Arhitecturi Paralele

Abordarea algoritmilor în mod paralel 2

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

Curs susținut în parteneriat cu Prof. Florin Pop

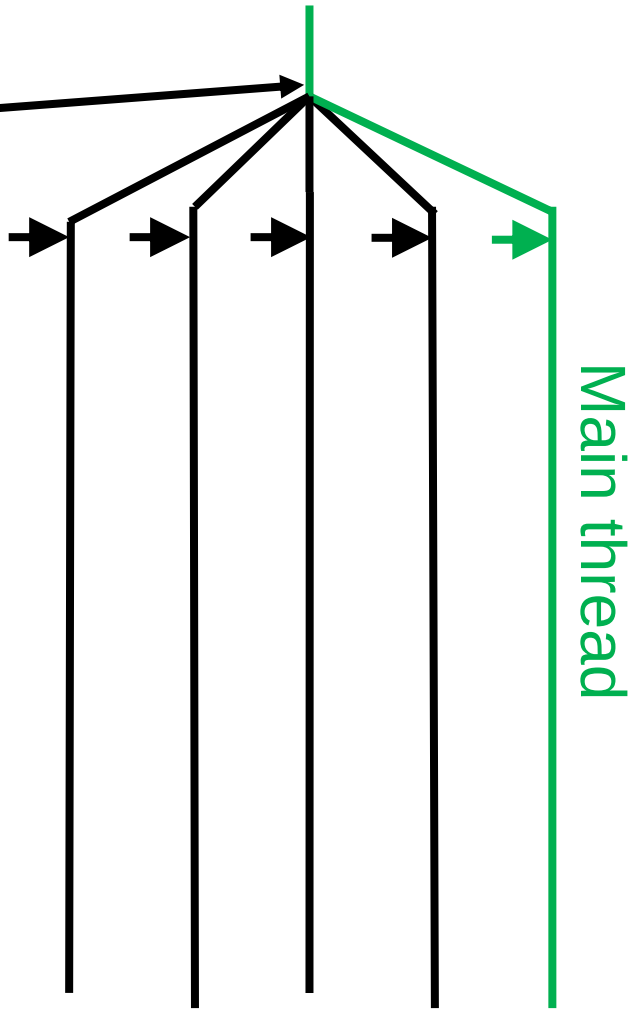
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Executor Service
sau
Replicated Workers
sau
Thread Pool

Abordare de probleme recursive în paralel

Replicated Workers

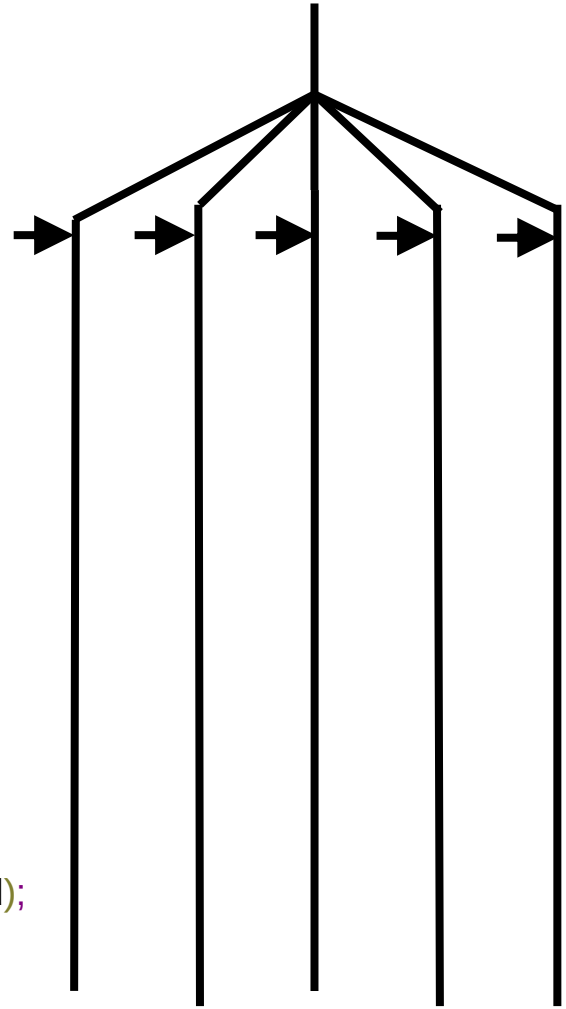
```
startWorkers();
```



Replicated Workers

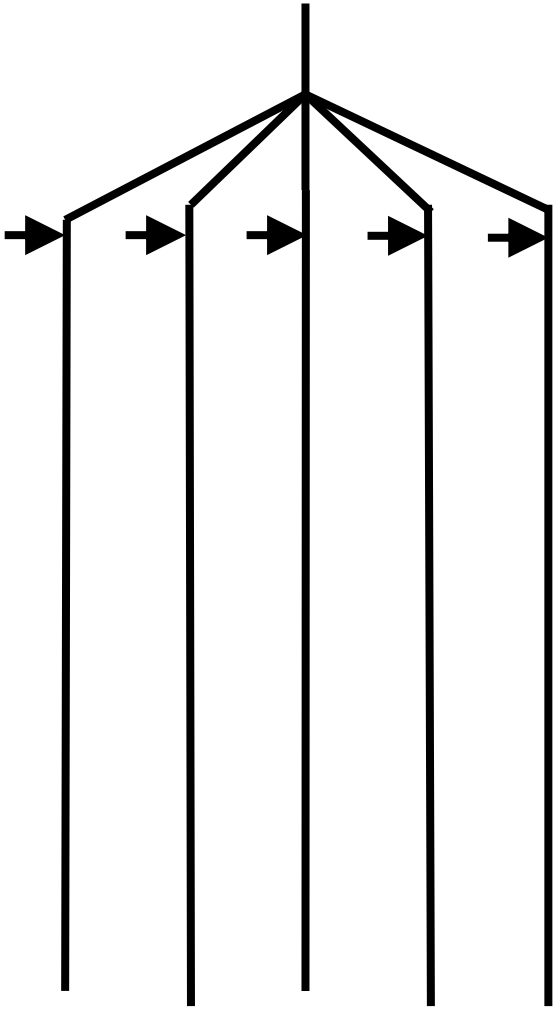
```
Task makeTask(int i)
{
        Task task;
        int * newData = (int*)malloc(sizeof(int));
        newData[0]=i;
        task.data=newData;
        task.runTask = printSomething;
        return task;
}

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));
}
```

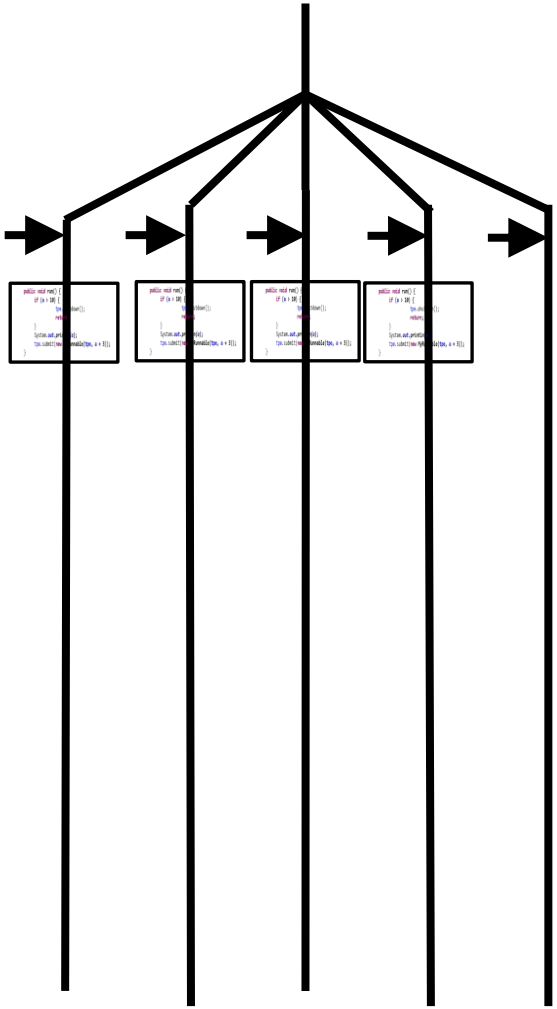


Replicated Workers

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



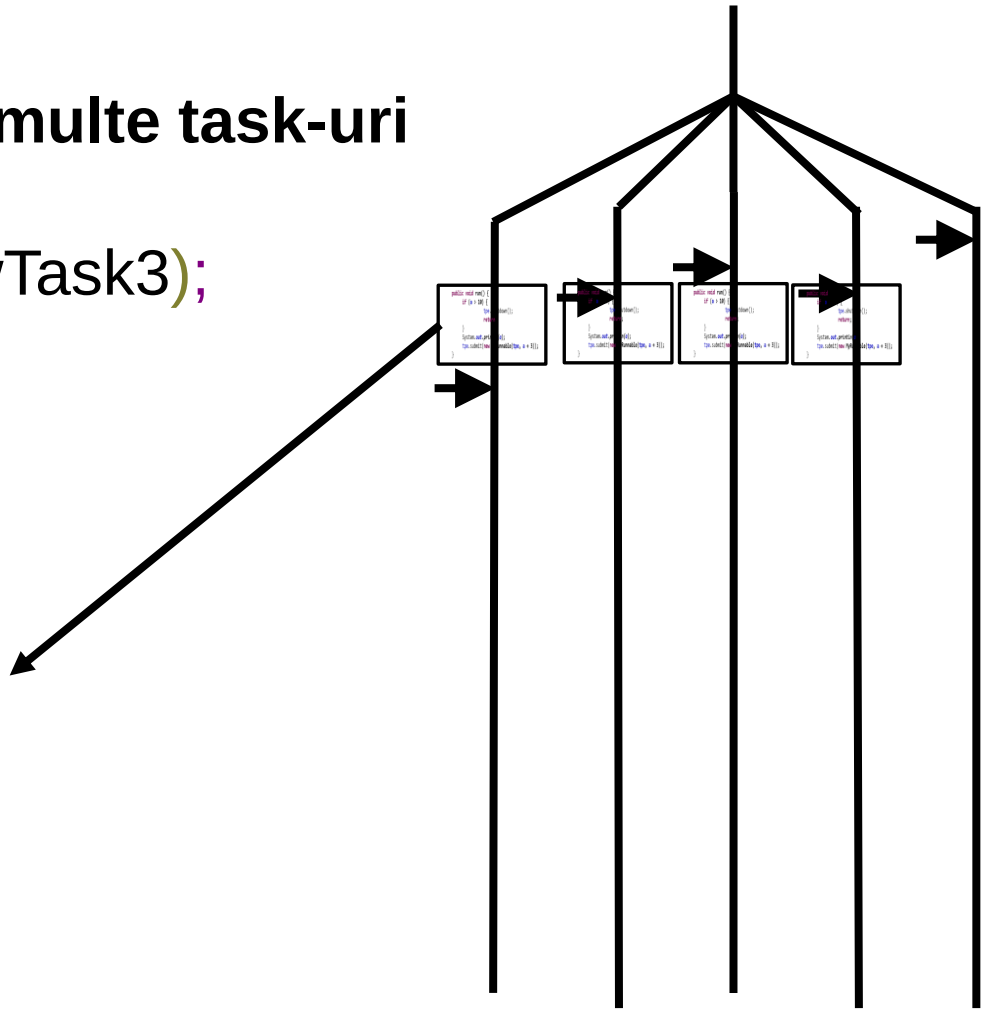
Replicated Workers



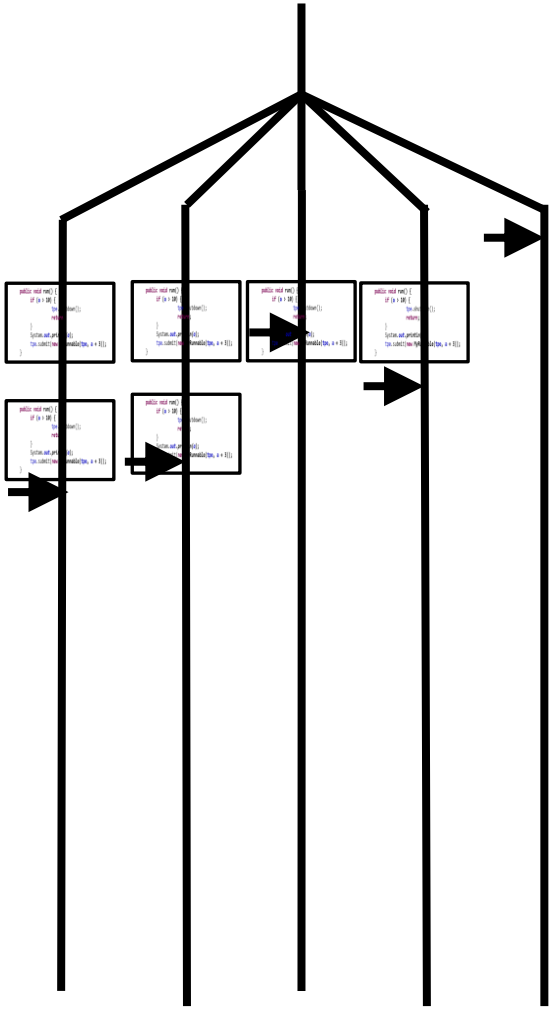
Replicated Workers

Task-urile pot crea mai multe task-uri

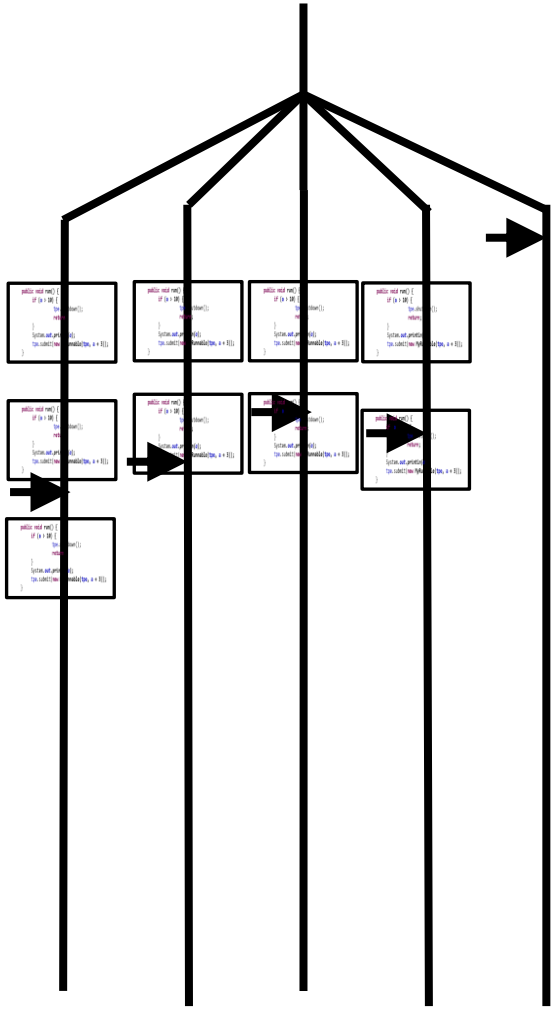
putTask(newTask3);



