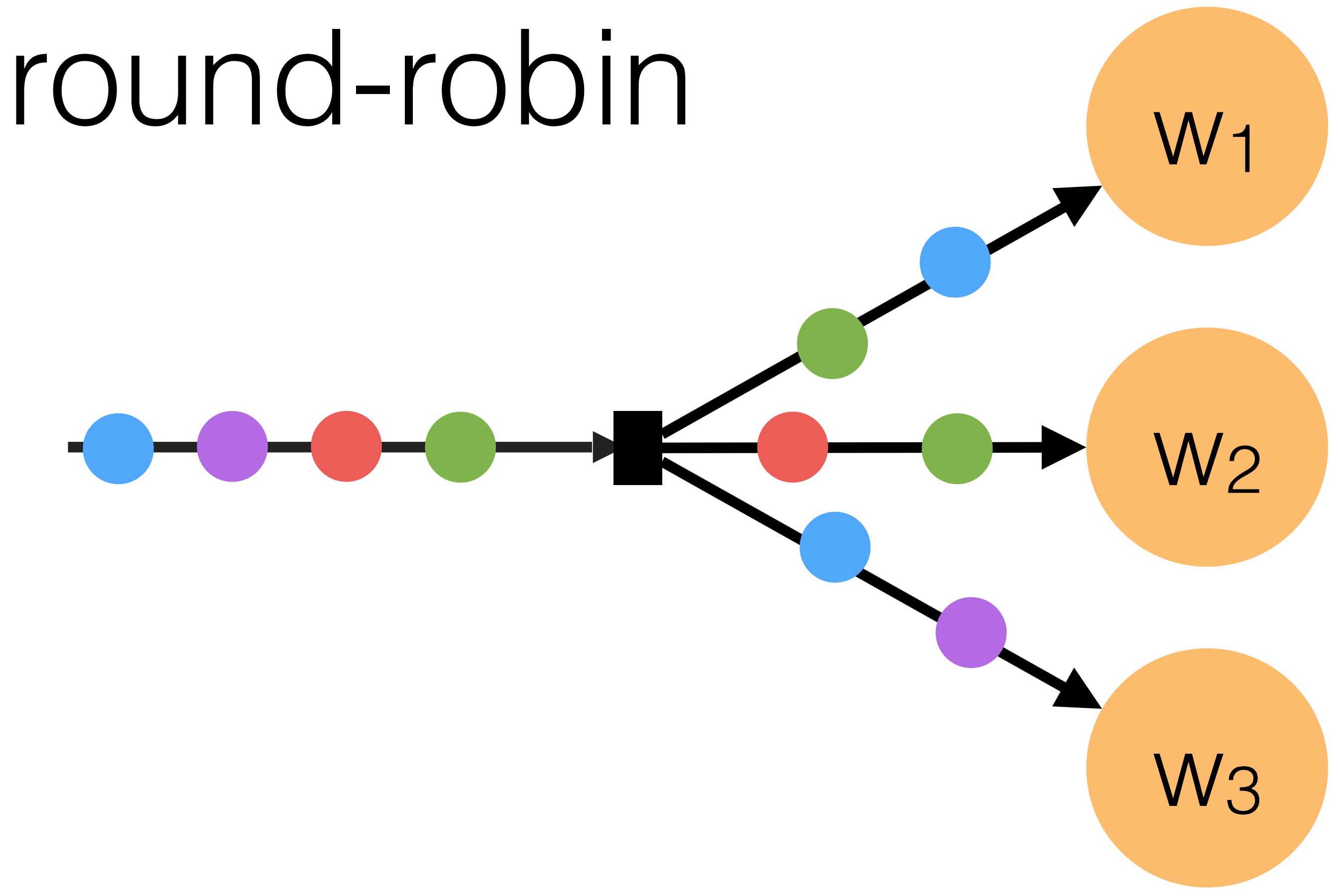
CS 591 K1: Data Stream Processing and Analytics

