# Funkcje haszujące

$$h : \Omega \longrightarrow \{0, \ldots, n-1\} = [n]$$

- kryptograficzne
  - MD5, SHA-...
  - $\{h(x_i) : i = 1 \ldots n\}$ → $[n]$
  - jedn. rozkład: $\Pr[h(x) = i] \cong \frac{1}{n}$
  - → Znając $y = h(x)$ łatwo trudno jest znaleźć $x$,
- nie-kryptograficzne

MurMur hash

- k-uniwersalność dla dużego k

{Mu ← multiply
{r ← rotate

# MurmurHash

```
scala:
import scala.util.hashing.{MurmurHash3 => MH3}
.....
val seed = 1353451;          (uint)
myHash(data) {
  val m = MH3.stringHash(data, seed)
  m
}
```

---

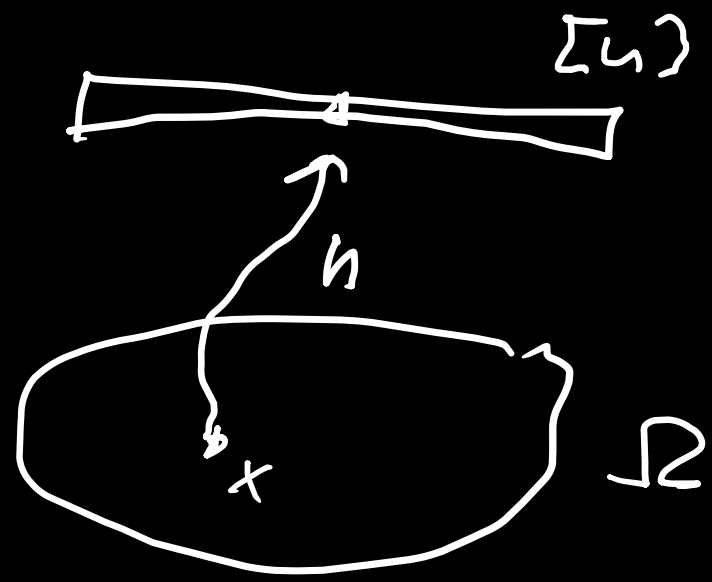
$h_1, \ldots, h_K$ : murmur. fuck, bash.

val = $[s_1, s_2, \ldots, s_K]$ ← seed's

Dobre hash functions,

$h \sim$ losowe niezależne

wybory $\in [n]$




- $X_1, X_2, X_3, \ldots$ – ciąg niez. zm. losowych o rozkł. $\in [n]$

$$L = \min\{k : X_k \in \{X_1, \ldots, X_{k-1}\}\}$$ ← moment pierwszej kolizji

$$E[L] \approx \sqrt{\frac{\pi \cdot n}{2}} + \frac{3}{2} + O\left(\frac{1}{\sqrt{n}}\right)$$

