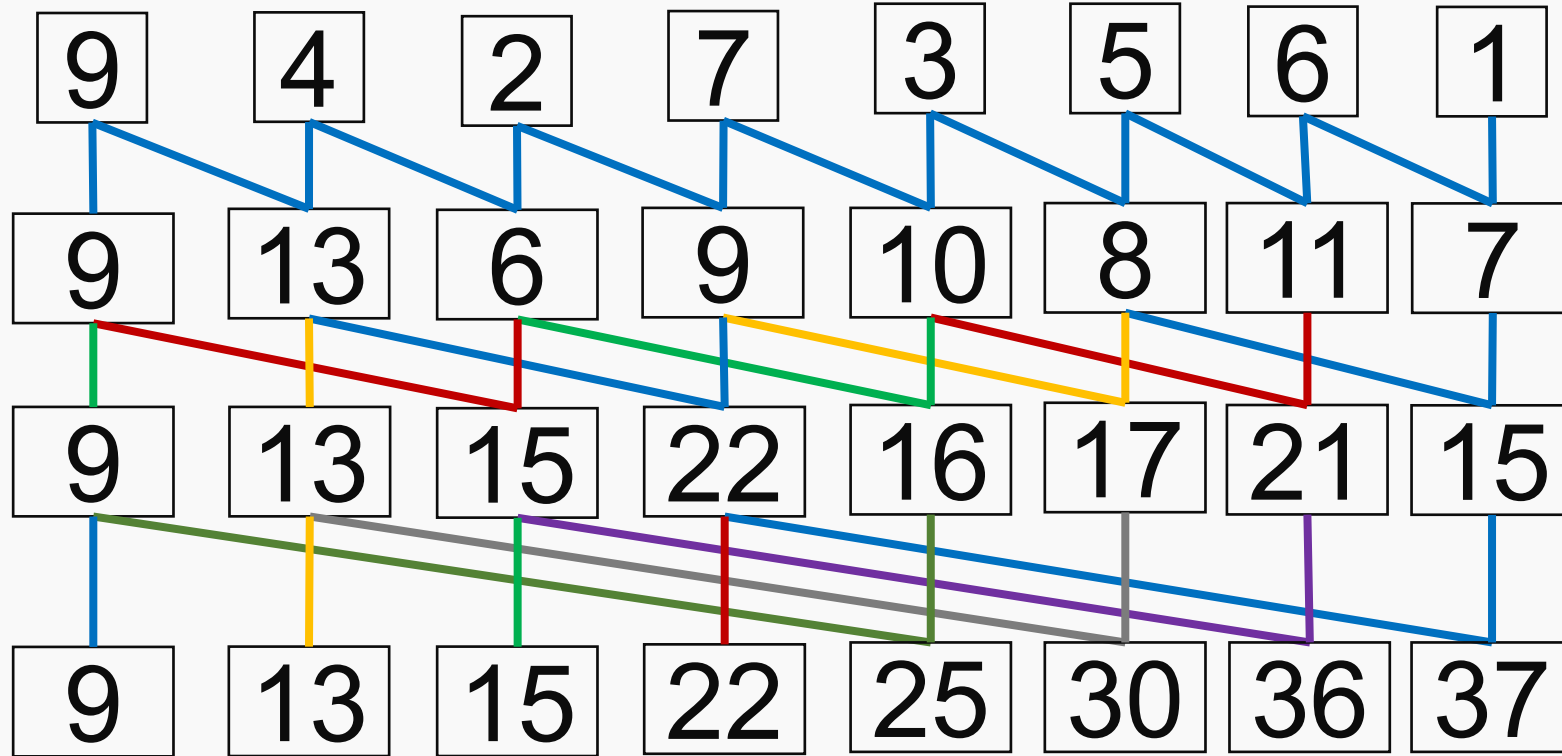


## Scan – Cum funcționează?





# Scan – Cum funcționează?





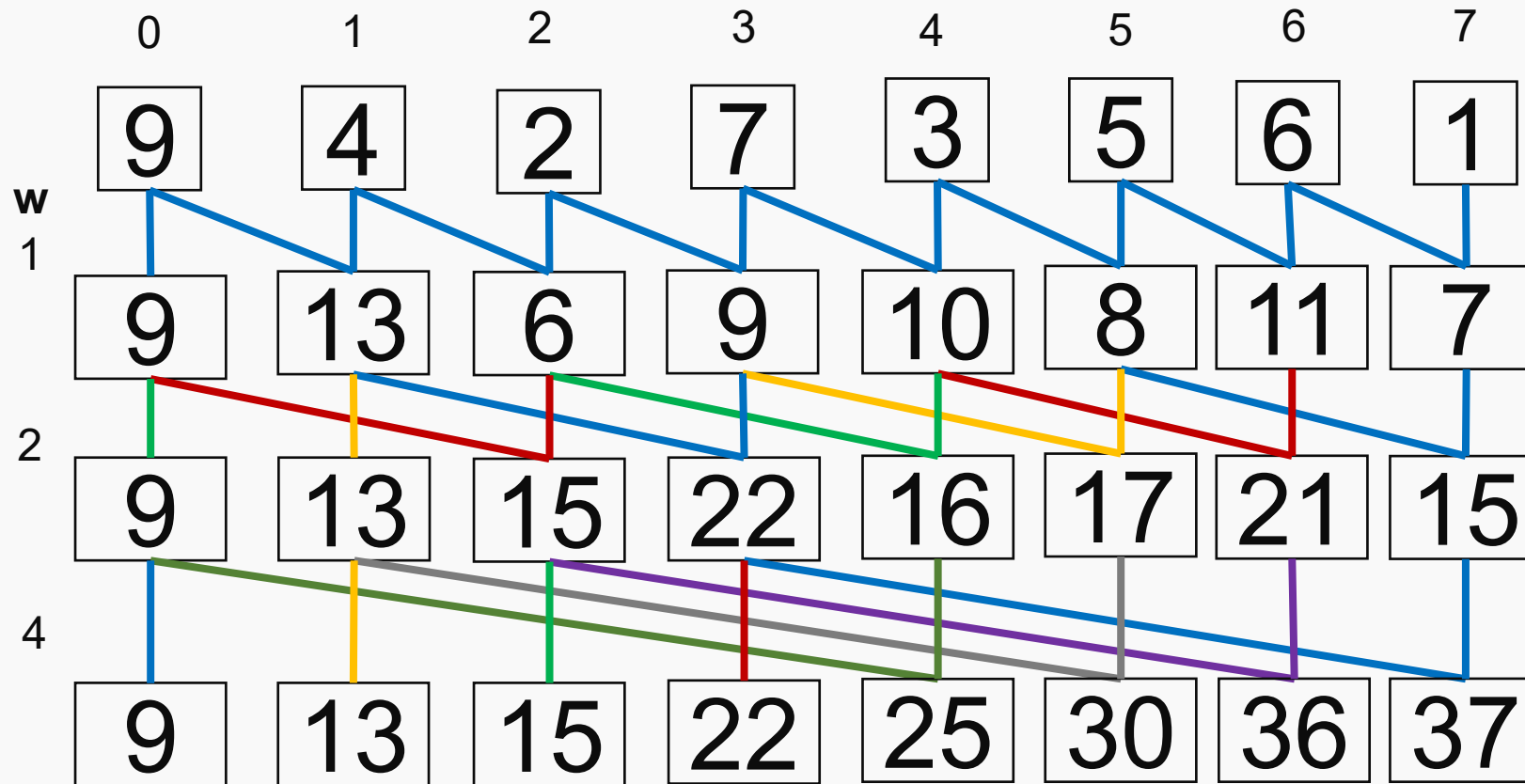
## Scan – Cum funcționează?