Spring 2020

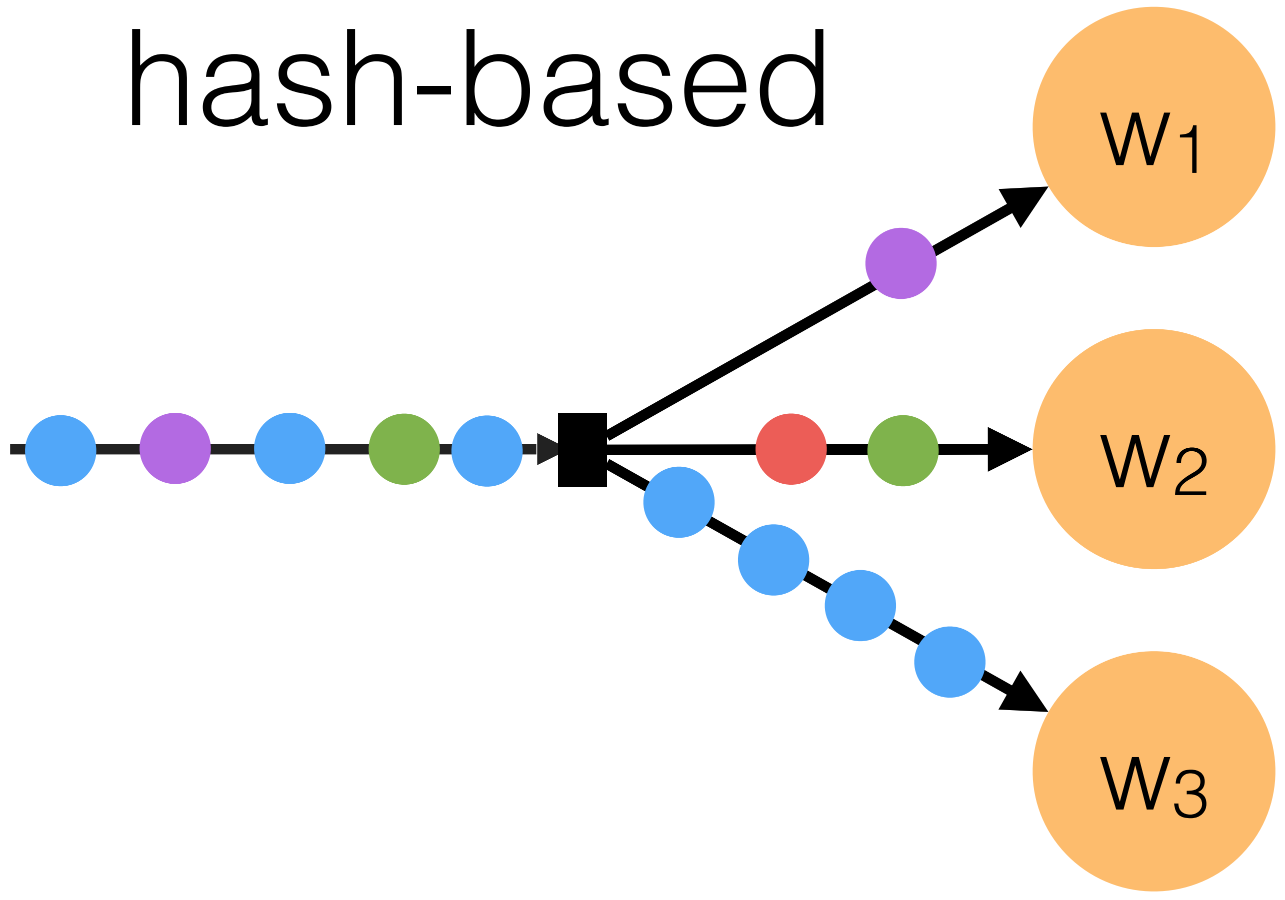
4/16: Skew mitigation

Vasiliki (Vasia) Kalavri
vkalavri@bu.edu

Key partitioning



- Items are perfectly balanced among workers
- No routing table required
- Key semantics are not preserved: values of the same key might be routed to different workers



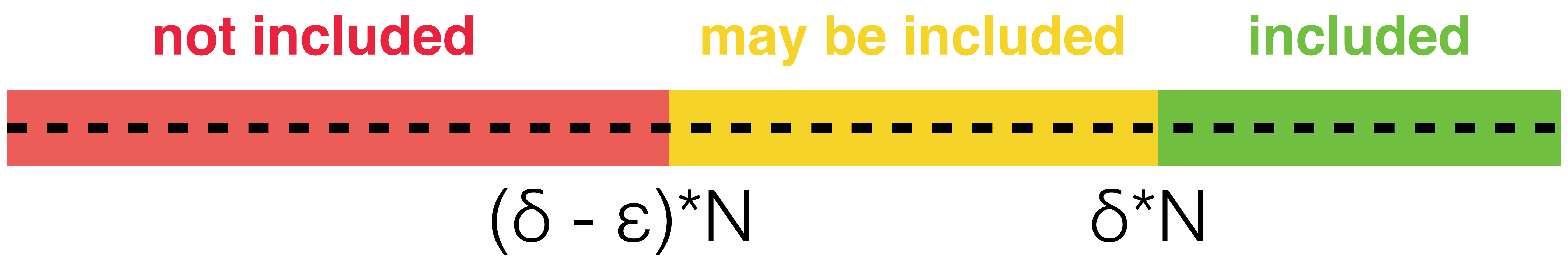
- Workers are responsible for roughly the same amount of keys
- No routing table is required
- Key semantics preserved: values of the same key are always processed by the same worker
- Popular keys cause imbalance

Addressing skew

- To address skew, the system needs to track the frequencies of the partitioning key values.
- We can then use a hybrid partitioning function that treats normal keys and popular keys differently.
- Keeping exact counts is impractical for large key domains, e.g. IP addresses.
    - …but we don’t need the exact frequencies, we only need to track the heavy hitters

Lossy Counting

- Find all items x in a data stream such that:
  - freq(x) > δ*N, where N is the number of stream elements
- The solution will not contain any item y with frequency:
  - freq(y) < (δ - ε)*N, for a user-chosen value ε



Notation (I)

Input: a stream of items

`N`: number of items in the stream

f_e: true frequency of the item e in the input stream

`f`: estimated frequency of item

δ: user-defined threshold, so that `freq(x)≥ δ*N,δ∈(0,1)`

ε: user-defined error

Output: All items with frequency greater than or equal to `δ*N`.