$$\approx \sqrt{n}$$

Birthday Paradox

- $C = \min\{k : \{X_1, \dots, X_k\} = [n]\}$

$$P[C = \infty] = 0$$

$$E[C] = nH_n \approx n \ln n$$

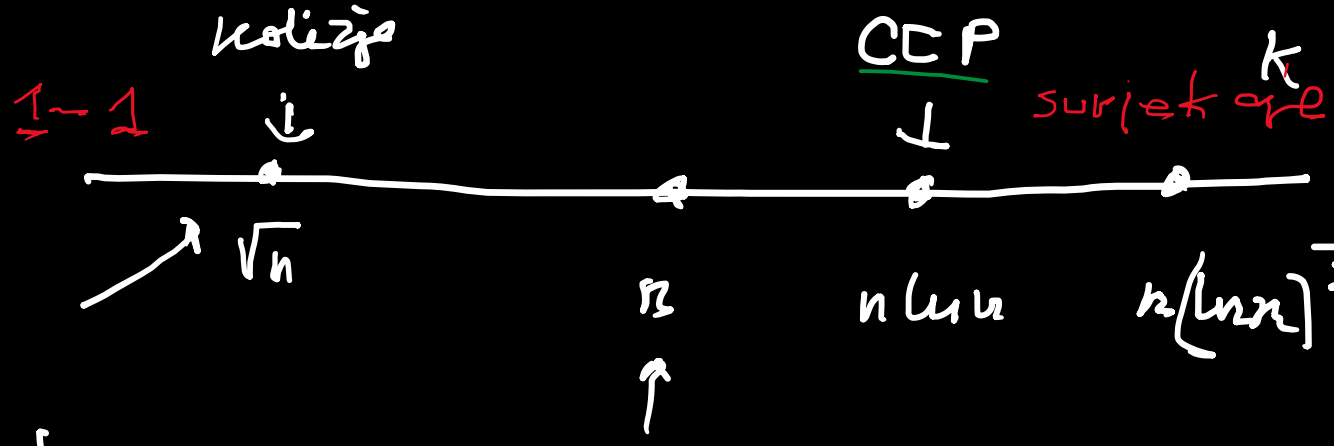
Coupon Collector Problem

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$$

$$H_n \approx \ln n + \gamma + O\left(\frac{1}{n}\right)$$