9	4	2	7	3	5	6	1
9	13	6	9	10	8	11	7
9	13	15	22	16	17	21	15
9	13	15	22	25	30	36	37



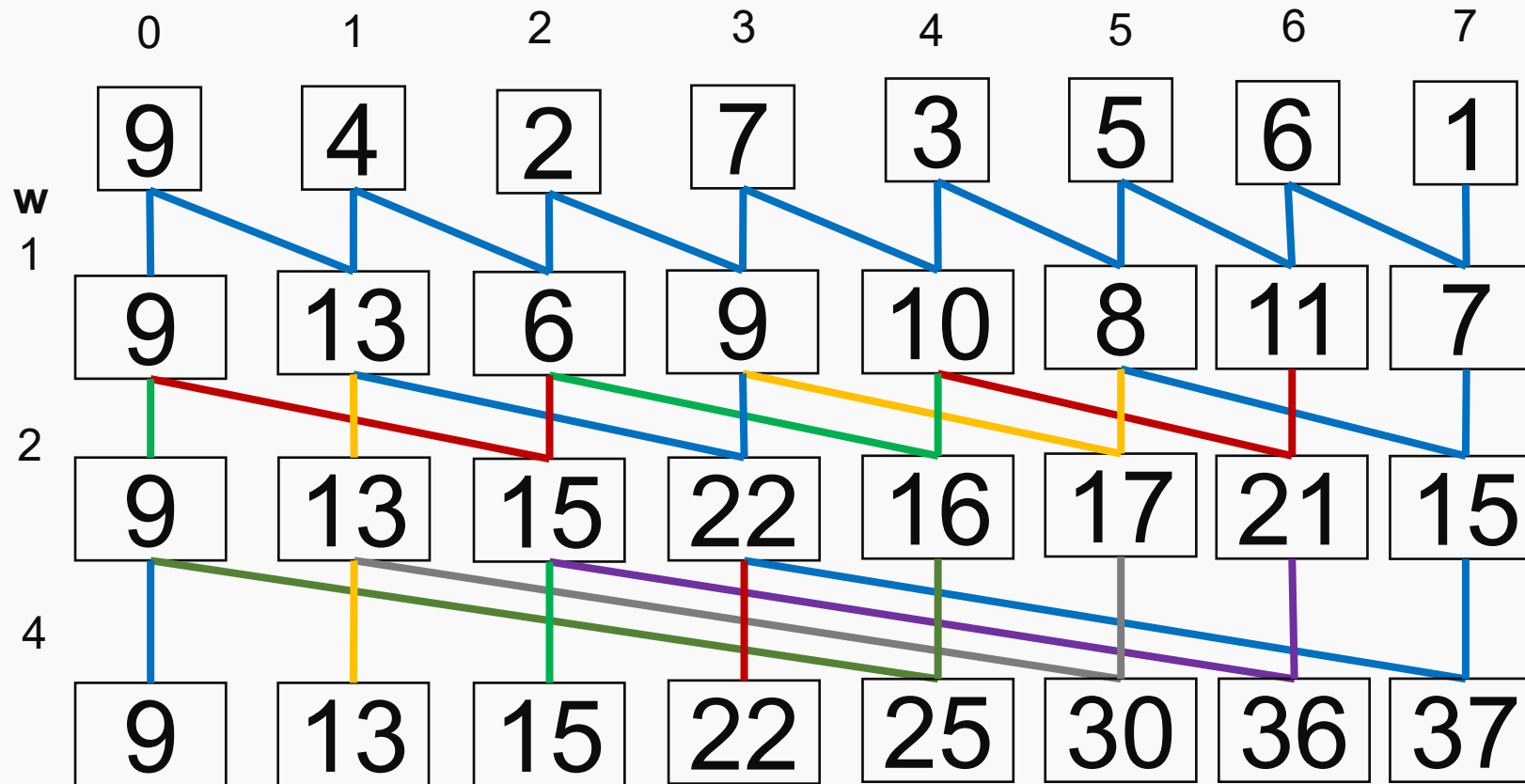


# Scan – implementare





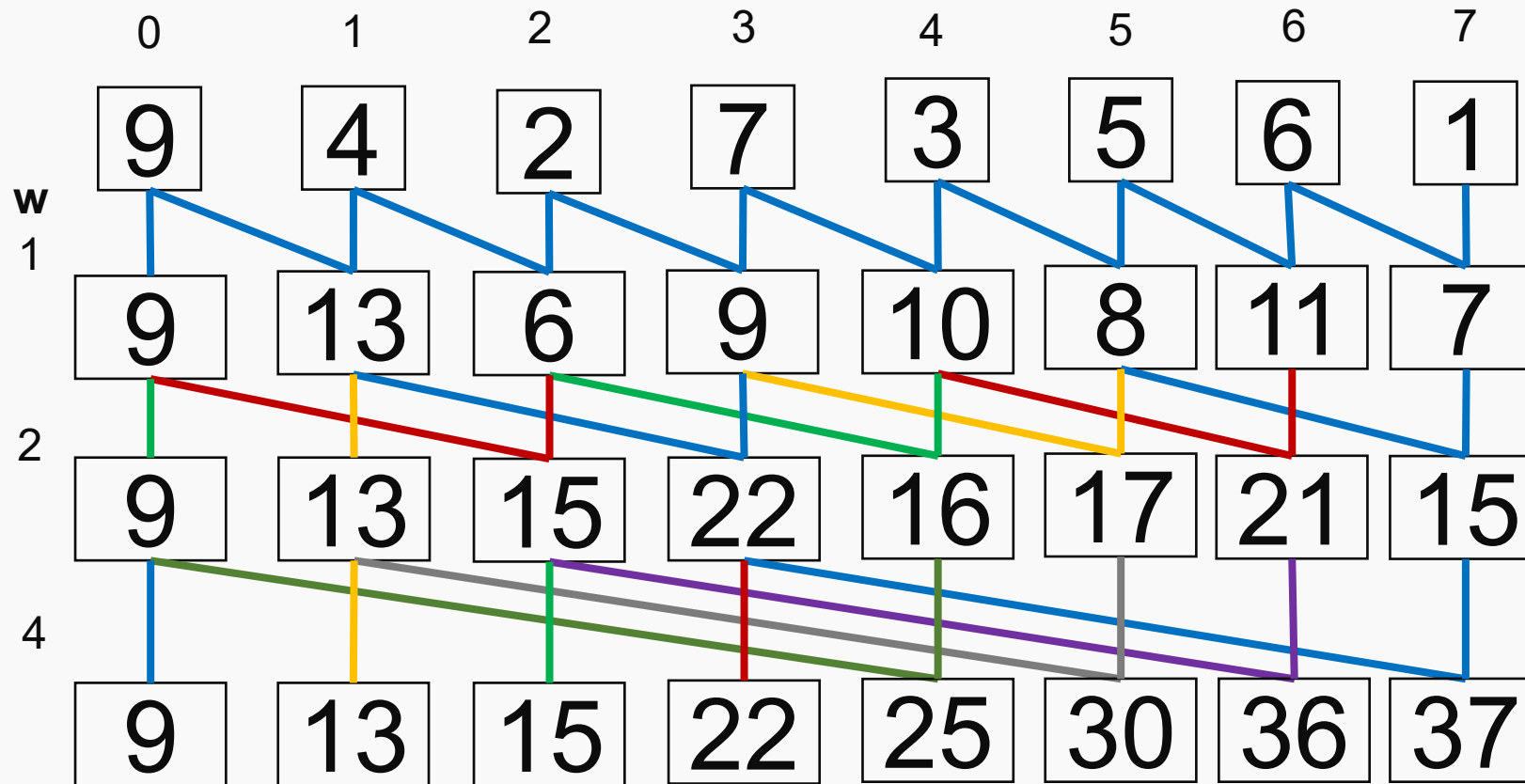