No item with frequency less than `(δ-ε)*N`.

Notation (II)

- We define windows of size `w = 1/ε` with increasing numeric ids, starting from 1.
  - e.g., if **ε**=0.2, `w`=5 (5 items per window)
- $\mathtt{w_{cur}}$: the current window id
- We keep a list `D` of element frequencies and their maximum associated error.
- Once a window fills up, we remove infrequent elements.

Lossy counting algorithm

```
$D = \{\}$ // empty list
$w_{cur} = 1$ // first window id
$N = 0$ // elements seen so far
```

Insert step

```
For each element x in $w_{cur}$:
    if $x \in D$, increase its frequency, $f_x = f_x + 1$
    else insert with frequency $f_x = 1$ and error $\varepsilon_x = w_{cur} - 1$
    $N = N + 1$
```

Delete step

```
Iterate over D and remove every element x with
    $f_x + \varepsilon_x \leq w_{cur}$
```

Output: elements in D with $f_x \geq (\delta - \varepsilon) * N$

Example

input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	**w_2**	**w_3**	**w_4**

Example

input stream *ε=0.2*

1 2 2 3 5

w_1

5 1 1 2 3

w_2

3 3 3 1 2

w_3

0 1 1 3 5

w_4

w_1

1 2 2 3 5

Example

input stream $\varepsilon=0.2$ →

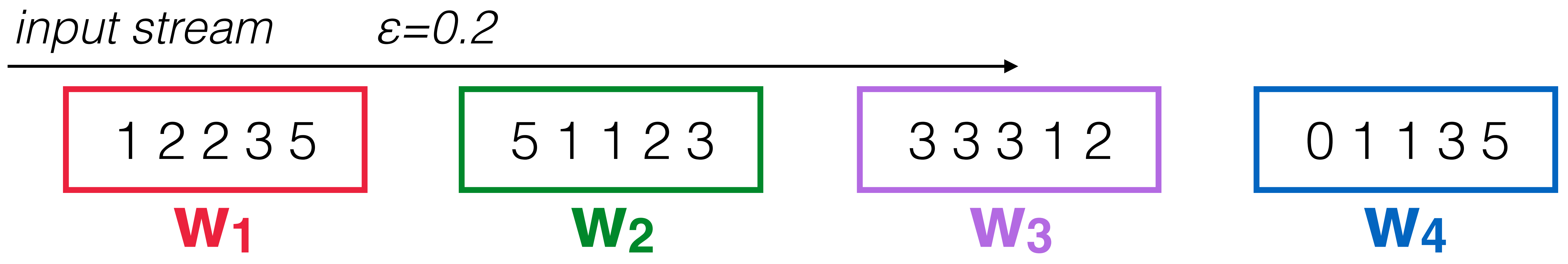
1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	w_2	w_3	w_4

w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
1	0	2	0	1	0	1	0
1		2		3		5	

Example



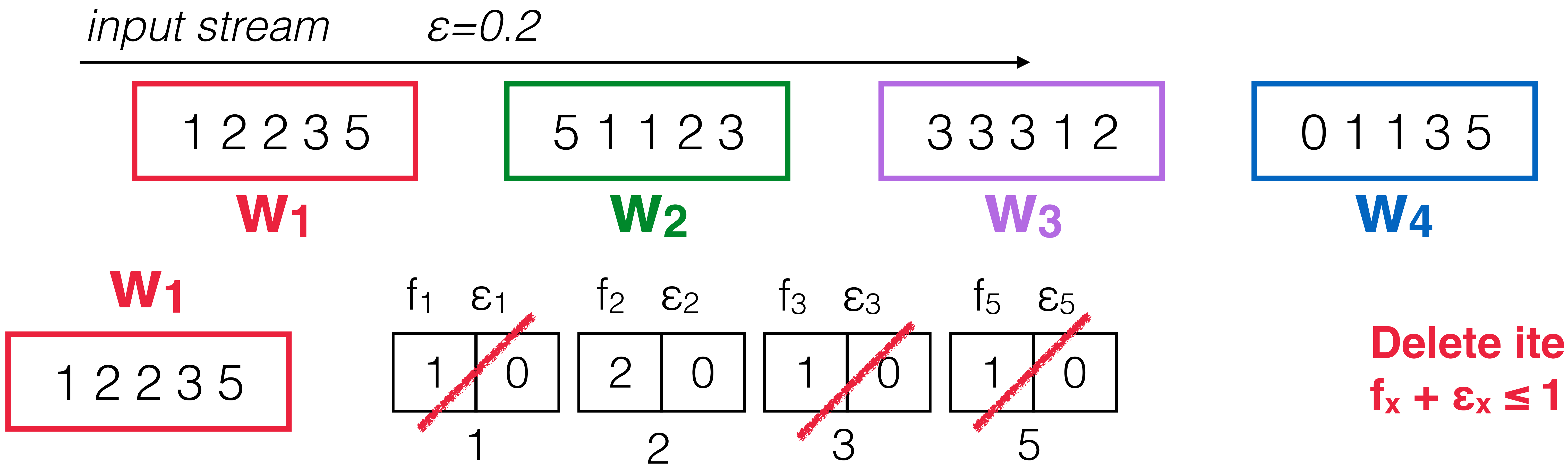
w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
1	0	2	0	1	0	1	0
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