$$\gamma \approx 0.57$$

$$\approx \ln n$$




Pop. urodzi.

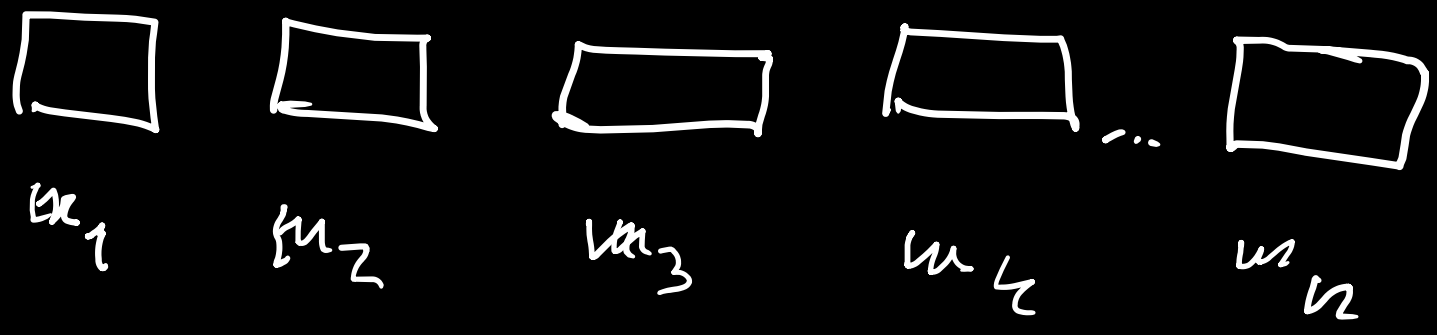
spośród pierwszych $\sqrt{n}$ ~ $\frac{\sqrt{n}}{e}$

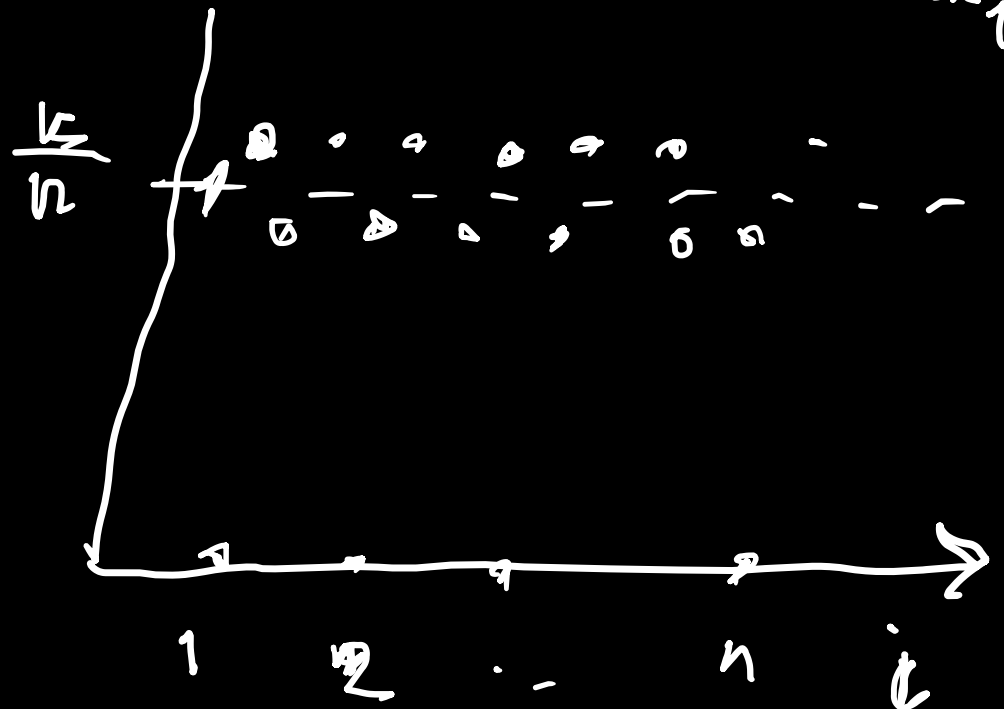
$k \geq n(\ln n)^3$ :

urny są prawie pełne.

wystarczy : $\approx \frac{k}{n}$ elem.