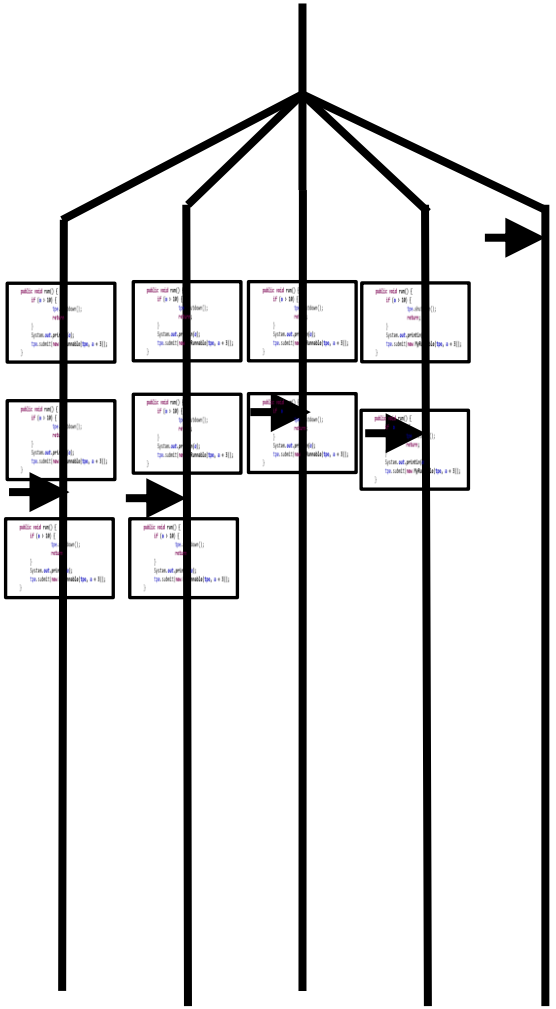
Replicated Workers



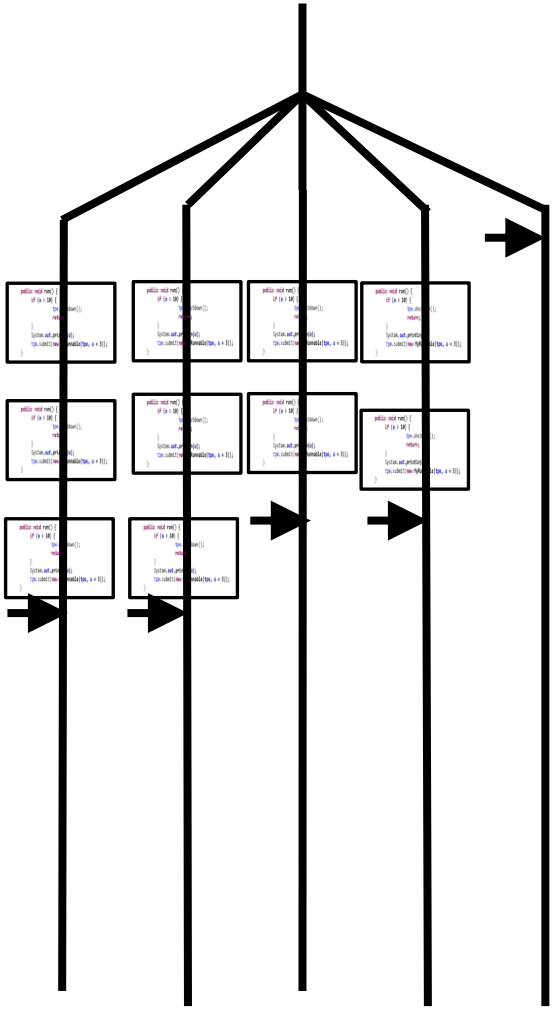
Replicated Workers



Replicated Workers

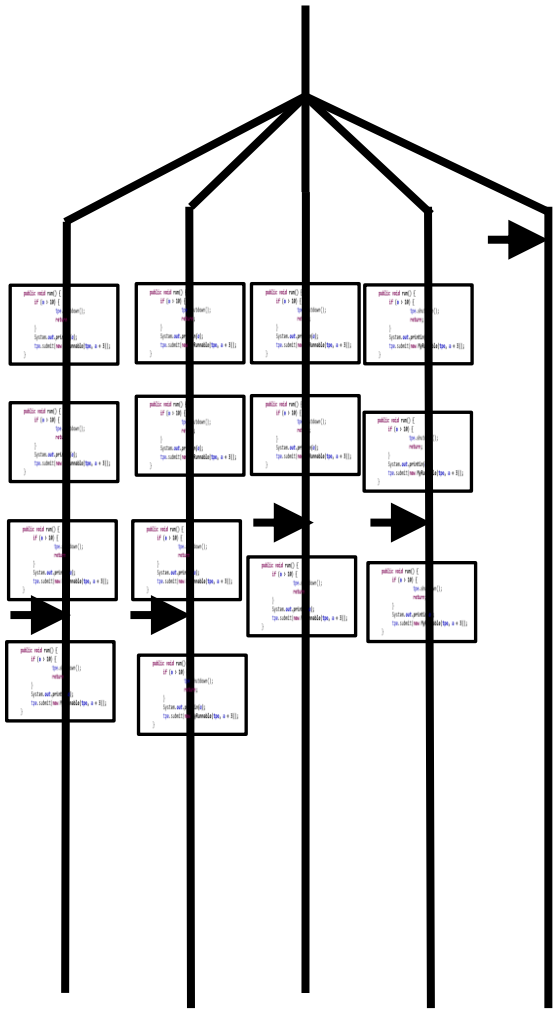


Replicated Workers



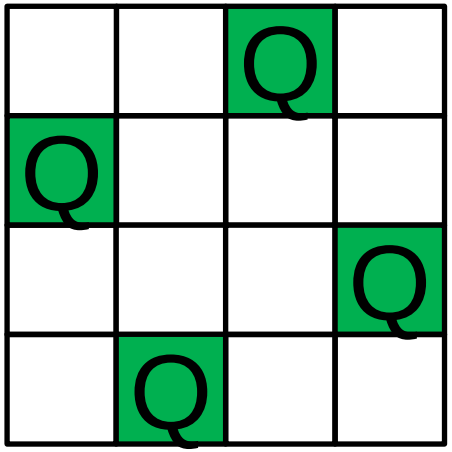
Replicated Workers

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();
```

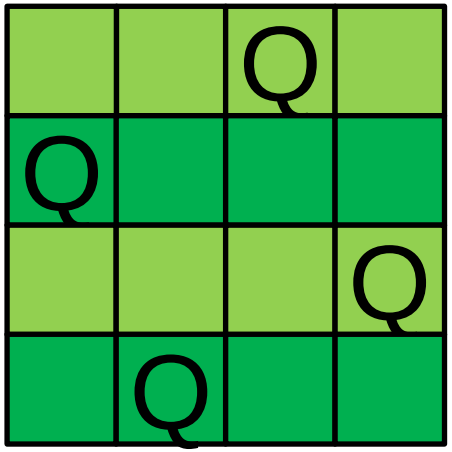
N Queens Problem



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

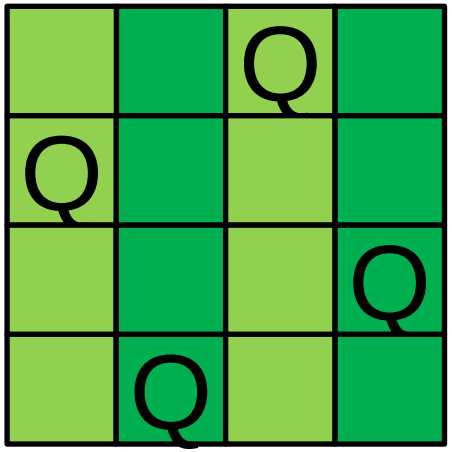
N Queens Problem

Nu avem voie mai mult de o regină pe linie



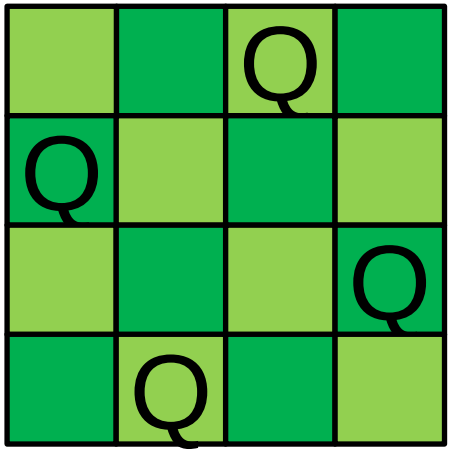
N Queens Problem

Nu avem voie mai mult de o regină pe coloană

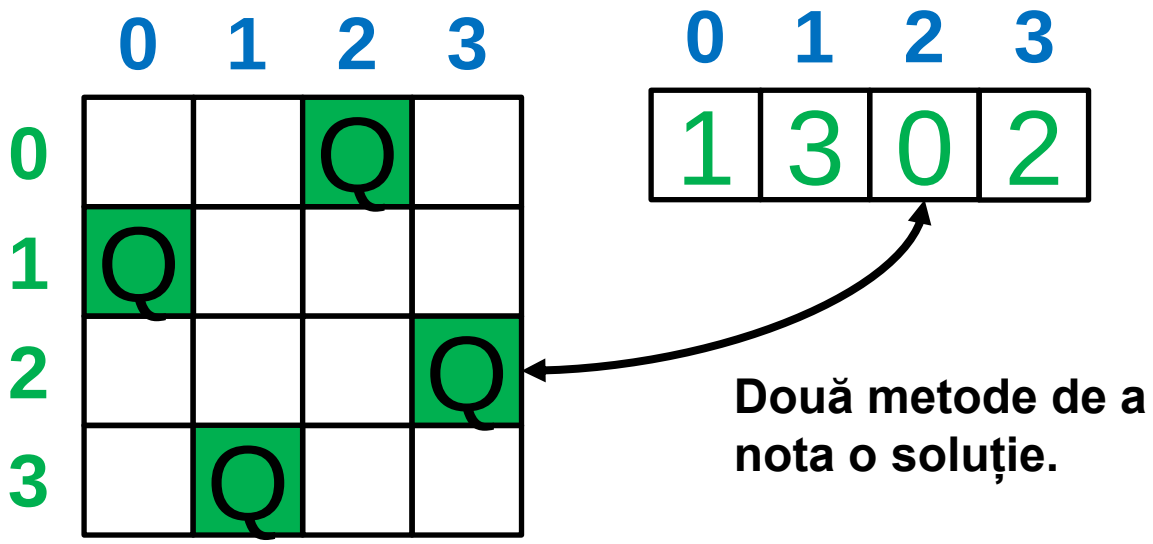


N Queens Problem

Nu avem voie mai mult de o regină pe diagonală



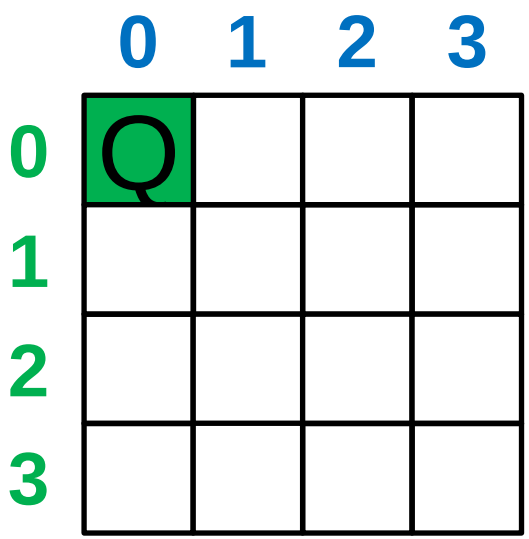
N Queens Problem

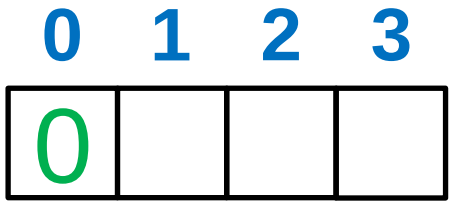


Poziția din vector reprezintă coloana pe care este așezată regina.

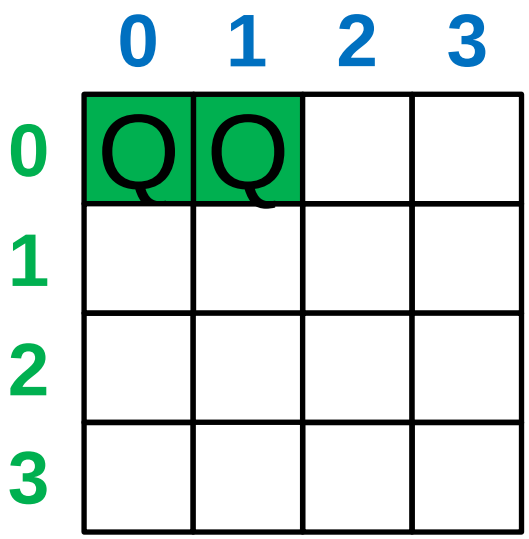
Valoarea din vector reprezintă linia pe care este așezată regina.

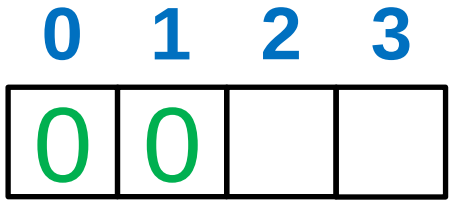
N Queens Problem – Soluție





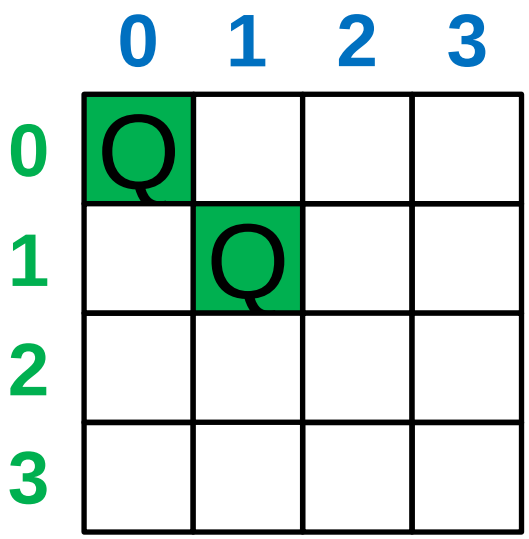
N Queens Problem – Soluție

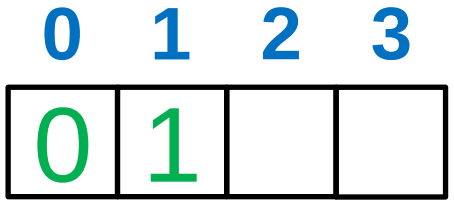




Conflict linie

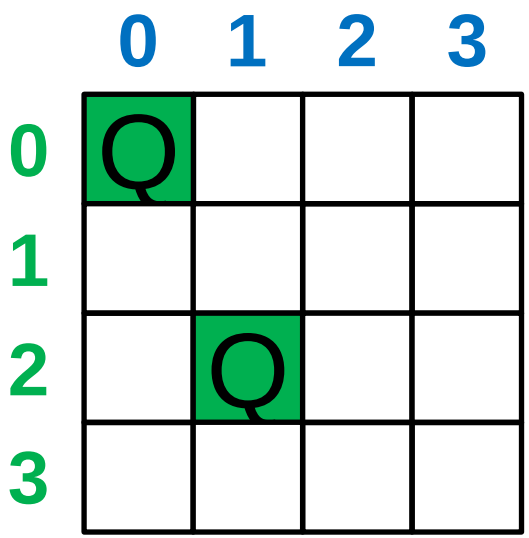
N Queens Problem – Soluție





Conflict diagonală

N Queens Problem – Soluție

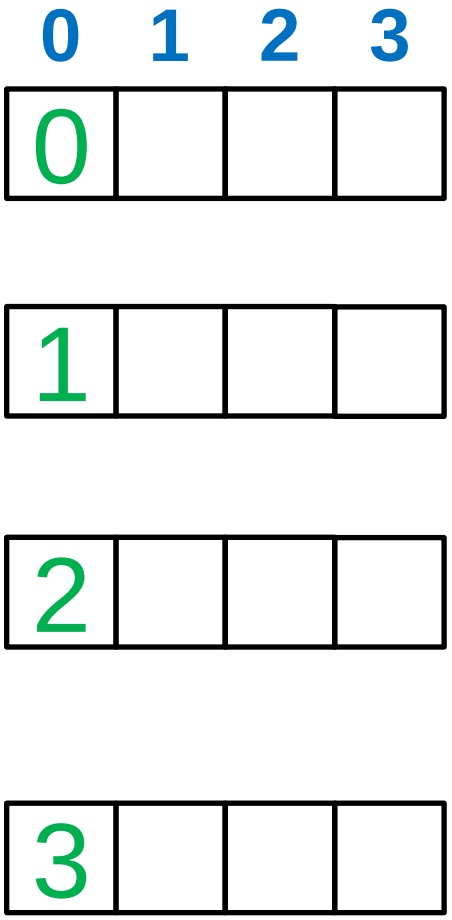


0	1	2	3
0	2		

OK.

Și tot așa până punem toate reginele

N Queens Problem – Soluție paralelă



N Queens Problem – Soluție paralelă