Example



input stream $\varepsilon=0.2$

w_1	**w_2**	**w_3**	**w_4**
1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5

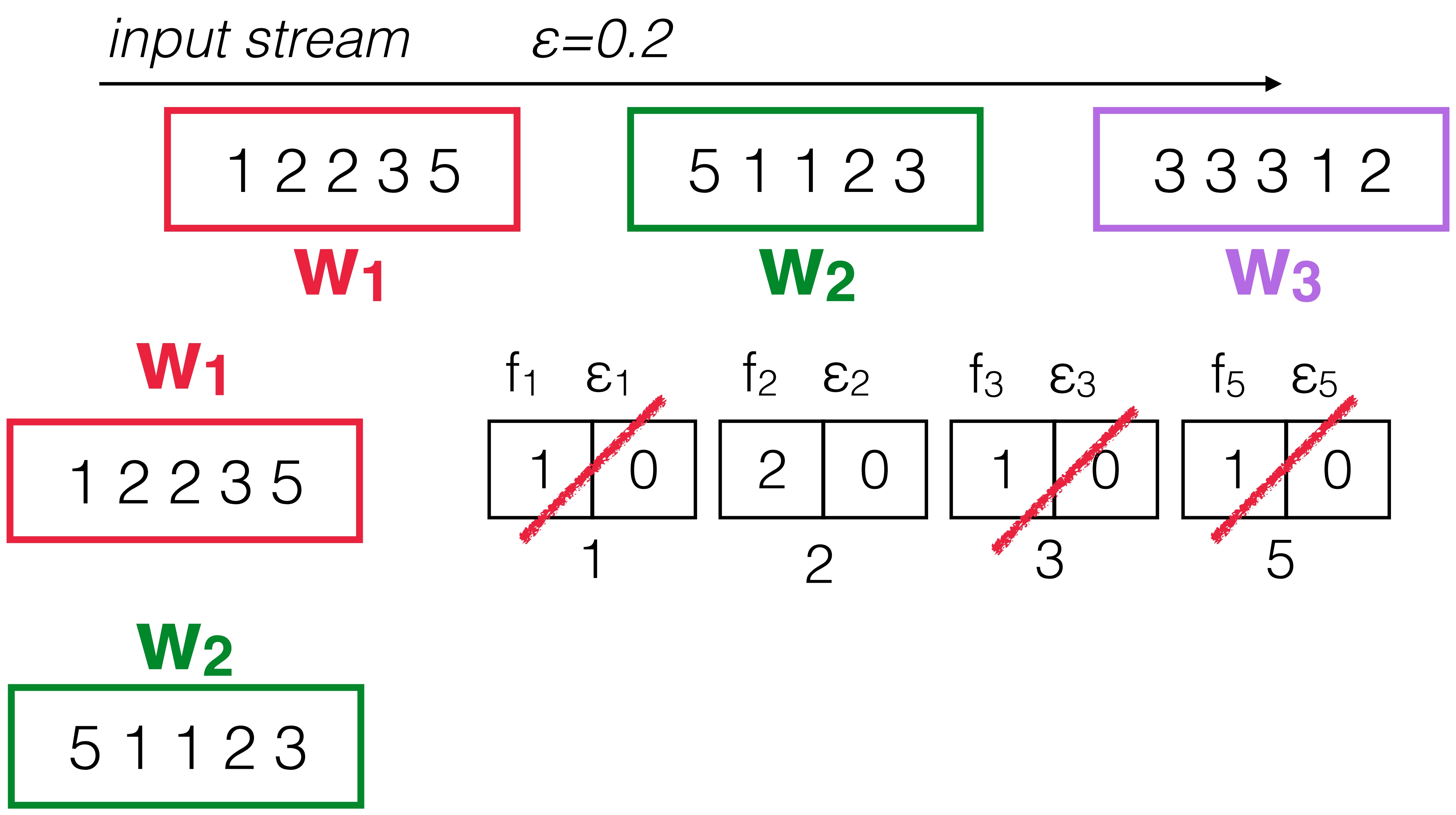
w_1

1 2 2 3 5

	1	2	3	5
f	~~1~~	2	~~1~~	~~1~~
ε	~~0~~	0	~~0~~	~~0~~

Delete items with
$f_x + \varepsilon_x \leq 1$

Example


input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
W1	**W2**	**W3**	**W4**