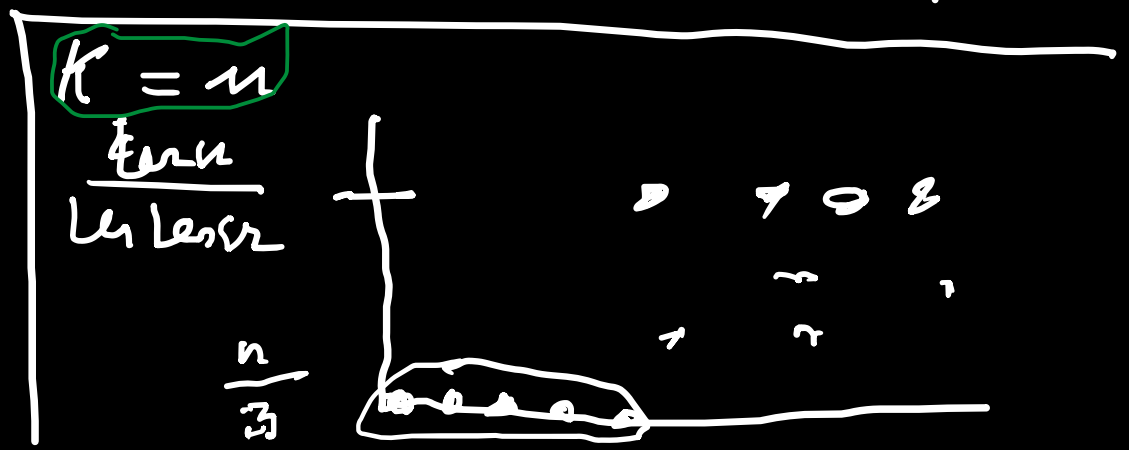
$k > n(\ln n)^3$





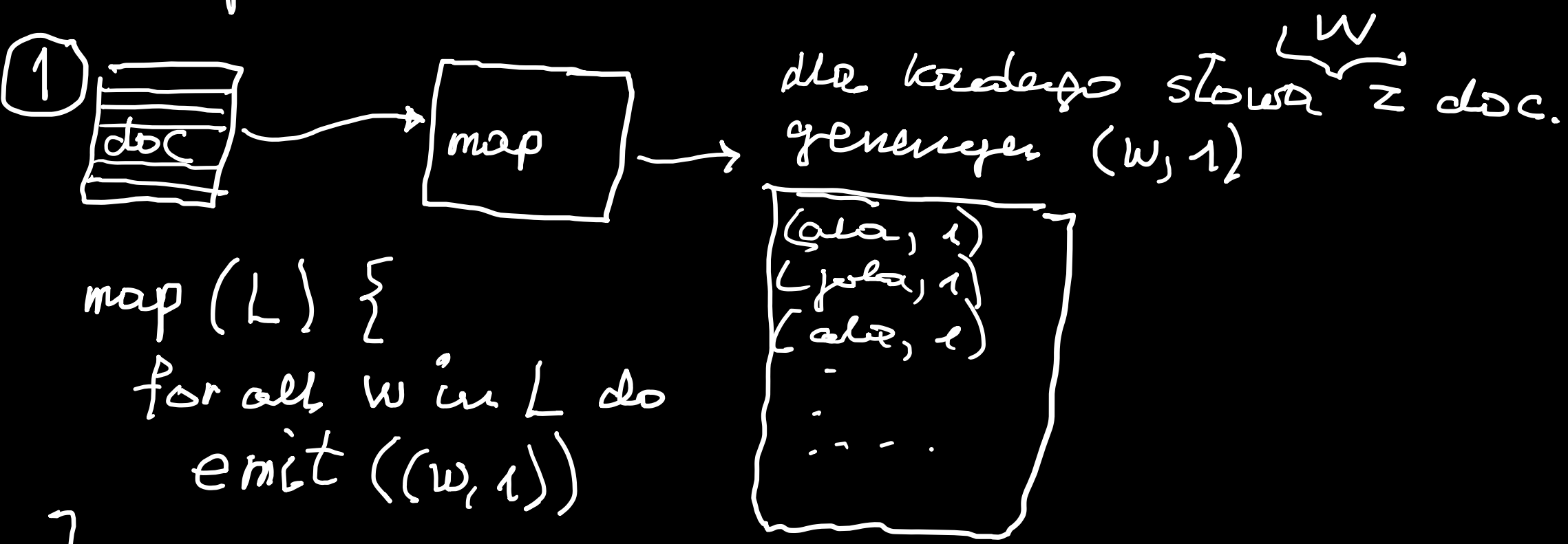

← mocno obliż dla podłogi.