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

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

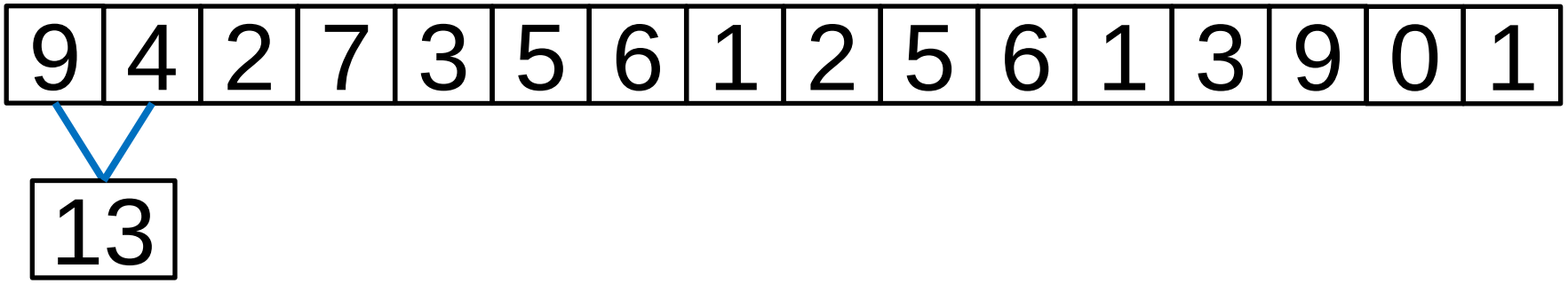
Și tot așa…

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

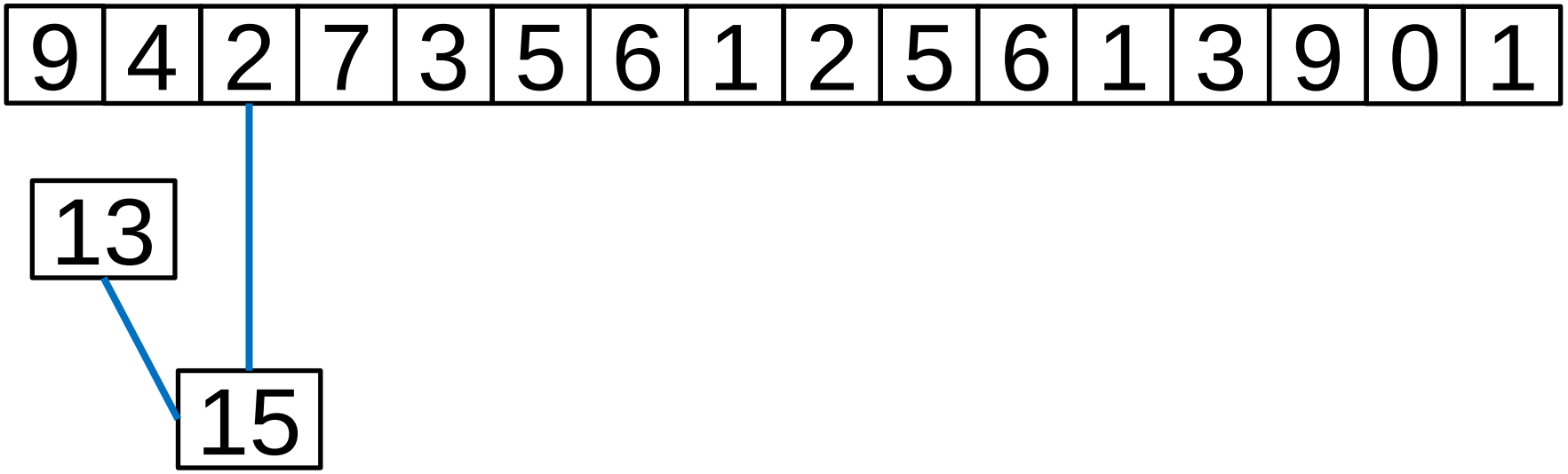
	0	1	2	3	
0	3	0			x
1	3	1			x
2	3	2			x
3	3	3			x

Abordarea log sau arbore

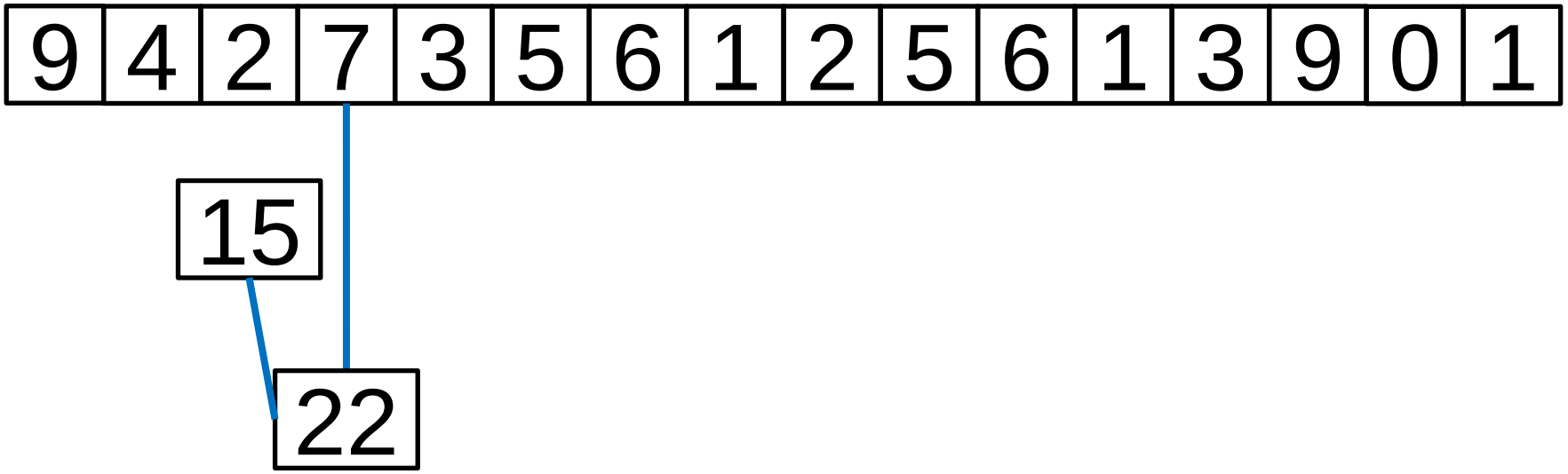
Sumă - clasic



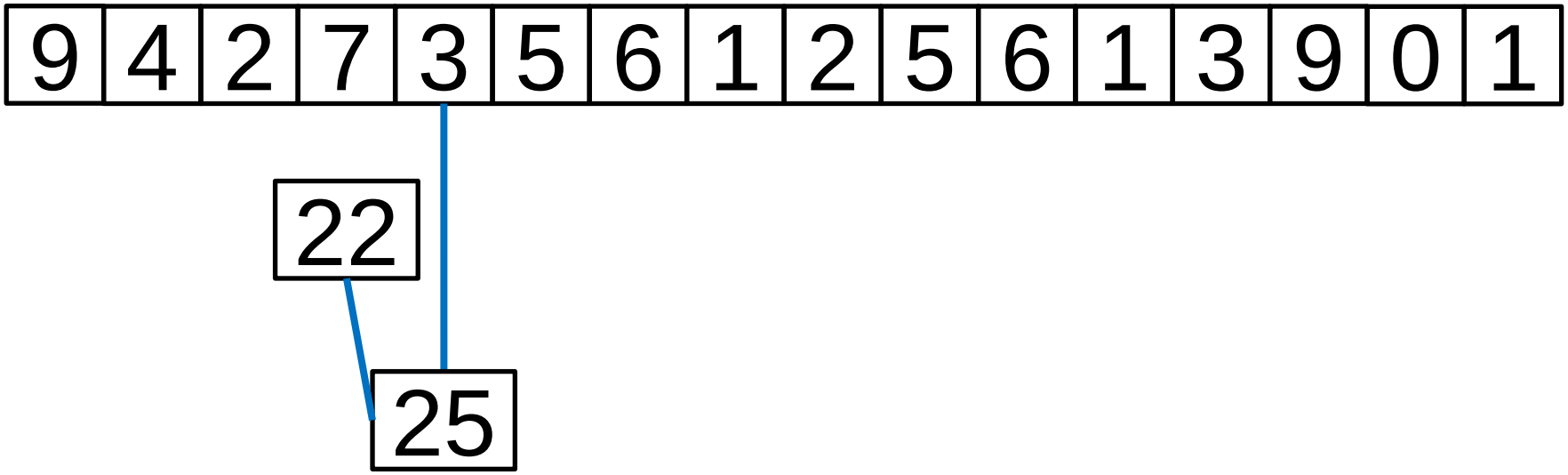
Sumă - clasic



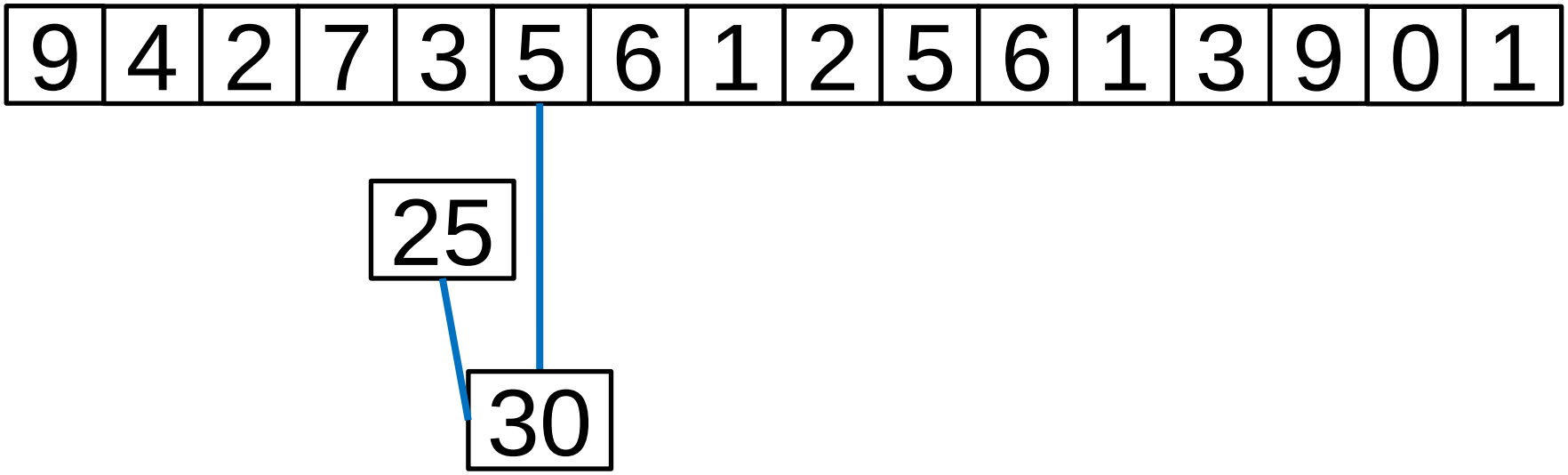
Sumă - clasic



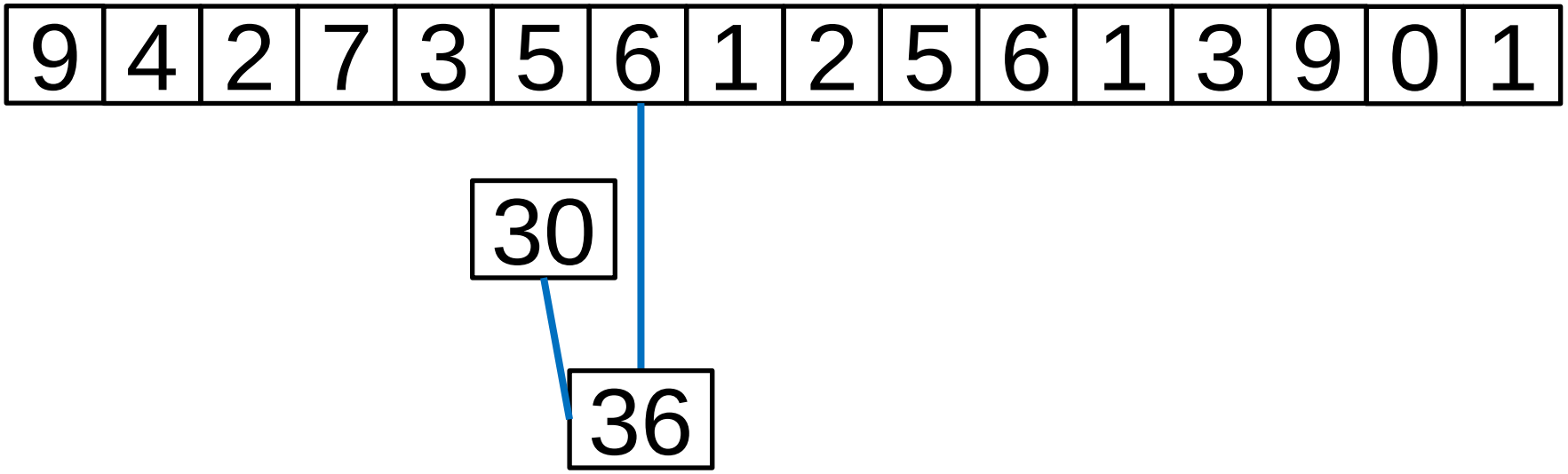
Sumă - clasic



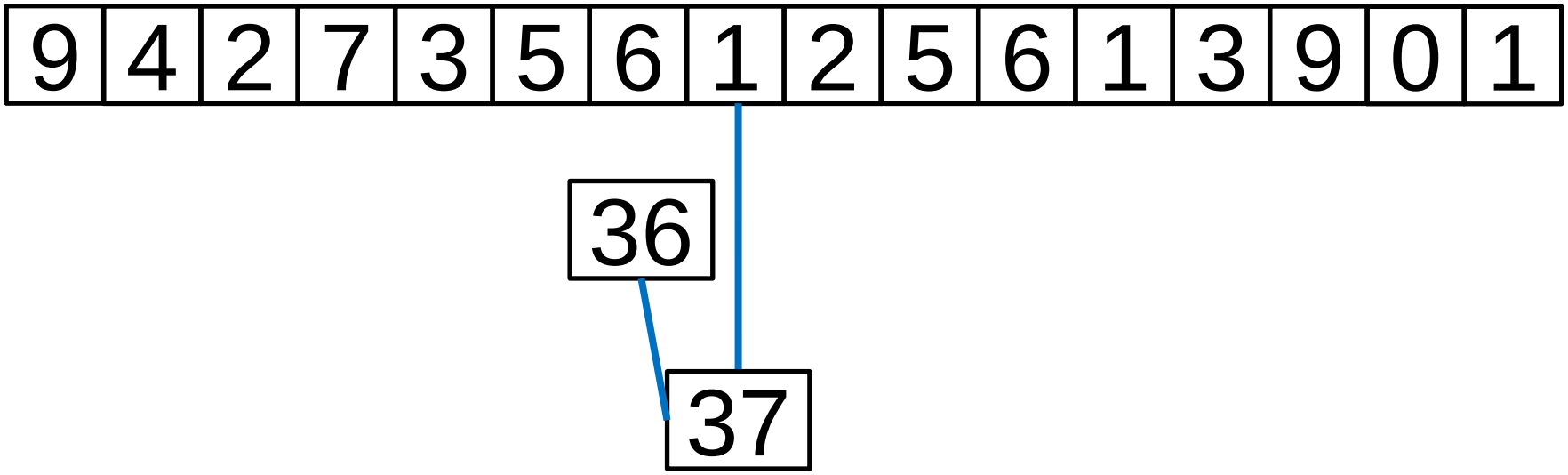
Sumă - clasic



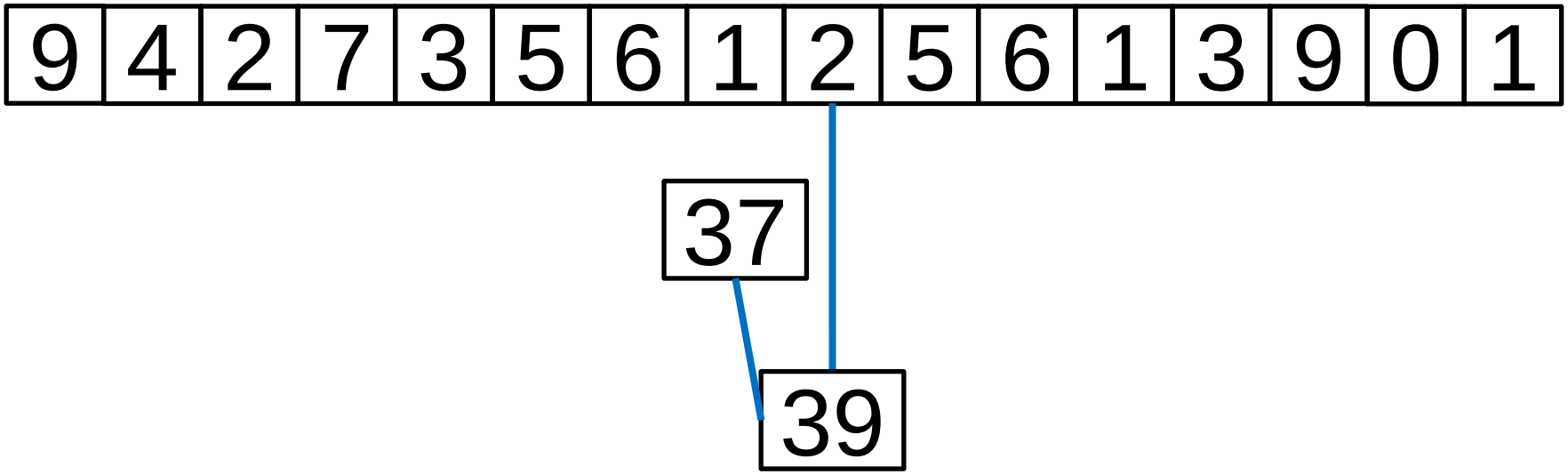
Sumă - clasic



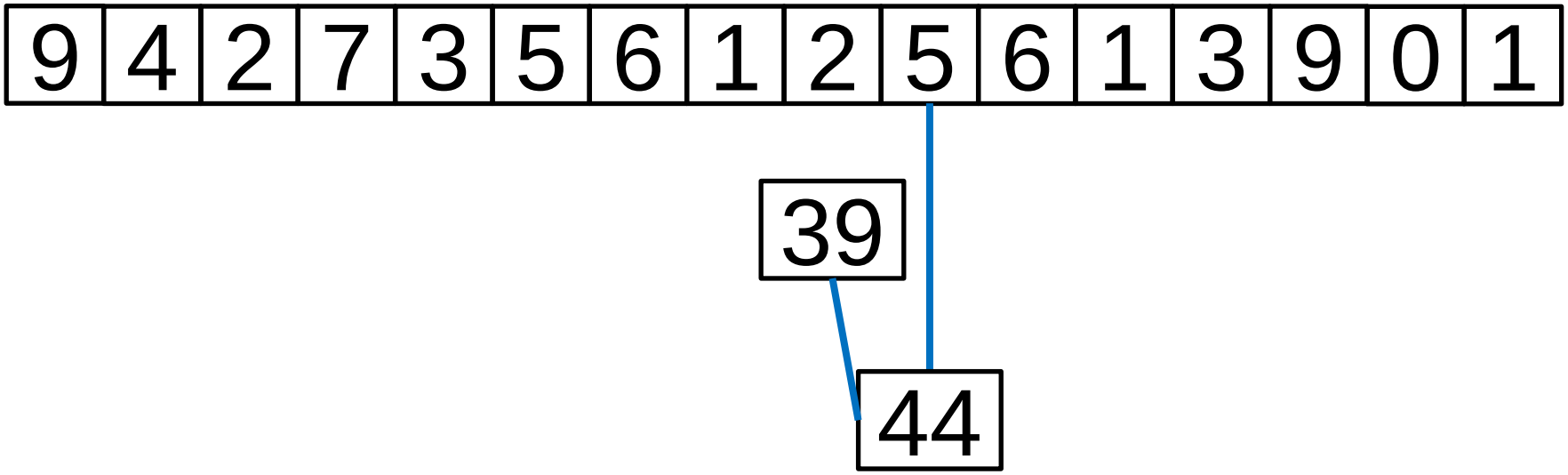
Sumă - clasic



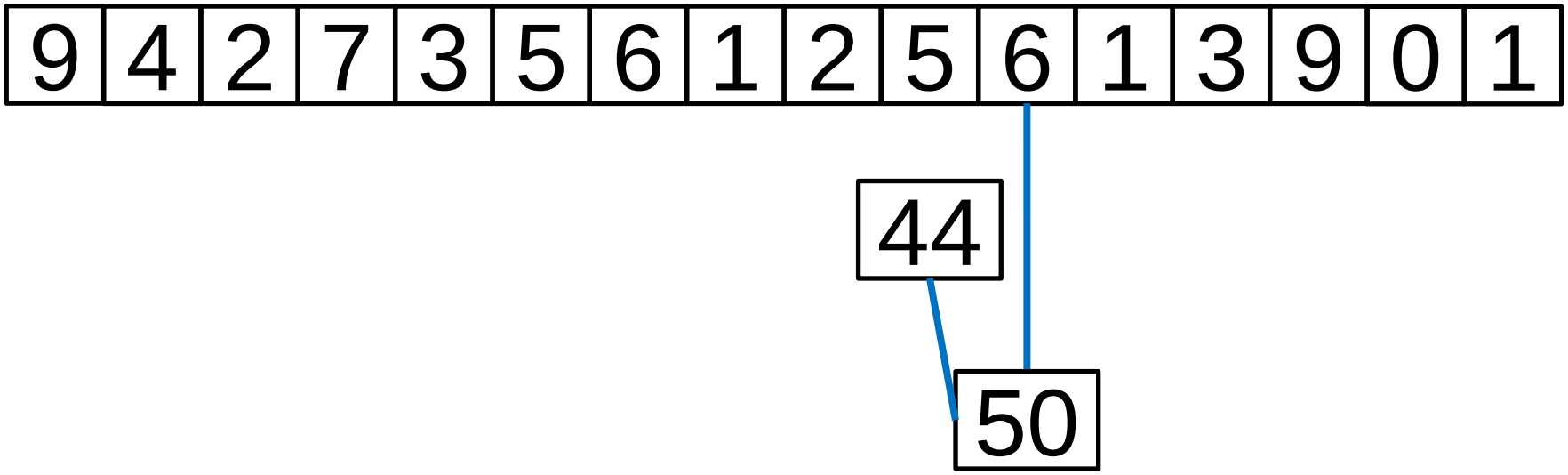
Sumă - clasic



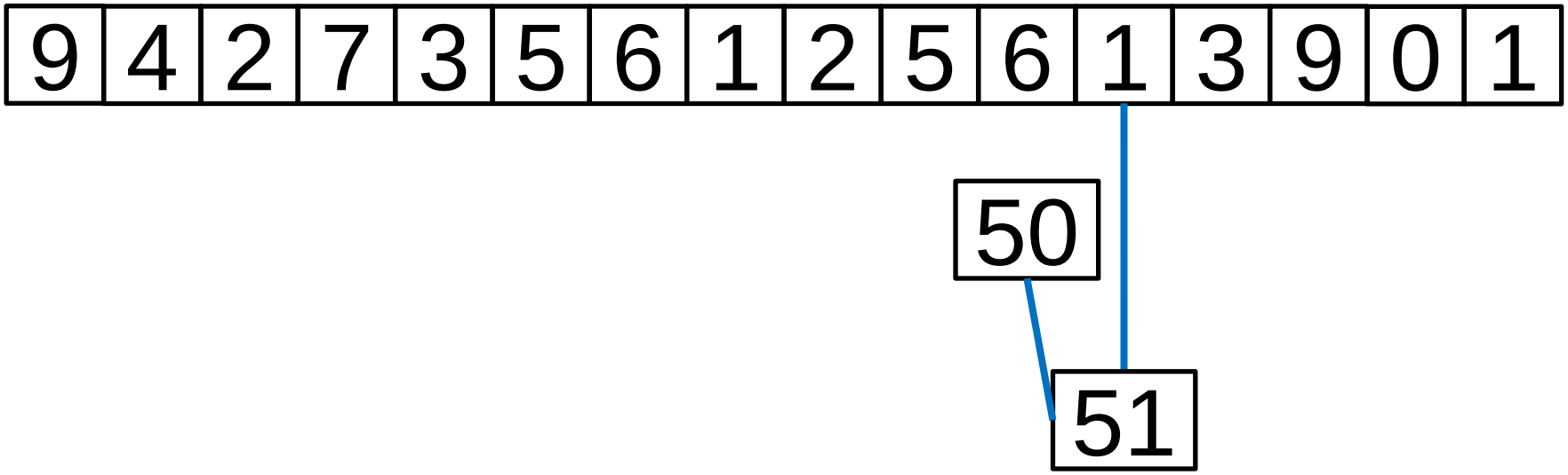
Sumă - clasic



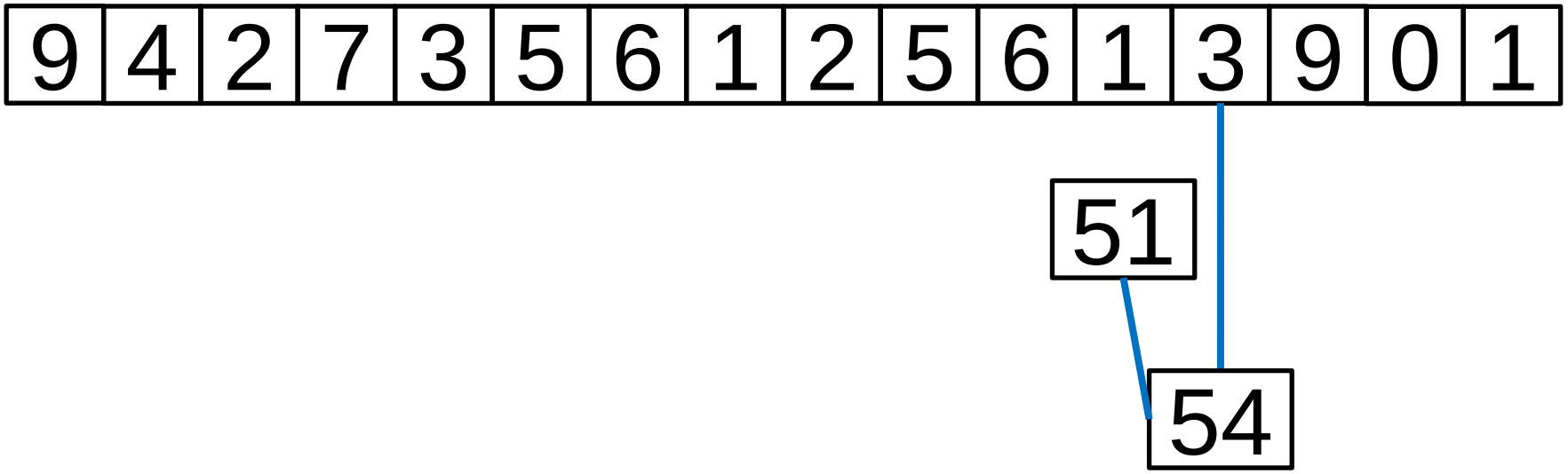
Sumă - clasic



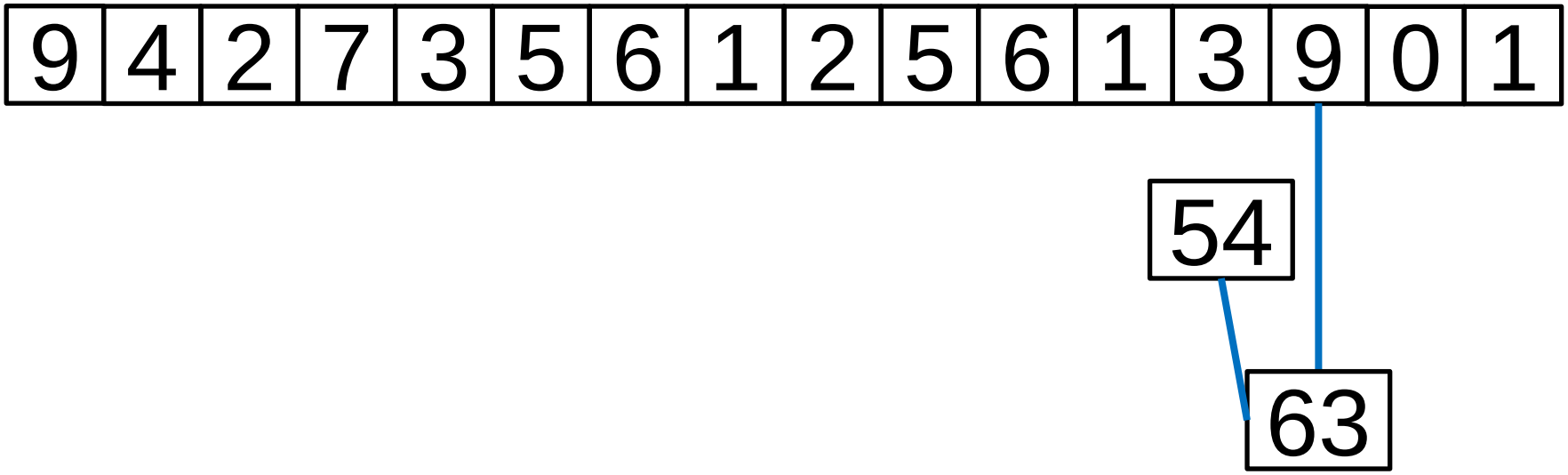
Sumă - clasic



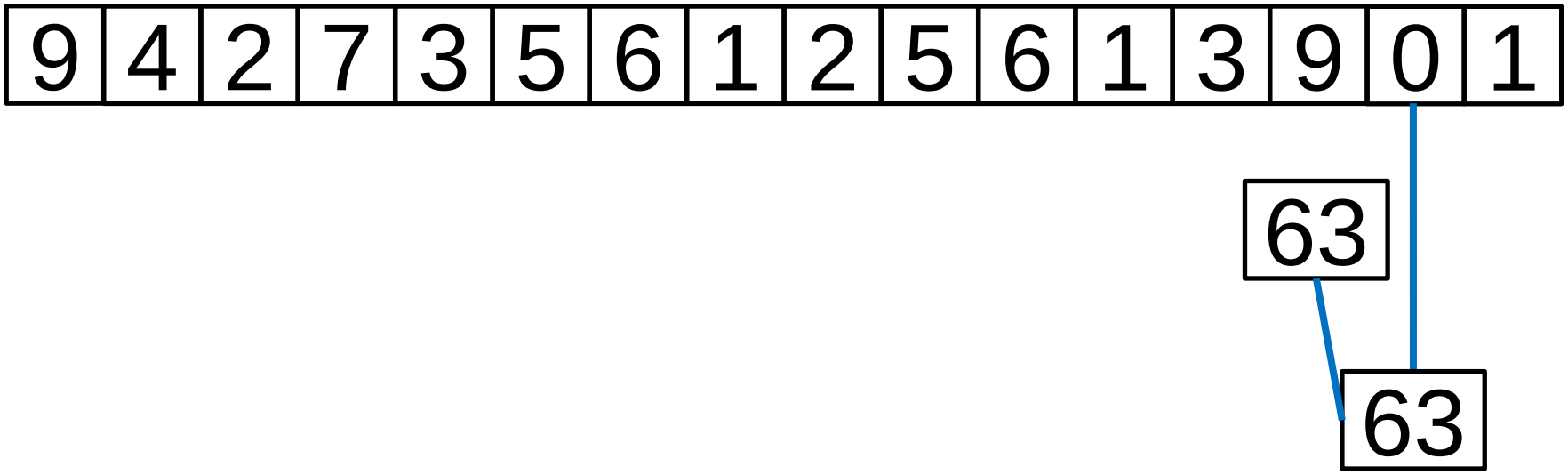
Sumă - clasic



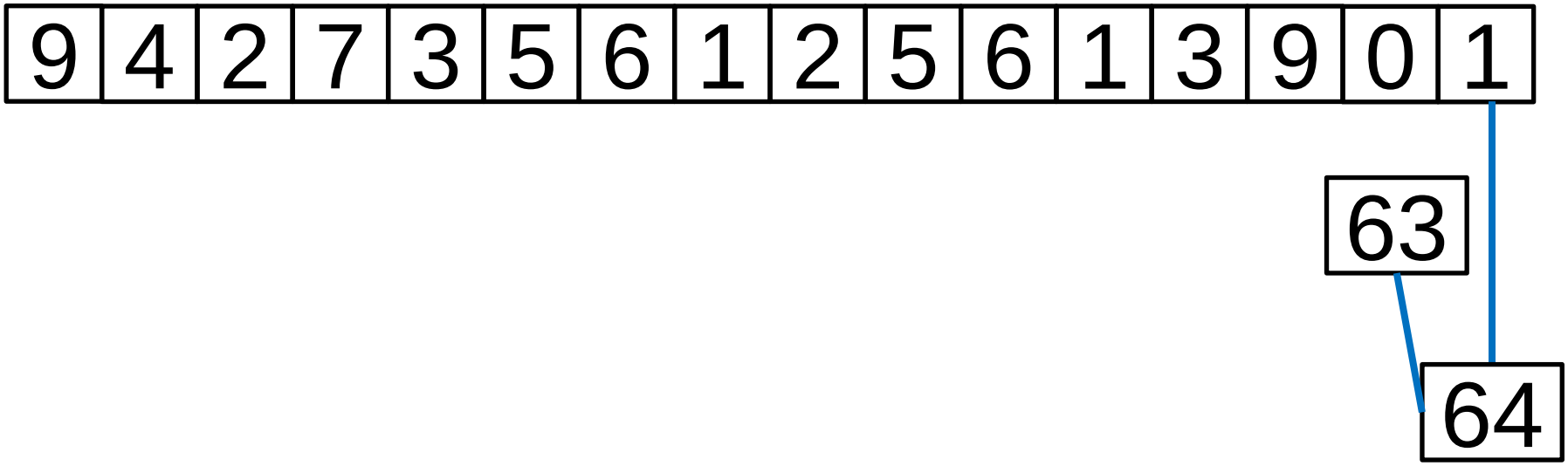
Sumă - clasic



Sumă - clasic



Sumă - clasic



Sumă - clasic

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

Complexitate O(N)

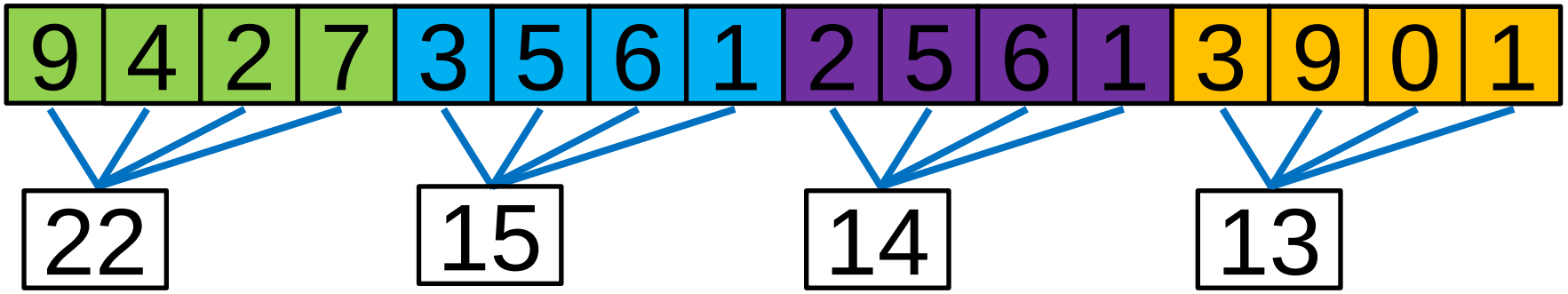
64

Sumă - paralelizare

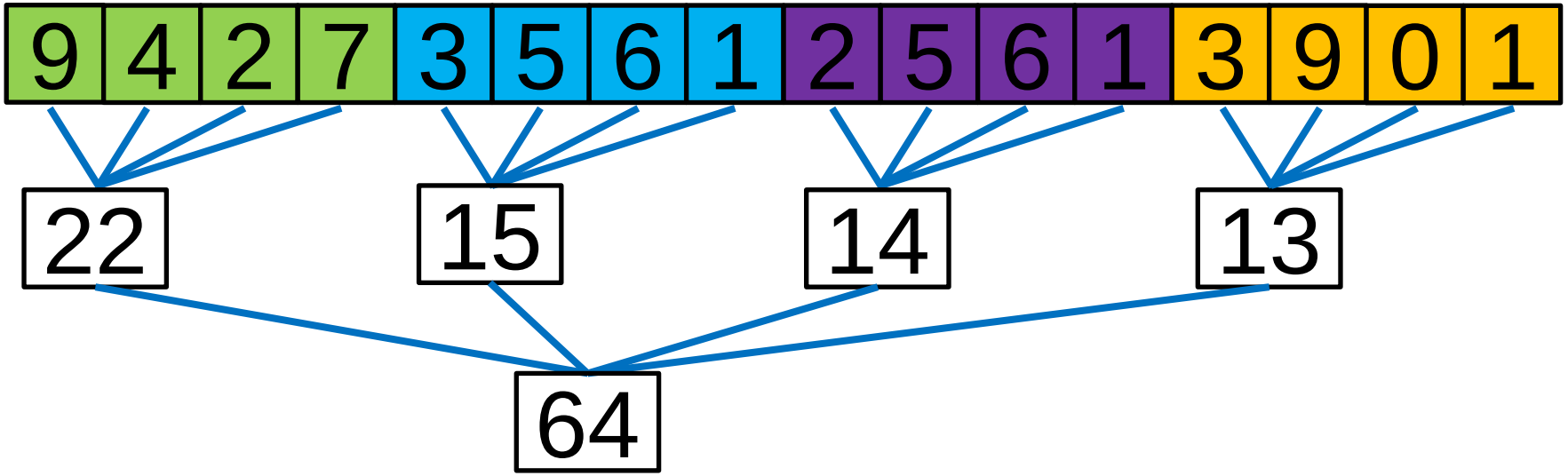
9	4	2	7	3	5	6	1	2	5	6	1	3	9	0	1

Sumă - paralelizare

Sumă - paralelizare

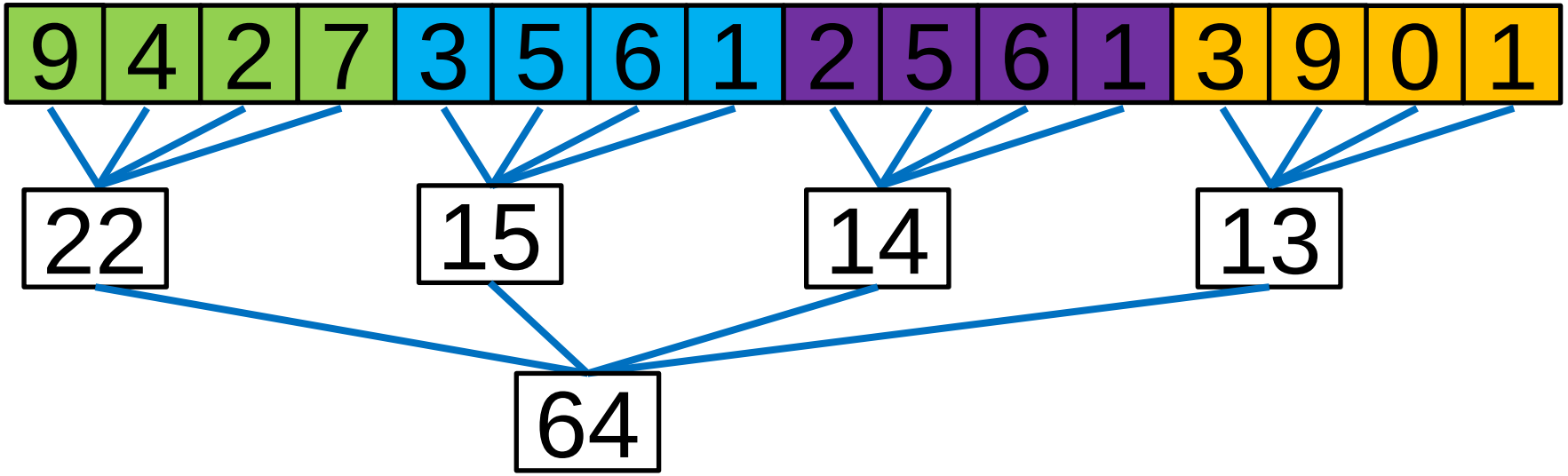


Sumă - paralelizare



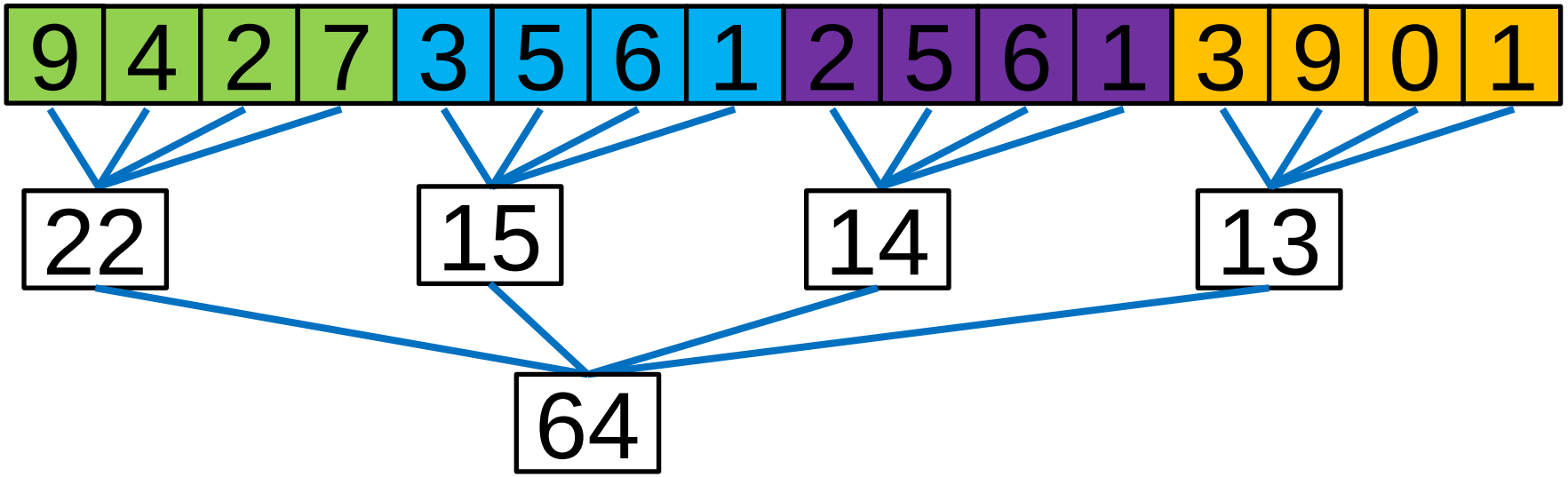
Complexitate?

Sumă - paralelizare



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