W1

1 2 2 3 5

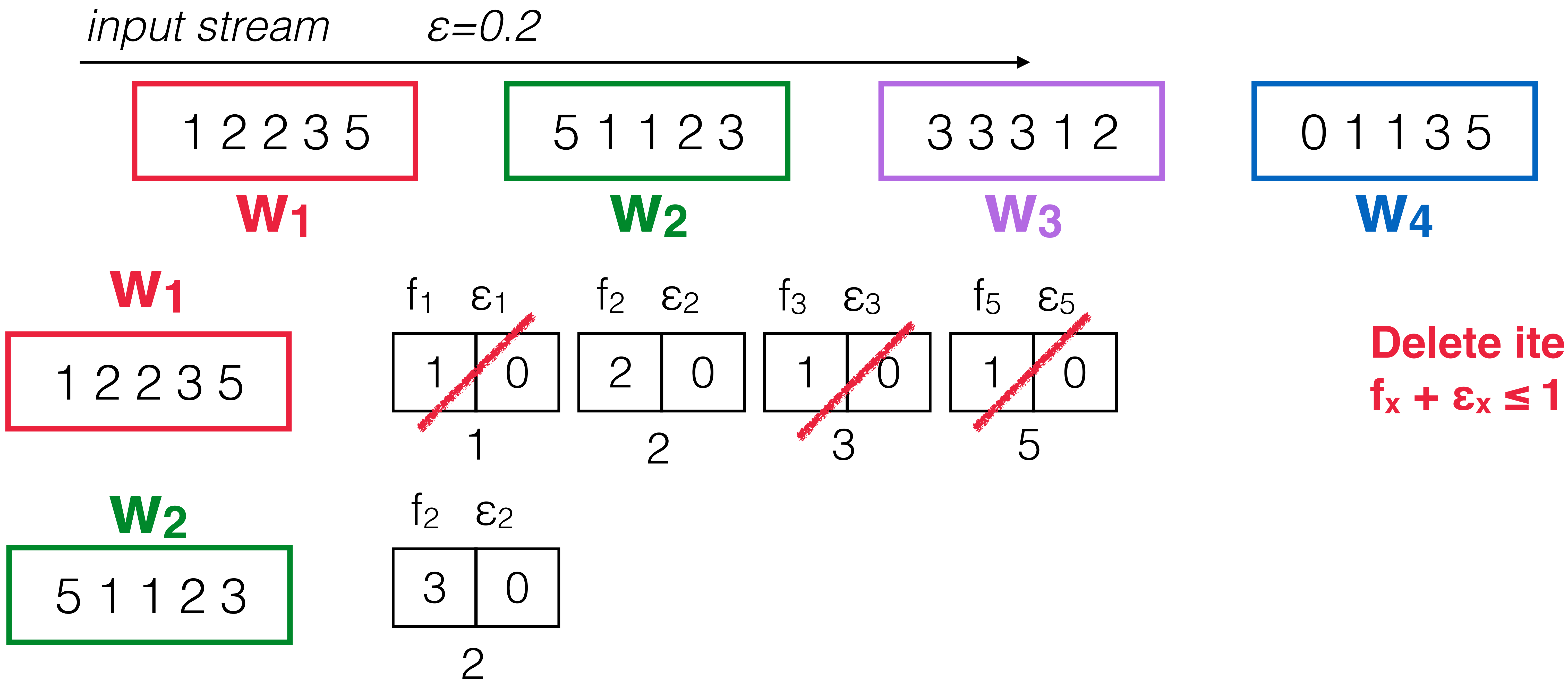
f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