to nie jest "logiczne",

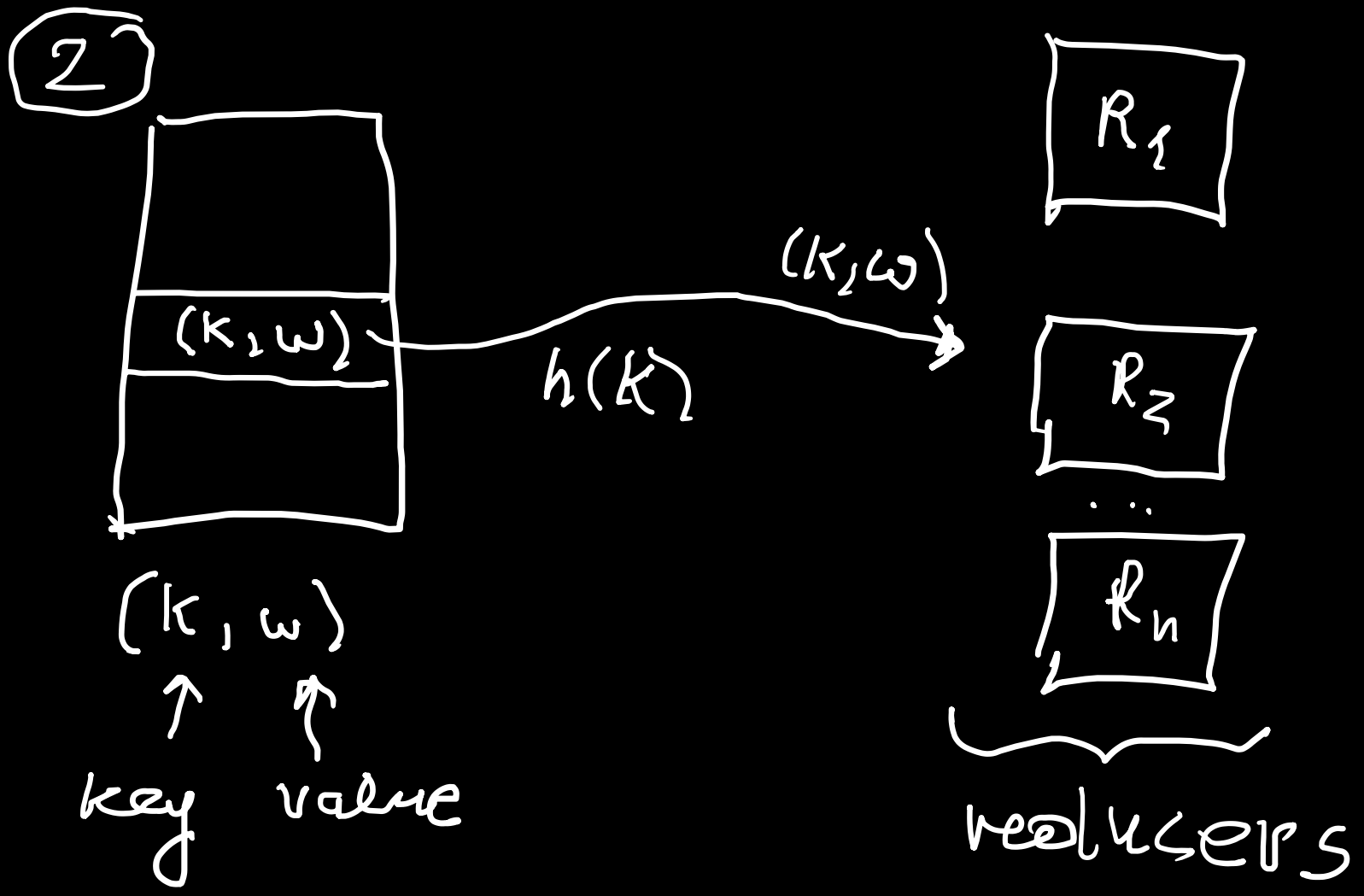


# Model Map-Reduce

Przykład: word count



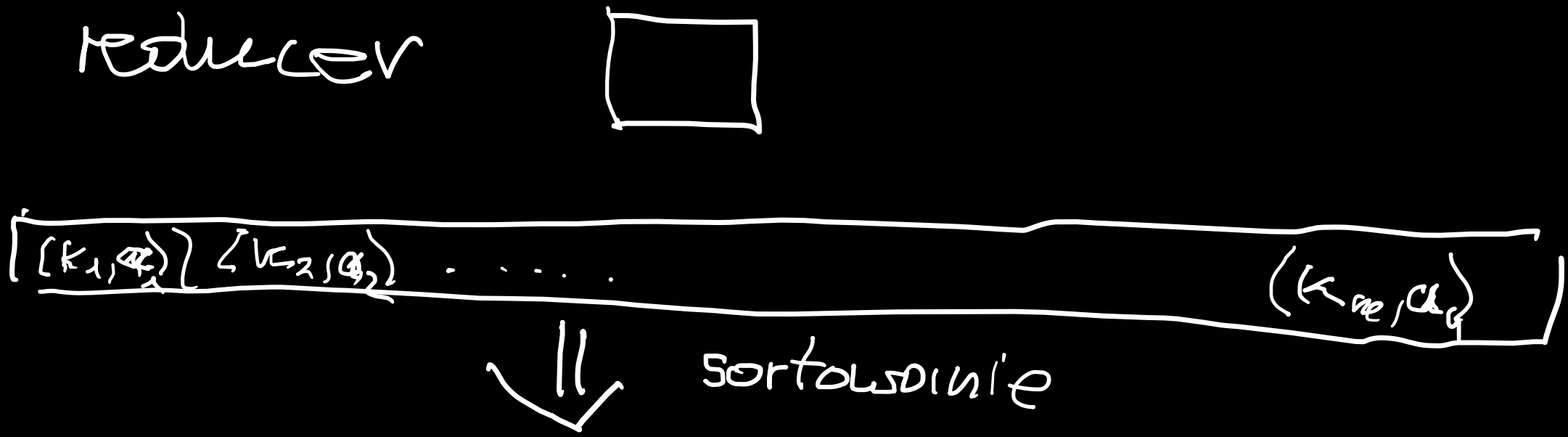

```
map (L) {
  for all w in L do
    emit ((w, 1))
}
```