# Scan – implementare – imagine globală



$$v[i] = v[i] + v[i - ?]$$



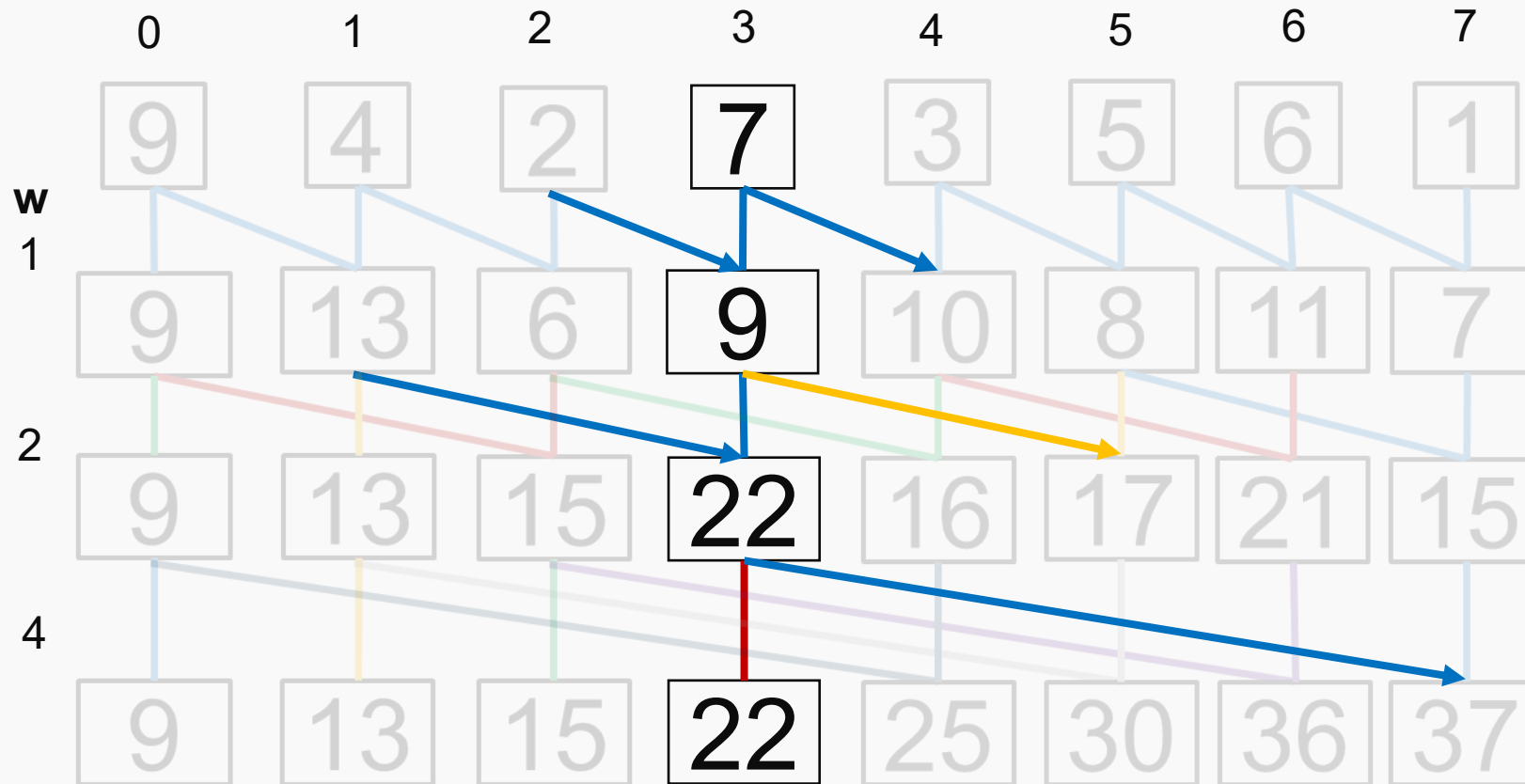
# Scan – implementare – imagine globală



$v[i] = v[i] + v[i - w]$   
Dar dacă  $i - w < 0$ ?