Sumă - paralelizare



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

Dar dacă P foarte mare?

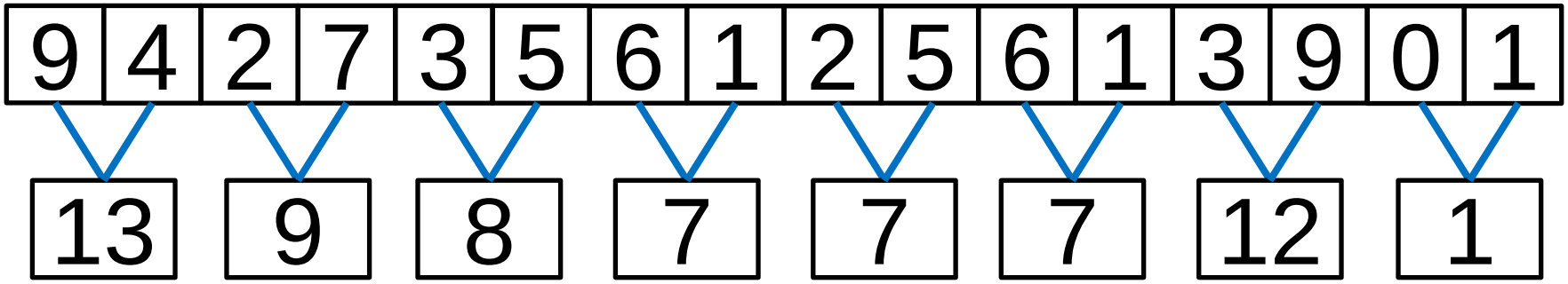
Reduce

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.

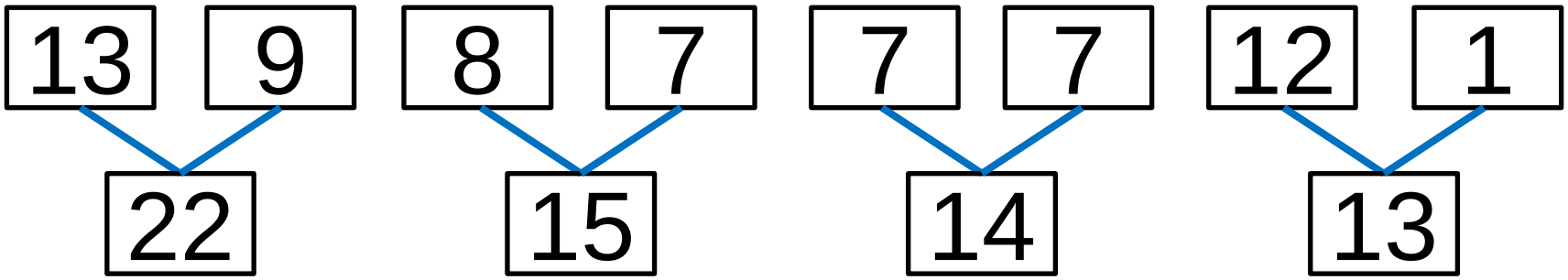
Operația trebuie să fie comutativă de exemplu:
+, *, min, max, and, ...

Reduce cu sumă



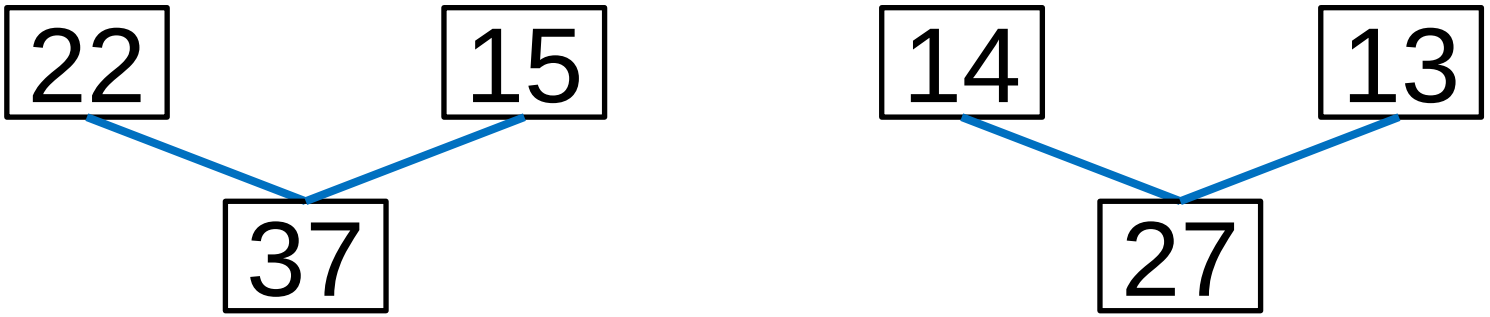
Pot fi executate în paralel

Reduce cu sumă



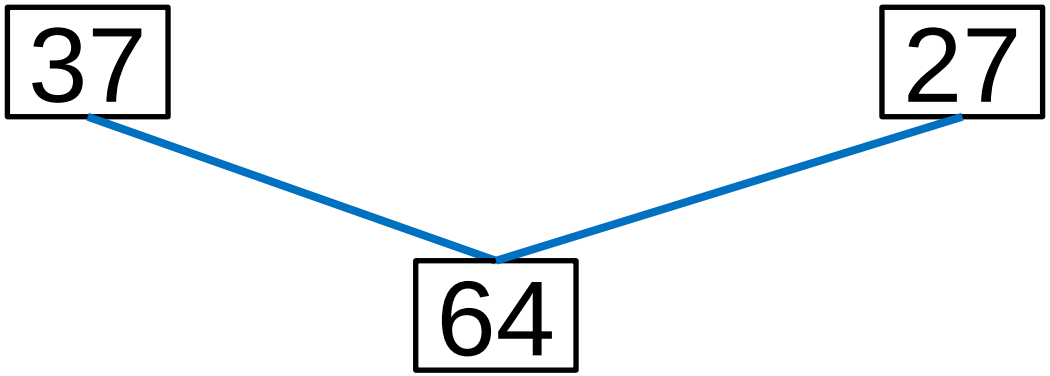
Pot fi executate în paralel

Reduce cu sumă

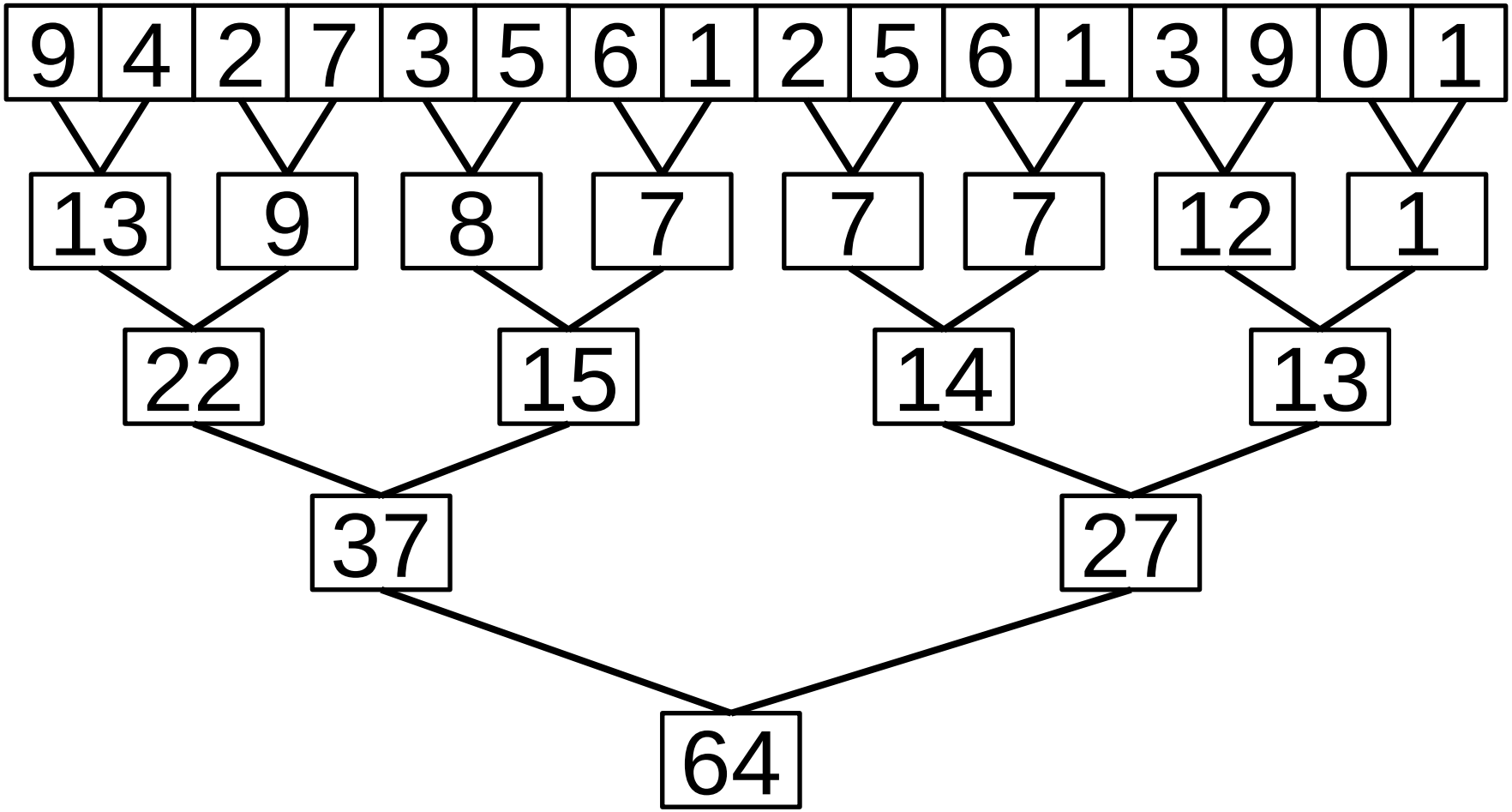


Pot fi executate în paralel

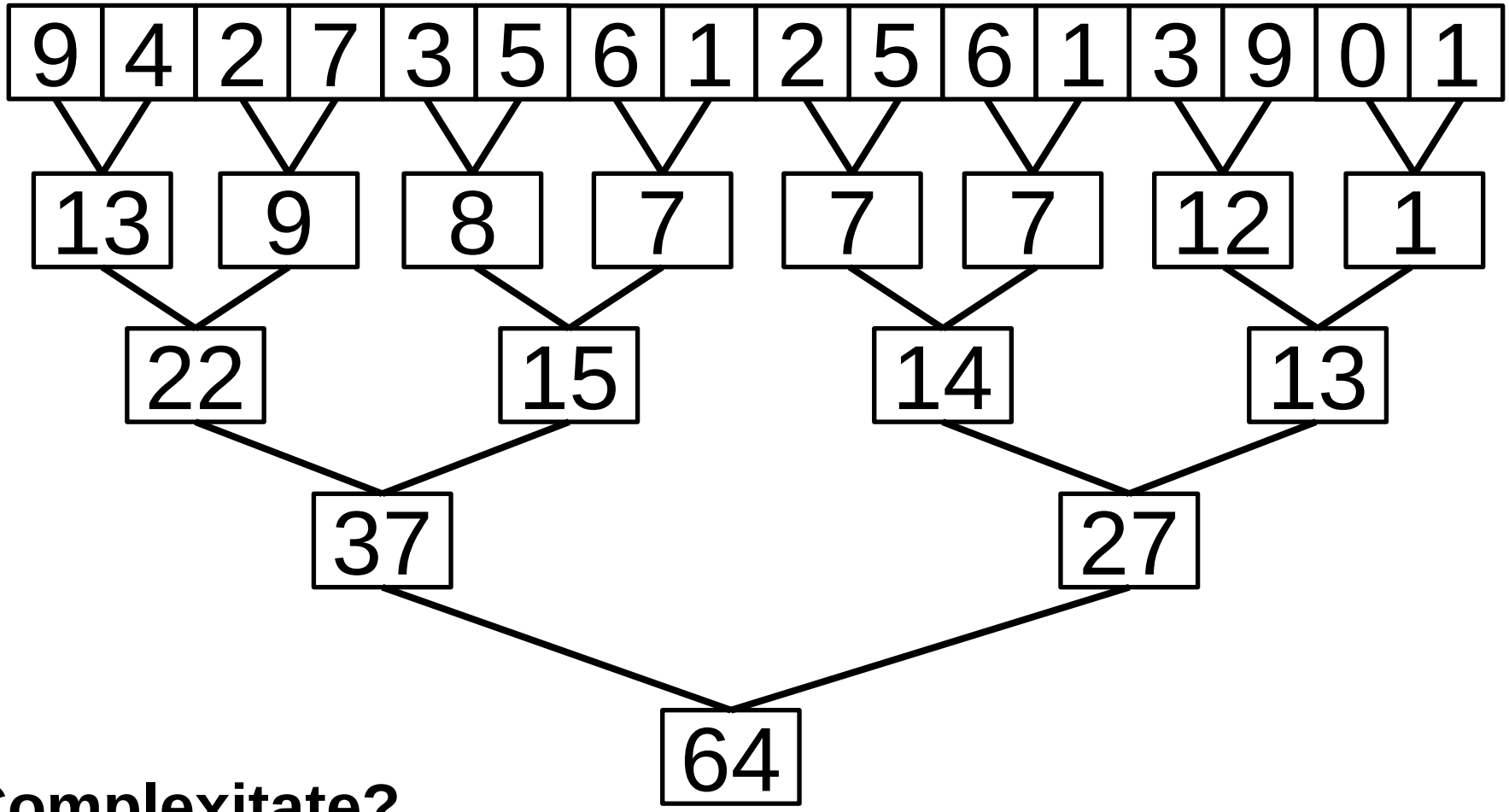
Reduce cu sumă



Reduce cu sumă

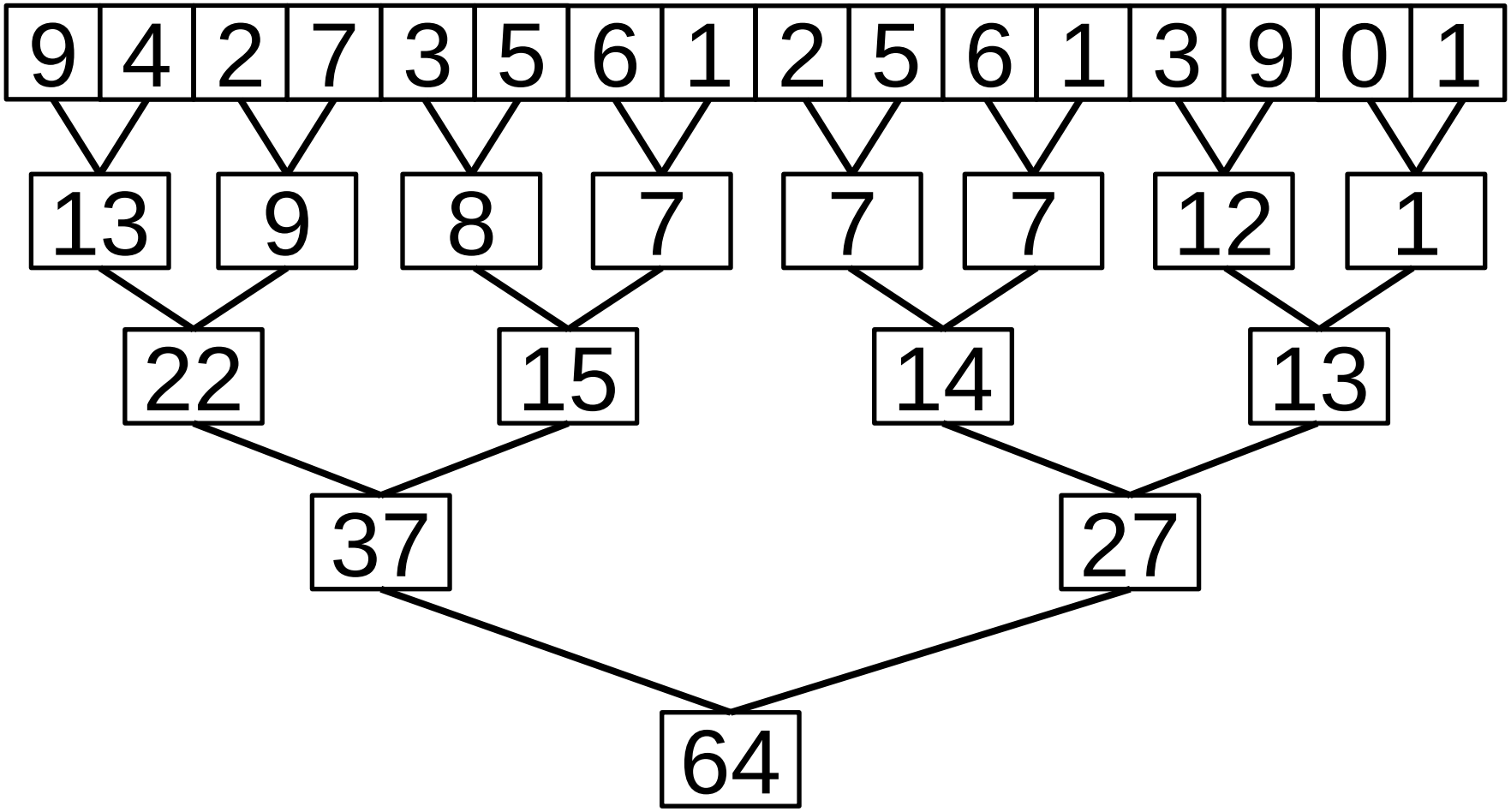


Reduce cu sumă



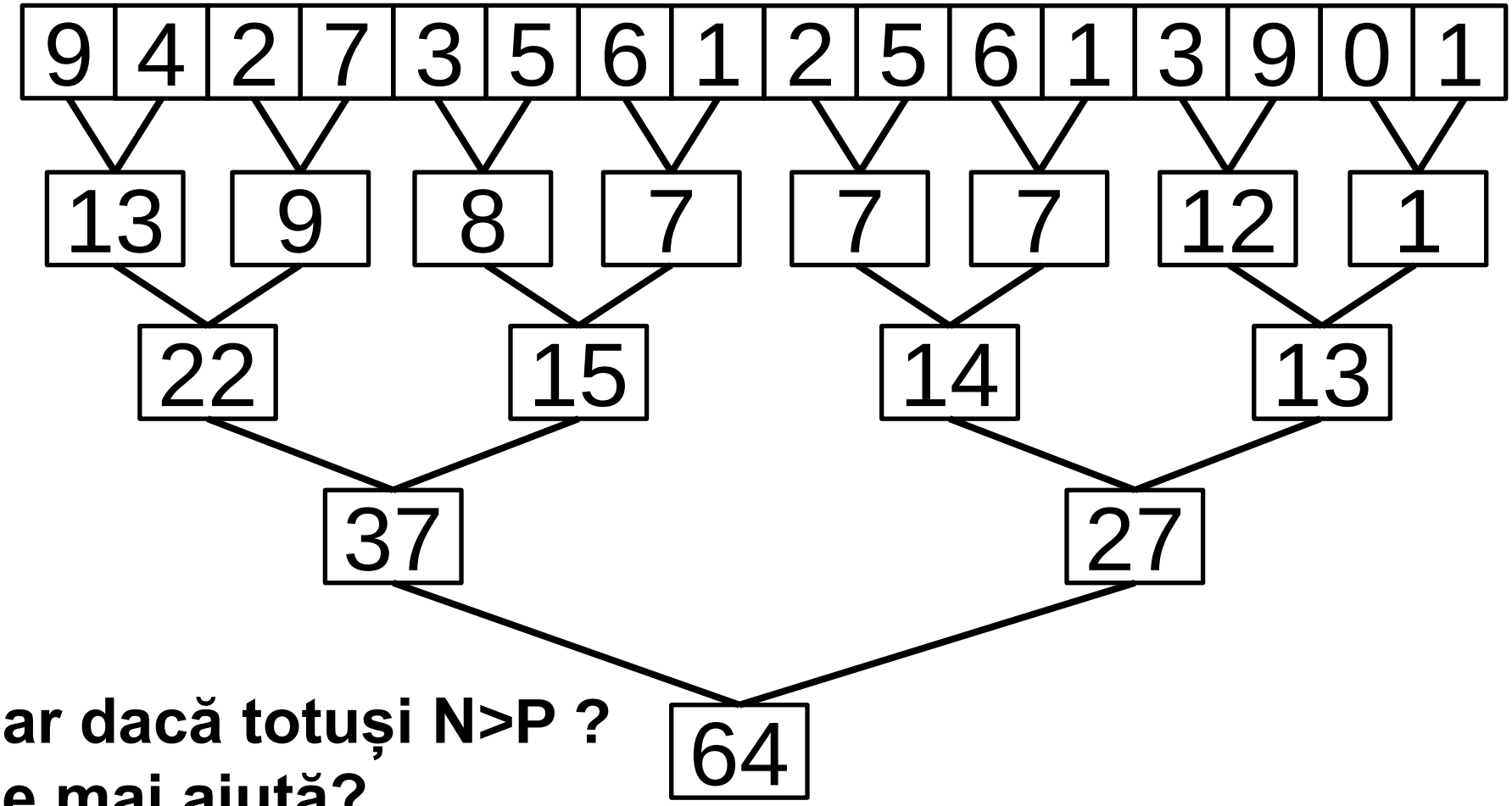
Complexitate?

Reduce cu sumă



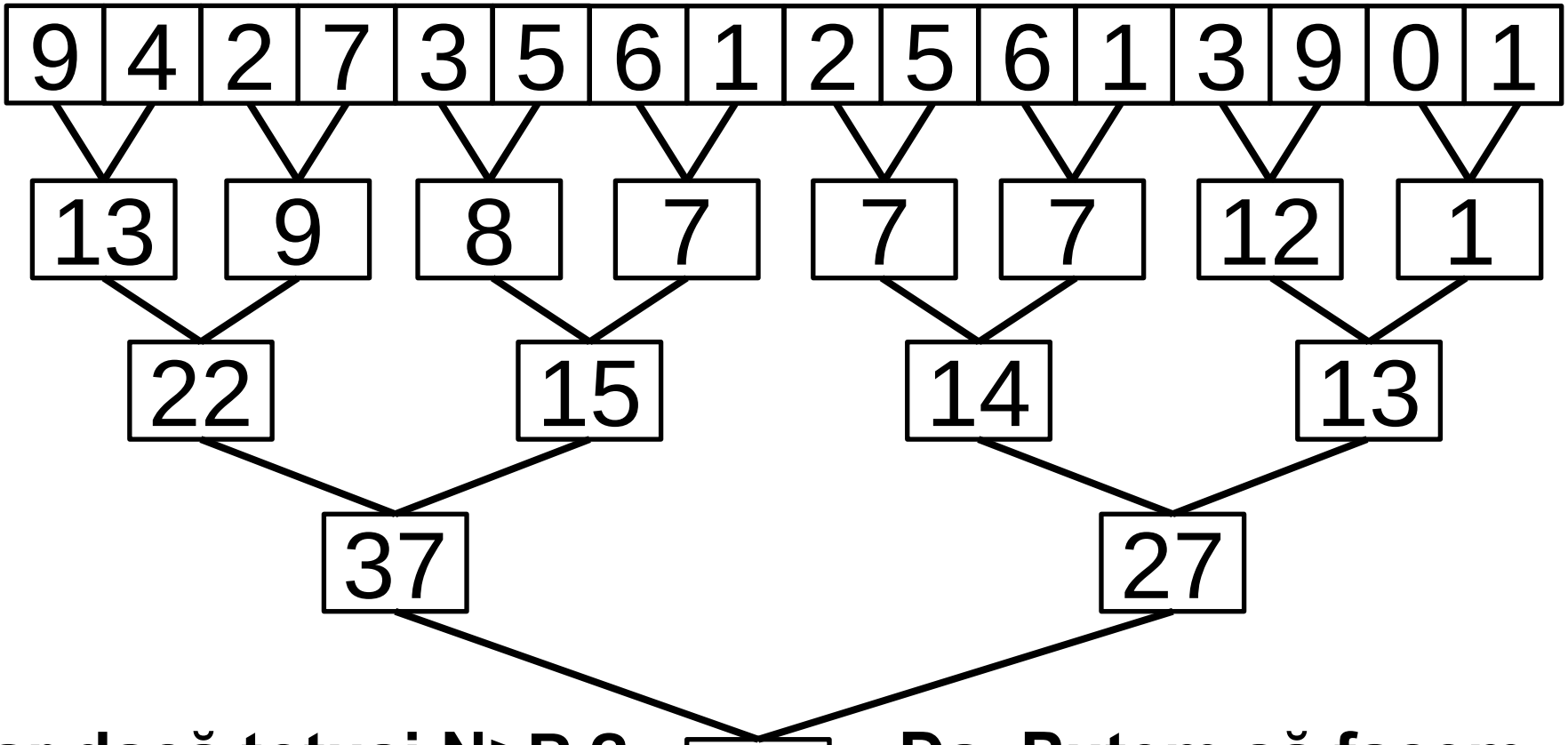
Complexitate: $O(log(N))$ dacă P=N

Reduce cu sumă



Dar dacă totuși N>P ?
Ne mai ajută?

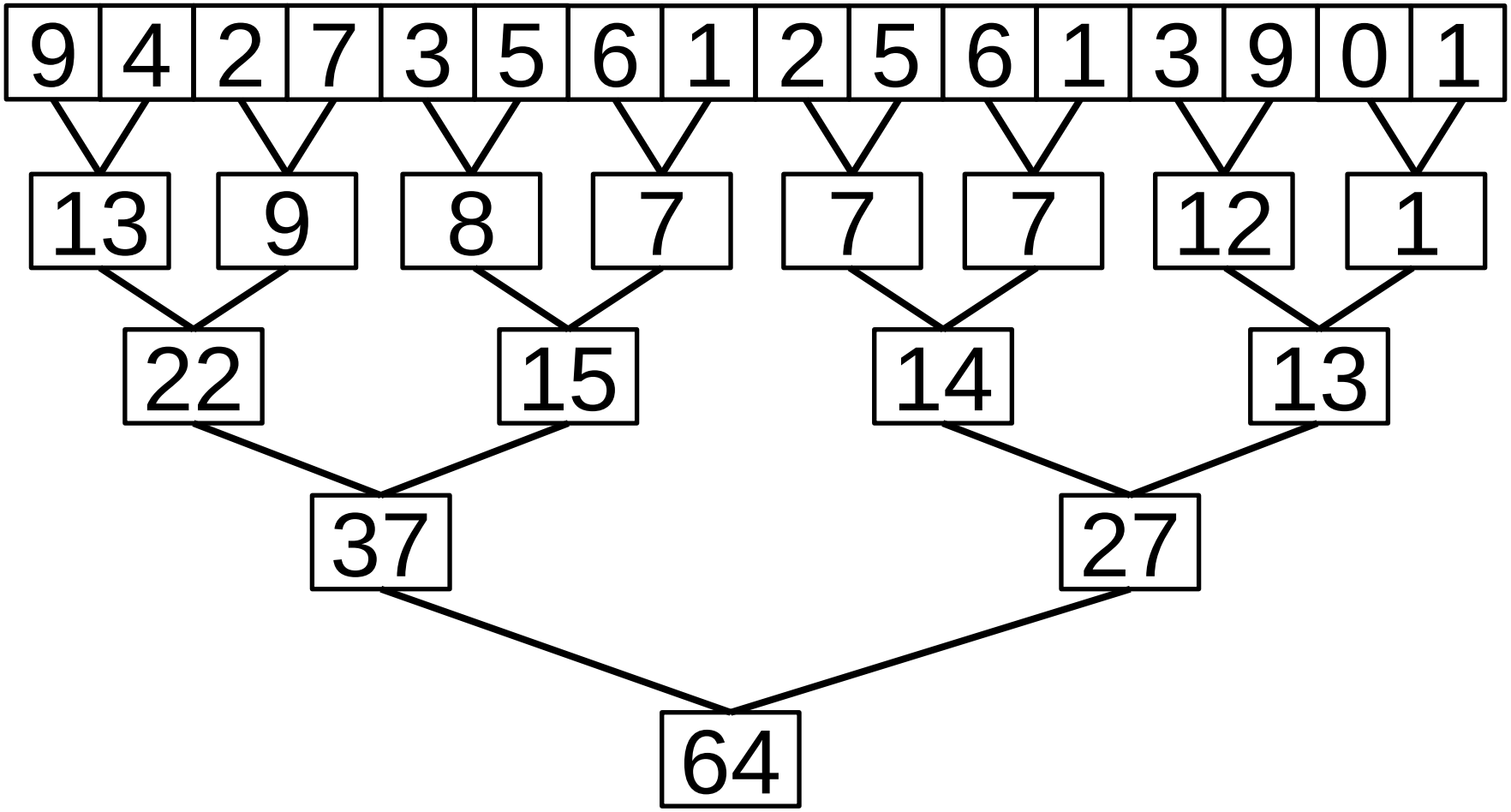
Reduce cu sumă



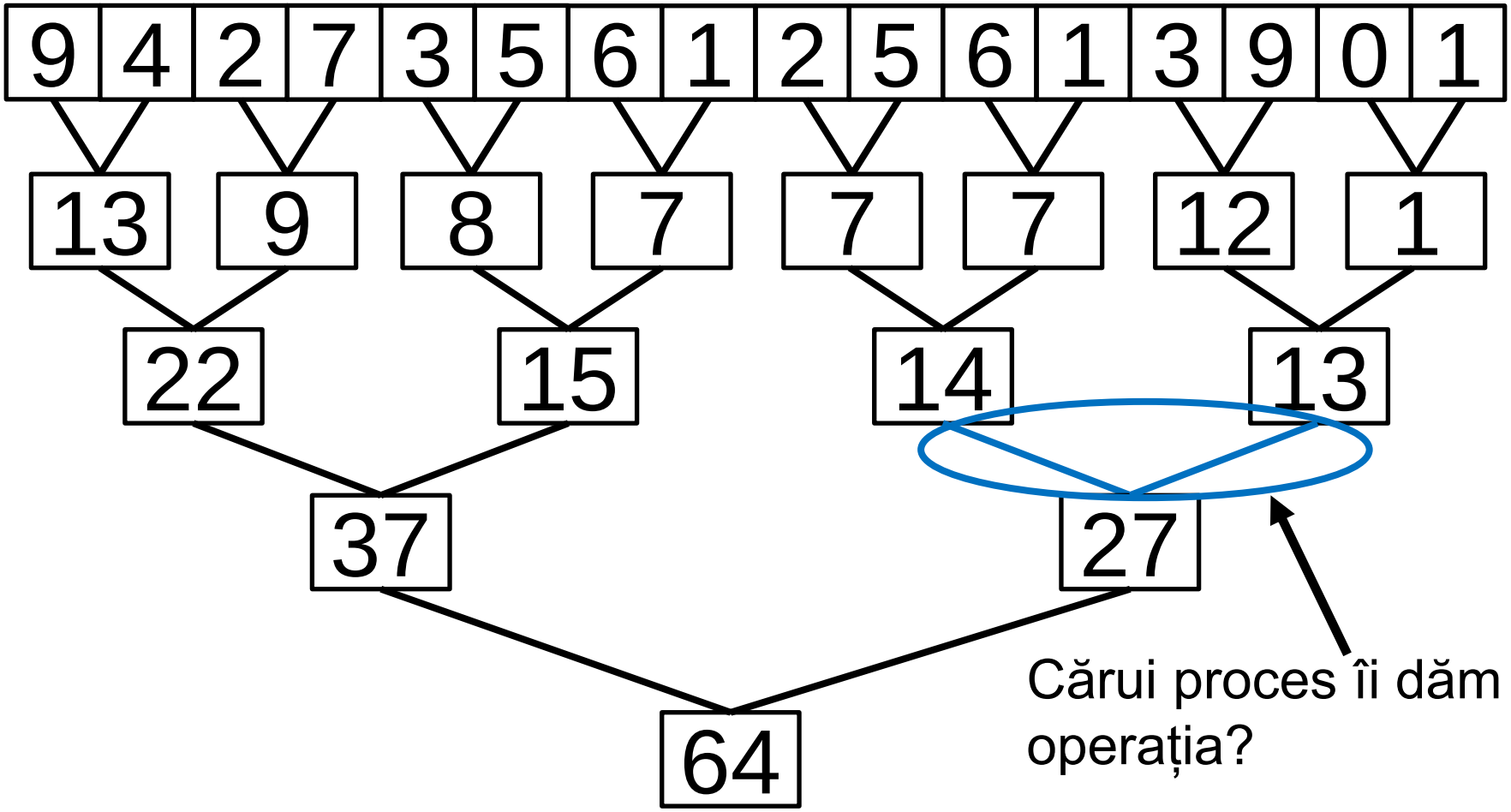
Dar dacă totuși N>P ?
Reducem la o problemă deja rezolvată

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

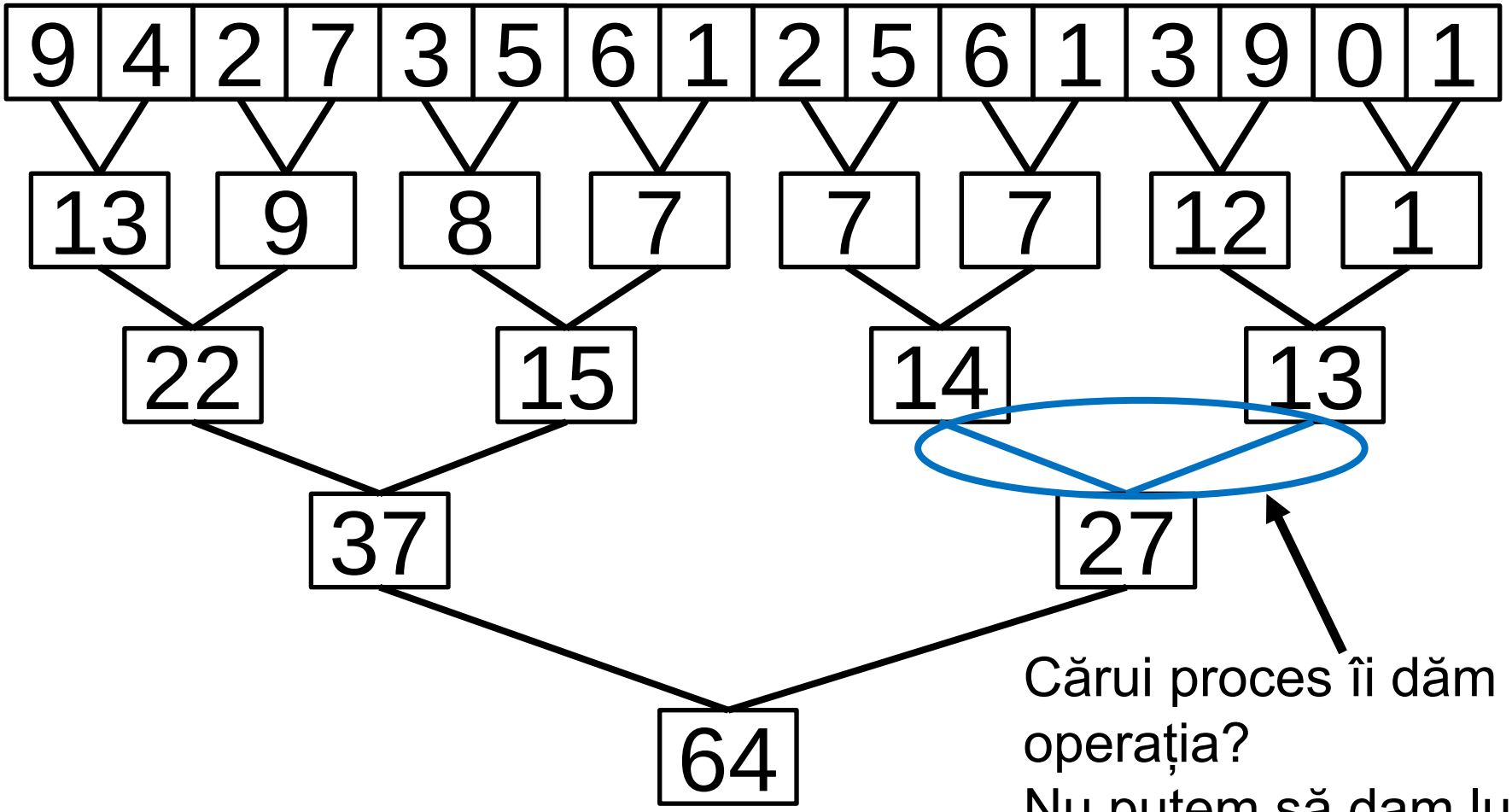
Reduce – implementare – model matematic



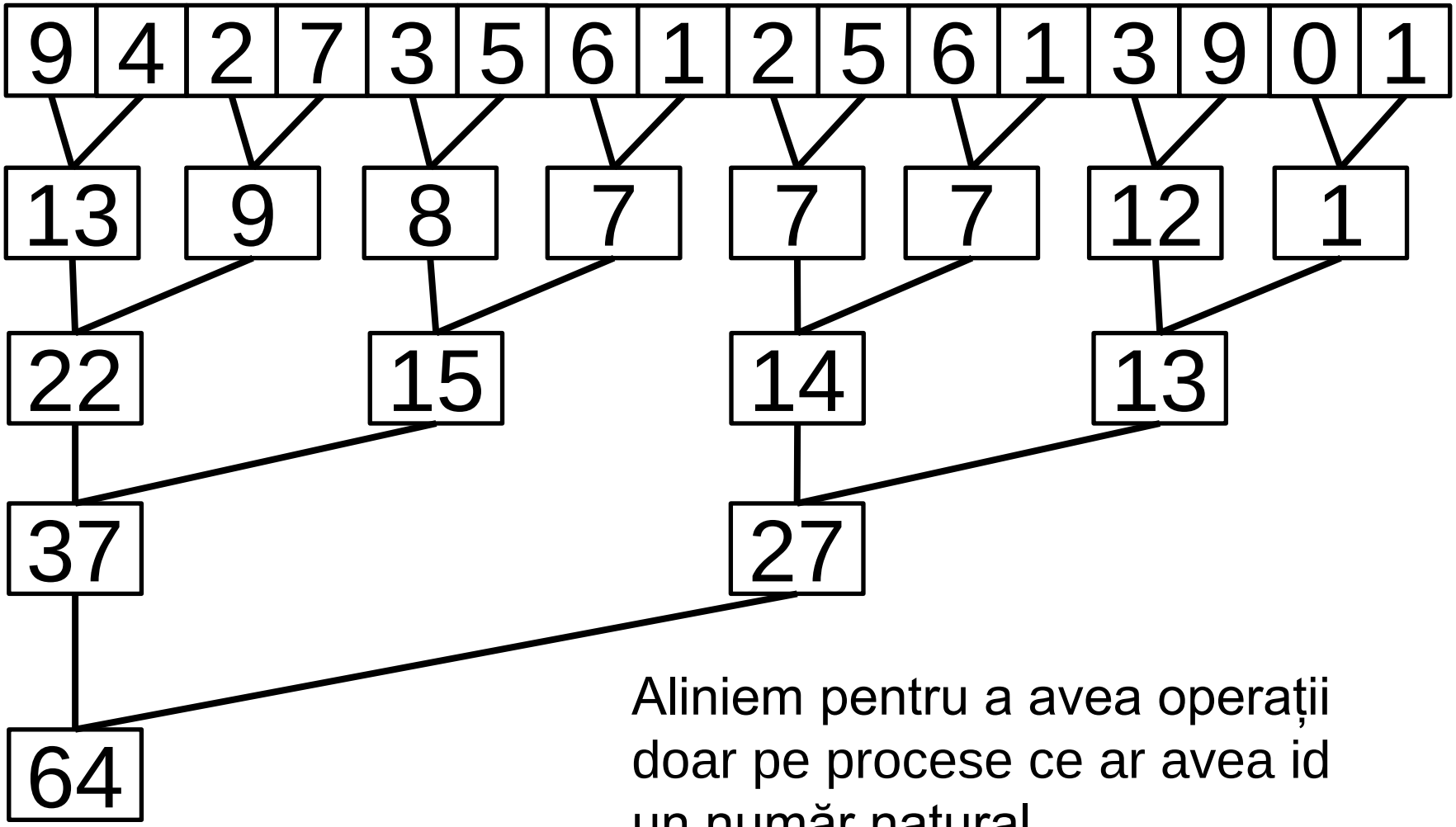
Reduce - implementare



Reduce - implementare

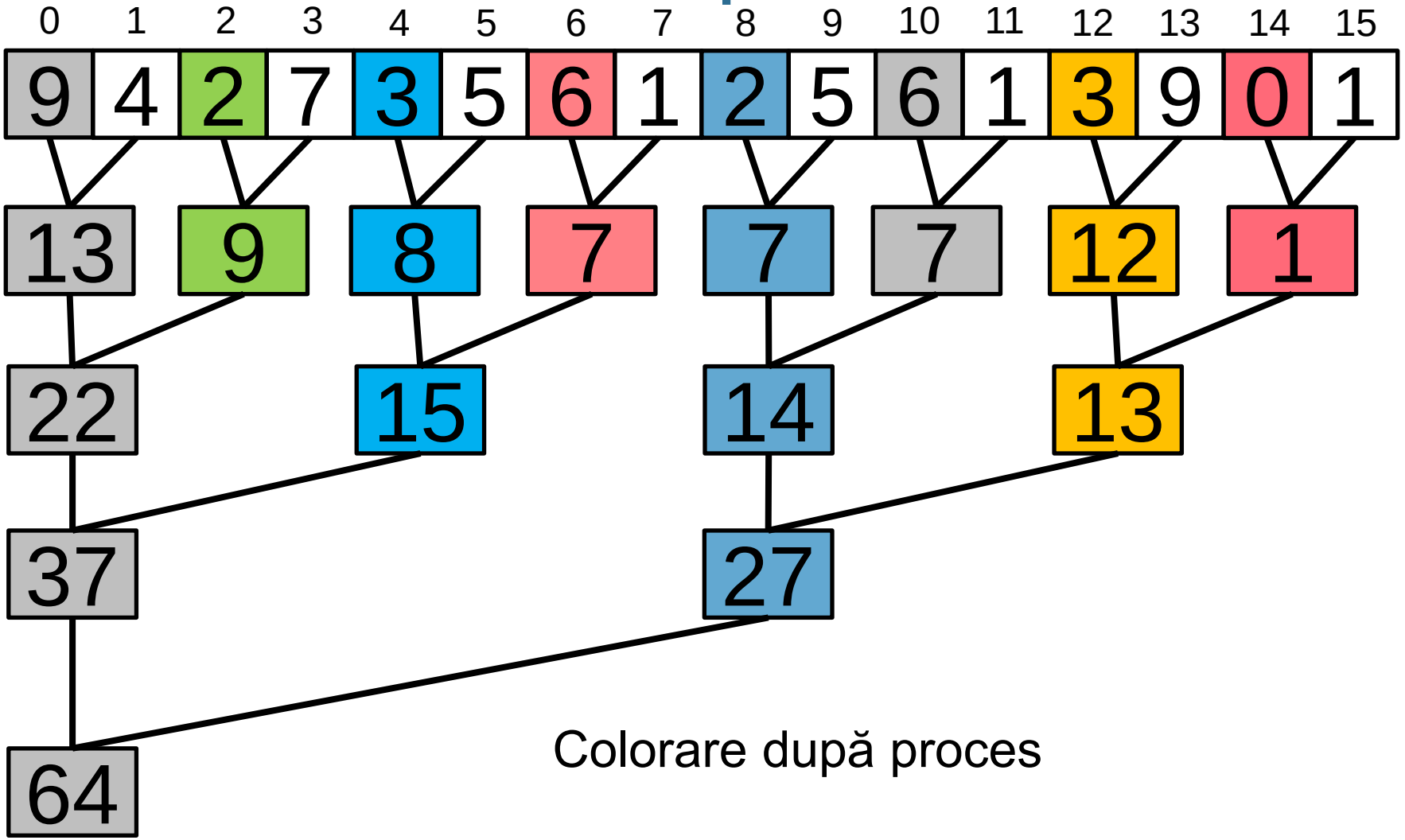


Reduce - implementare

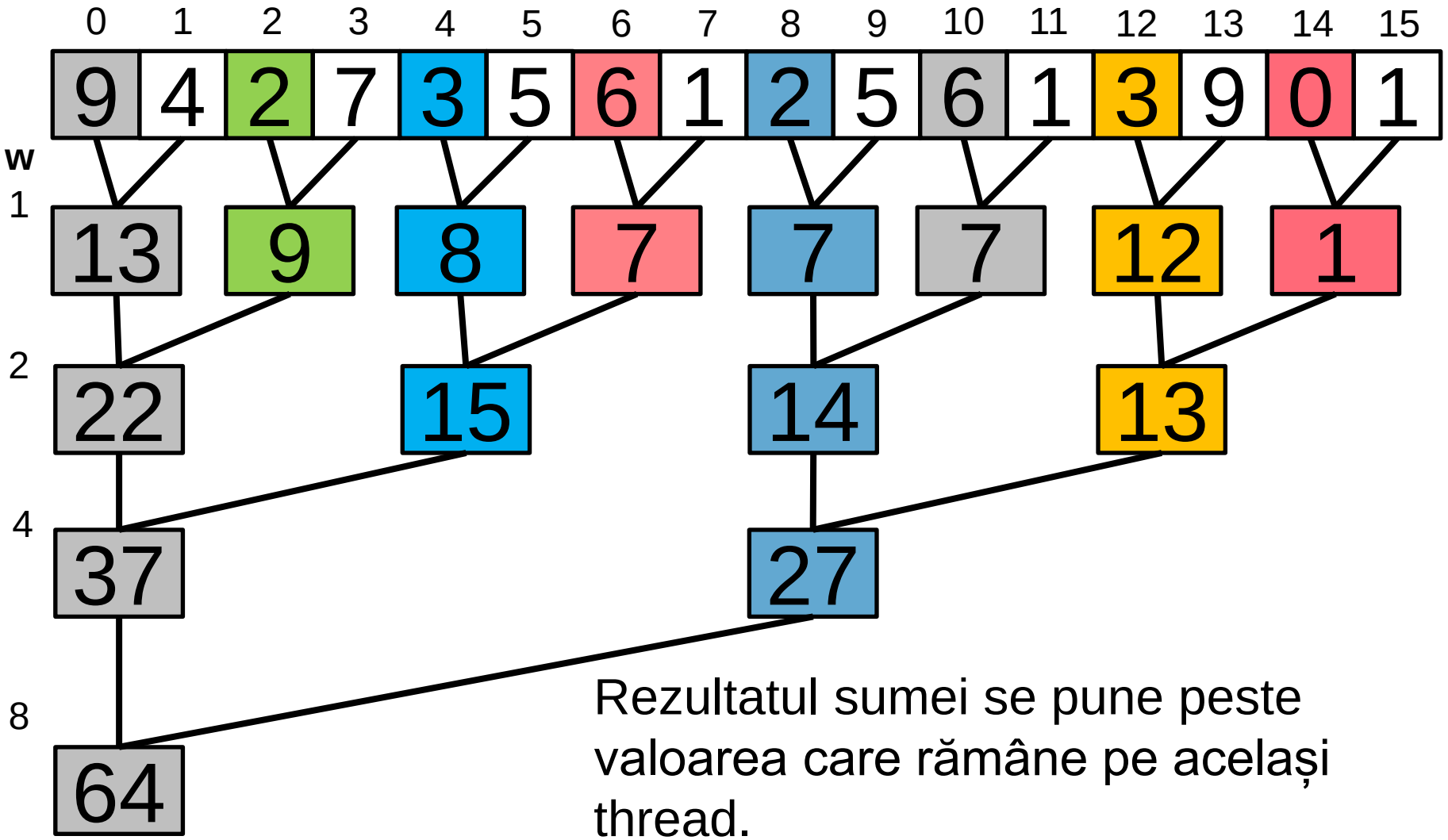


Aliniem pentru a avea operații doar pe procese ce ar avea id un număr natural

Reduce - implementare

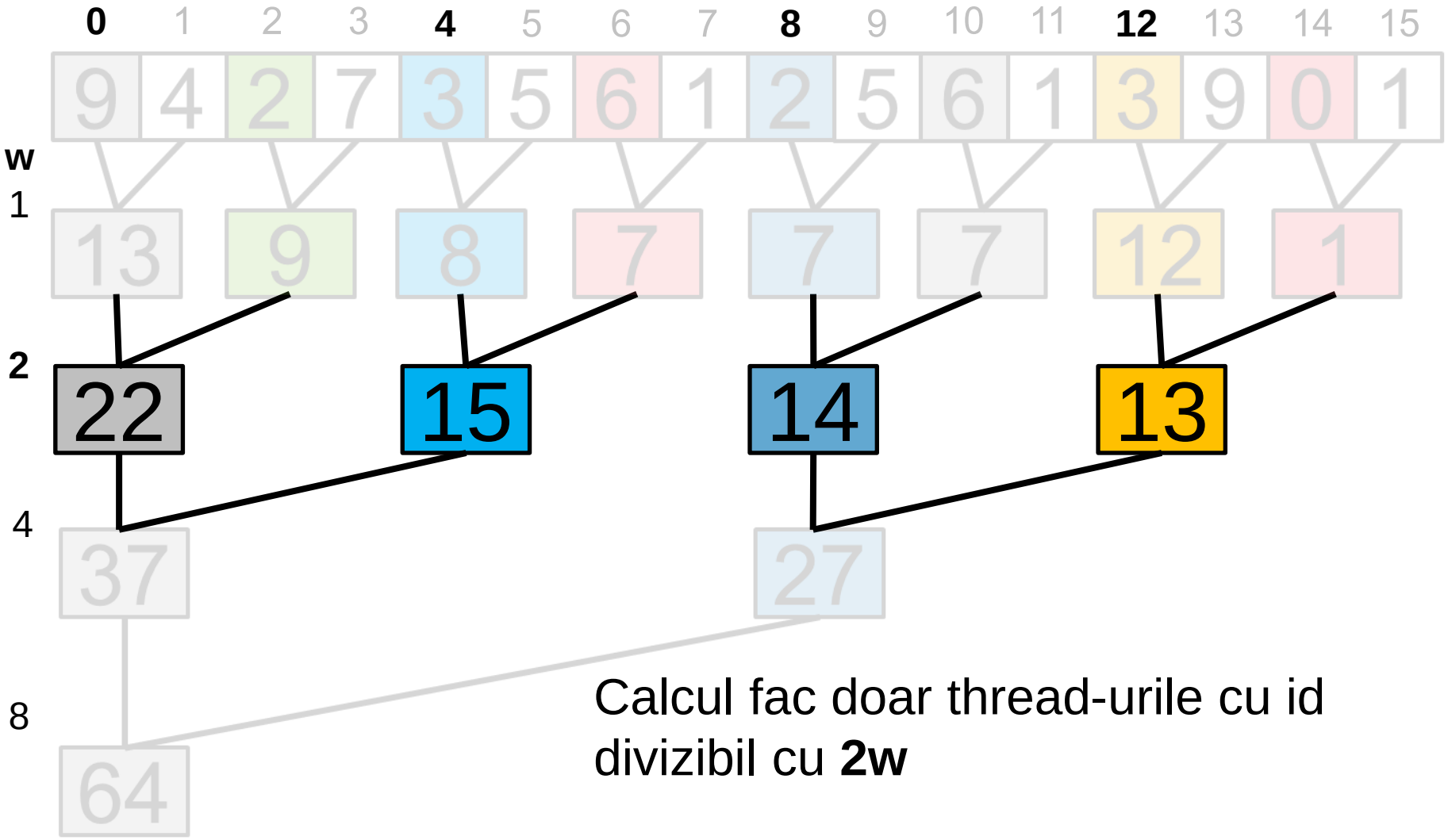