$h : \Omega \to \{1, \ldots, n\}$

Fix

## ③ Reducer



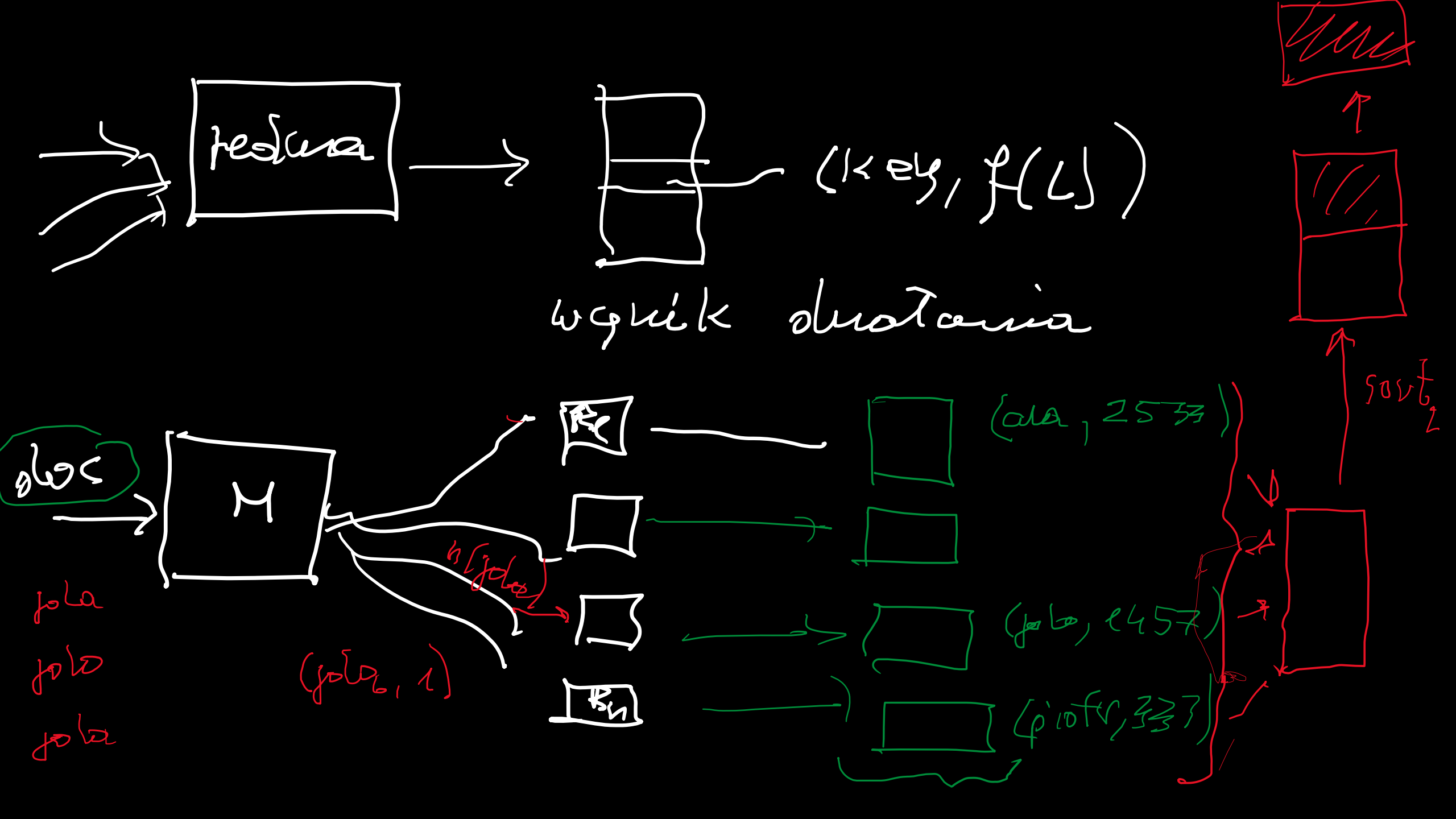

$$[[k_1, [a_{i_1} \dots a_{i_1}]], \dots, [k_r, [a_{j_1} \dots a_{j_r}]]]$$

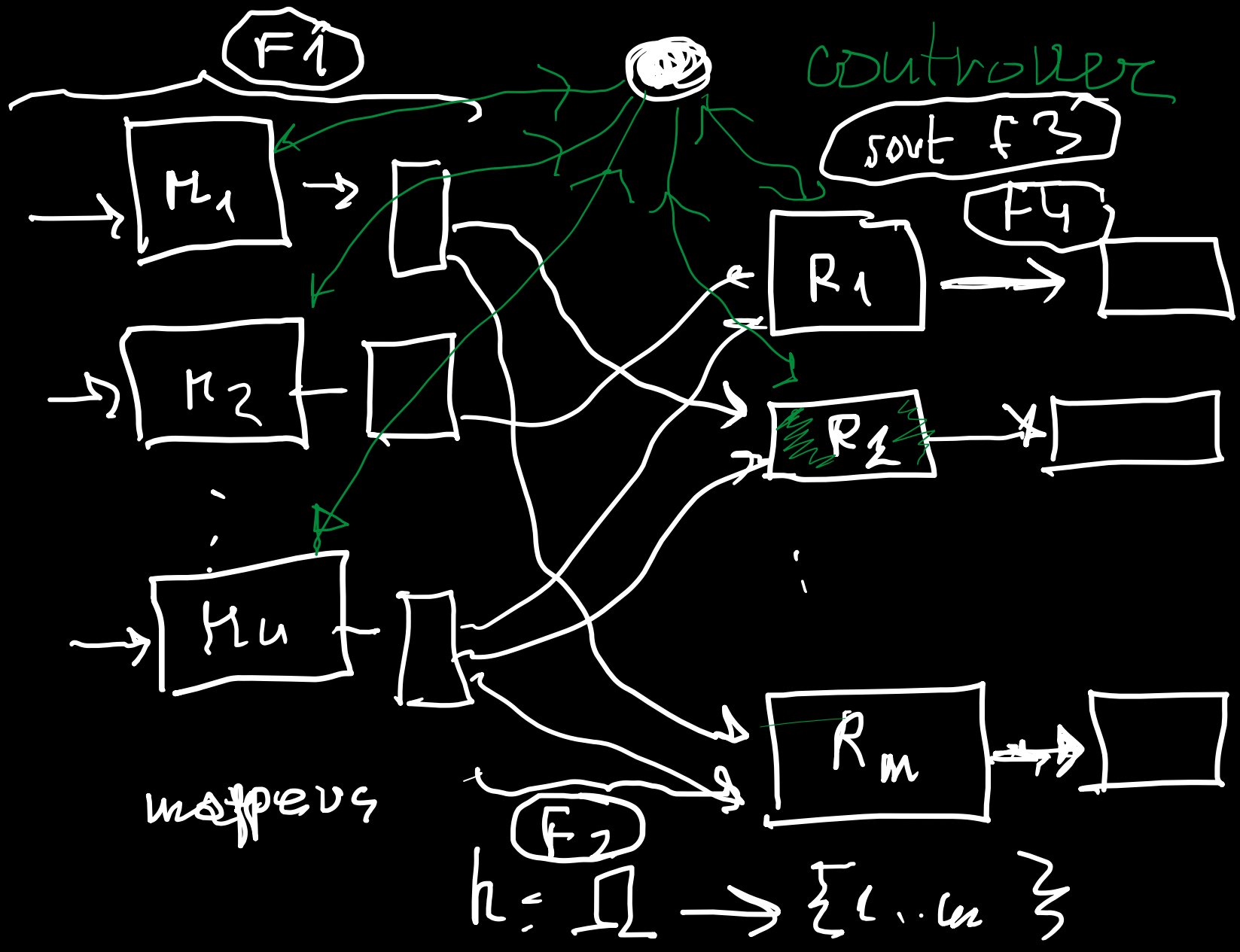
[alex, [1, 1, 1, .. 1]]

```
reduce (key, L) {
    m = f(L);
} emit((k, m))
```

word - count