W2

5 1 1 2 3

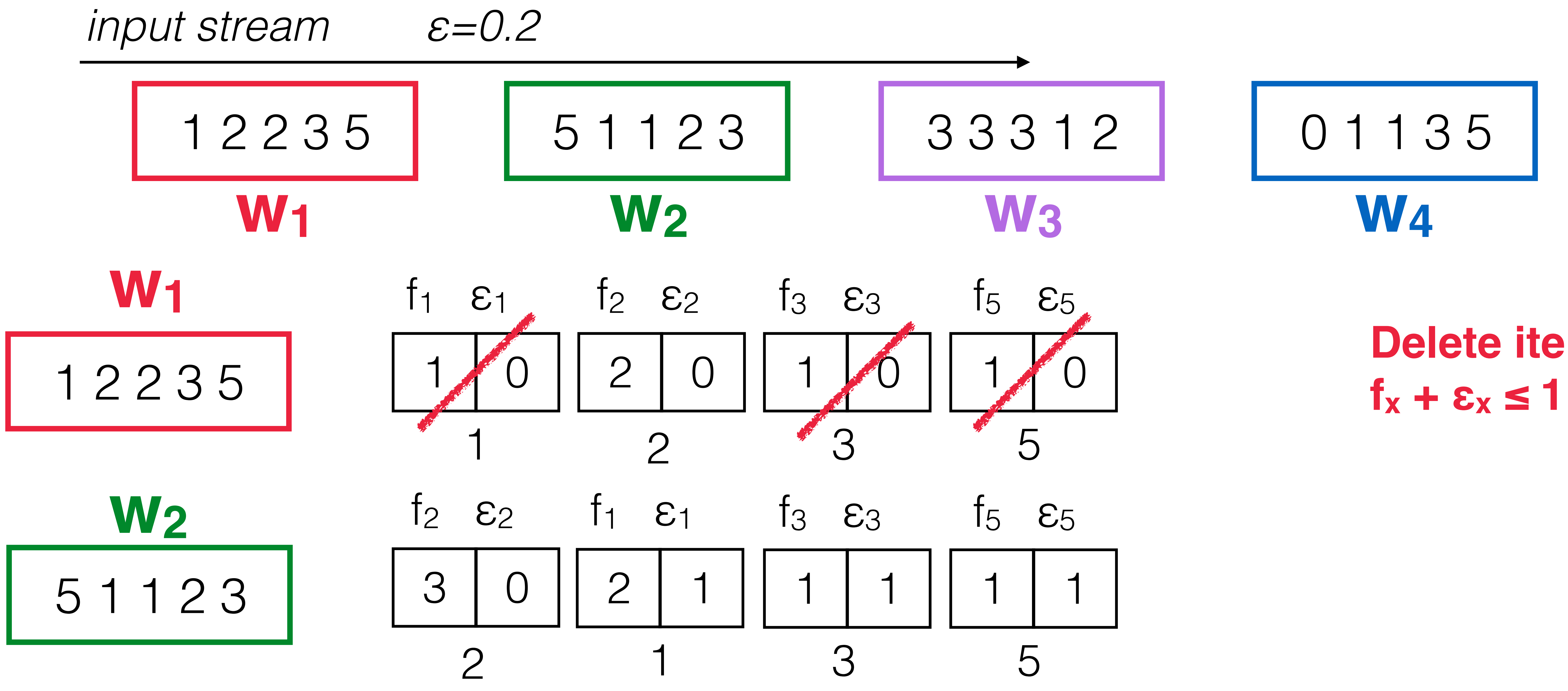
Delete items with $f_x + \varepsilon_x \leq 1$

Example



Delete items with
$f_x + \varepsilon_x \le 1$

Example



input stream *ε=0.2*

w_1	**w_2**	**w_3**	**w_4**
1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5

w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

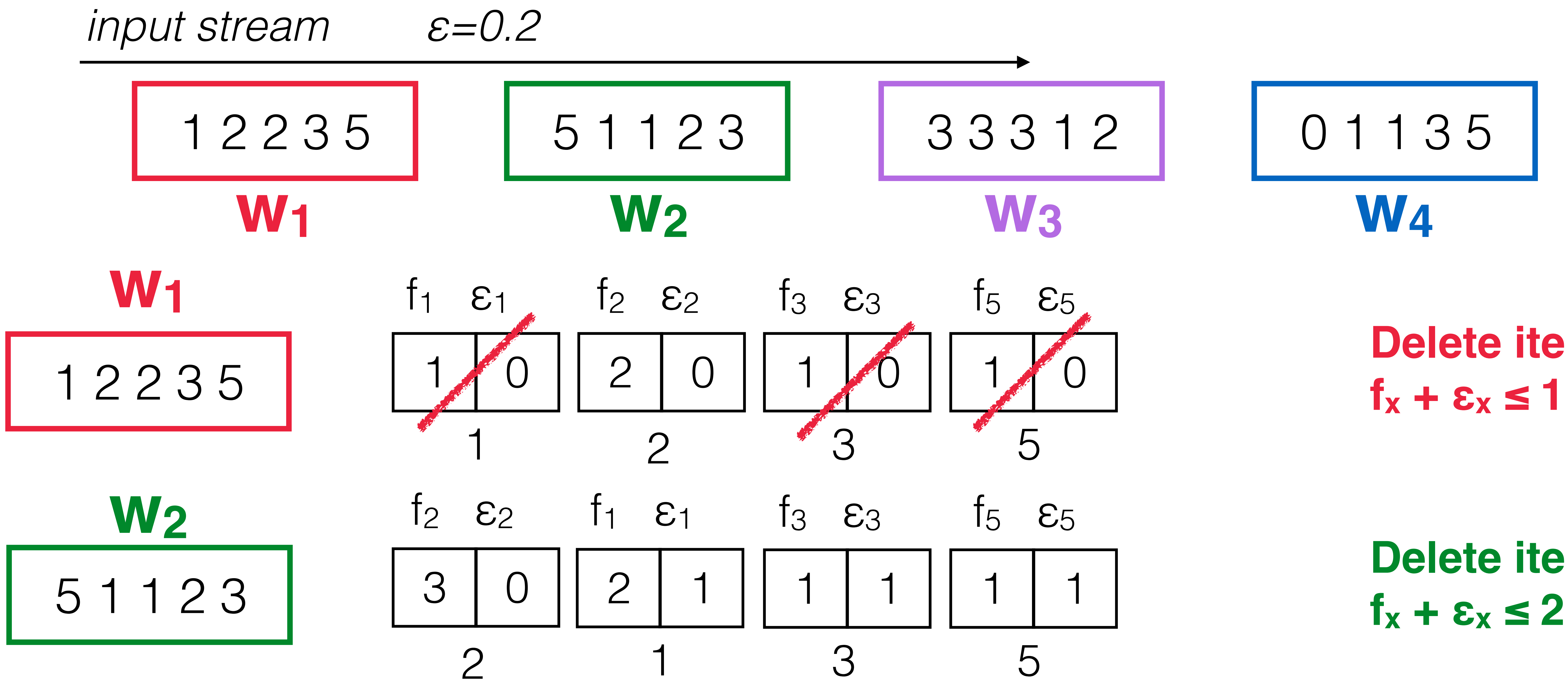
w_2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	1	1	1	1
2		1		3		5	