Reduce – implementare – imagine globală



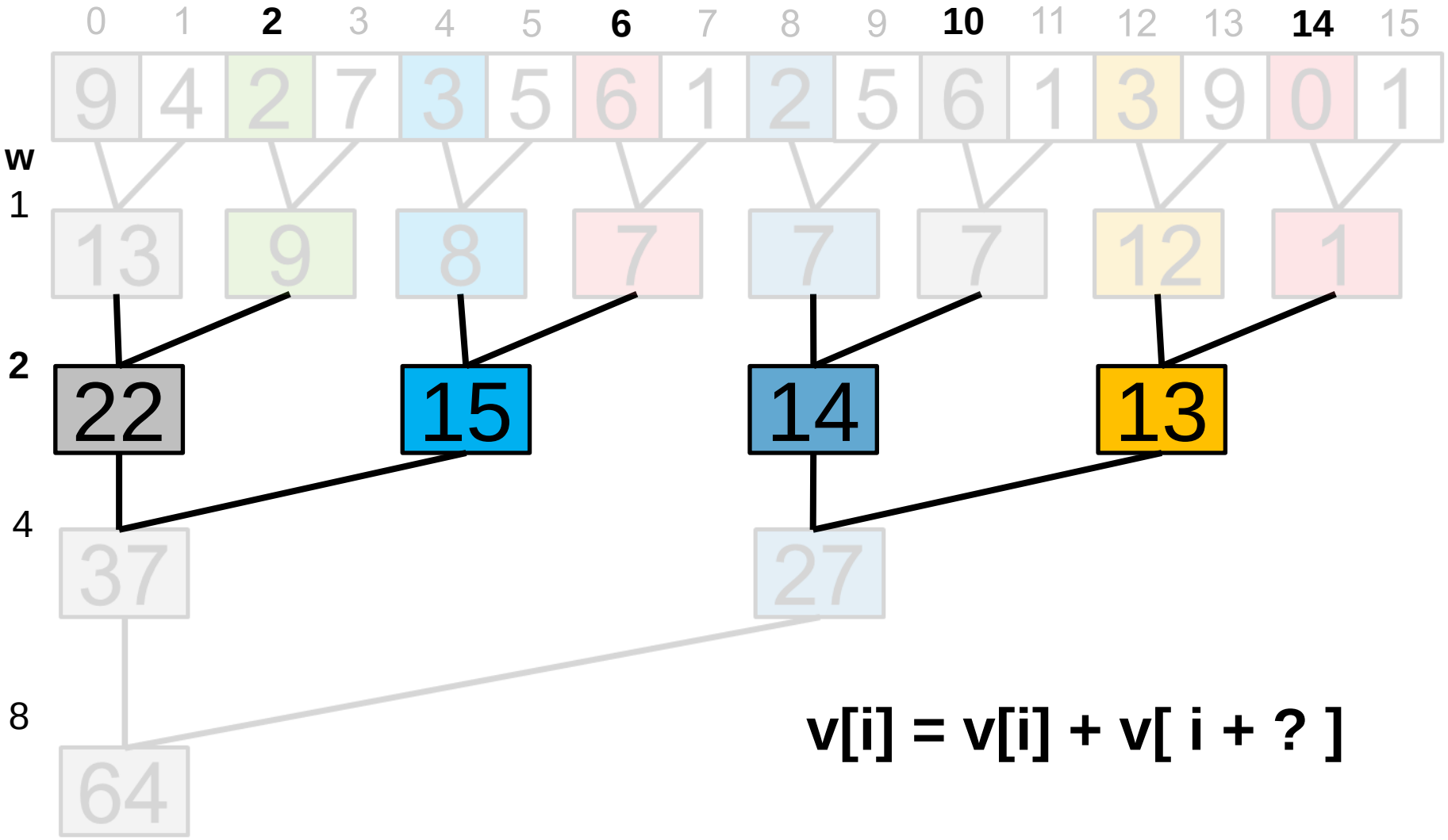
Rezultatul sumei se pune peste valoarea care rămâne pe același thread.

Reduce – implementare – imagine globală

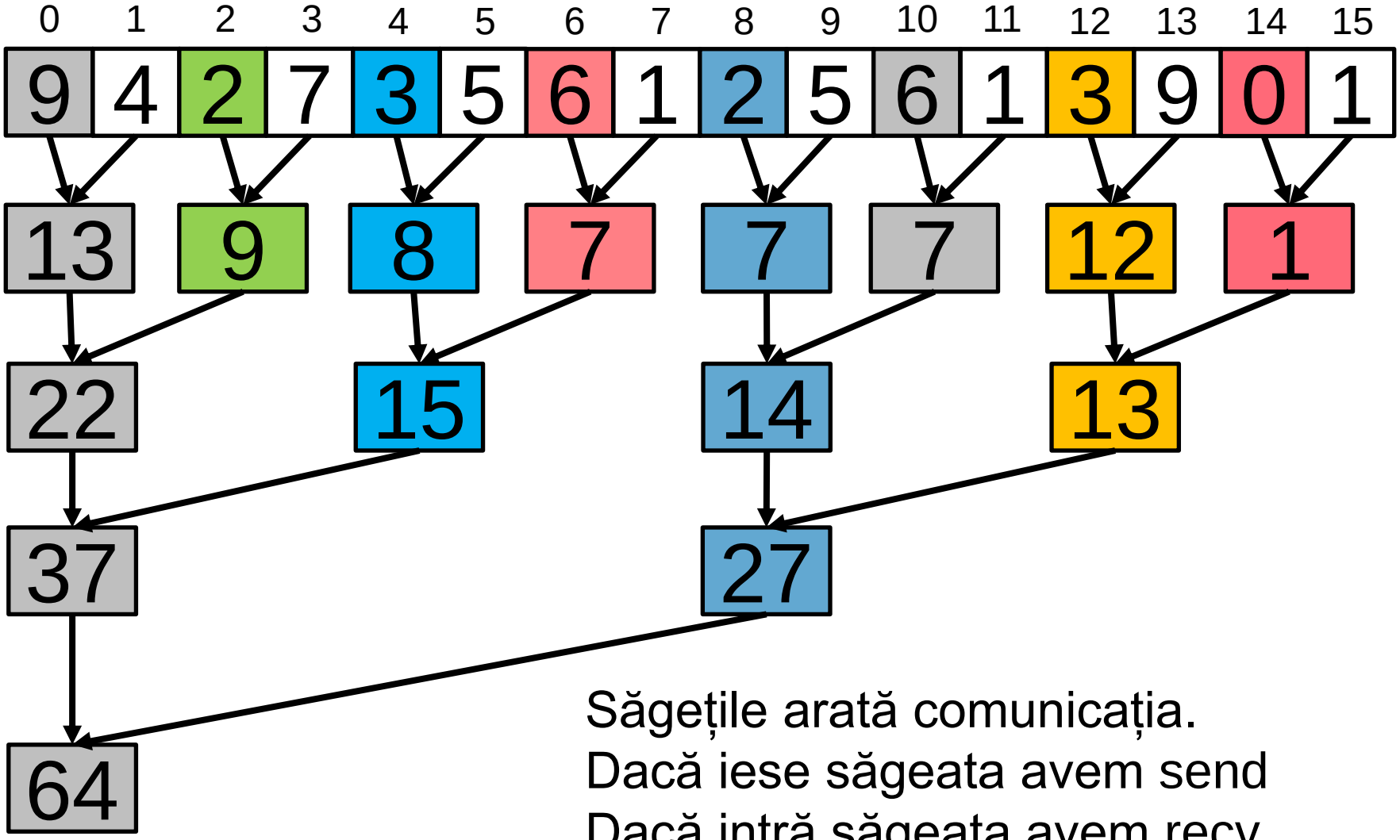


Calcul fac doar thread-urile cu id divizibil cu **2w**

Reduce – implementare – imagine globală

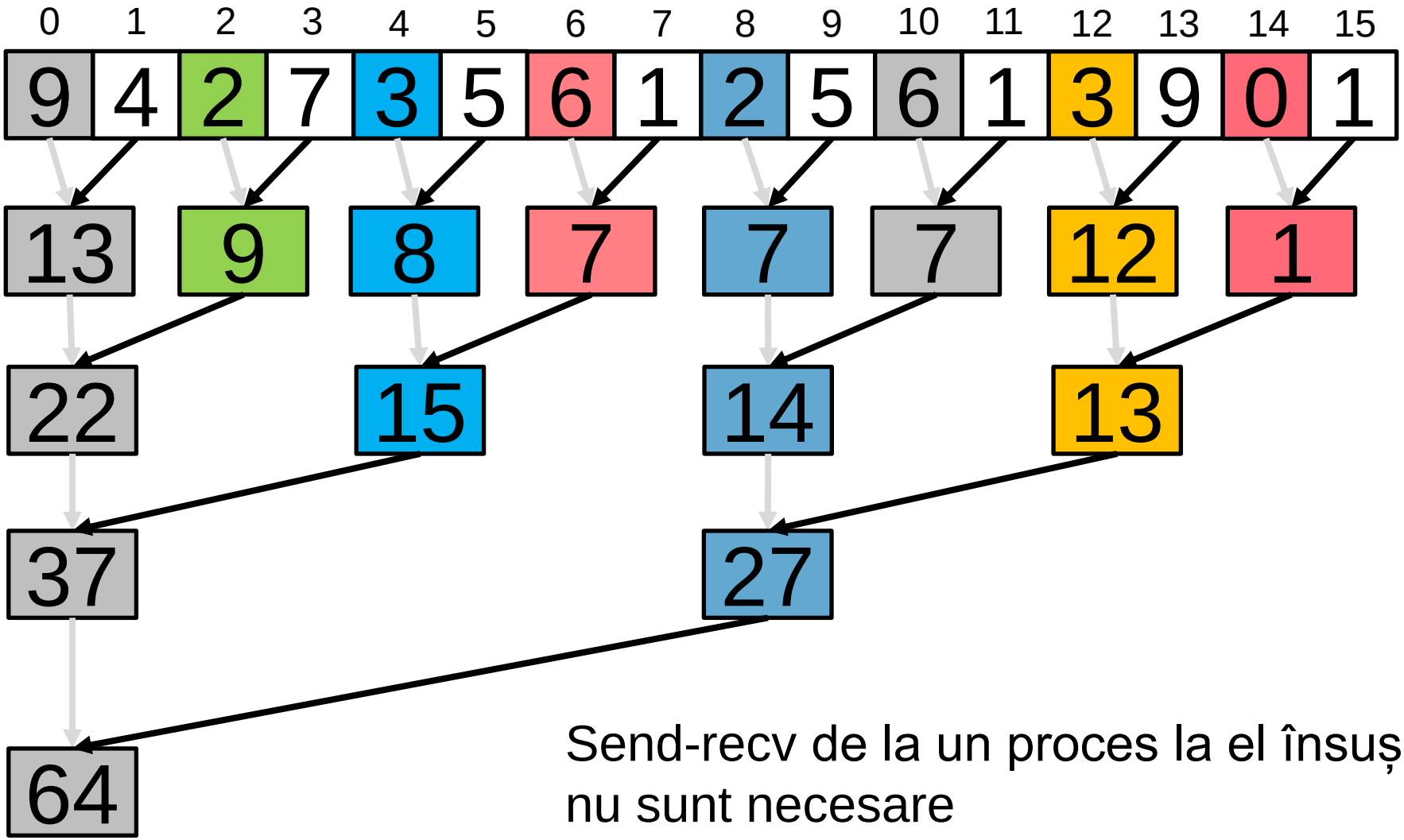


Reduce – implementare – model comunicație



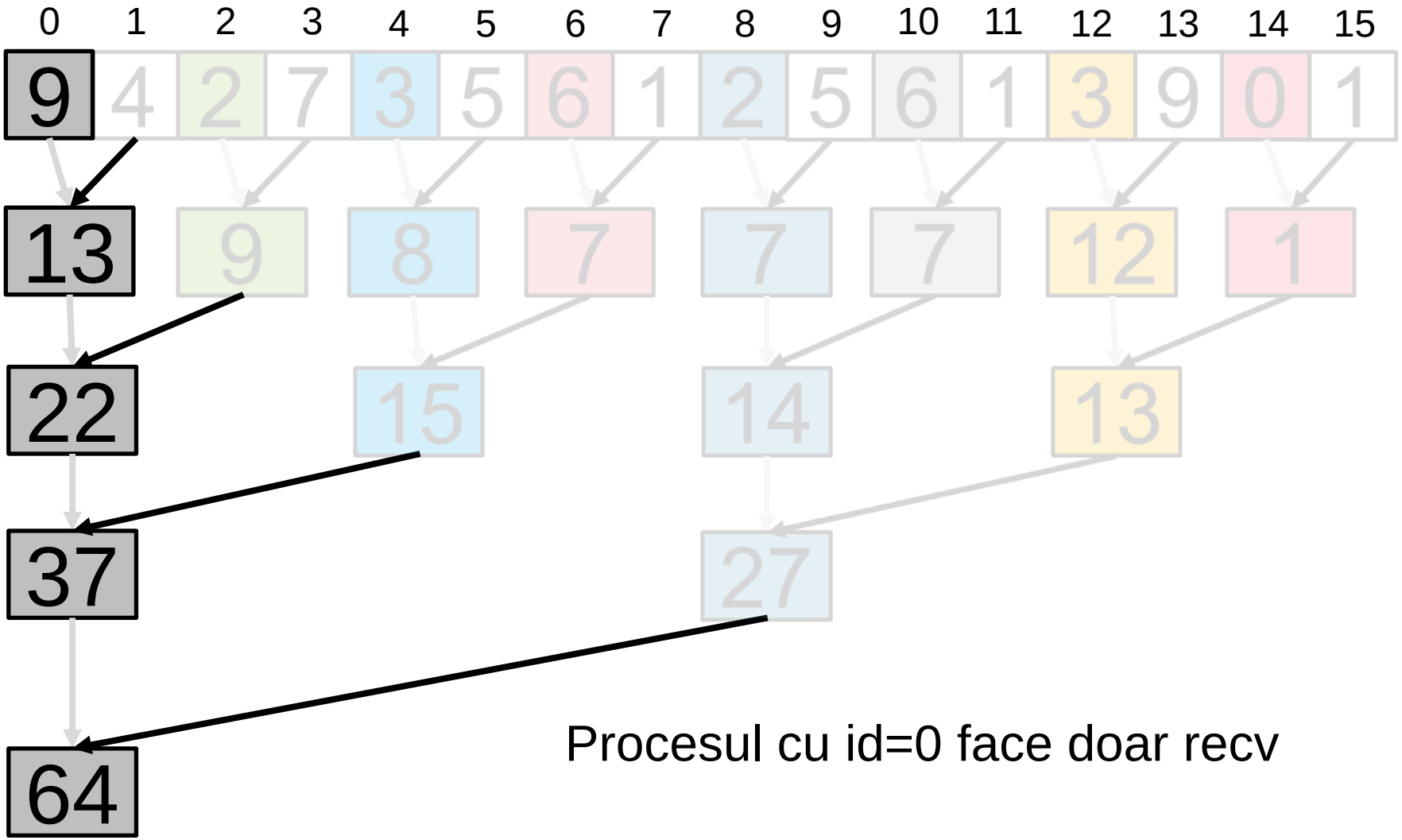
Săgețile arată comunicația.
Dacă iese săgeata avem send
Dacă intră săgeata avem recv

Reduce – implementare – imagine locală



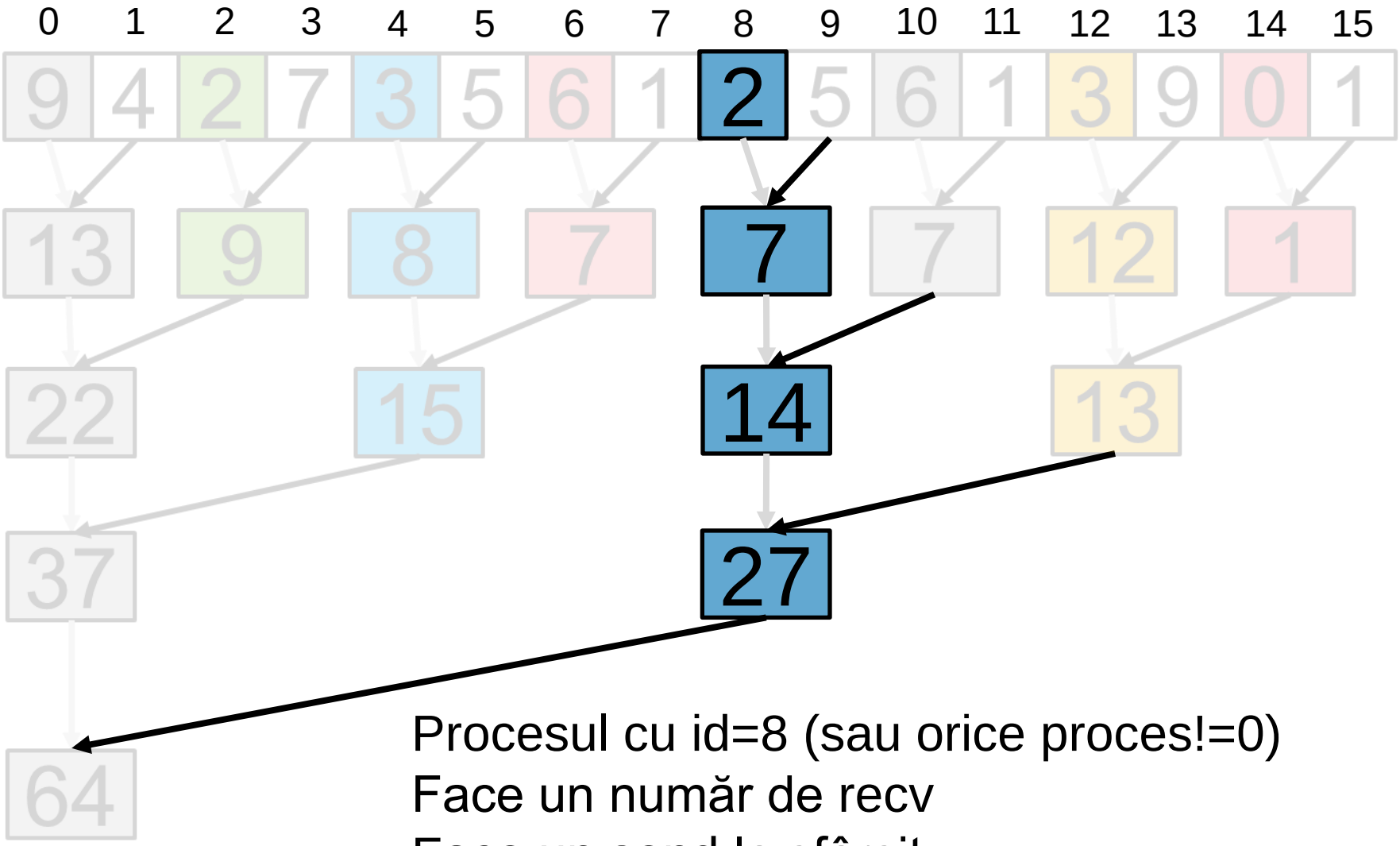
Send-recv de la un proces la el însuși nu sunt necesare

Reduce – implementare – imagine locală



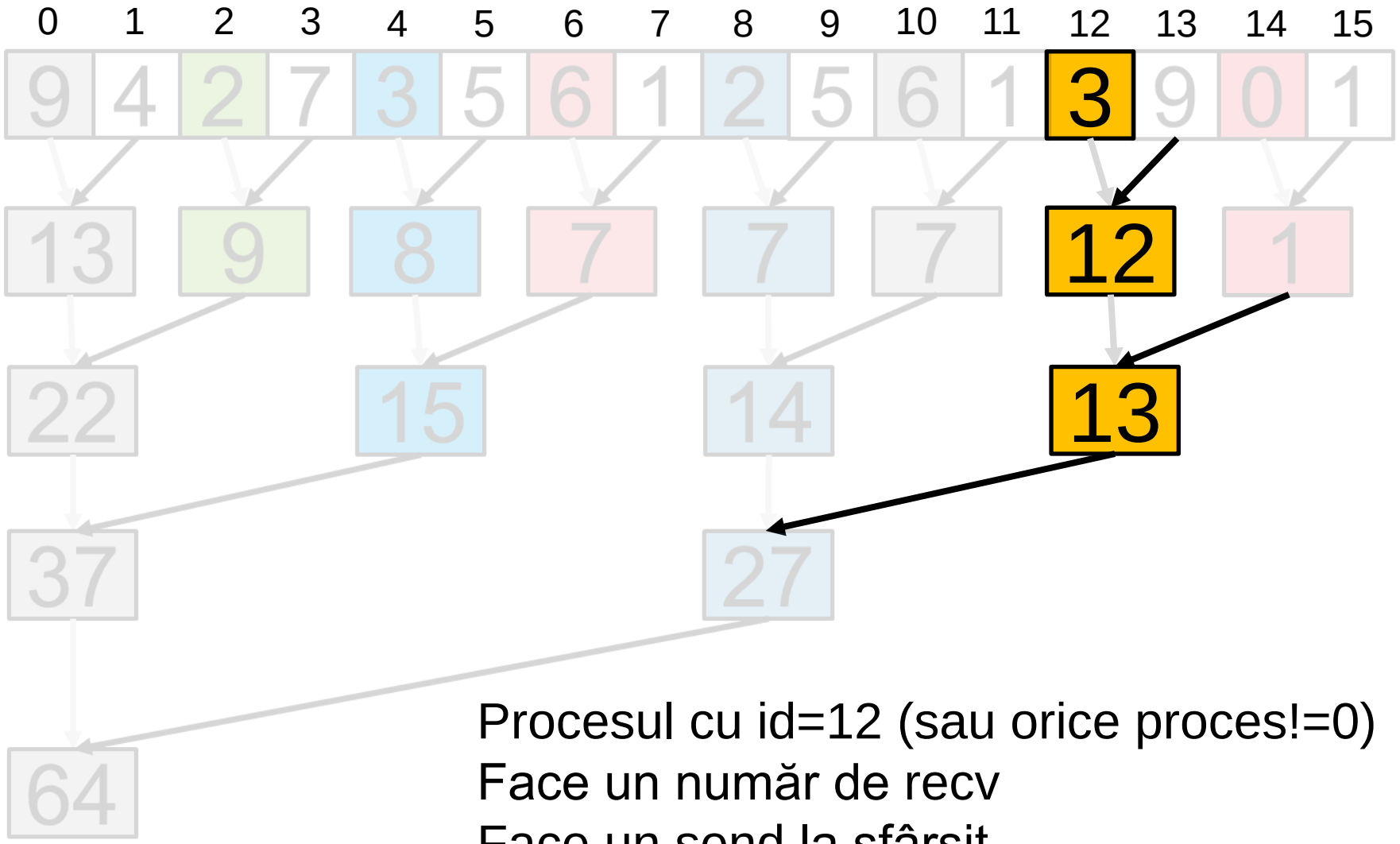
Procesul cu id=0 face doar recv

Reduce – implementare – imagine locală



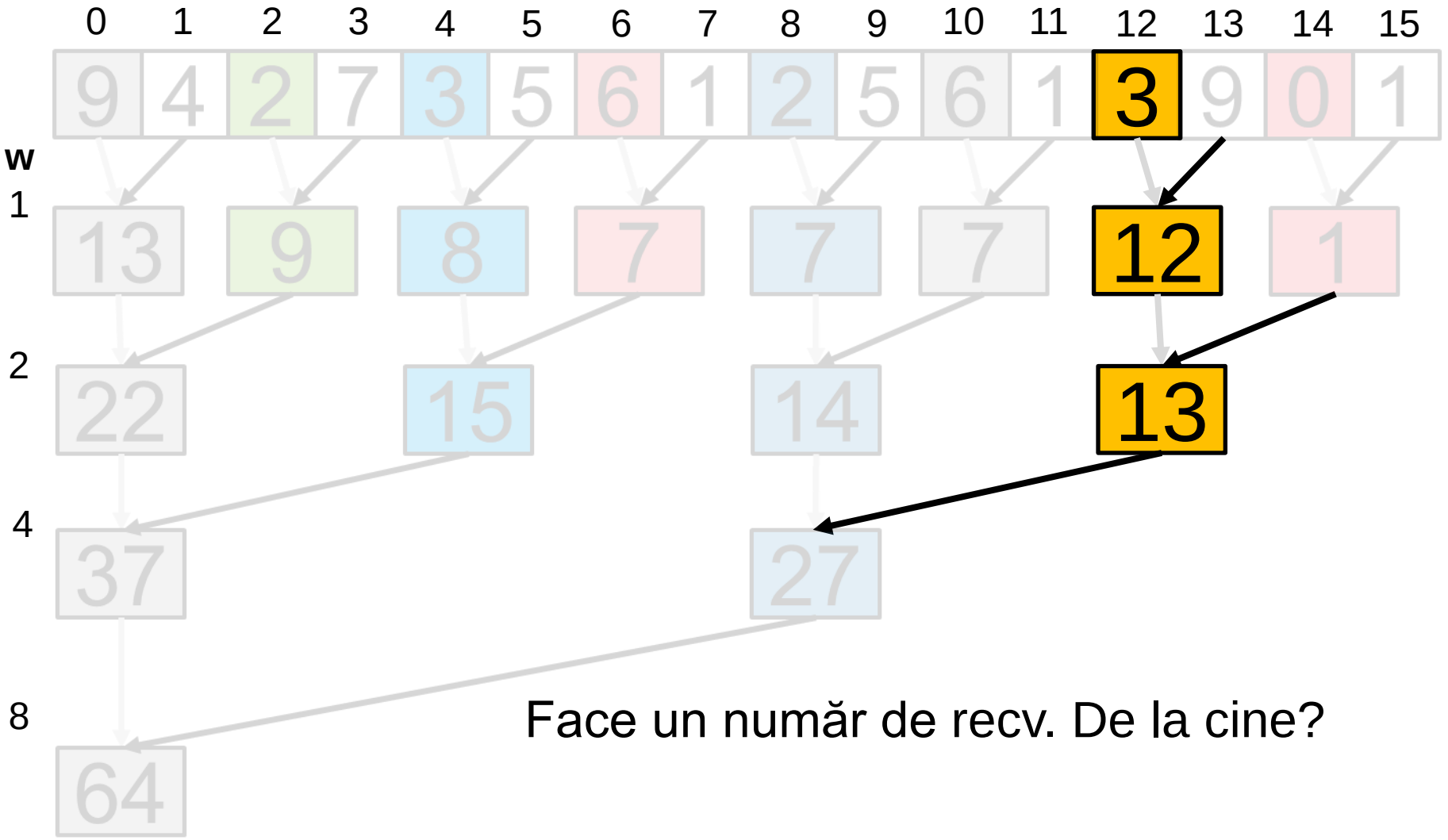
Procesul cu id=8 (sau orice proces!=0)
Face un număr de recv
Face un send la sfârșit

Reduce – implementare – imagine locală



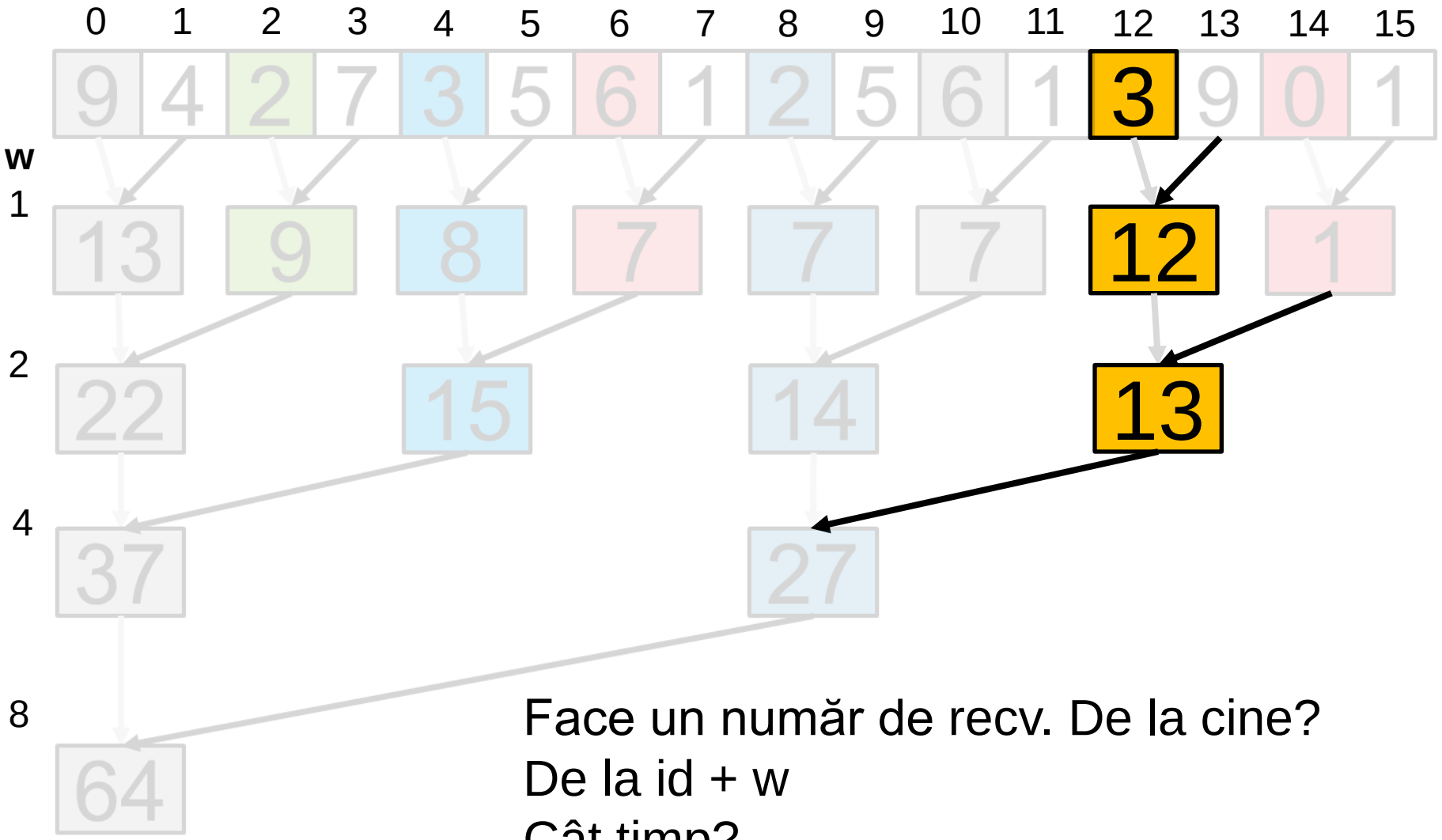
Procesul cu id=12 (sau orice proces!=0)
Face un număr de recv
Face un send la sfârșit

Reduce – implementare – imagine locală



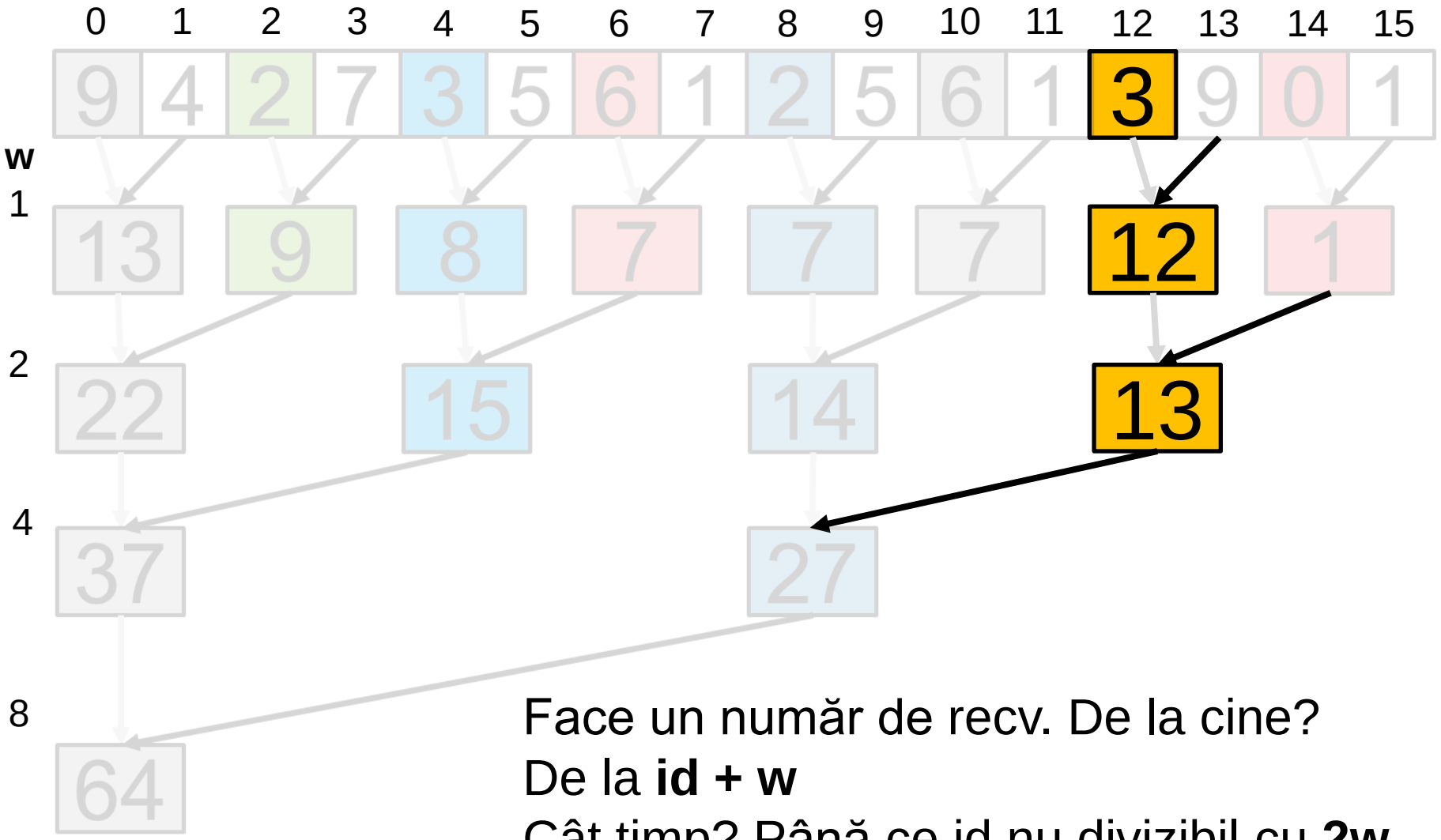
Face un număr de recv. De la cine?

Reduce – implementare – imagine locală



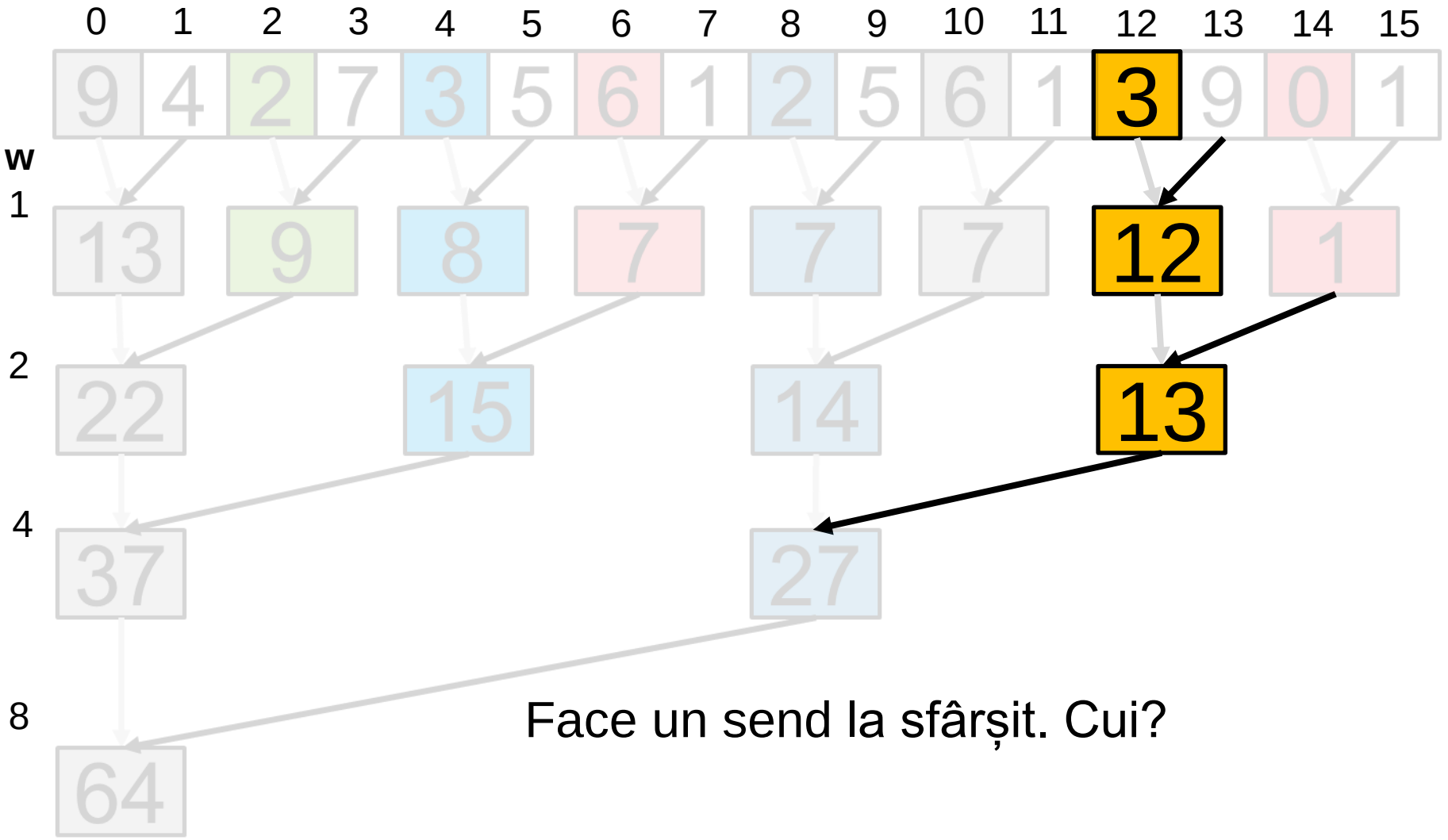
Face un număr de recv. De la cine?
De la id + w
Cât timp?

Reduce – implementare – imagine locală



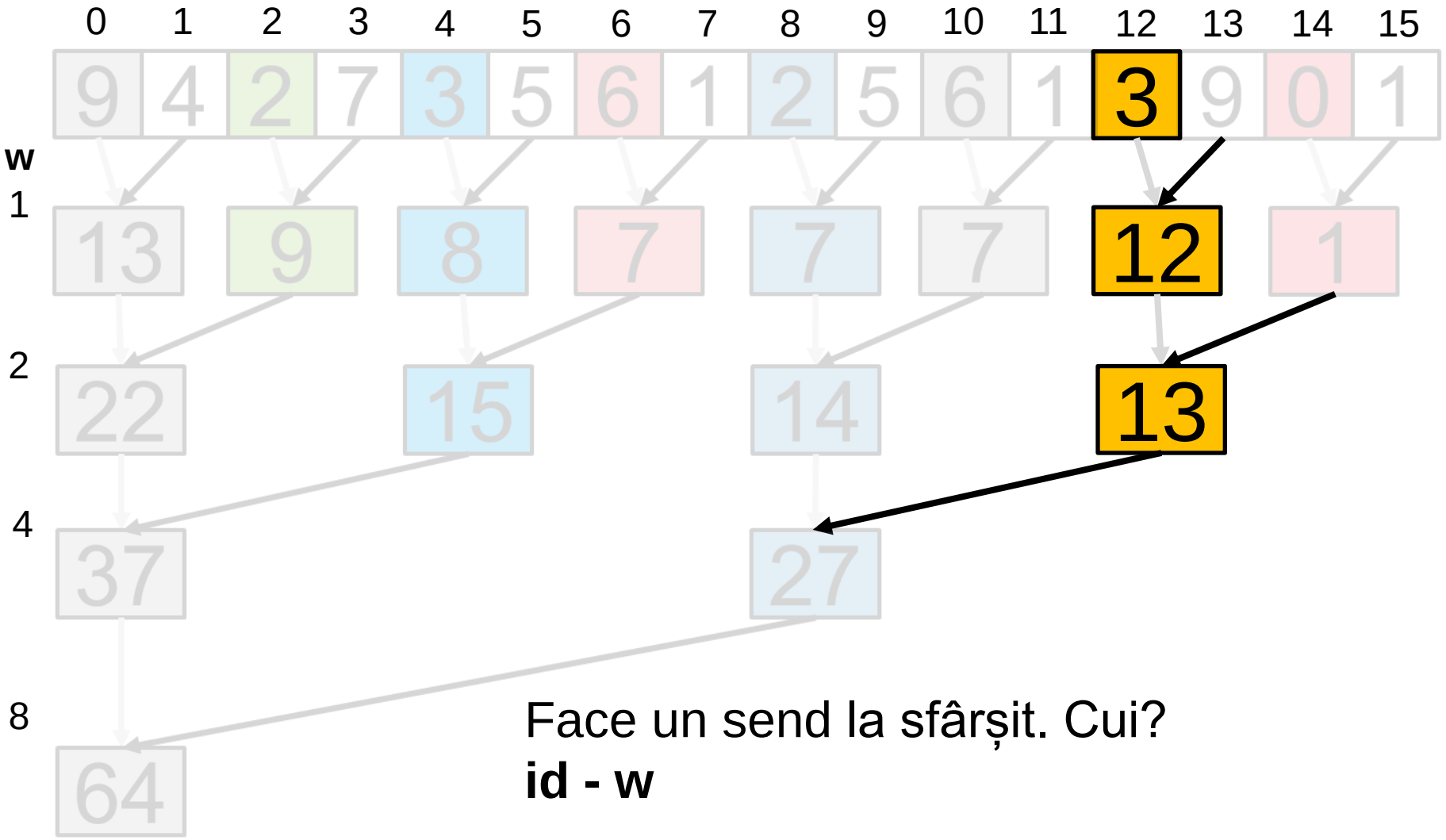
Face un număr de recv. De la cine?
De la **id + w**