# Scan – implementare – imagine locală

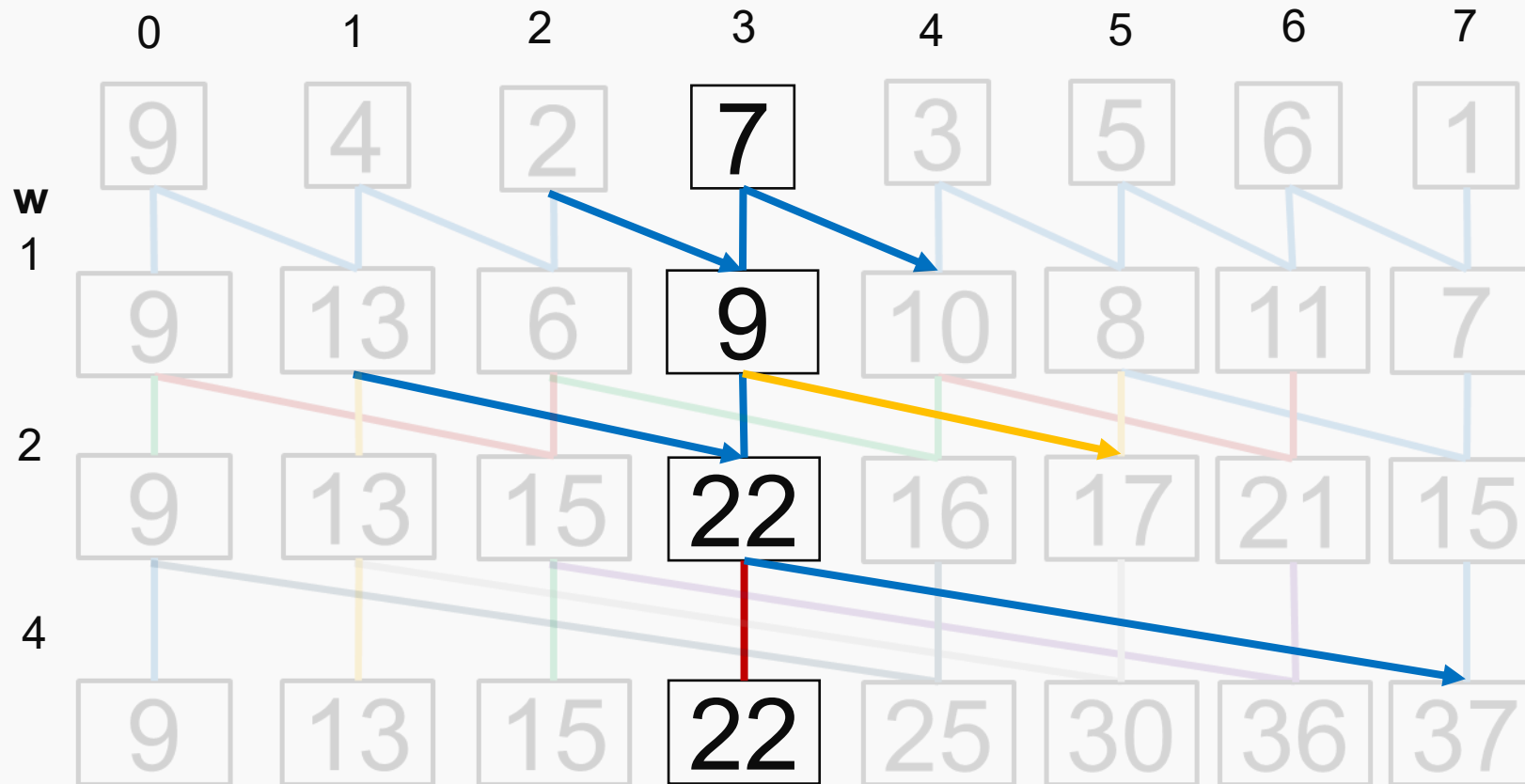


De unde se primesc valori?  
Unde se trimit valori?