Delete items with
$f_x + \varepsilon_x \leq 1$

Example



input stream *ε=0.2*

W_1	**W_2**	**W_3**	**W_4**
1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5

W_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

W_2

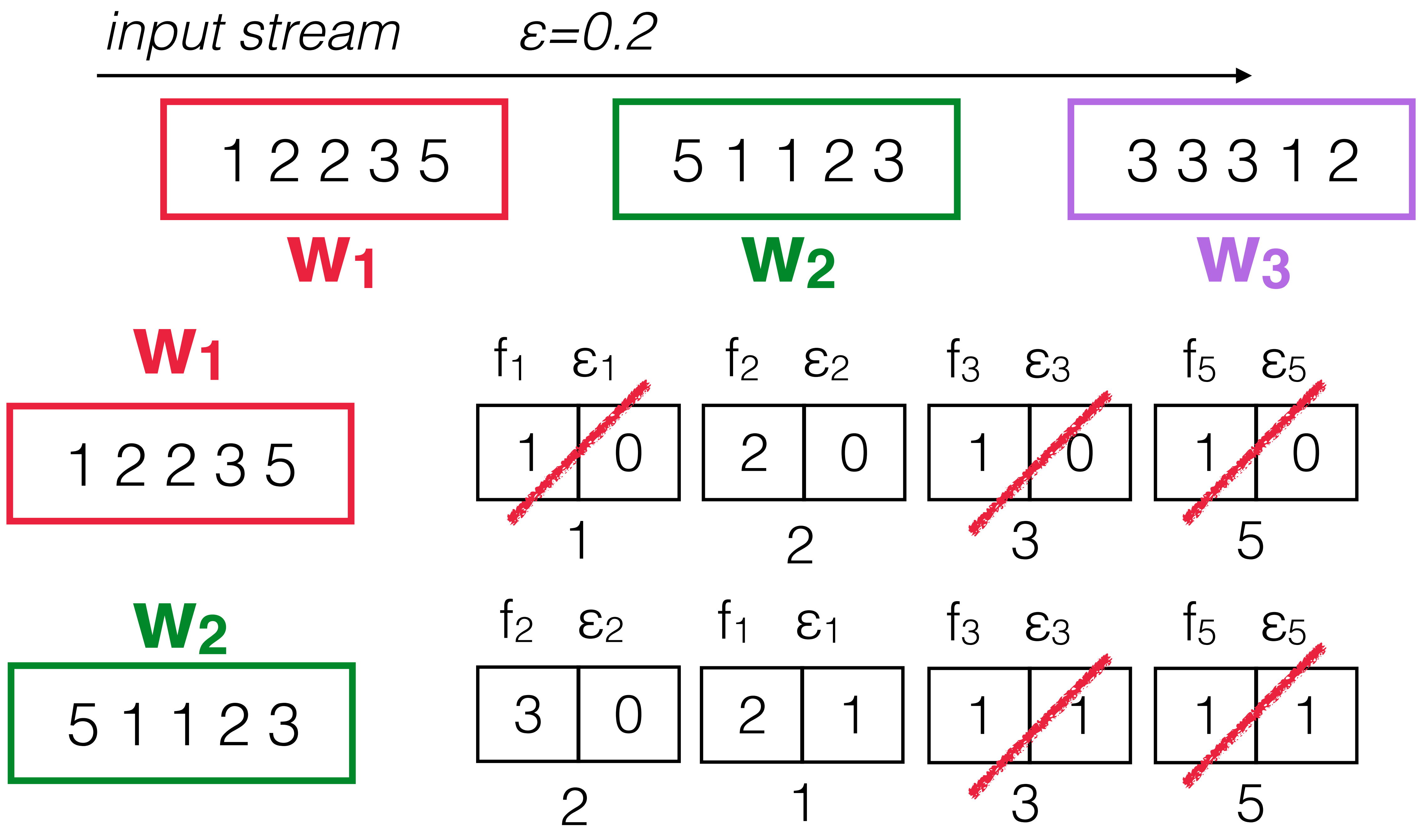
5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	1	1	1	1
2		1		3		5	