Cât timp? Până ce id nu divizibil cu **2w**

Reduce – implementare – imagine locală



Face un send la sfârșit. Cui?

Reduce – implementare – imagine locală



Face un send la sfârșit. Cui?

id - w

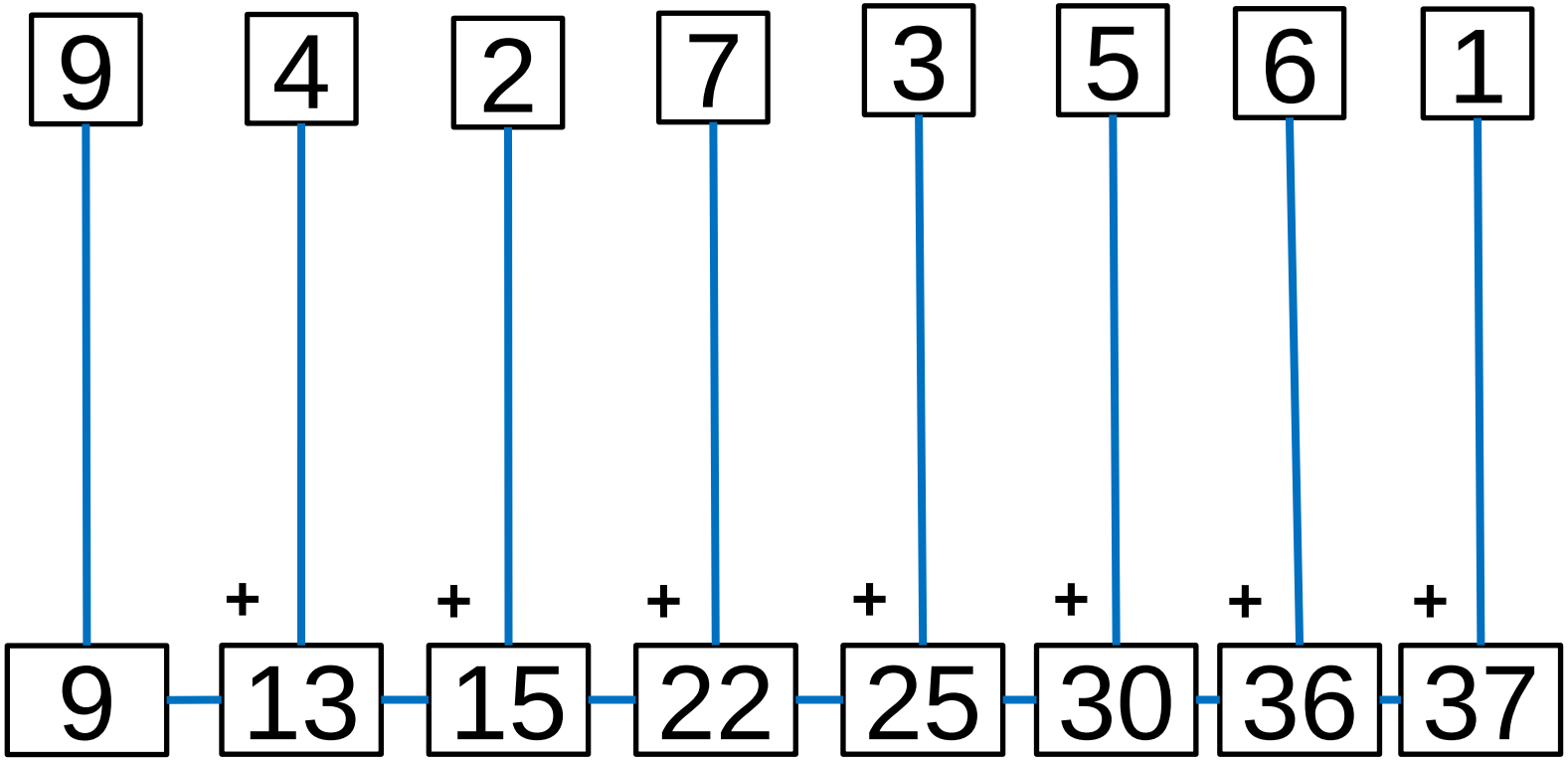
Scan

Similar cu reduce dar se păstrează toate rezultatele intermediare.

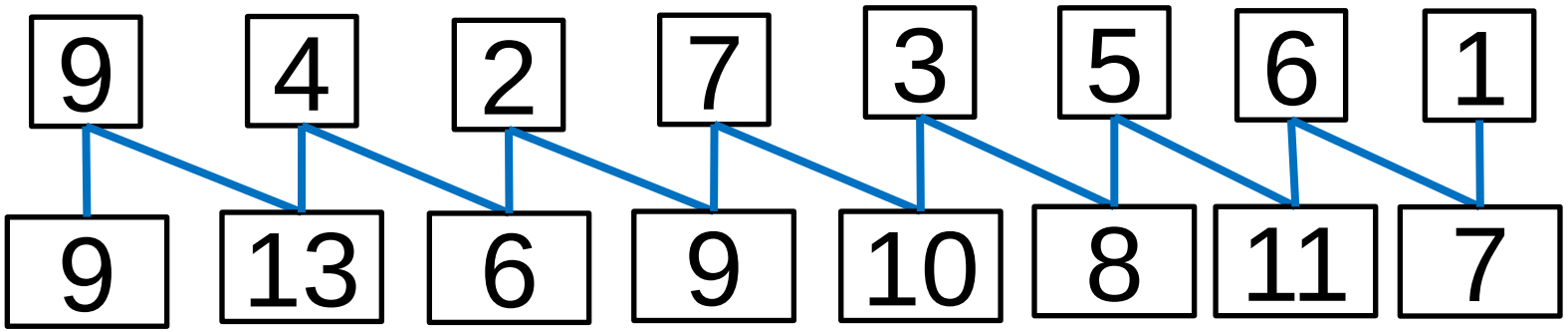
Se poate executa în $O(log(N))$ pe N procesoare organizând calculele într-o formă arborescentă.

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

Scan cu sumă

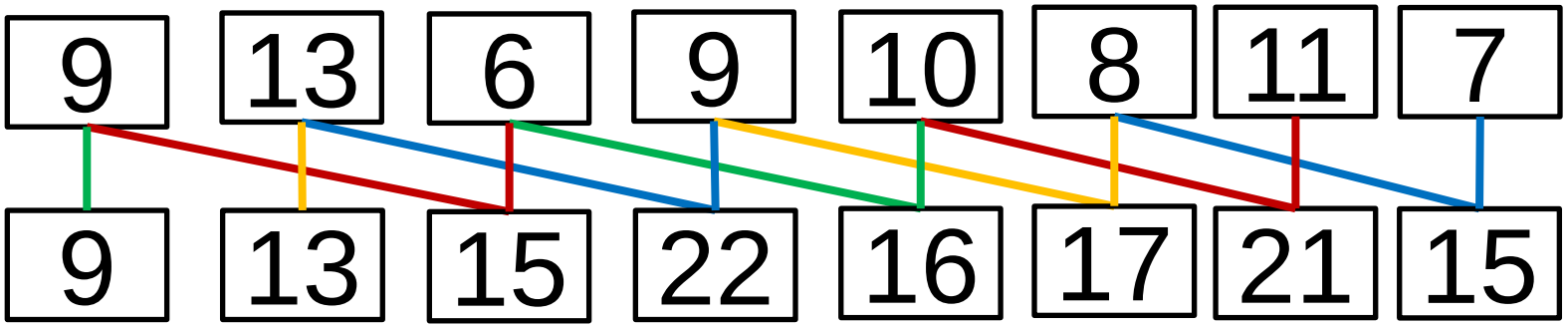


Scan cu sumă



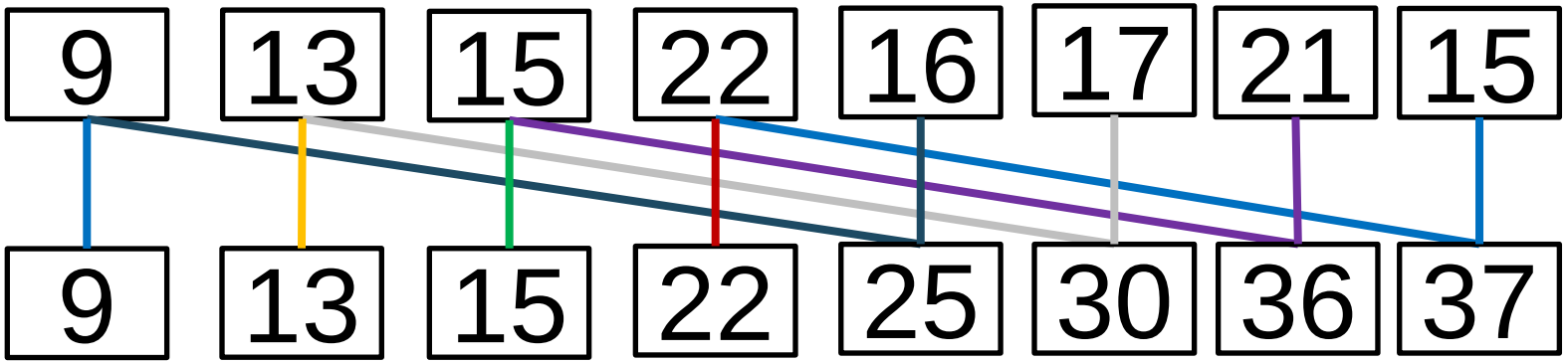
Pot fi executate în paralel

Scan cu sumă



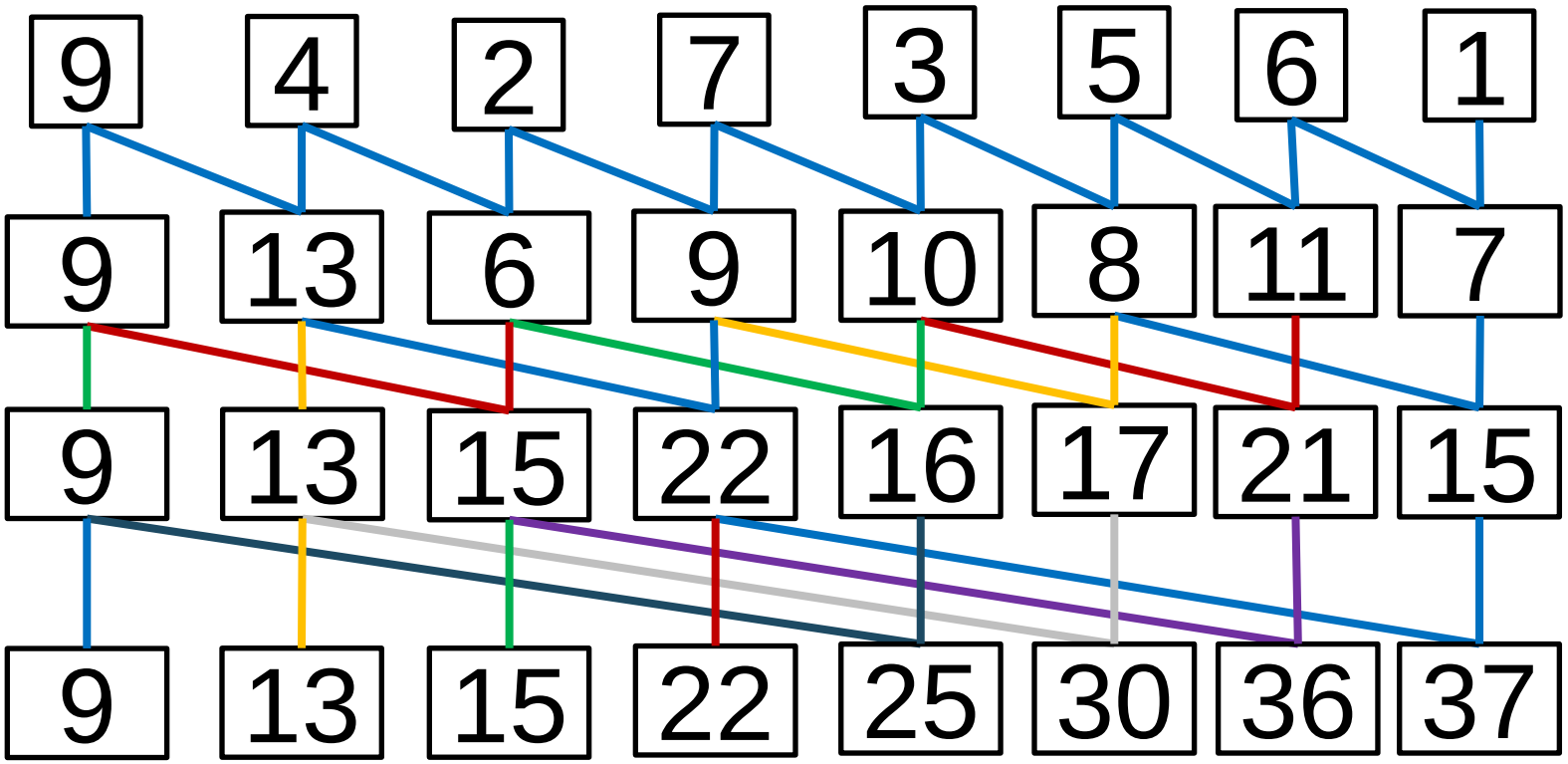
Pot fi executate în paralel

Scan cu sumă

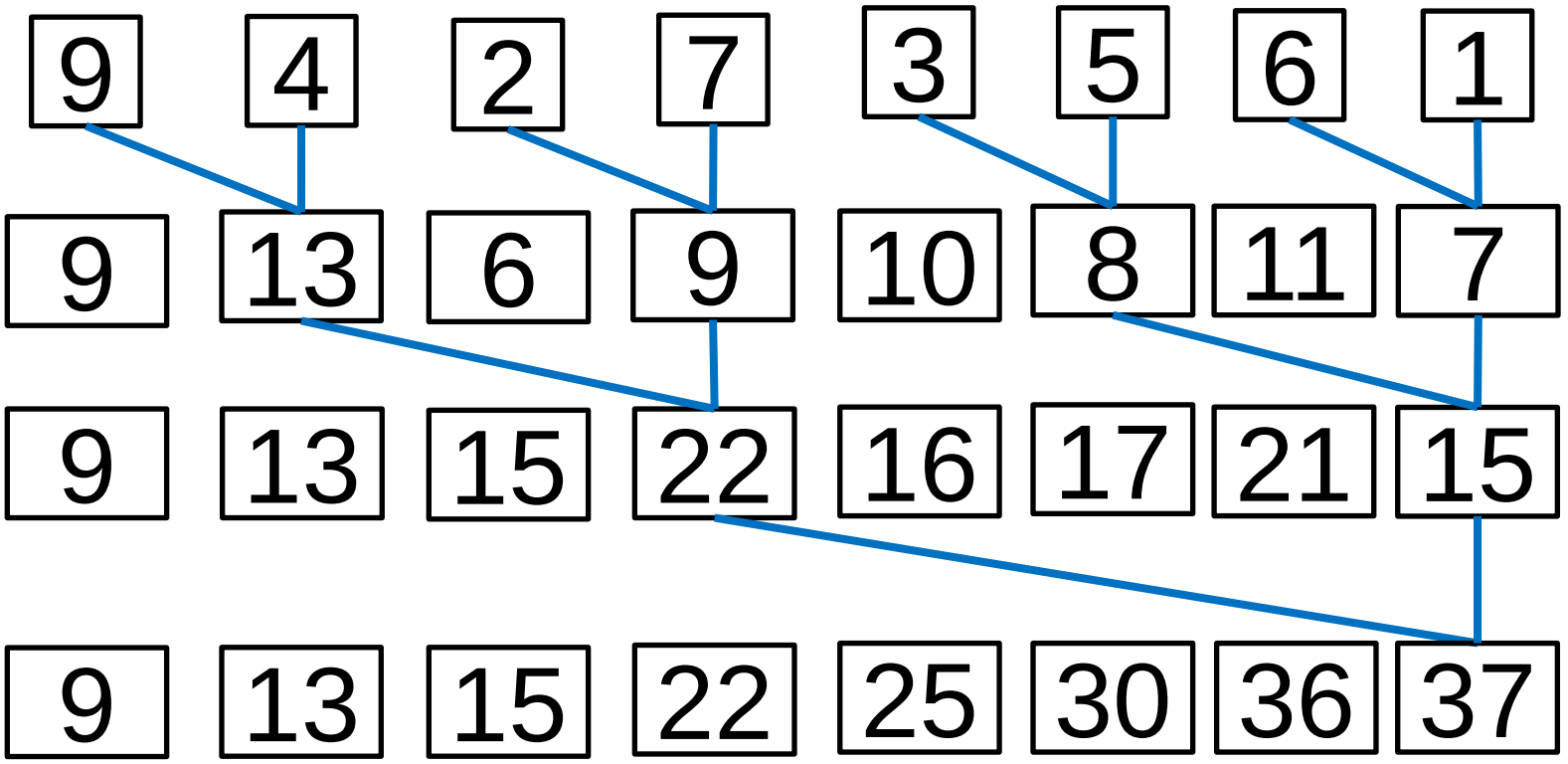


Pot fi executate în paralel

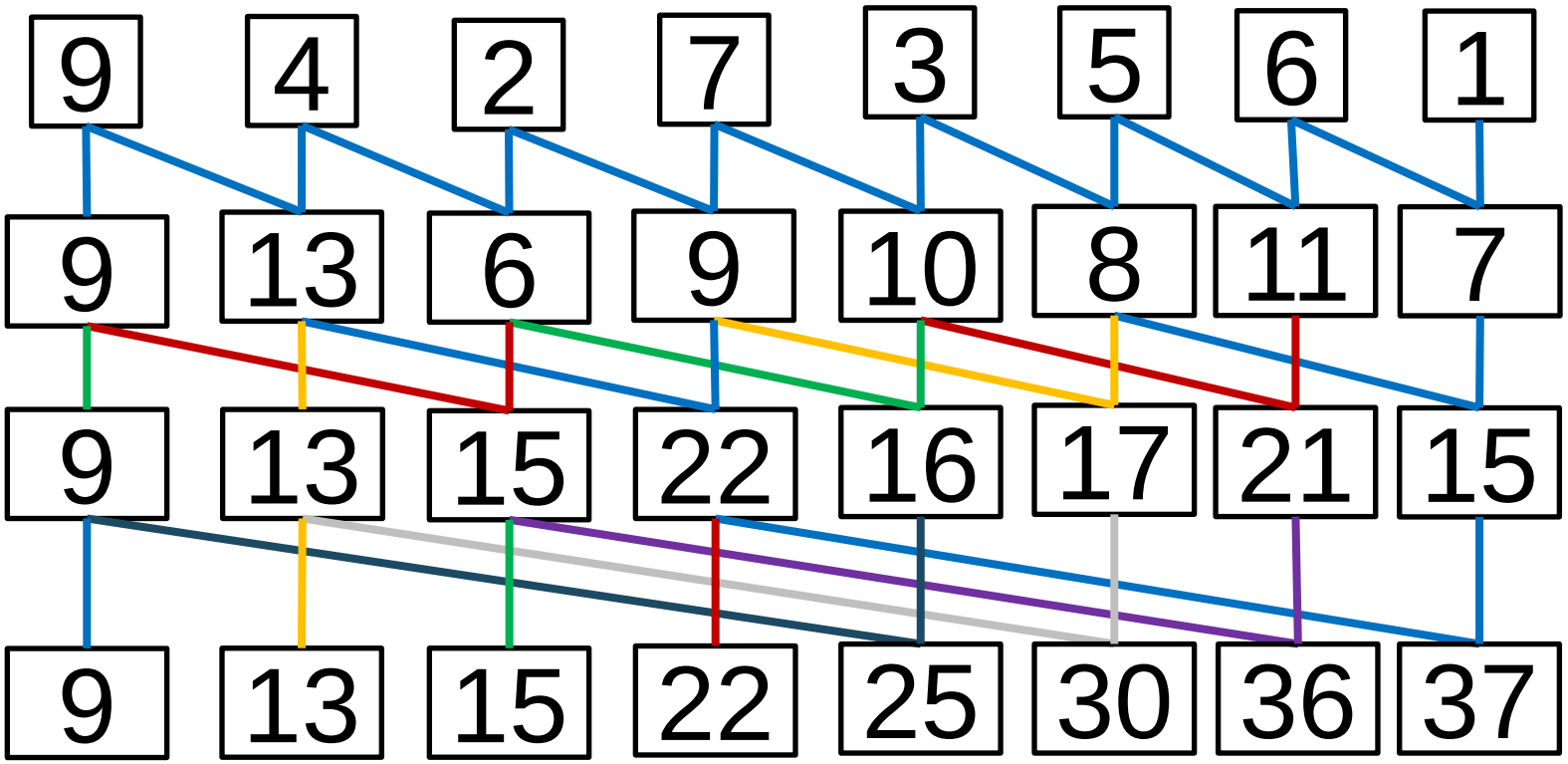
Scan – Cum funcționează?



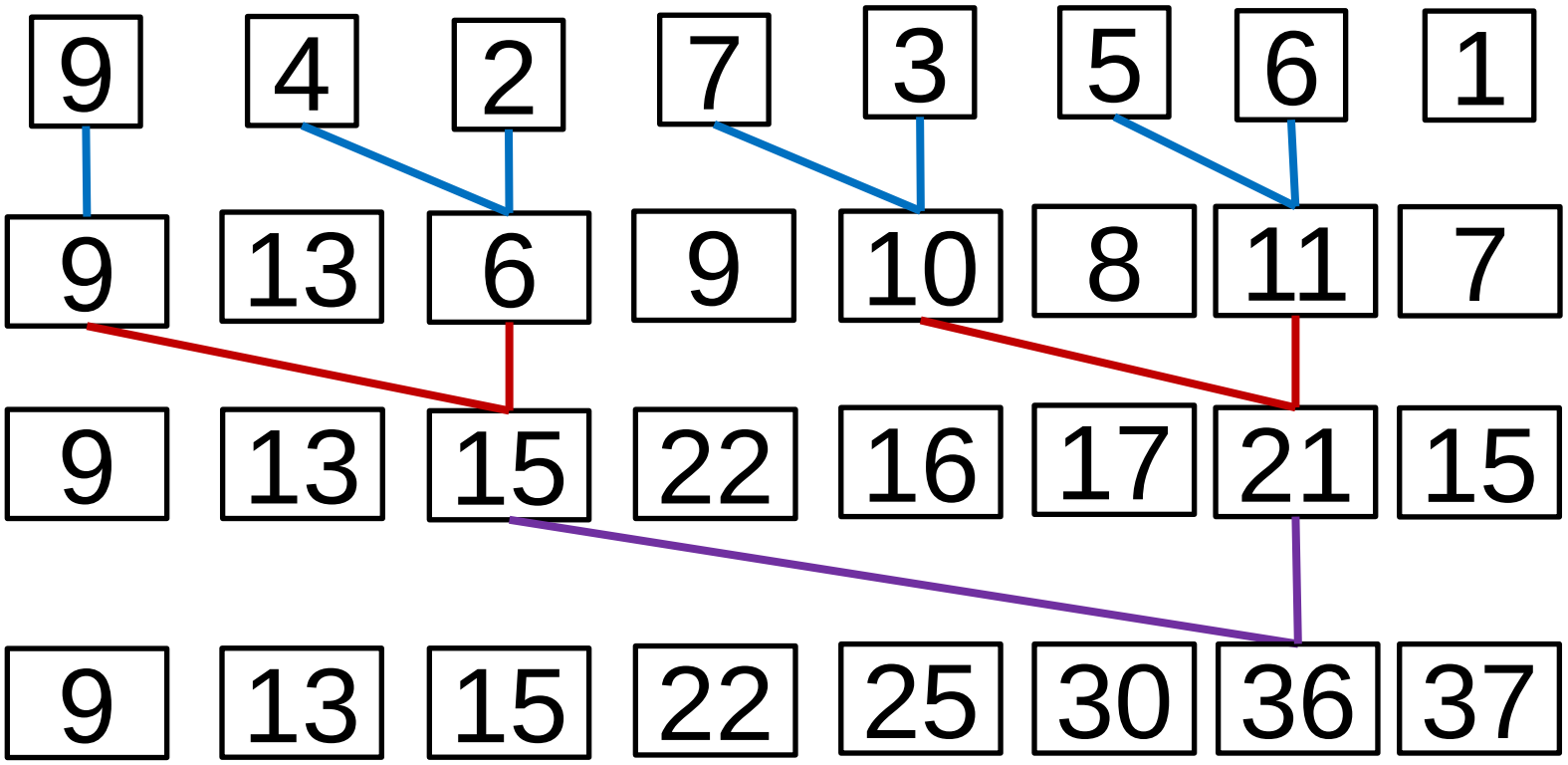
Scan – Cum funcționează?



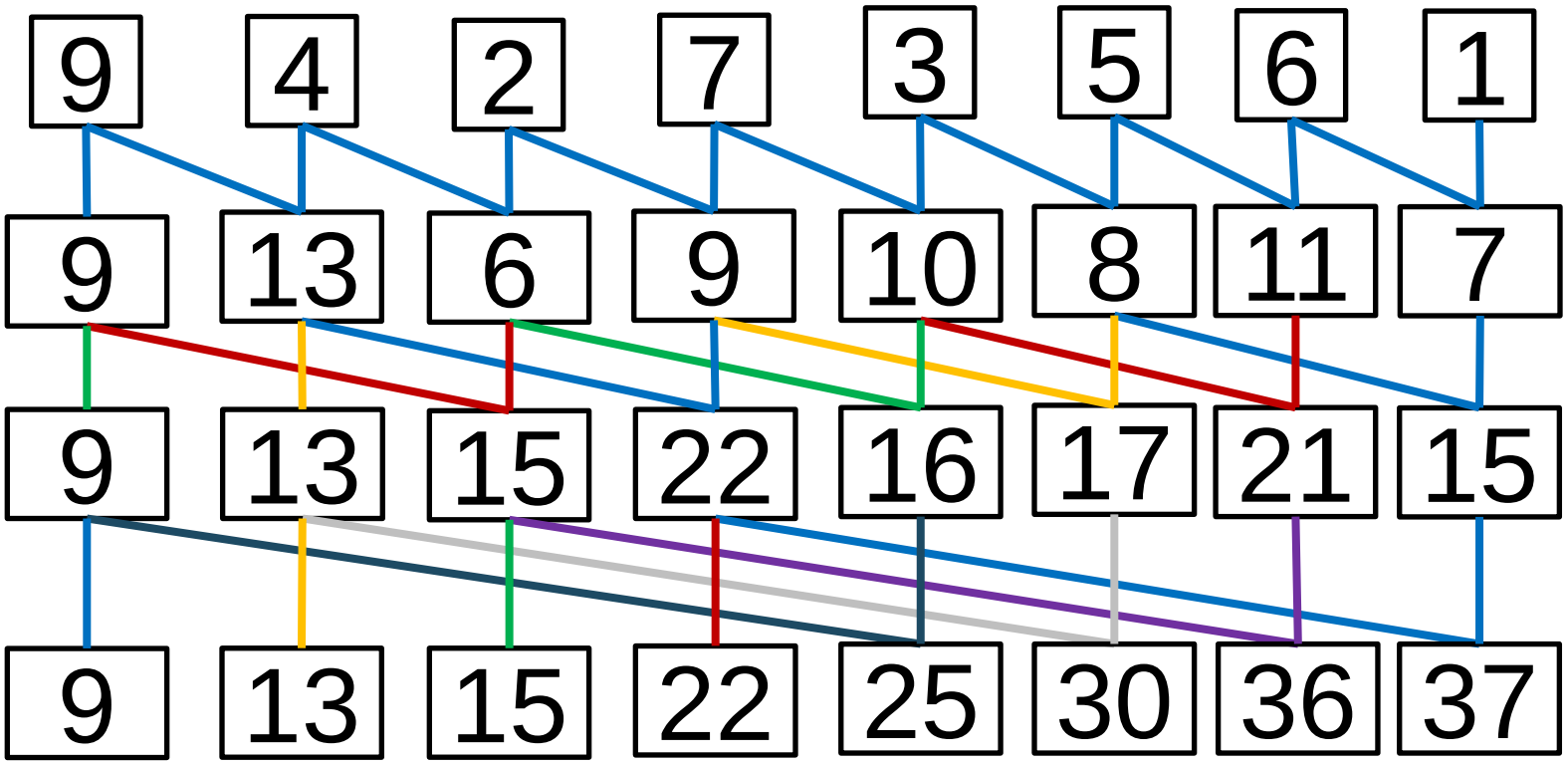
Scan – Cum funcționează?



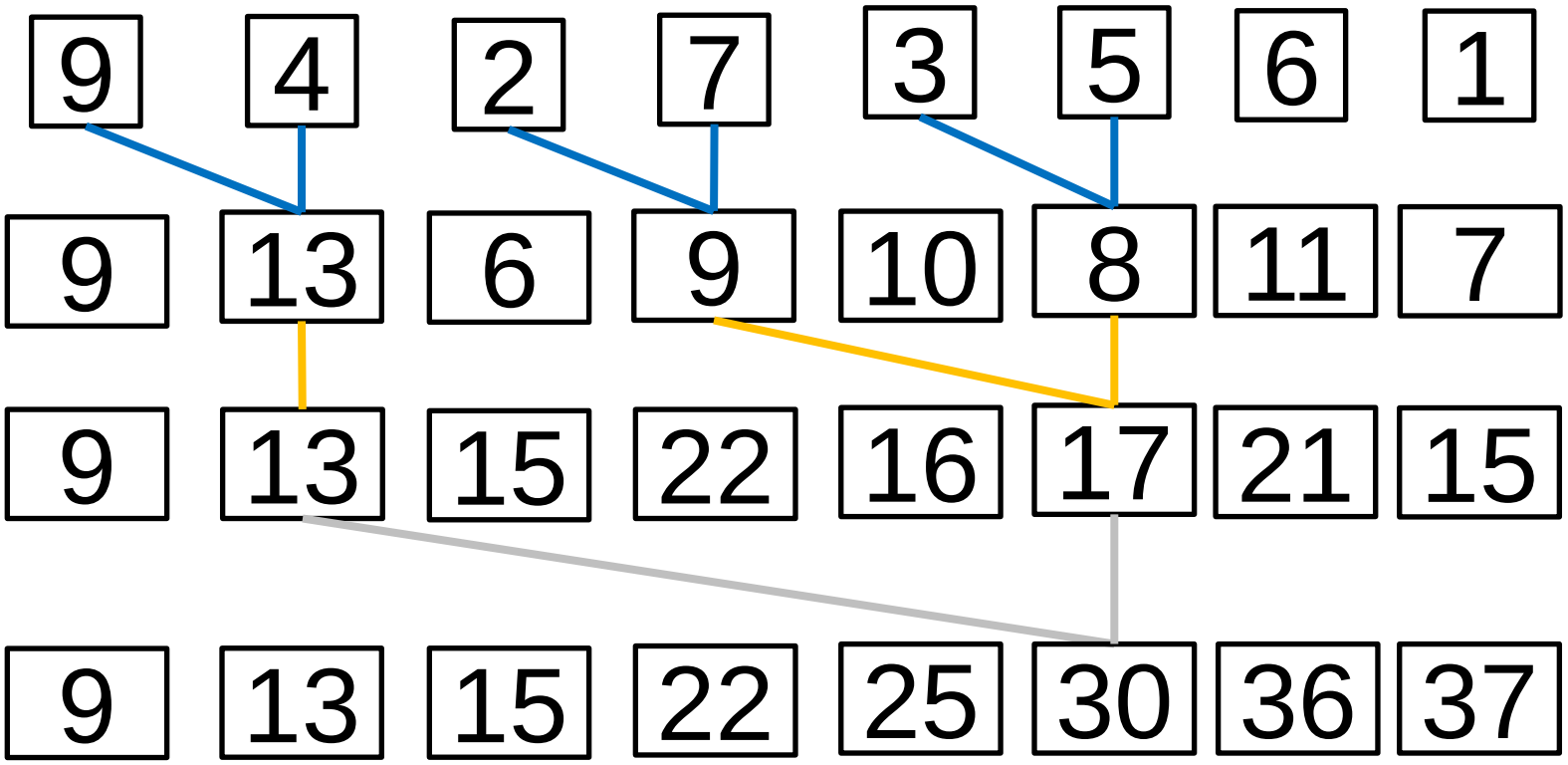
Scan – Cum funcționează?



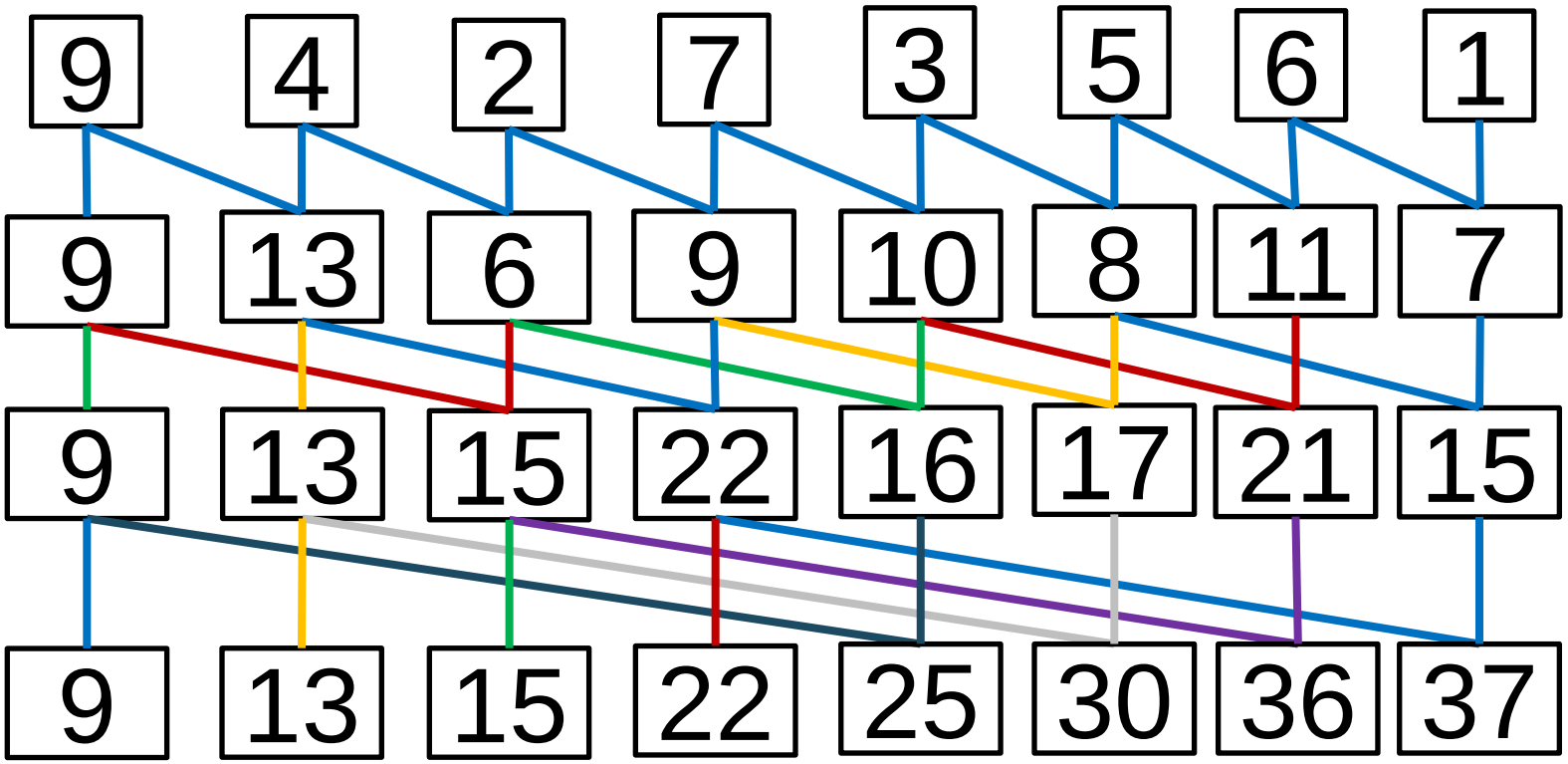
Scan – Cum funcționează?



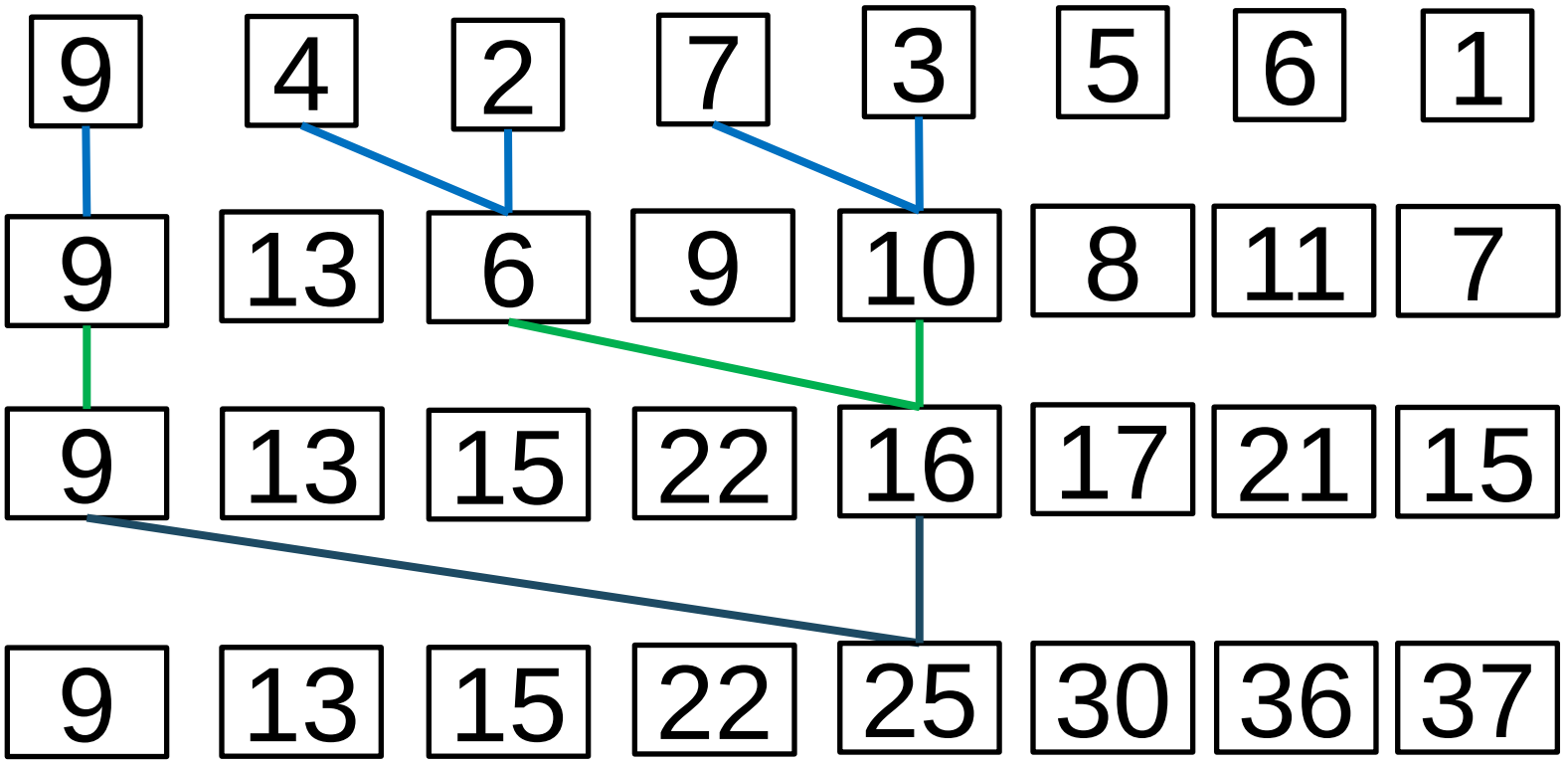
Scan – Cum funcționează?



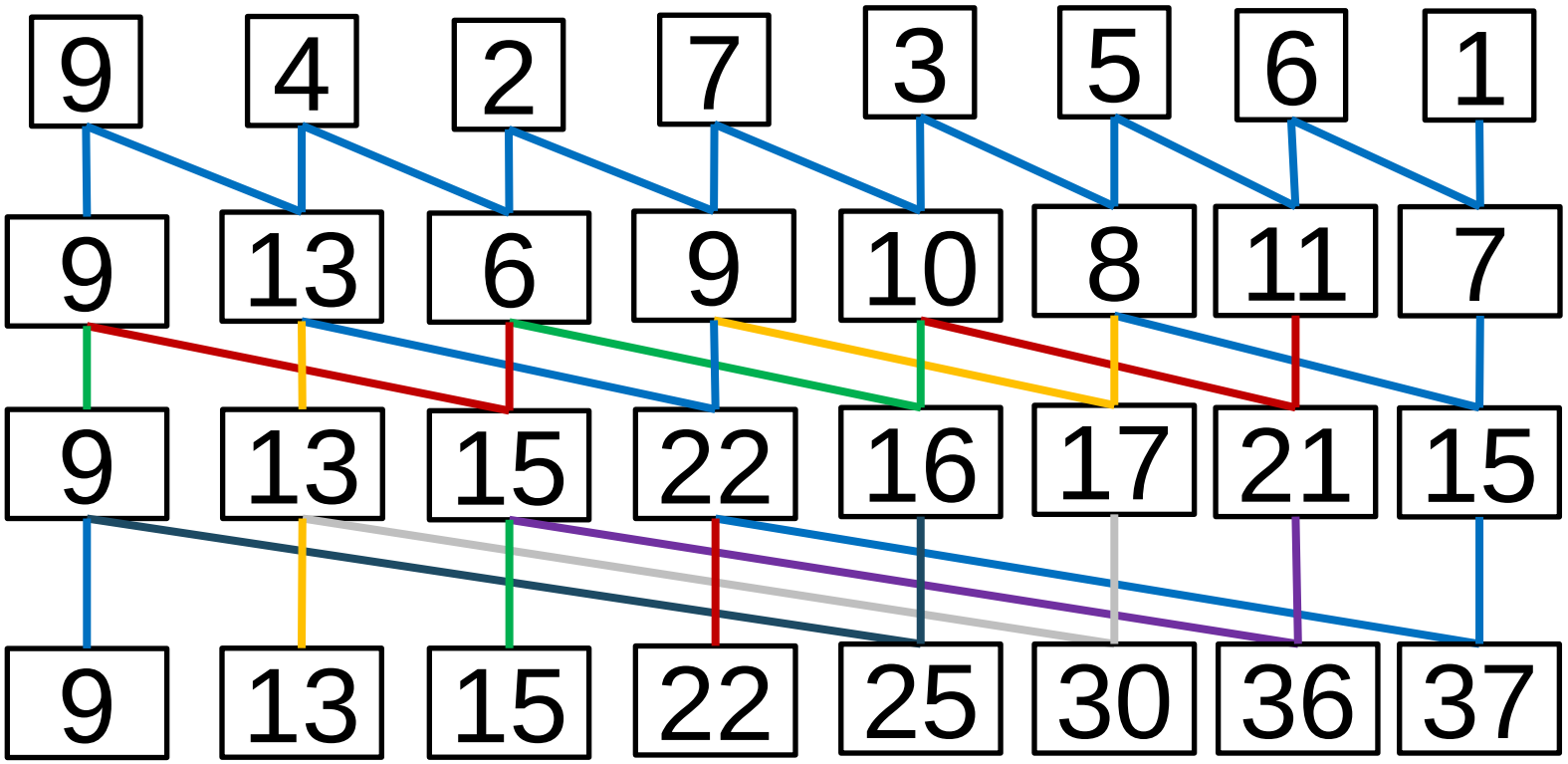
Scan – Cum funcționează?



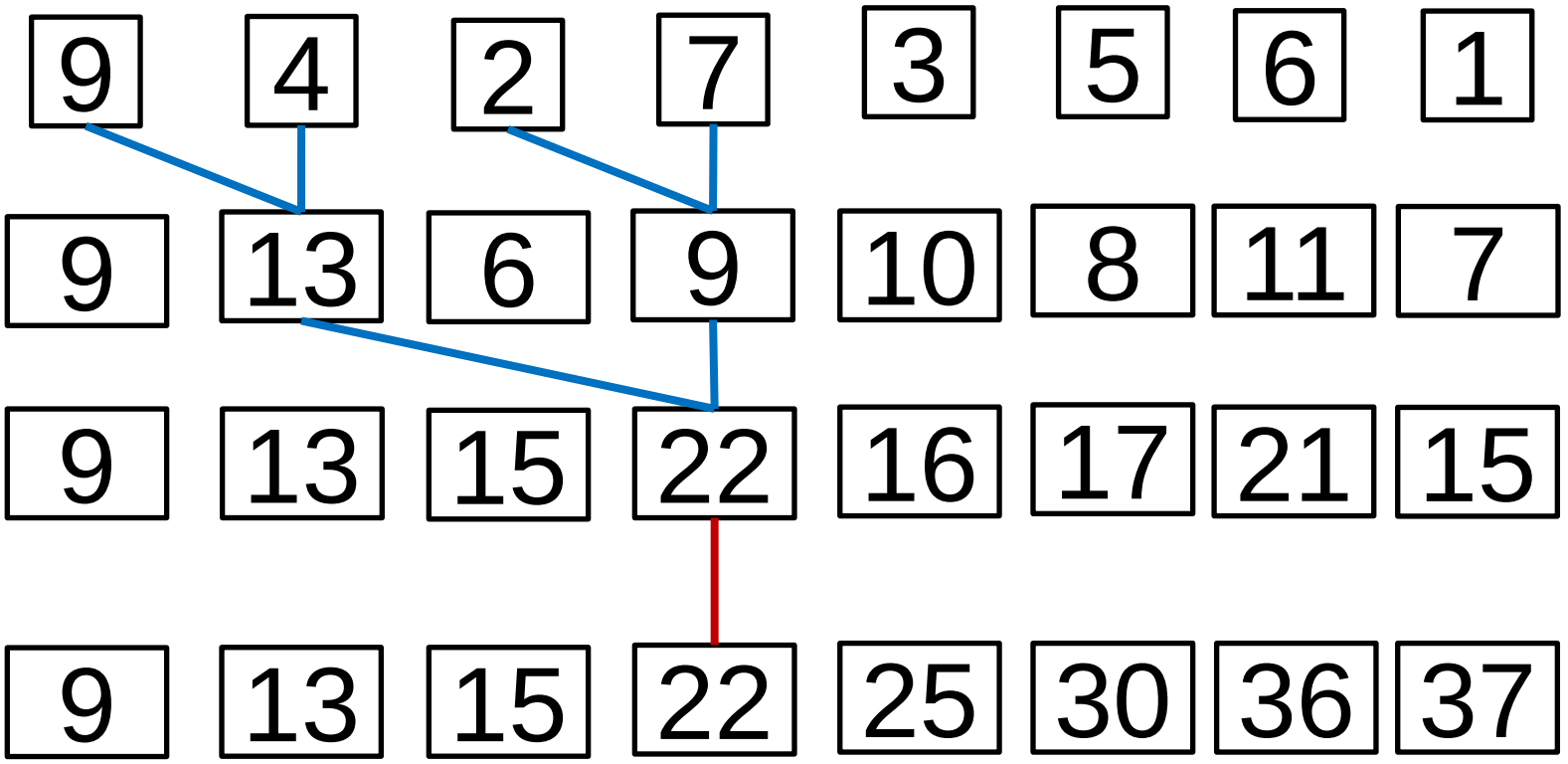
Scan – Cum funcționează?



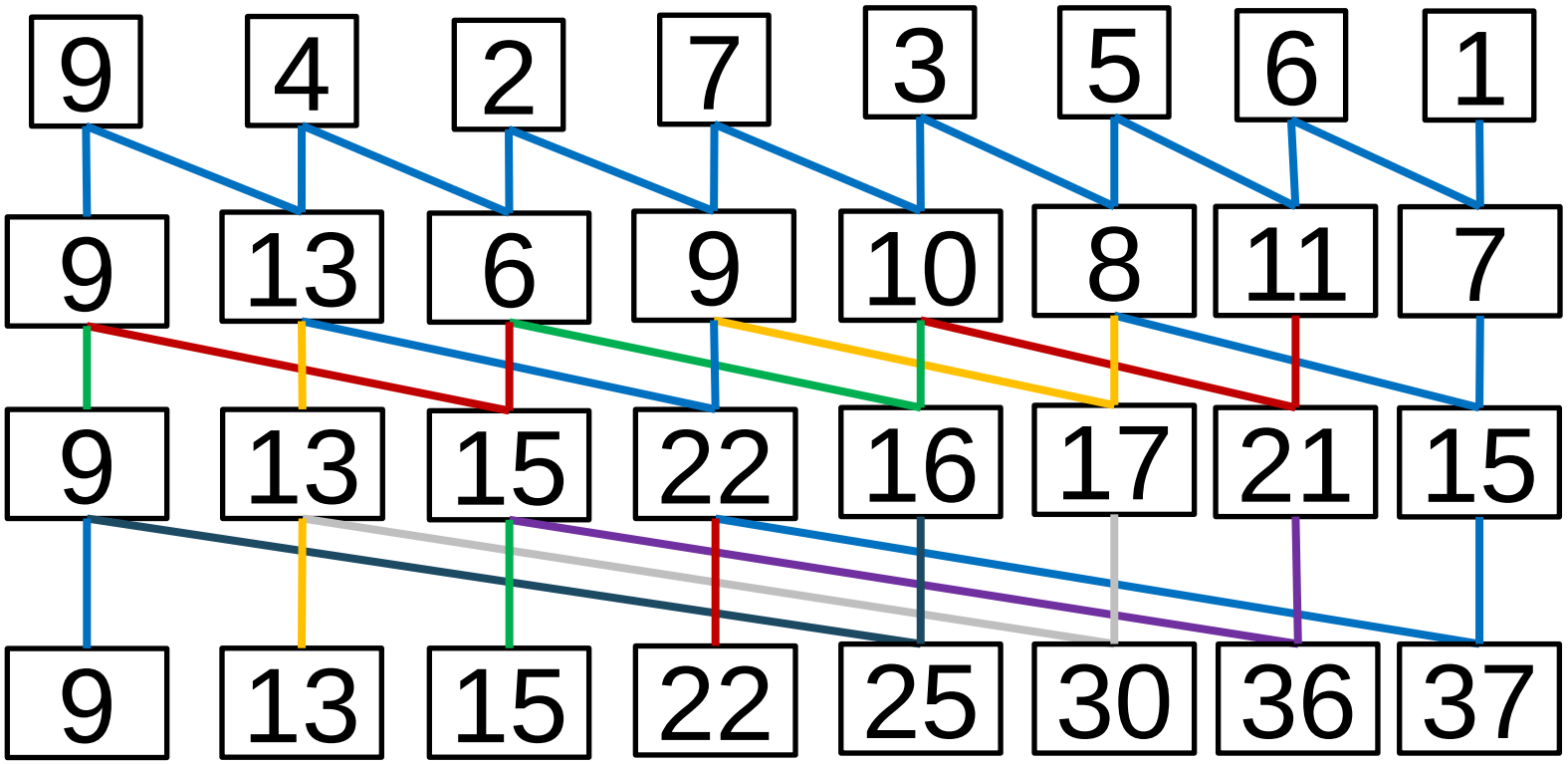
Scan – Cum funcționează?



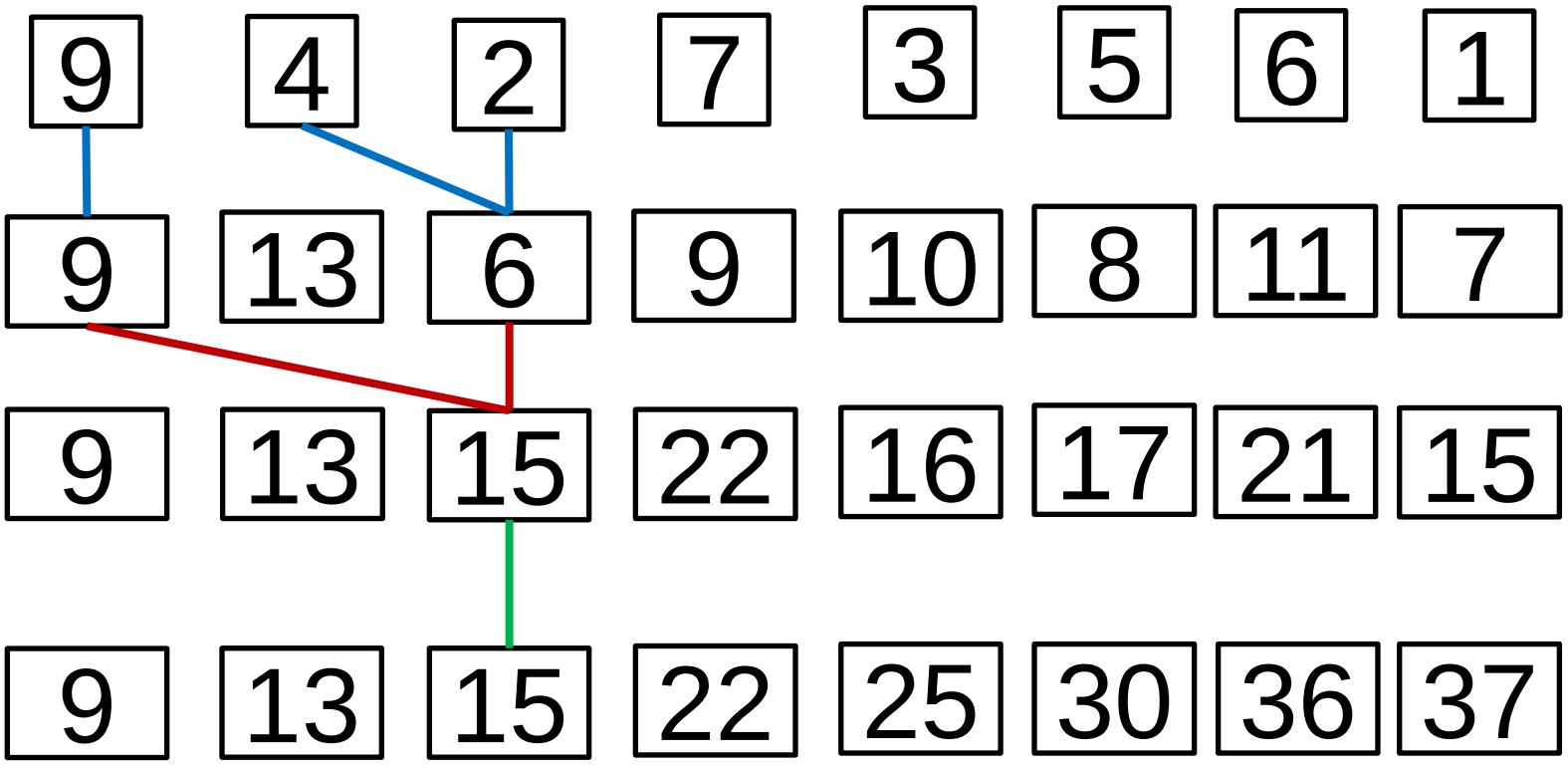
Scan – Cum funcționează?



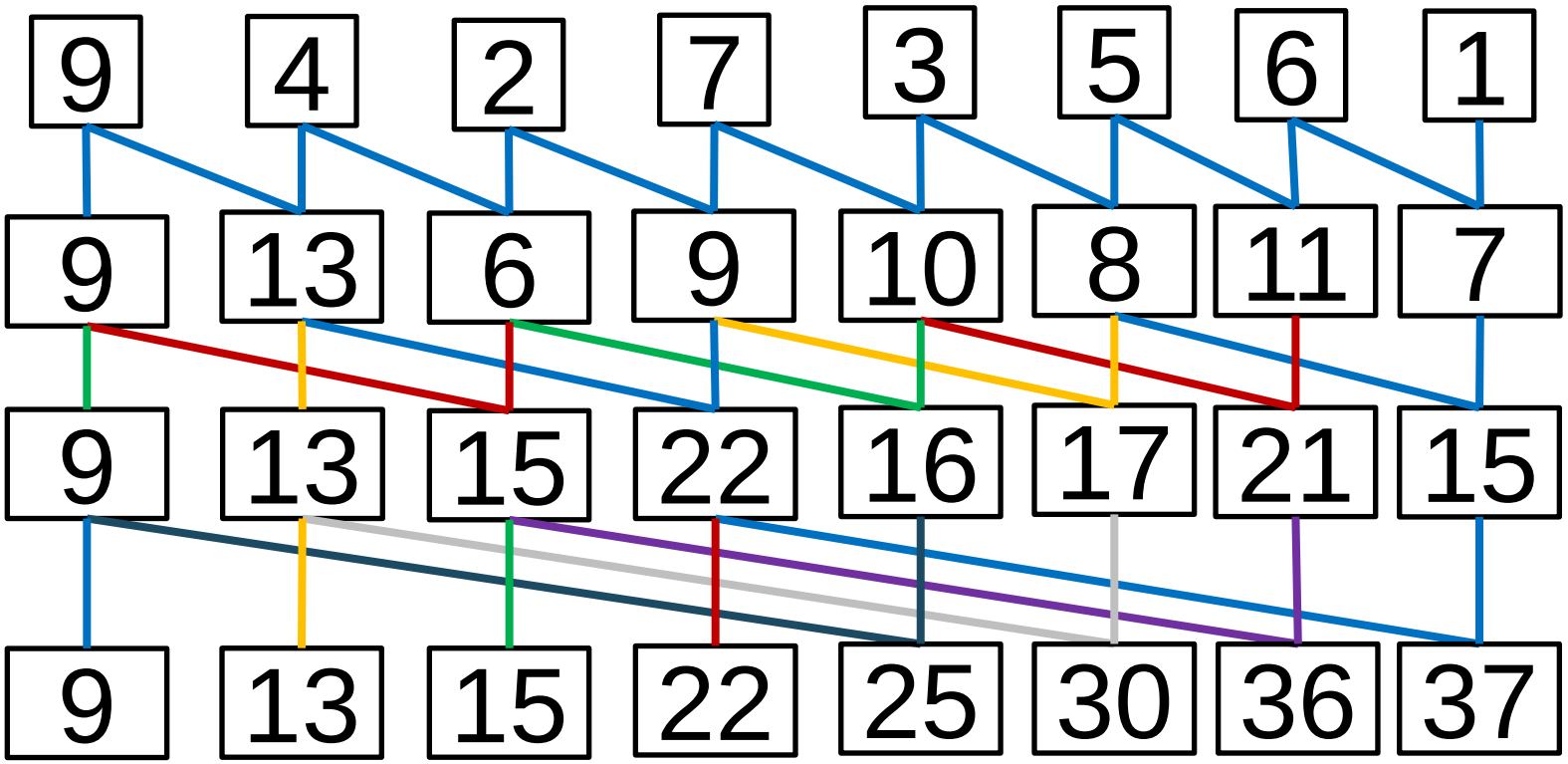
Scan – Cum funcționează?