```
reduce (key, L) {
    emit((key, length(L))
}
```


reduce
(key, f(L))
wynik akumulacja
sort
blok
M
jola
jola
jola
(jola, 1)
(jola, 253)
(jola, 457)
(piotr, 337)



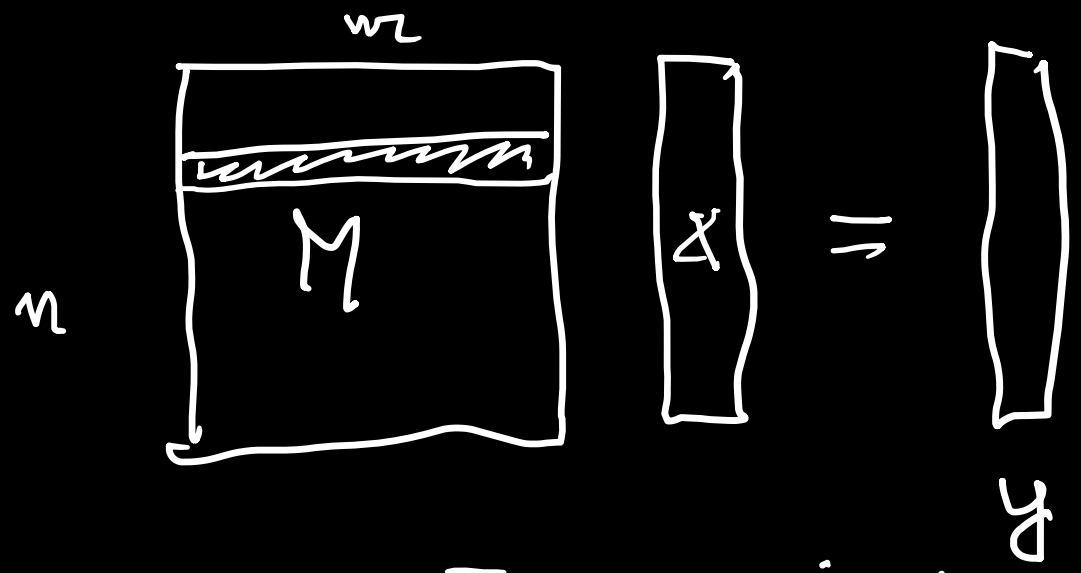

```
map (data){
  .....
  (k_i, w_i) ..(.)
}

reduce (key, L){
  emit (key, f(L))
}
```