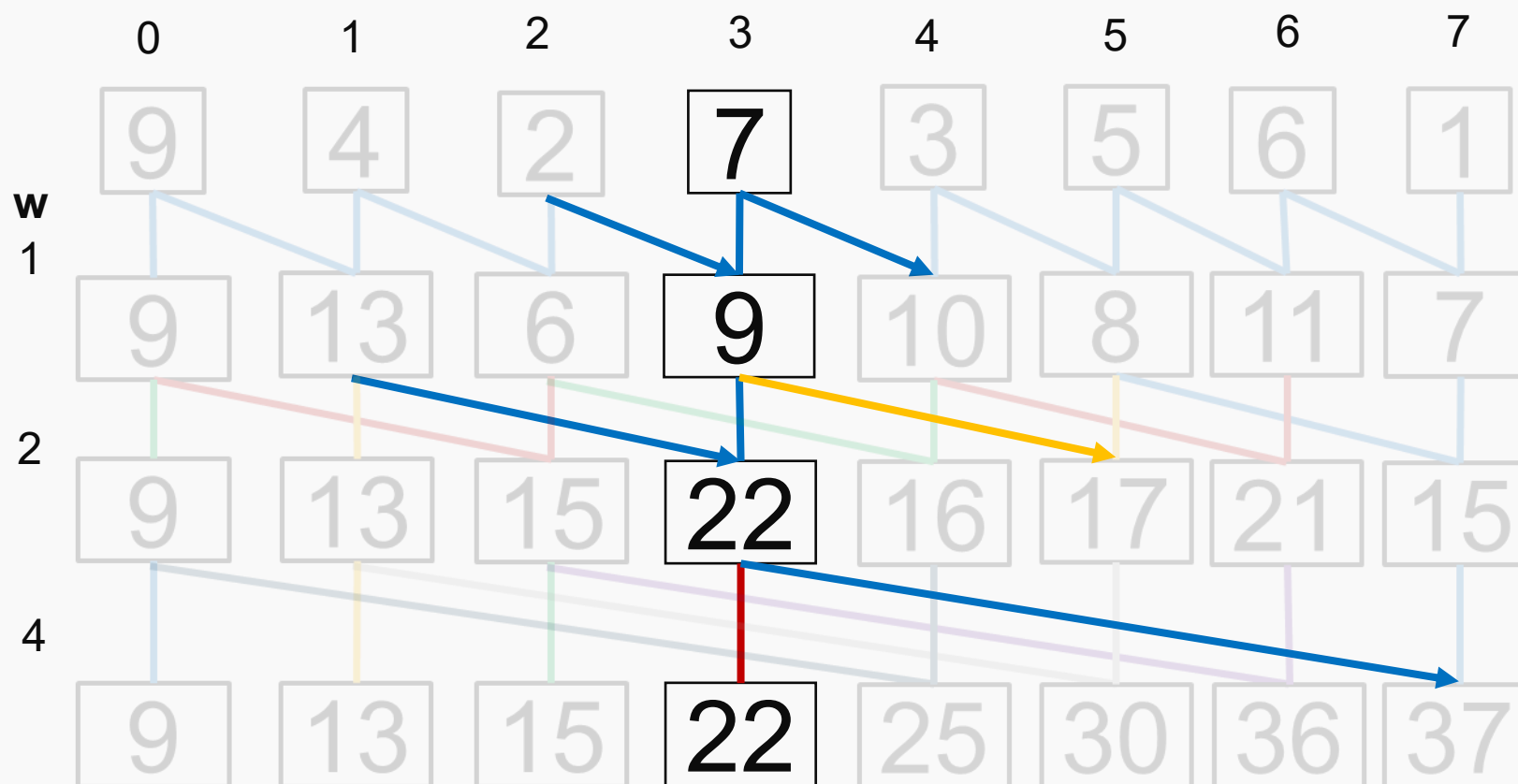
# Scan – implementare – imagine locală



De unde se primesc valori?  $id - w$   
Unde se trimit valori?  $id + w$



# Scan – implementare – imagine locală



De unde se primesc valori?  $id - w$

Unde se trimit valori?

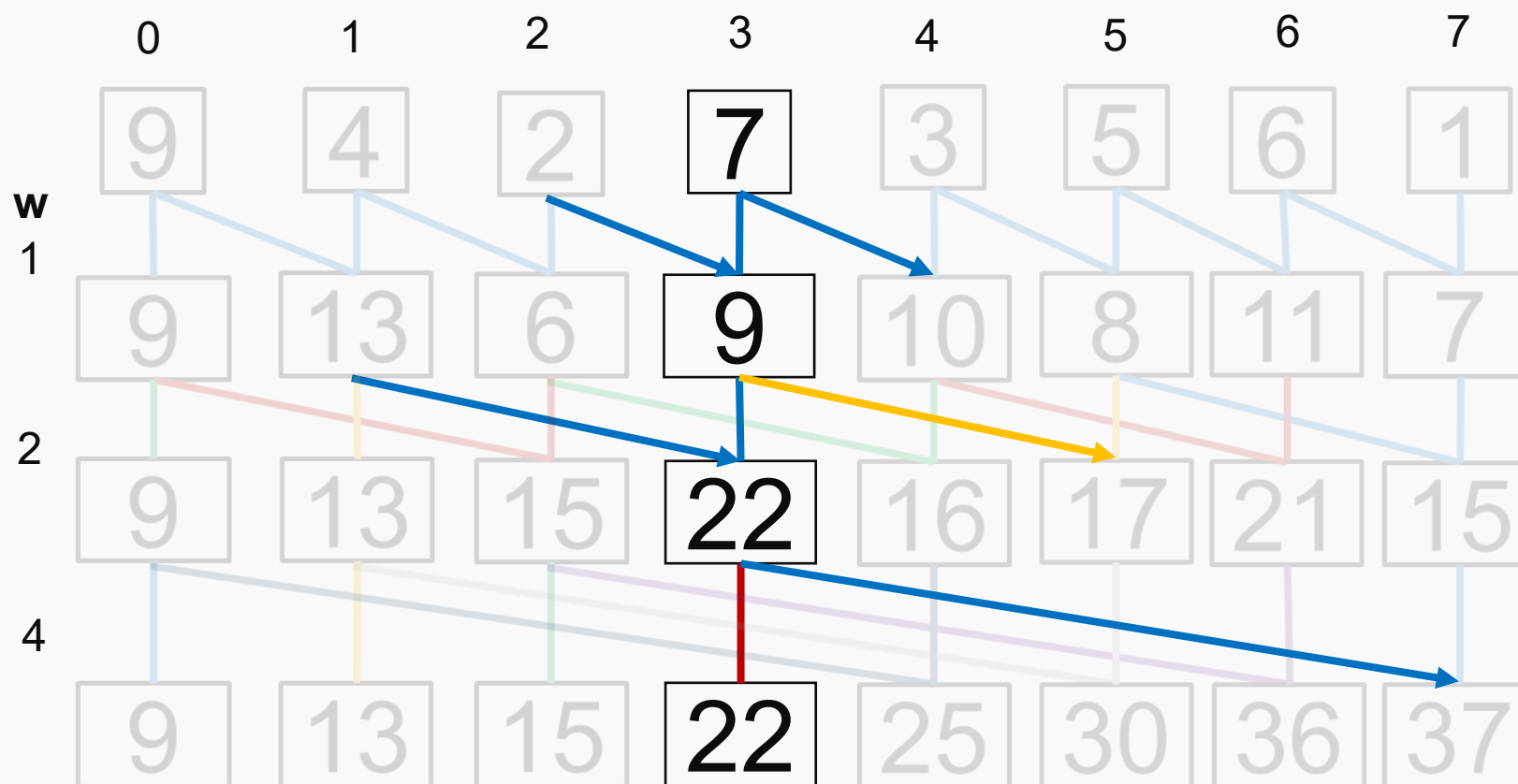
$id + w$

Când se primesc valori?

Când se trimit valori?