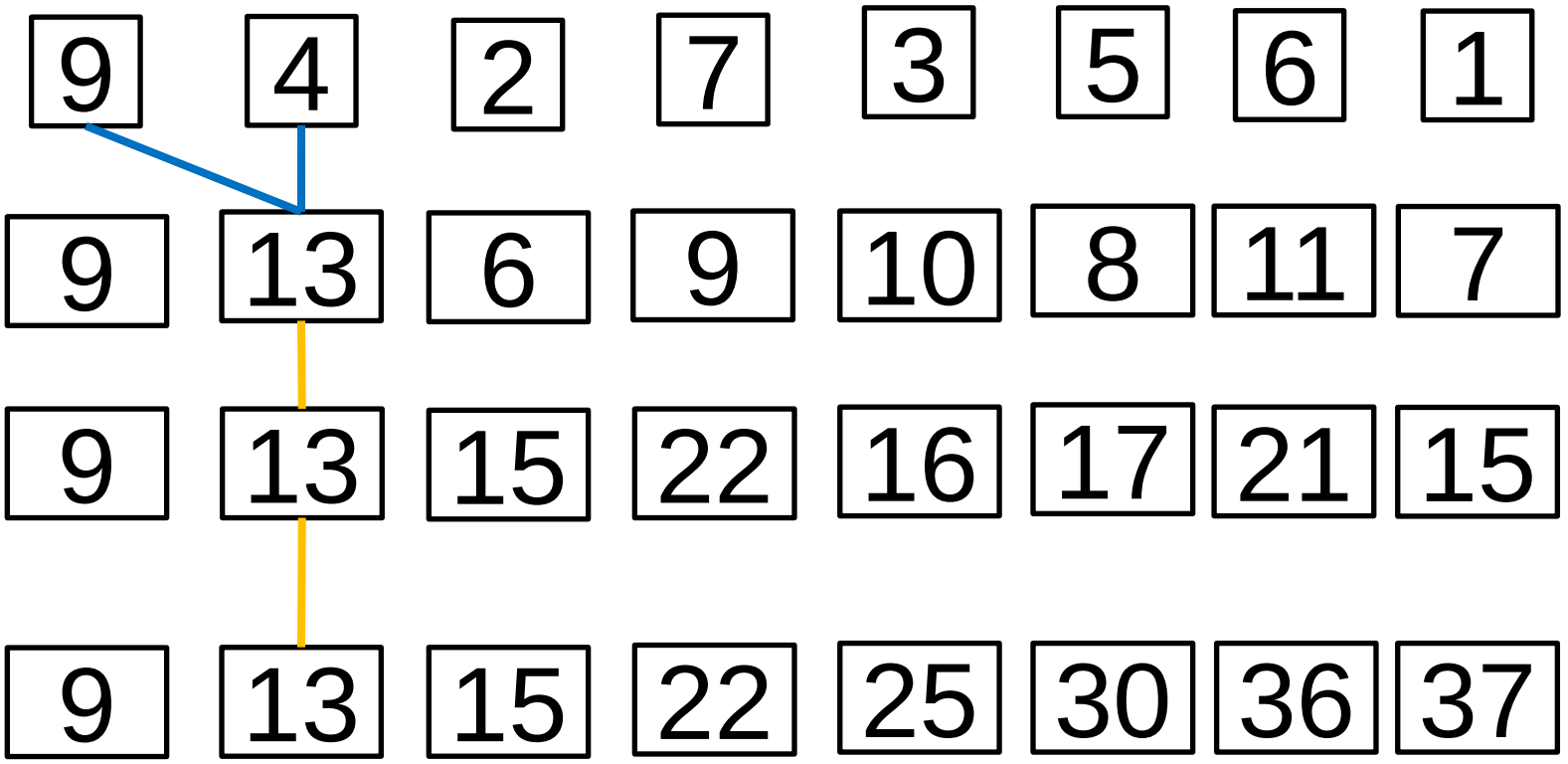
Scan – Cum funcționează?



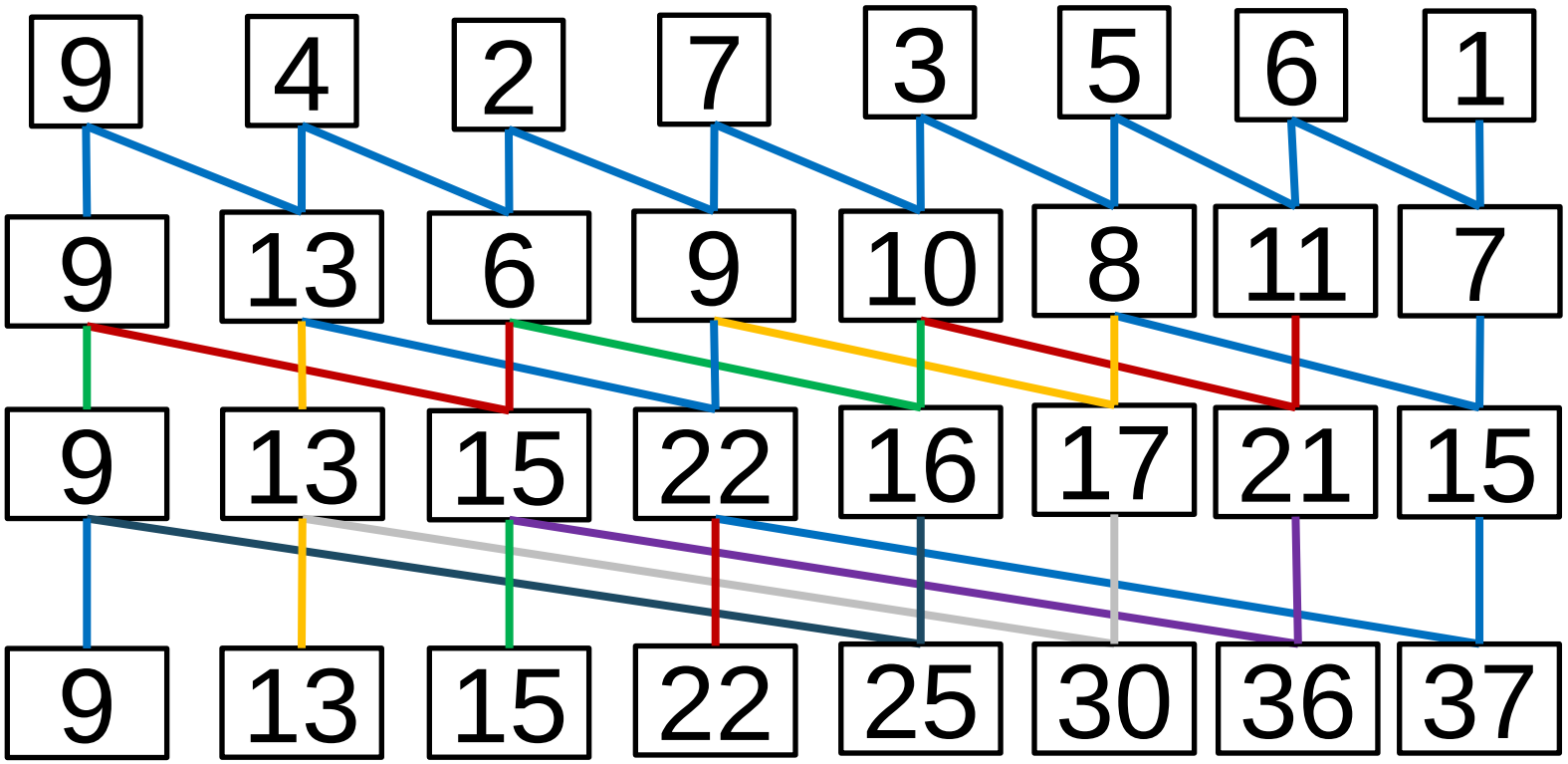
Scan – Cum funcționează?



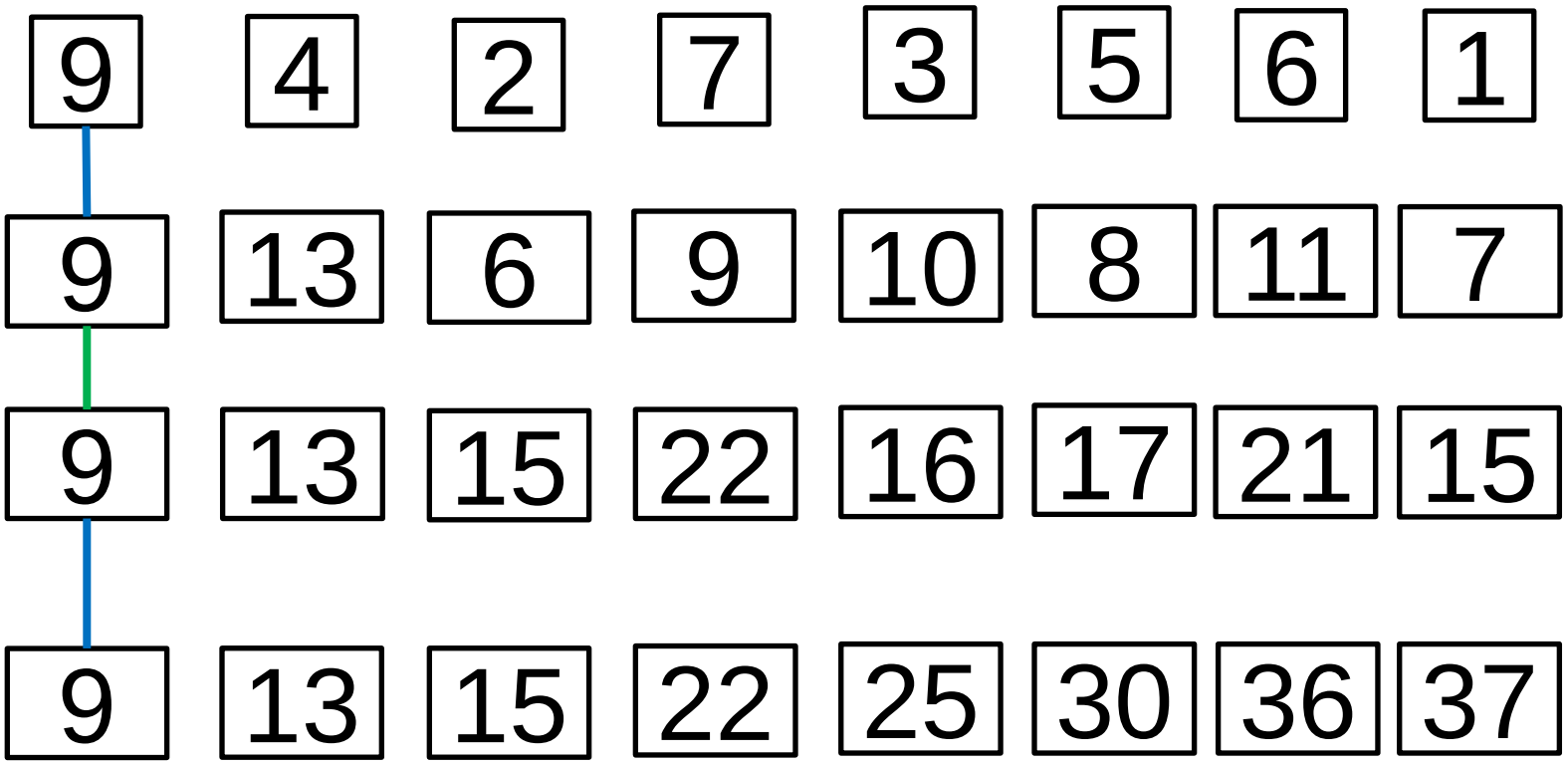
Scan – Cum funcționează?



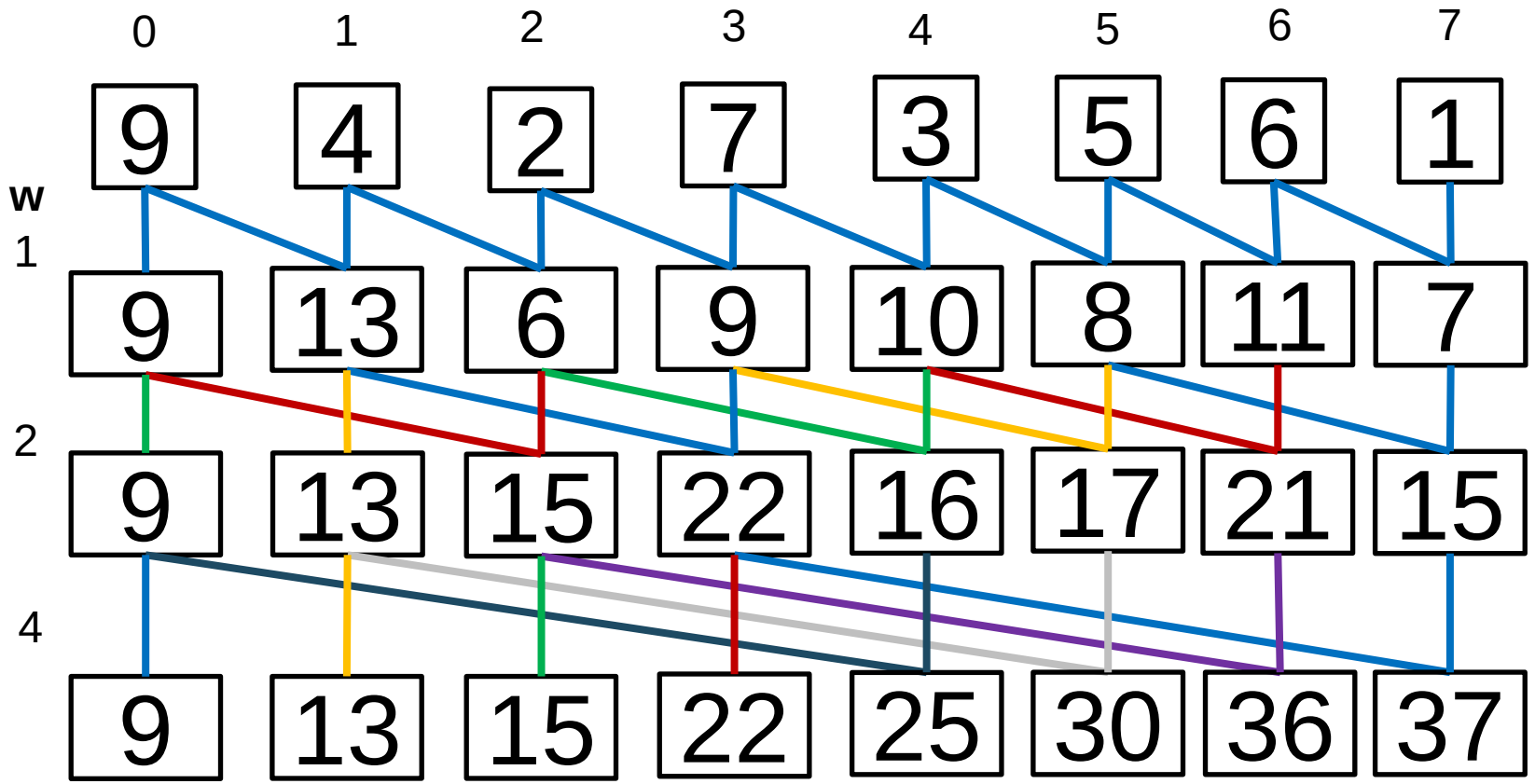
Scan – Cum funcționează?



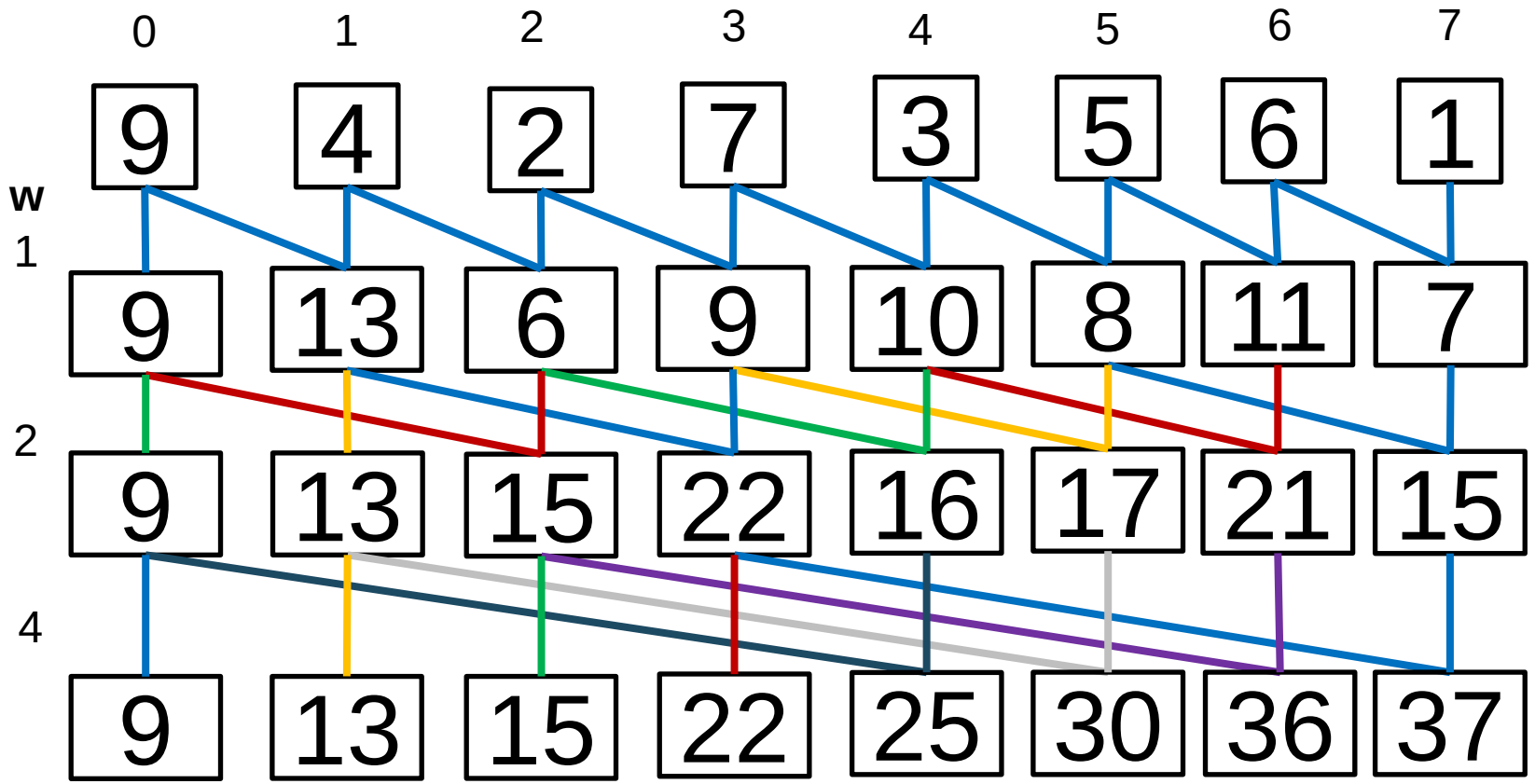
Scan – Cum funcționează?



Scan – implementare

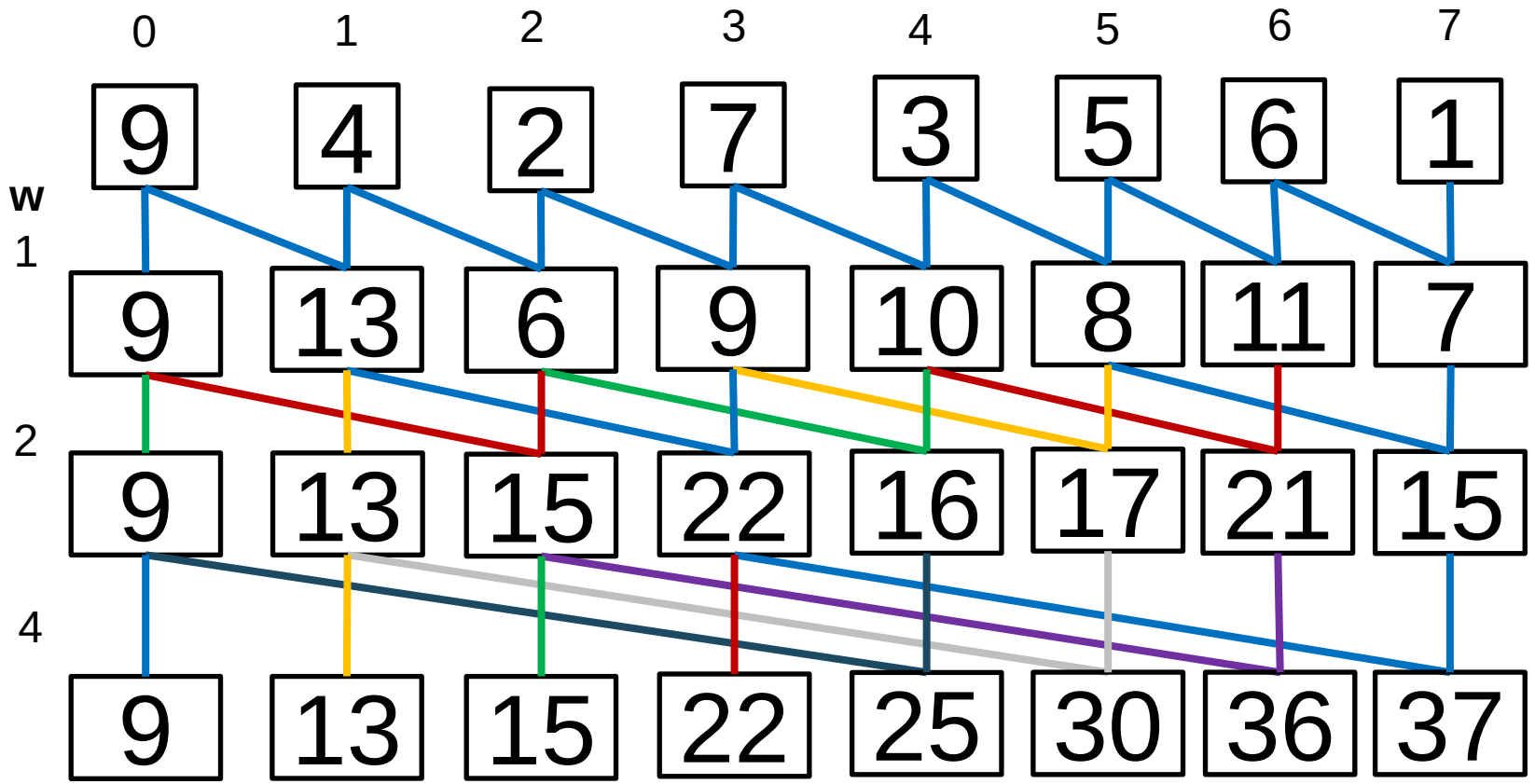


Scan – implementare – imagine globală



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

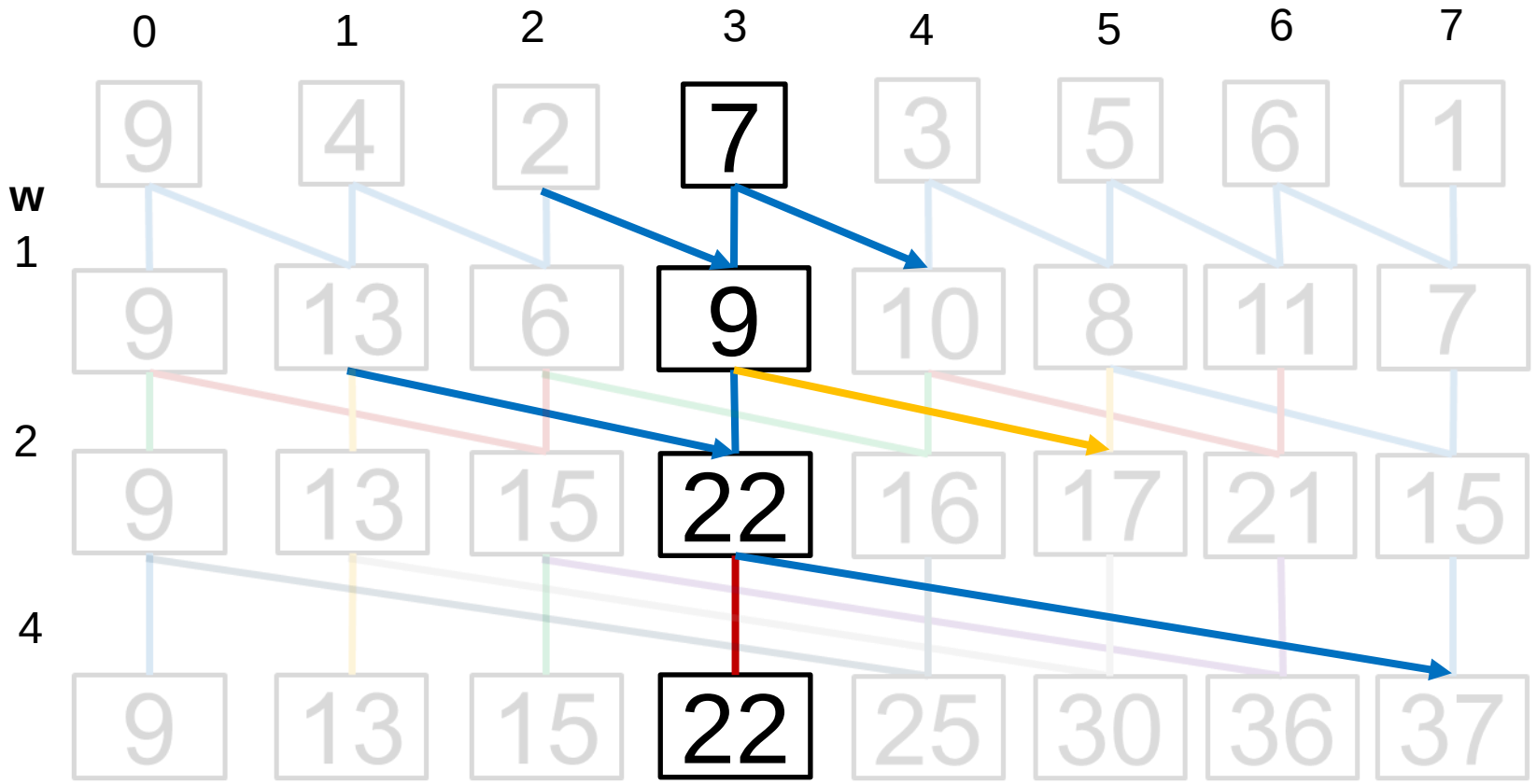
Scan – implementare – imagine globală



v[i] = v[i] + v[i - w]

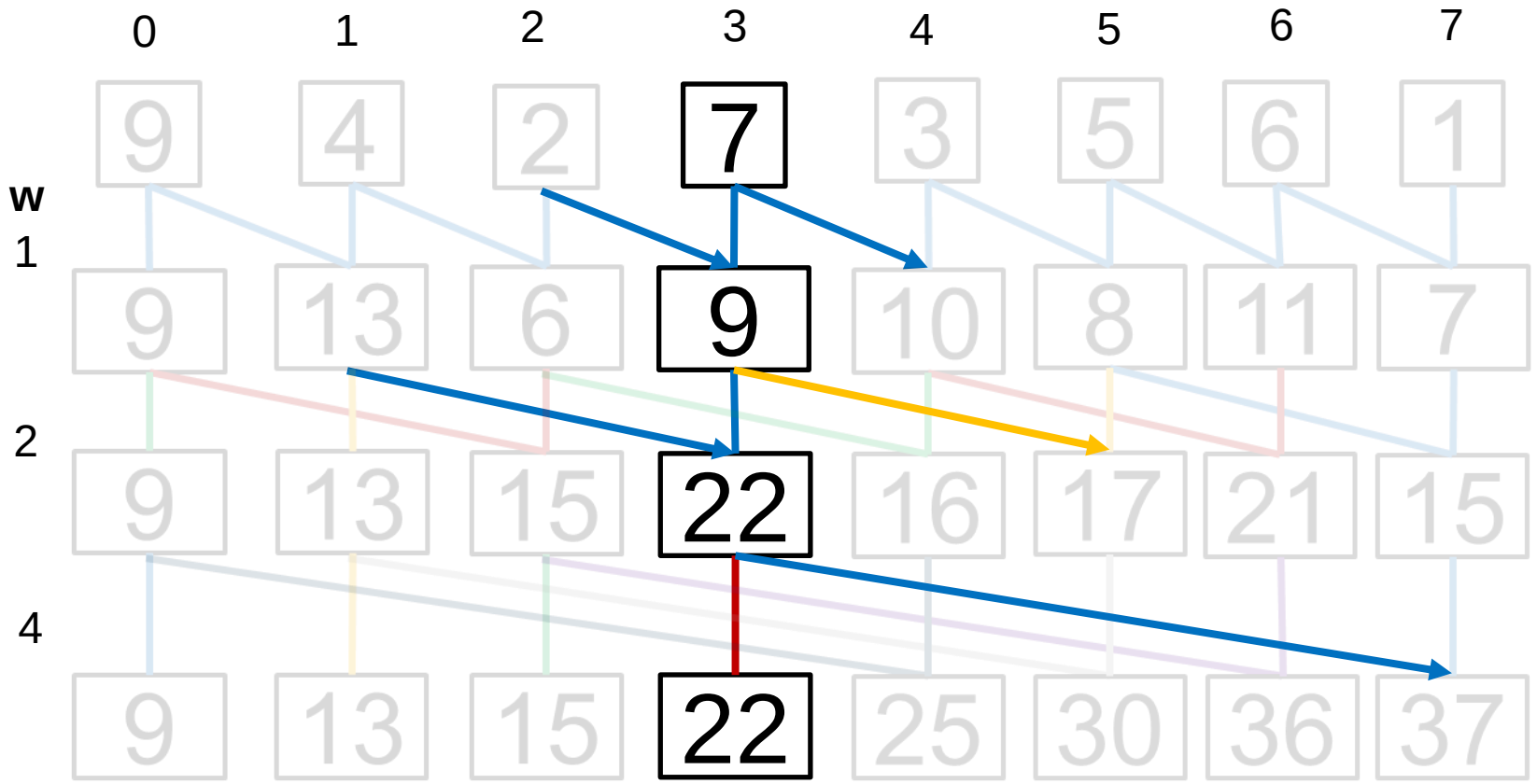
Dar dacă i - w < 0?

Scan – implementare – imagine locală



De unde se primesc valori?
Unde se trimit valori?

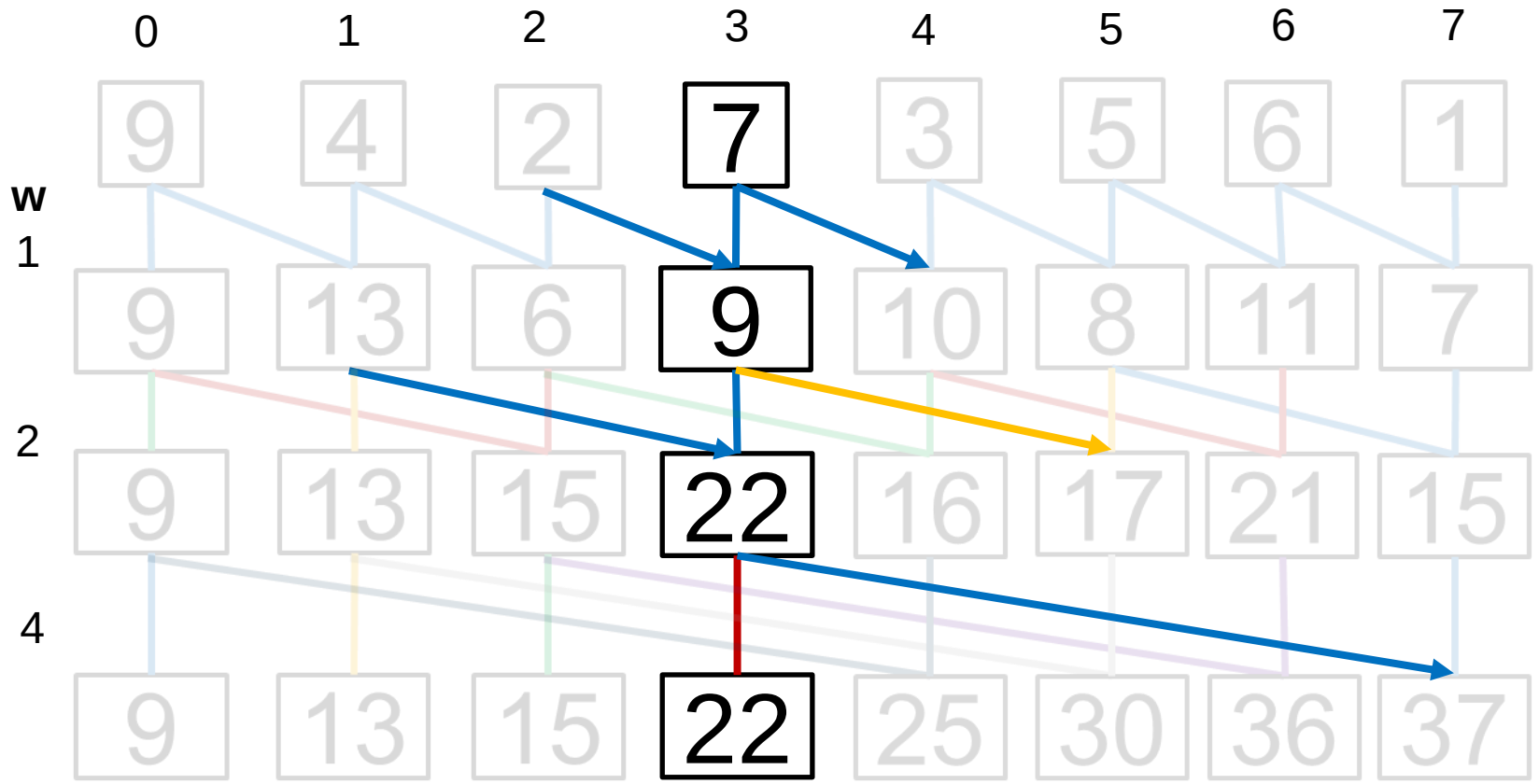
Scan – implementare – imagine locală



De unde se primesc valori? id - w

Unde se trimit valori? id + w

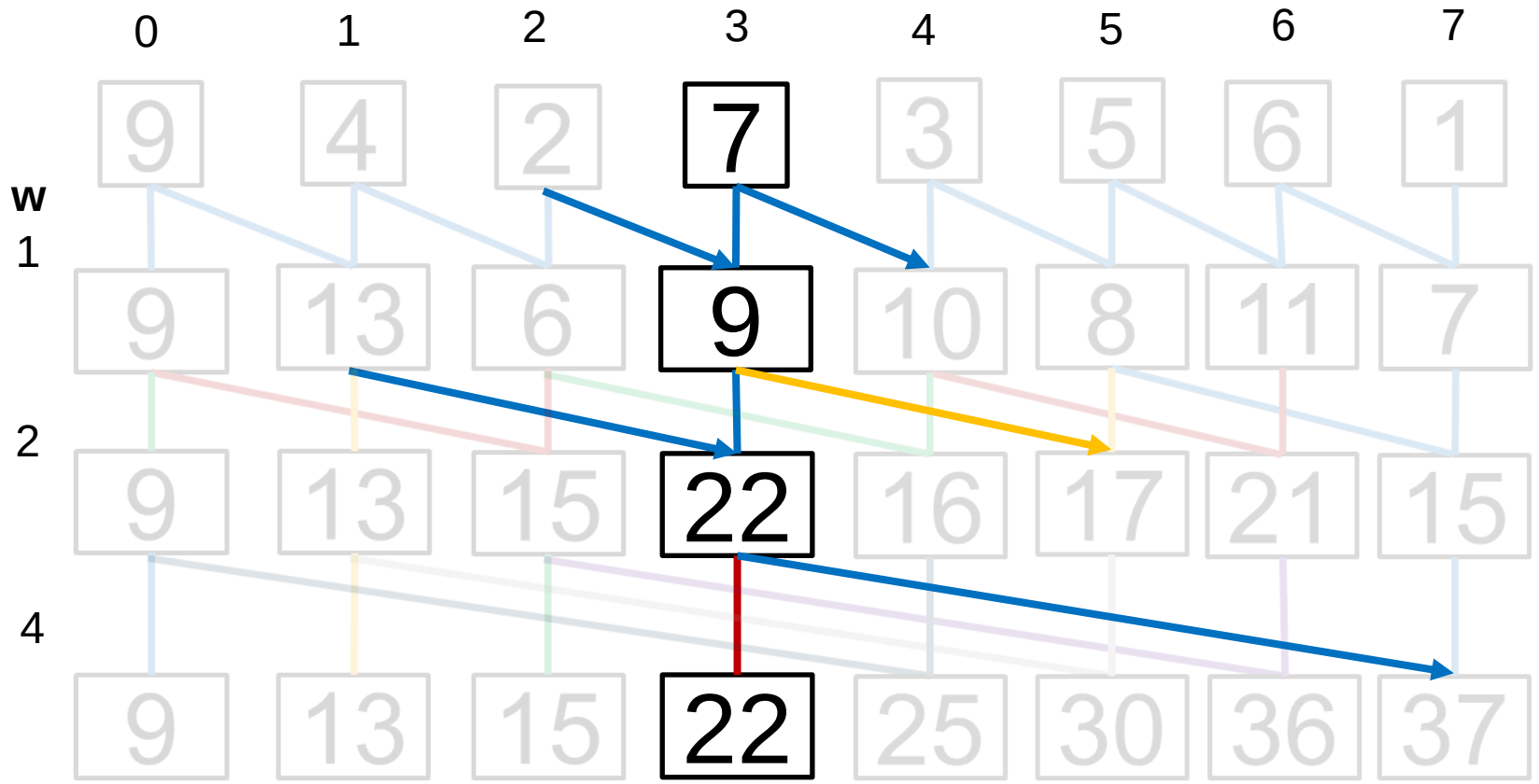
Scan – implementare – imagine locală



De unde se primesc valori? id - w Când se primesc valori?

Unde se trimit valori? id + w Când se trimit valori?

Scan – implementare – imagine locală

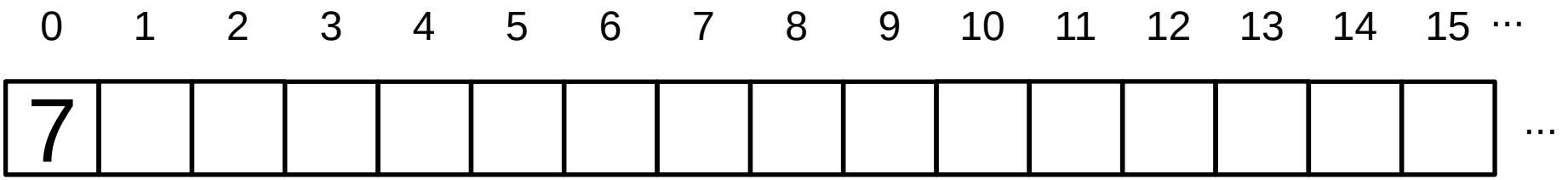


De unde se primesc valori? id - w Când se primesc valori? >= 0

Unde se trimit valori? id + w Când se trimit valori? < N

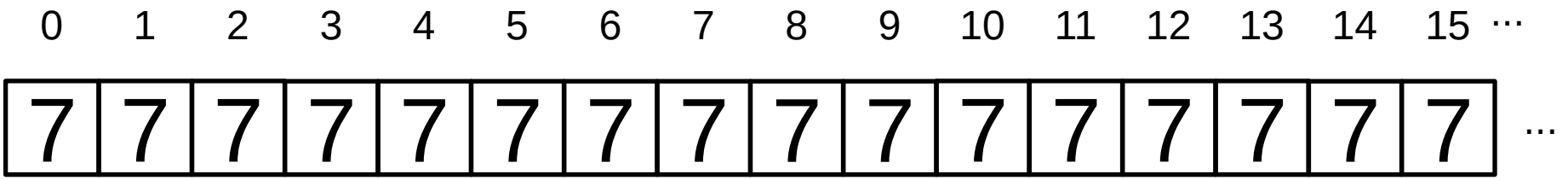
Broadcast

Start



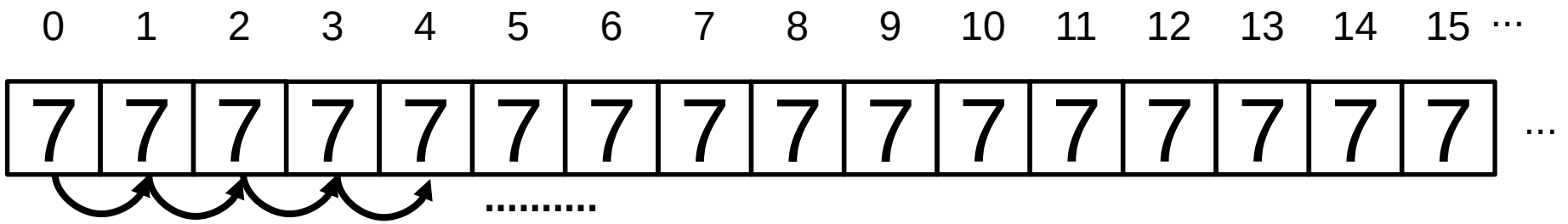
Broadcast

End



Broadcast ineficient

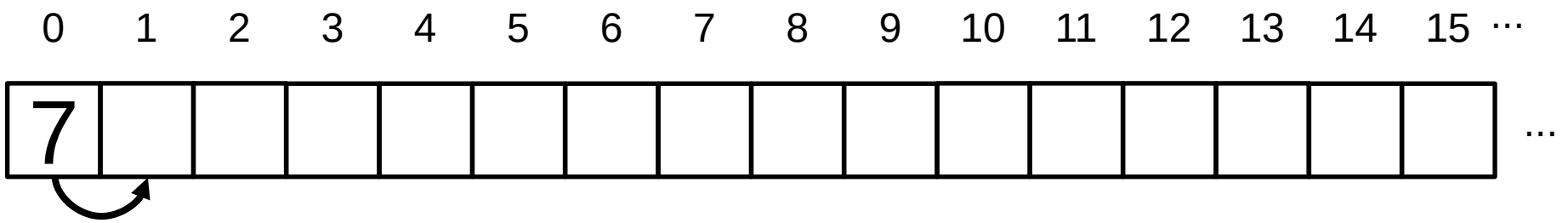
Complexitate: $O(N)$



Broadcast

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

