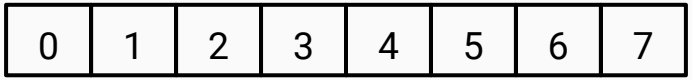
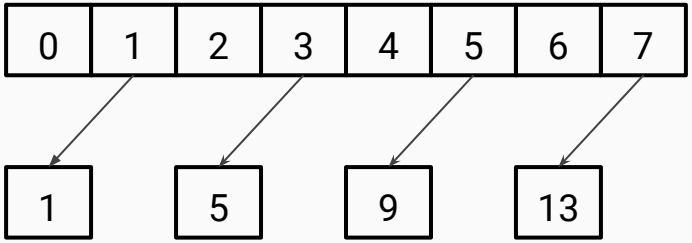
Reduce



În acest exemplu, fiecare proces are câte un element (8 elemente - 8 procese)

Reduce

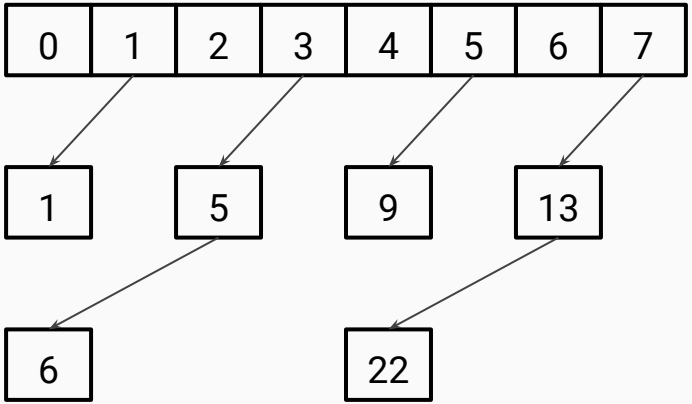


- procesul 1 trimite procesului 0, care adună valoarea primită (1 + 0 = 1)
- procesul 3 trimite procesului 2, care adună valoarea primită (3 + 2 = 5)
- procesul 5 trimite procesului 4, care adună valoarea primită (5 + 4 = 9)
- procesul 7 trimite procesului 6, care adună valoarea primită (7 + 6 = 13)

Deci:

- procesul 0 are valoarea 1
- procesul 2 are valoarea 5
- procesul 4 are valoarea 9
- procesul 6 are valoarea 13

Reduce

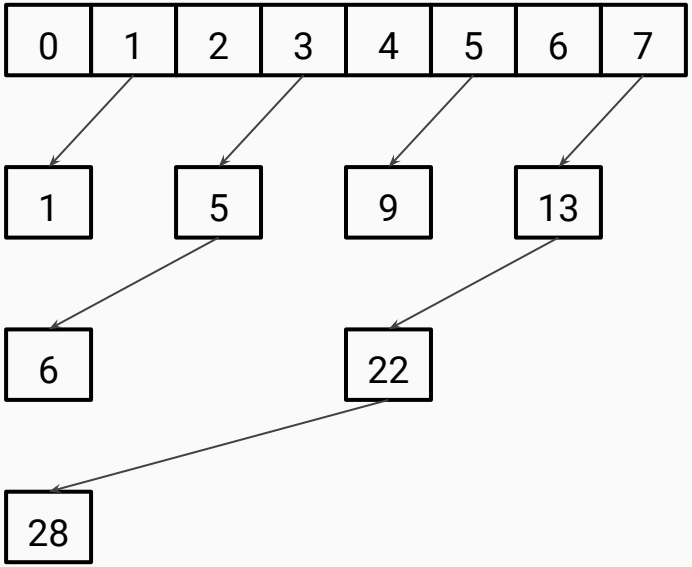


- procesul 2 trimite procesului 0, care adună valoarea primită (5 + 1 = 6)
- procesul 6 trimite procesului 4, care adună valoarea primită (13 + 9 = 22)

Deci:

- procesul 0 are valoarea 6
- procesul 4 are valoarea 22

Reduce



- procesul 4 trimite procesului 0, care adună valoarea primită (22 + 6 = 28)

Deci:

- procesul 0 are valoarea 28

Reduce - pseudocod

```
for (pas = 2; pas <= nr_procese; pas *= 2)
    if (rank % pas == 0)
        primește la procesul cu rank-ul [rank + (pas / 2)]
        adună
    else if (rank % (pas / 2) == 0)
        trimite la procesul cu rank-ul [rank - (pas / 2)]
```

Reduce - pseudocod - rulare

pas = 2:

- 1 (1 % (2 / 2) == 0) trimite lui 0 (1 - (2 / 2))
- 0 (0 % 2 == 0) primește de la 1 (0 + (2 / 2))

- 3 (3 % (2 / 2) == 0) trimite lui 2 (3 - (2 / 2))
- 2 (2 % 2 == 0) primește de la 3 (2 + (2 / 2))

- 5 (5 % (2 / 2) == 0) trimite lui 4 (5 - (2 / 2))
- 4 (4 % 2 == 0) primește de la 5 (4 + (2 / 2))

- 7 (7 % (2 / 2) == 0) trimite lui 6 (7 - (2 / 2))
- 6 (6 % 2 == 0) primește de la 7 (6 + (2 / 2))

Reduce - pseudocod - rulare

pas = 4:

- 2 (2 % (4 / 2) == 0) trimite lui 0 (2 - (4 / 2))
- 0 (0 % 4 == 0) primește de la 2 (0 + (4 / 2))

- 6 (6 % (4 / 2) == 0) trimite lui 4 (6 - (4 / 2))
- 4 (0 % 4 == 0) primește de la 6 (4 + (4 / 2))

pas = 8:

- 4 (4 % (8 / 2) == 0) trimite lui 0 (4 - (8 / 2))
- 0 (0 % 8 == 0) primește de la 4 (0 + (8 / 2))