# Scan – implementare – imagine locală



De unde se primesc valori?  $id - w$

Când se primesc valori?  $\geq 0$

Unde se trimit valori?  $id + w$

Când se trimit valori?  $< N$





# Broadcast

Start

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



# Broadcast

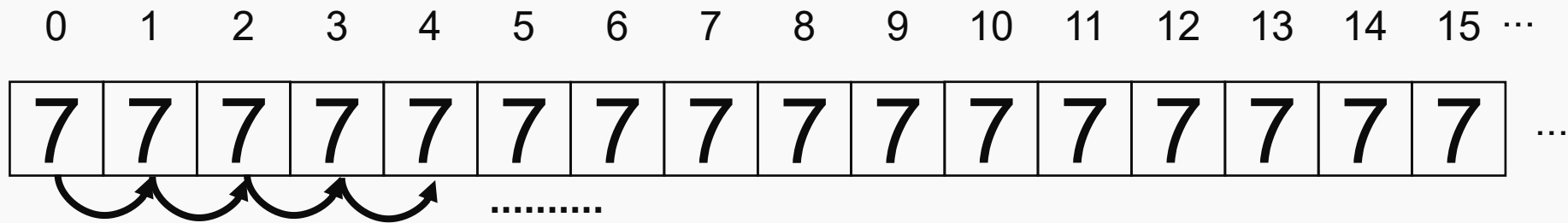
End

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



# Broadcast inefficient

Complexitate:  $O(N)$

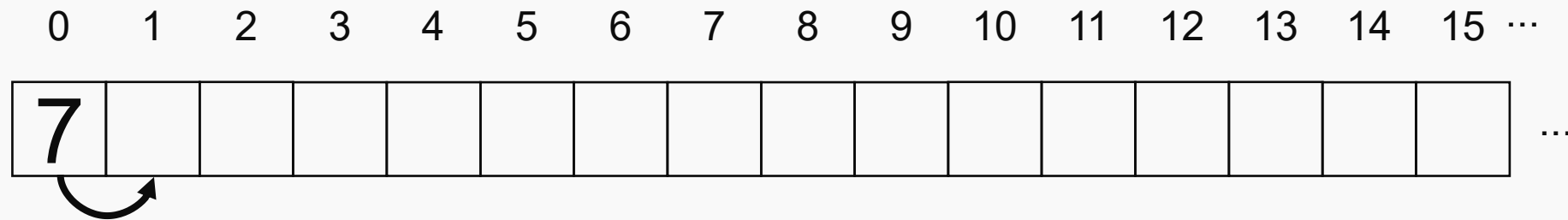




# Broadcast

Fiecare element care are valoarea  
o copiază la poziția sa + w

$$w = 1$$



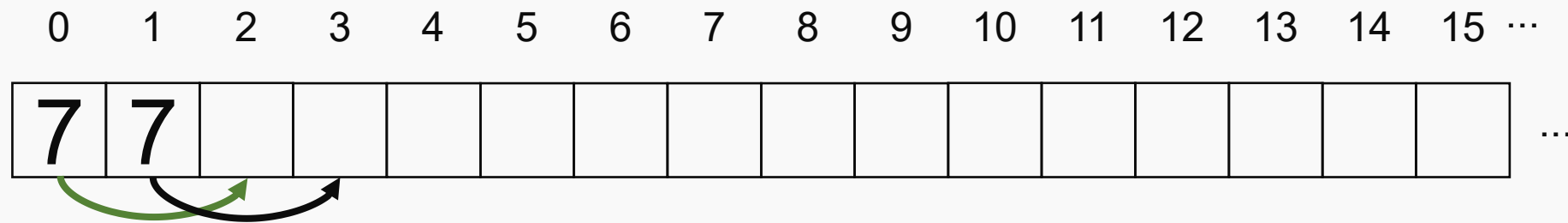




# Broadcast

Fiecare element care are valoarea  
o copiază la poziția  $s + w$

$$w = 2$$



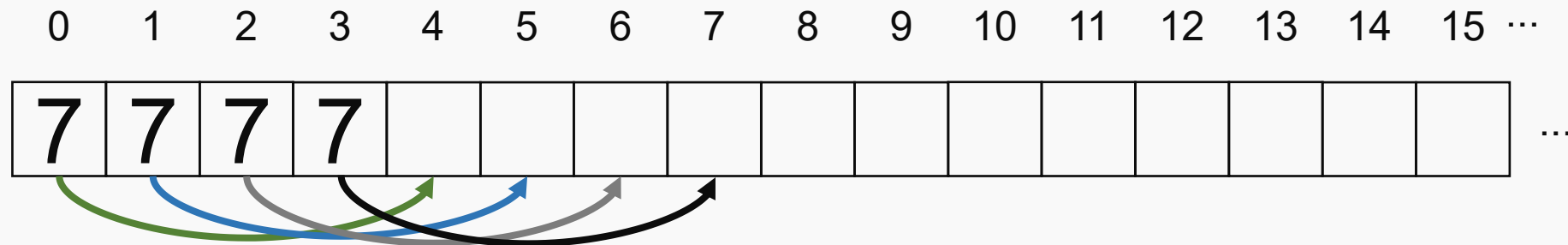
Aceste operații pot fi executate în paralel



# Broadcast

Fiecare element care are valoarea  
o copiază la poziția sa + w

$$w = 4$$



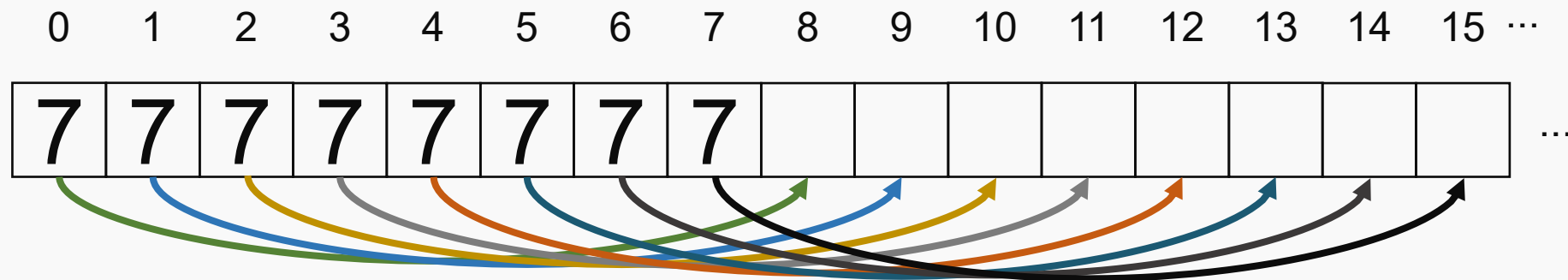
Aceste operații pot fi executate în paralel