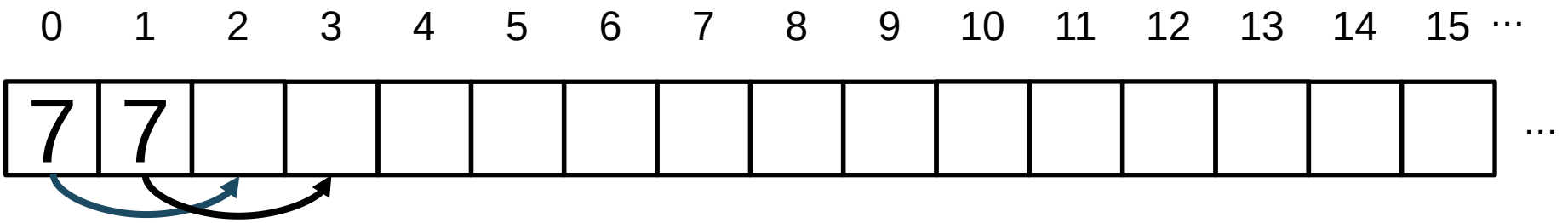
w = 1



Broadcast

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

w = 2

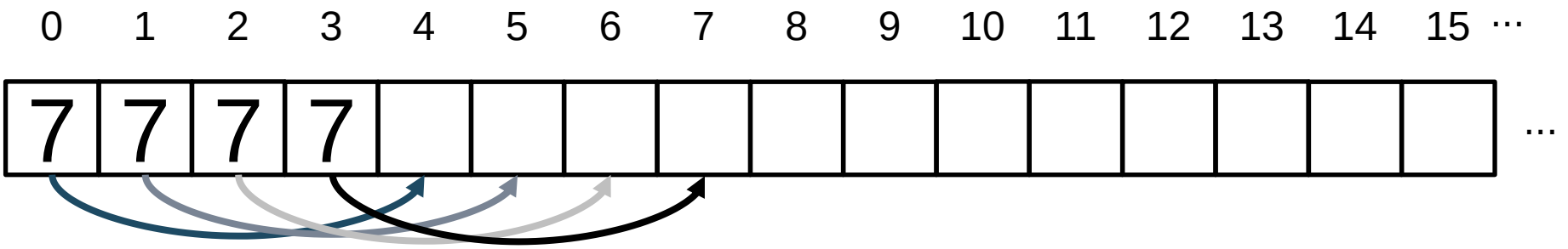


Aceste operații pot fi executate în paralel

Broadcast

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

w = 4

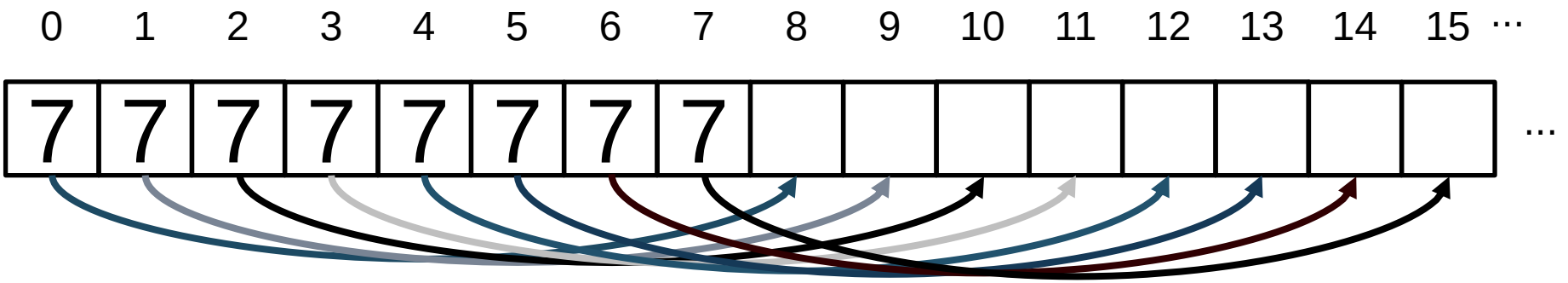


Aceste operații pot fi executate în paralel

Broadcast

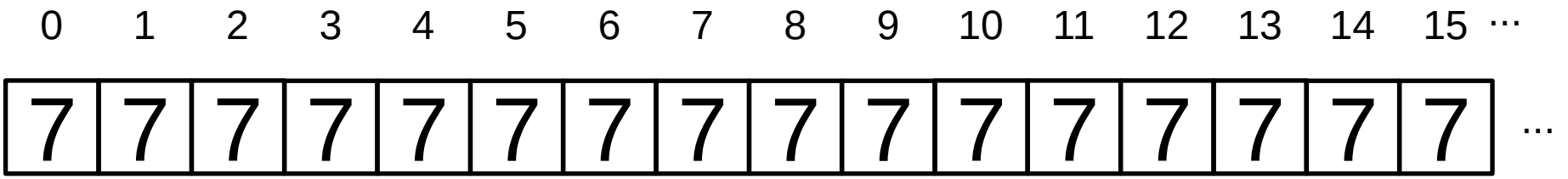
Fiecare element care are valoarea
o copiază la poziția sa + w

w = 8

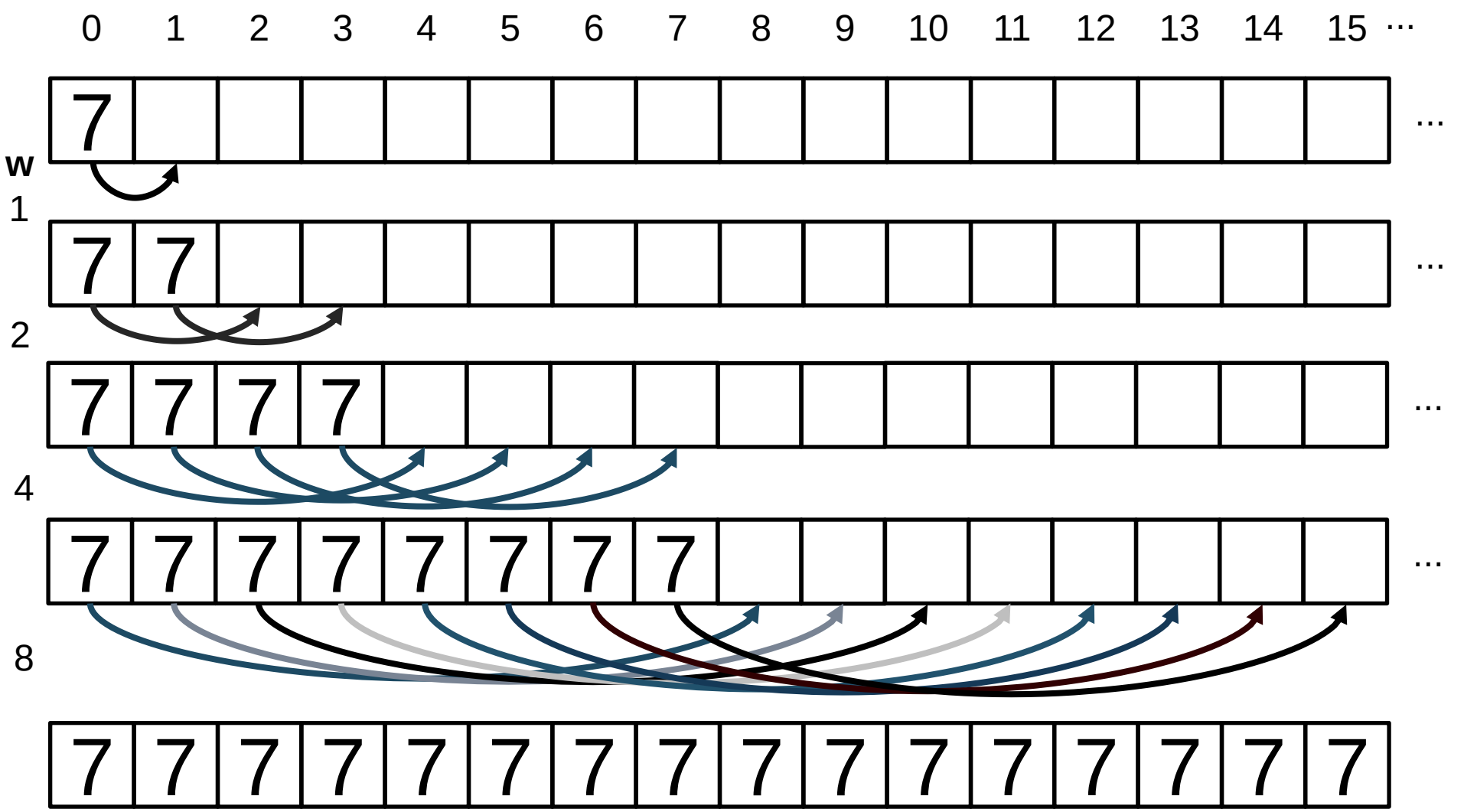


Aceste operații pot fi executate în paralel

Broadcast

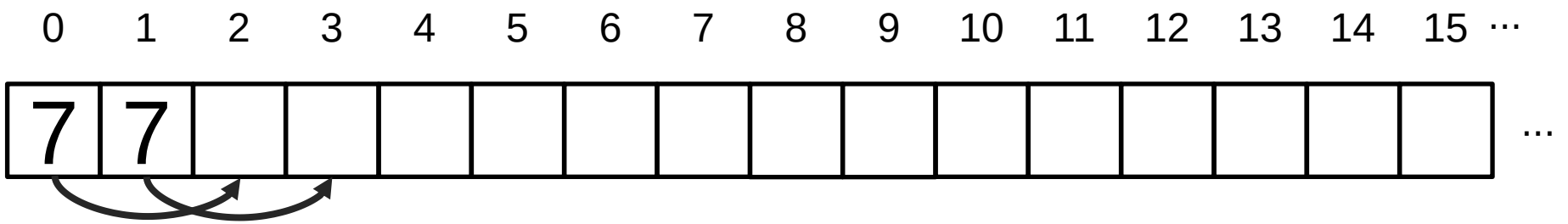


Broadcast


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

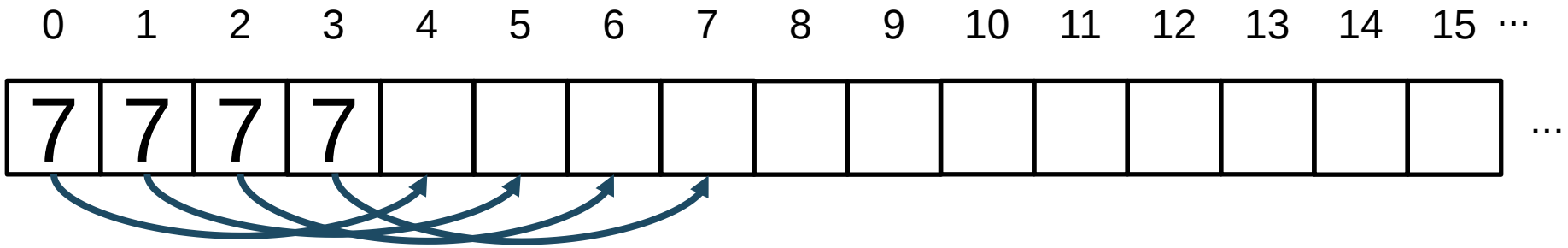
Broadcast

w = 2



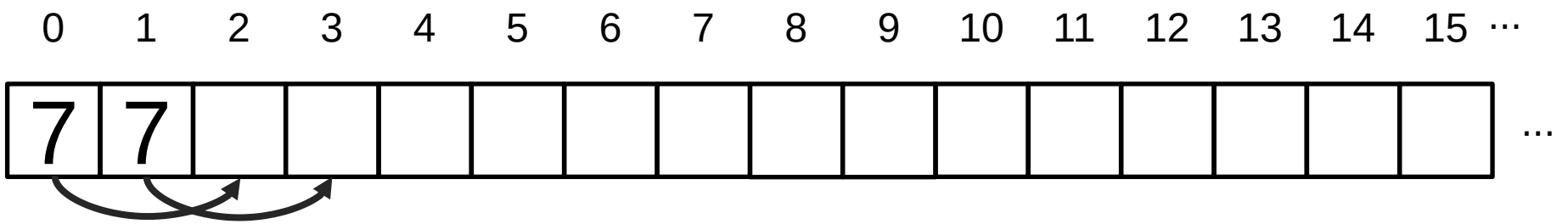
Aceste operații **NU** pot fi executate în paralel

w = 4

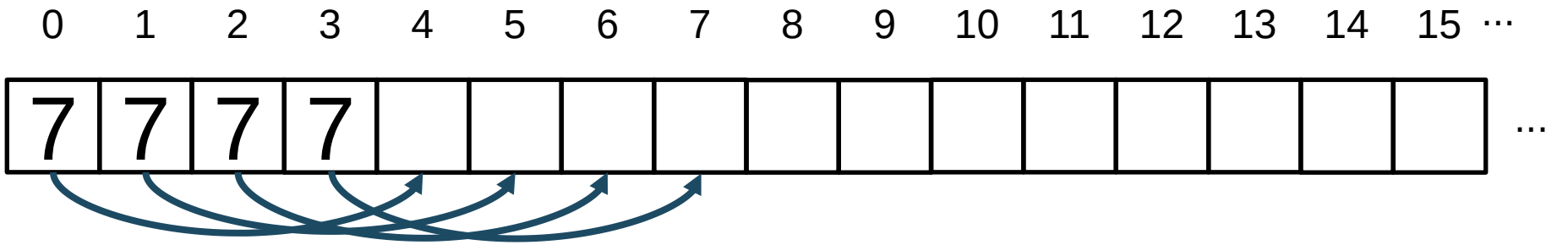


Broadcast

w = 2

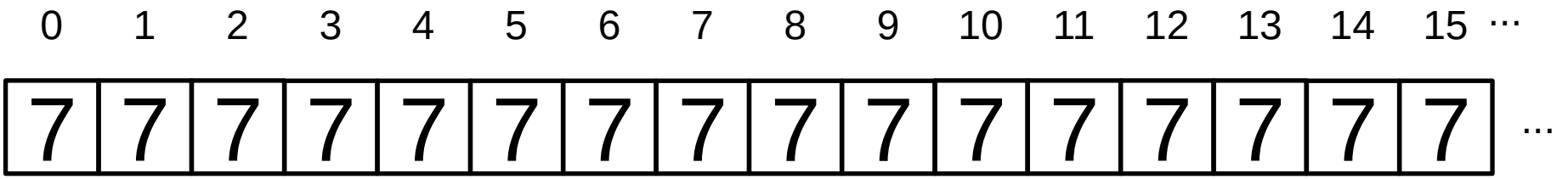


w = 4



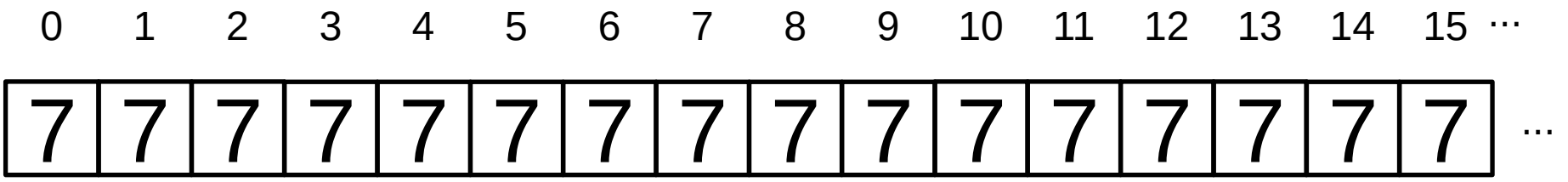
Broadcast

Complexitate?



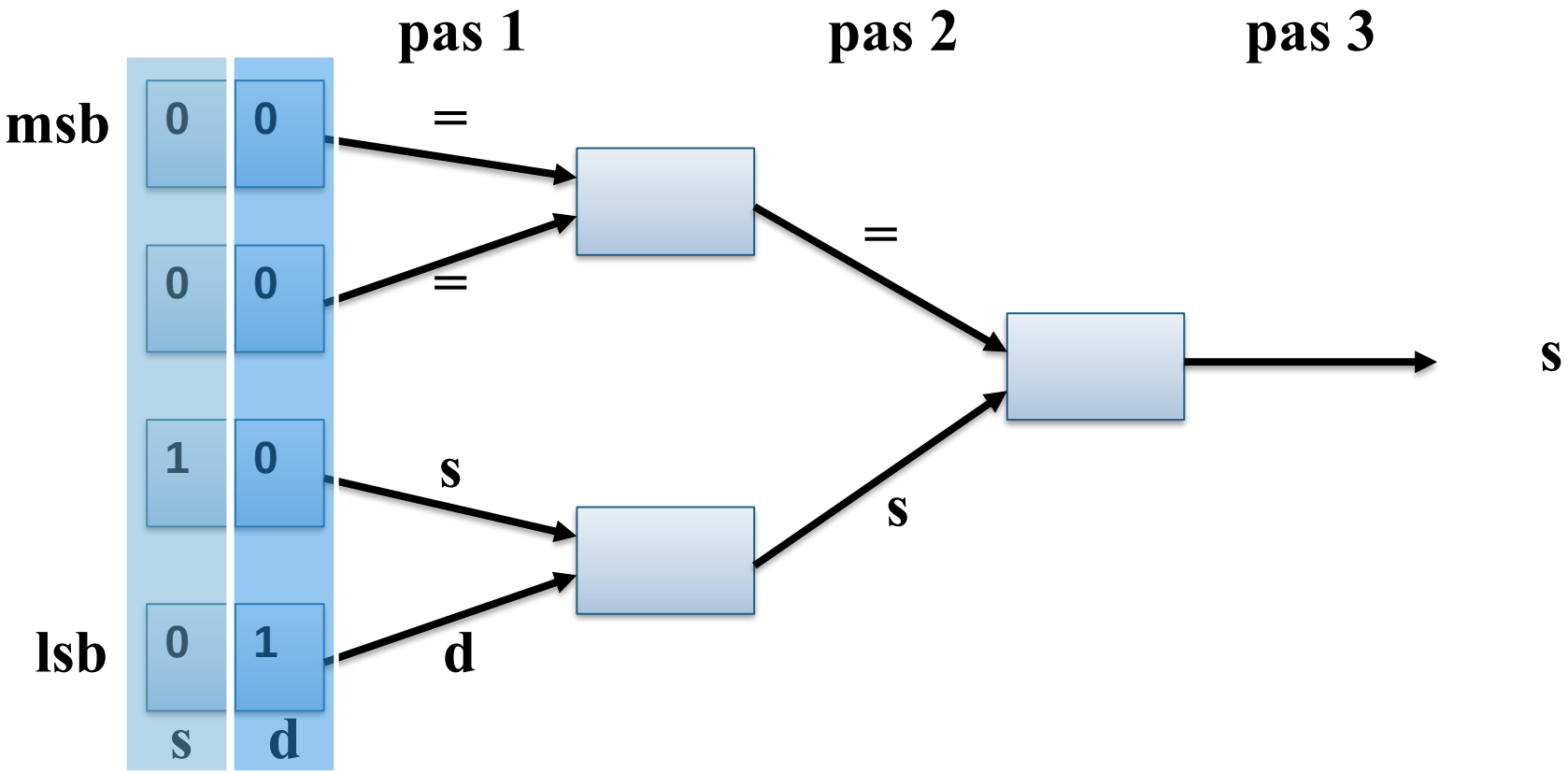
Broadcast

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



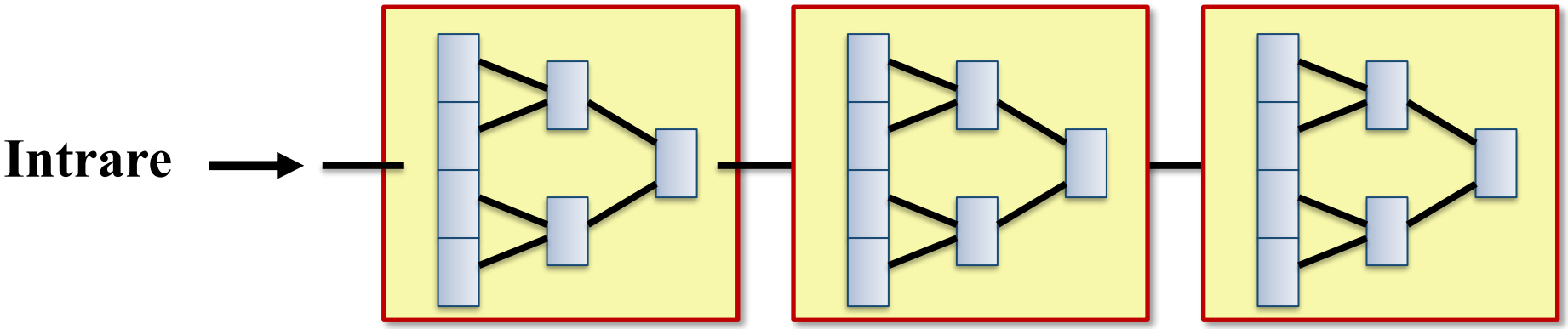
Exemplu complex – multiple tehnici

- Operație pe biți – comparație a două numere.



Exemplu complex – multiple tehnici

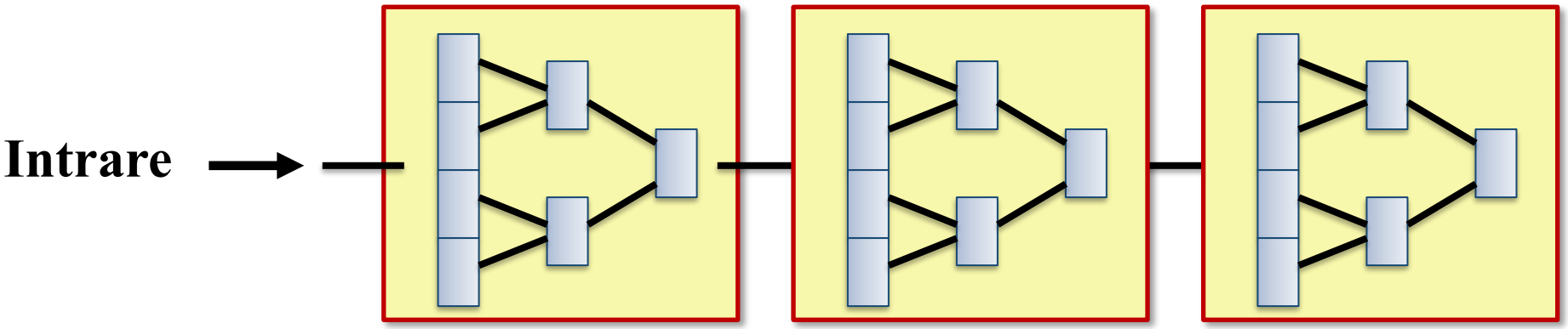
Algoritmul de sortare cu pipeline devine:



Complexitate?

Exemplu complex – multiple tehnici

Algoritmul de sortare cu pipeline devine:



Complexitate?

$$O(N * \log(numbiți))$$