{ Hadoop & Apache Spark

NIEZAWODNOŚĆ: duplikacja

(P5) Mnożenie macierzy przez wektor



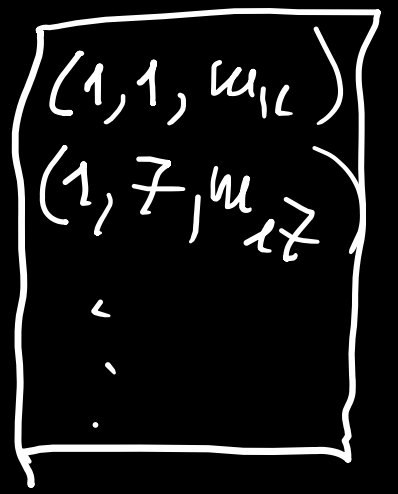

$$y_i = \sum_{j=1}^{n} m_{ij} \cdot x_j$$

$\overline{X}$ ← miejsce się w pamięci,

M ← za dużo

repr. macierzy: $(i, j, m_{ij})$

(mamy założ. że $m_{ij} \neq 0$)



```
map ((i, j, m)) {
    emit ((i, m · x_j))
}
```

$m_{i,j}$

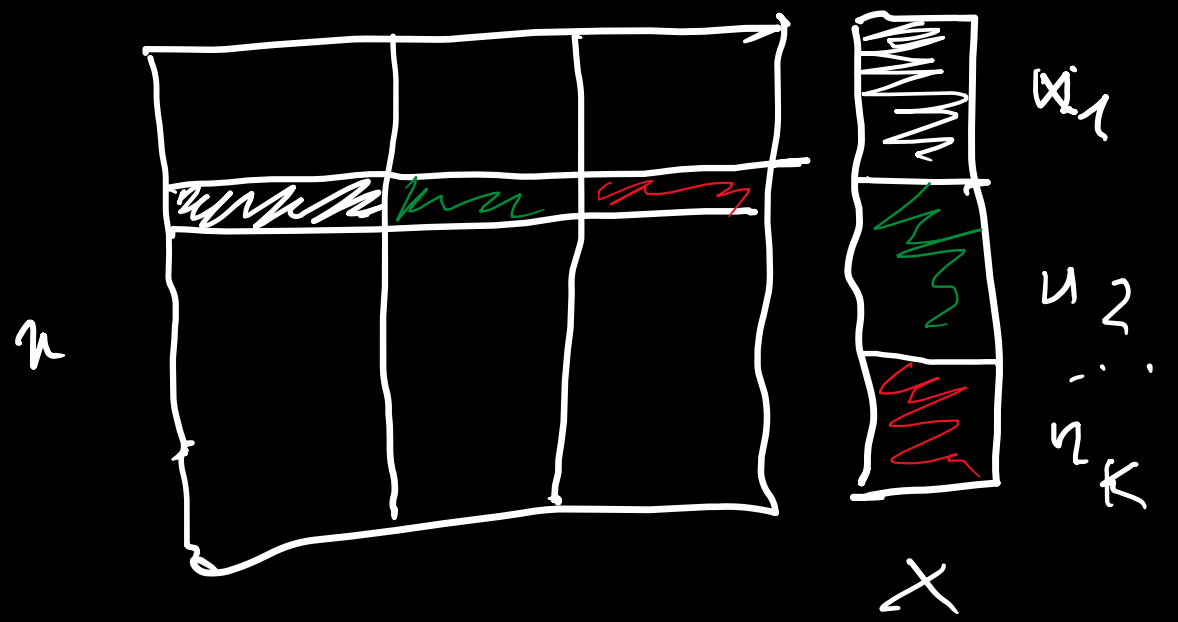
$m_{i,j}$

$$y_i = \sum_j m_{ij} x_j$$

```
reduce (i, L) {
    emit (i, Σ L)
}
```

$$\left[1, [m_{1,3} \cdot x_3, m_{1,5} \cdot x_5, \dots]\right]$$

# Za duży wektor ?




$$n_1 + n_2 + \ldots n_K = n$$

jak to zrobić w M-R ?