# Broadcast

Fiecare element care are valoarea  
o copiază la poziția sa + w

$$w = 8$$



Aceste operații pot fi executate în paralel

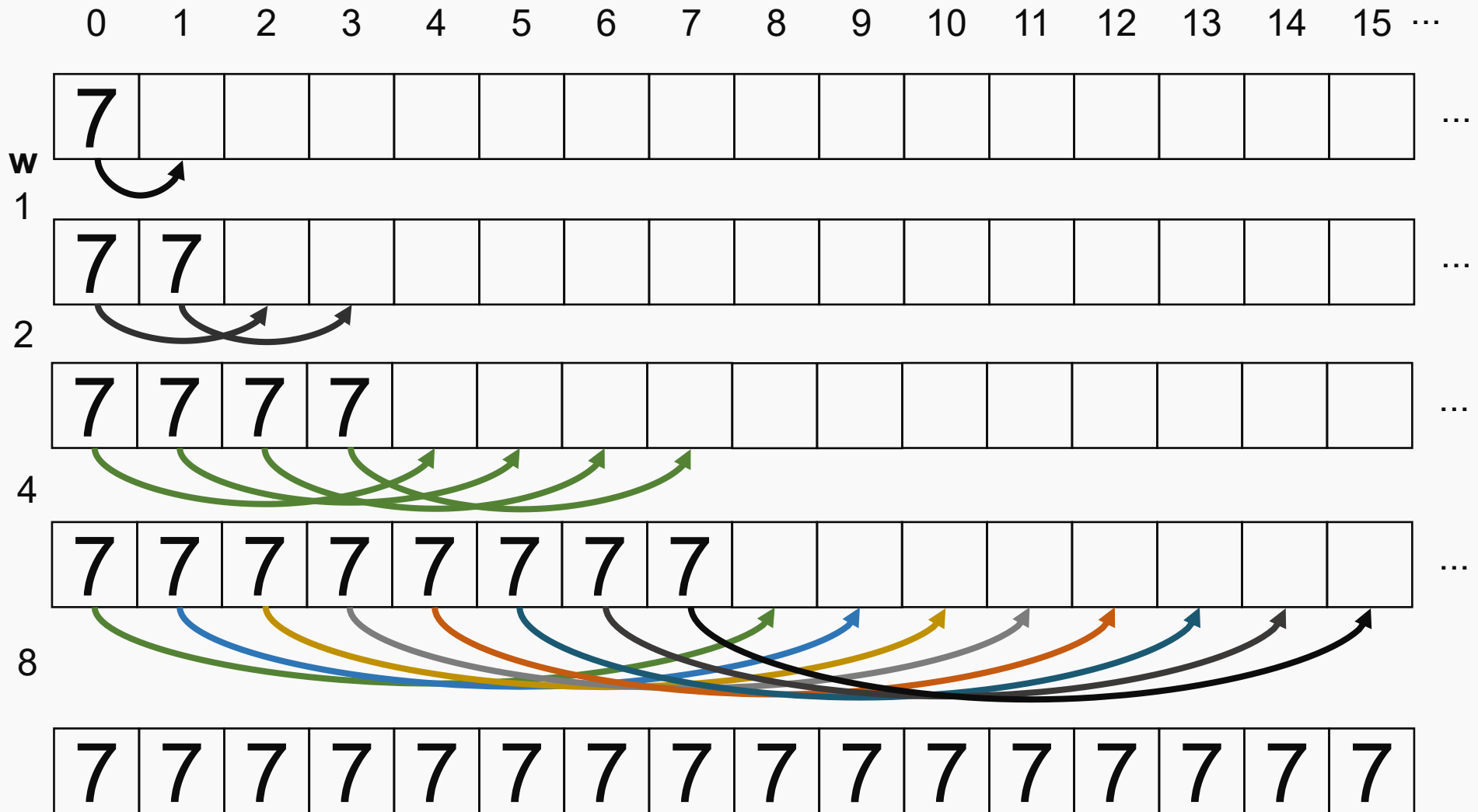


# Broadcast

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



# Broadcast

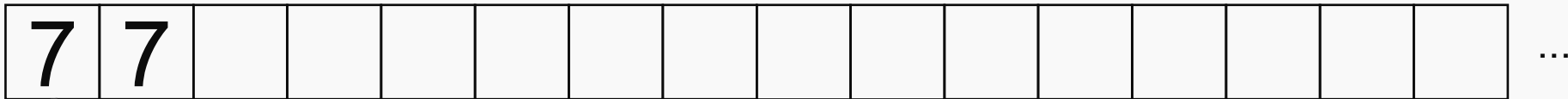




# Broadcast

$$w = 2$$

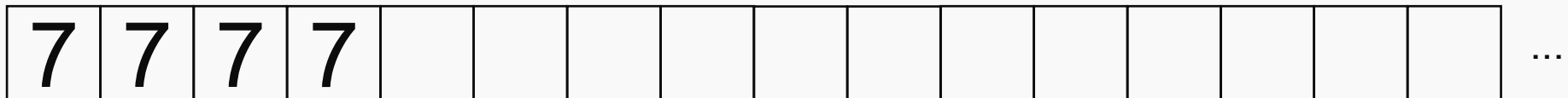
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 ...



Aceste operații **NU** pot fi executate în paralel

$$w = 4$$

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

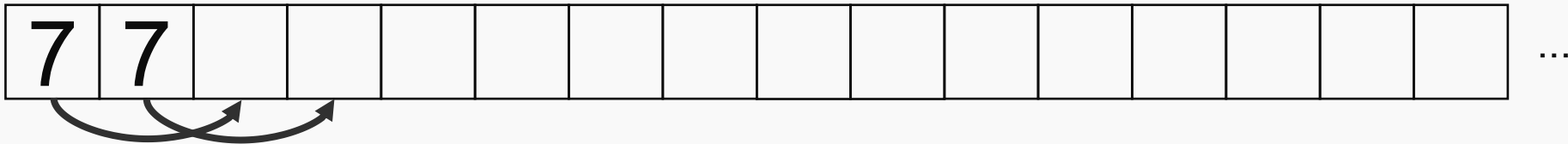




# Broadcast

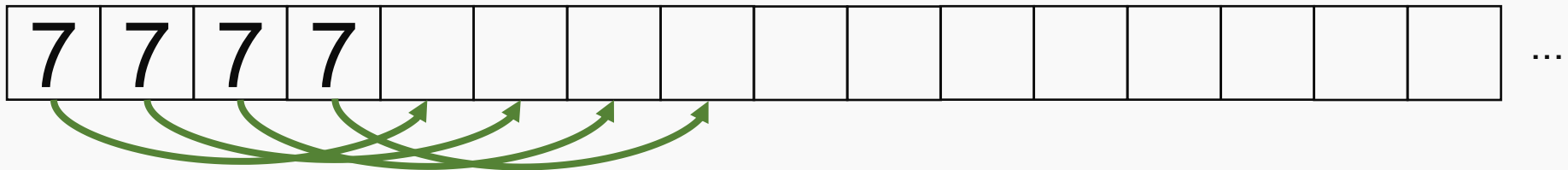
$w = 2$

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



$w = 4$

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





# Broadcast

**Complexitate?**

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





# Broadcast

**Complexitate?  $O(\log(N))$  pentru  $N=2P$**

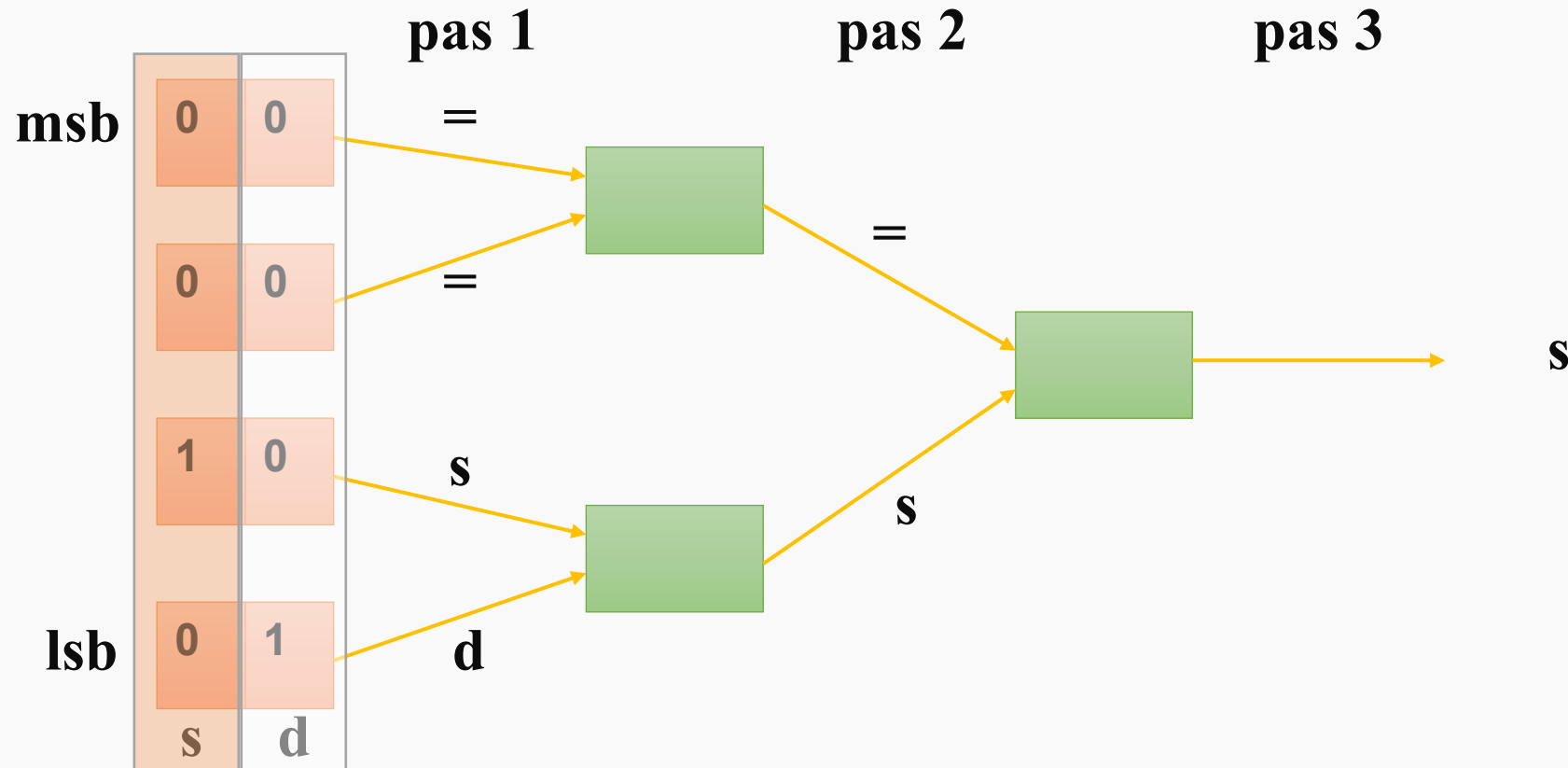
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	...





## Exemplu complex – multiple tehnici

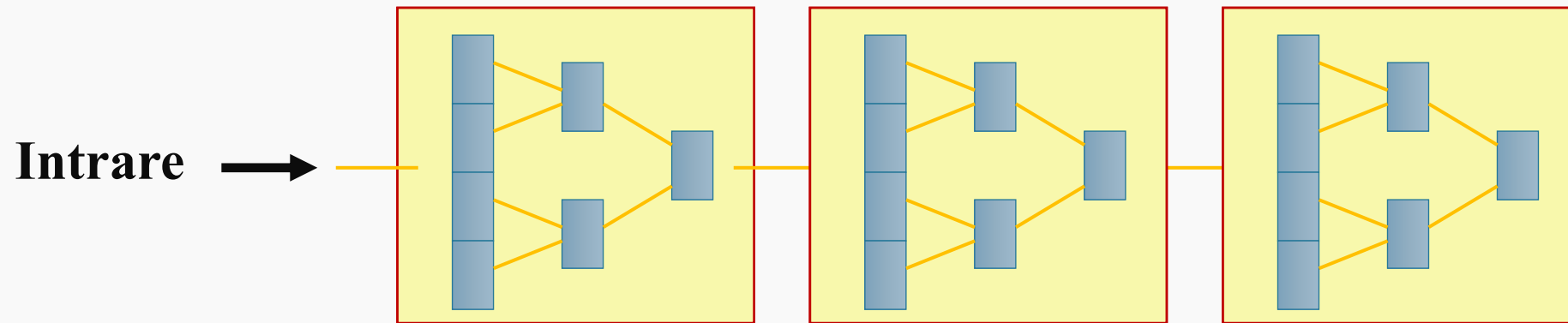
- Operație pe biți – comparație a două numere.





## Exemplu complex – multiple tehnici

Algoritmul de sortare cu pipeline devine:

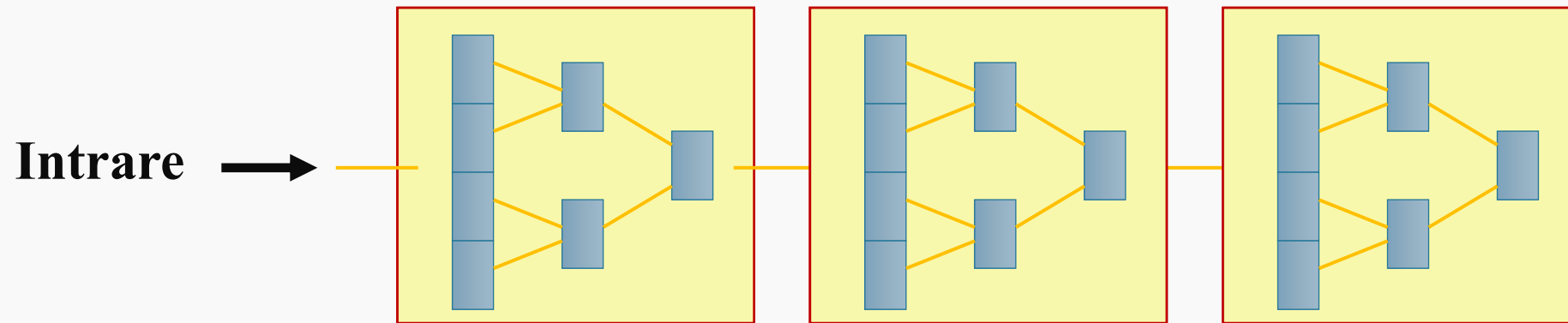


Complexitate?



## Exemplu complex – multiple tehnici

Algoritmul de sortare cu pipeline devine:



Complexitate?

$$O(N * \log(\text{numbiți}))$$