Delete items with
$f_x + \varepsilon_x \leq 1$

Delete items with
$f_x + \varepsilon_x \leq 2$

Example



Delete items with
$f_x + \varepsilon_x \leq 1$

Delete items with
$f_x + \varepsilon_x \leq 2$

Example

input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	**w_2**	**w_3**	**w_4**

w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

w_2

5 1 1 2 3

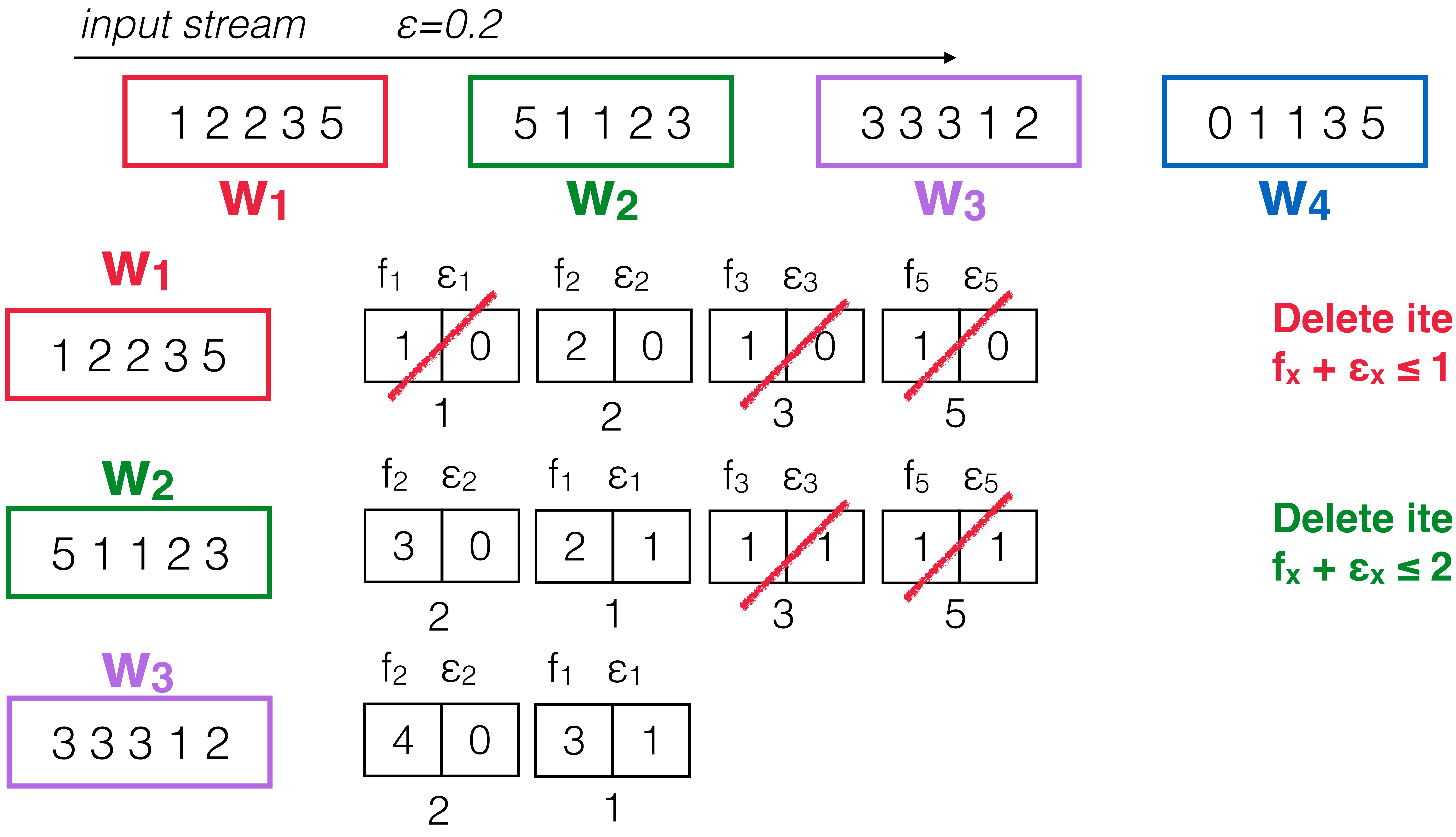
f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

Delete items with $f_x + \varepsilon_x \leq 2$

w_3

3 3 3 1 2

Example



input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	**w_2**	**w_3**	**w_4**

w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

w_2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

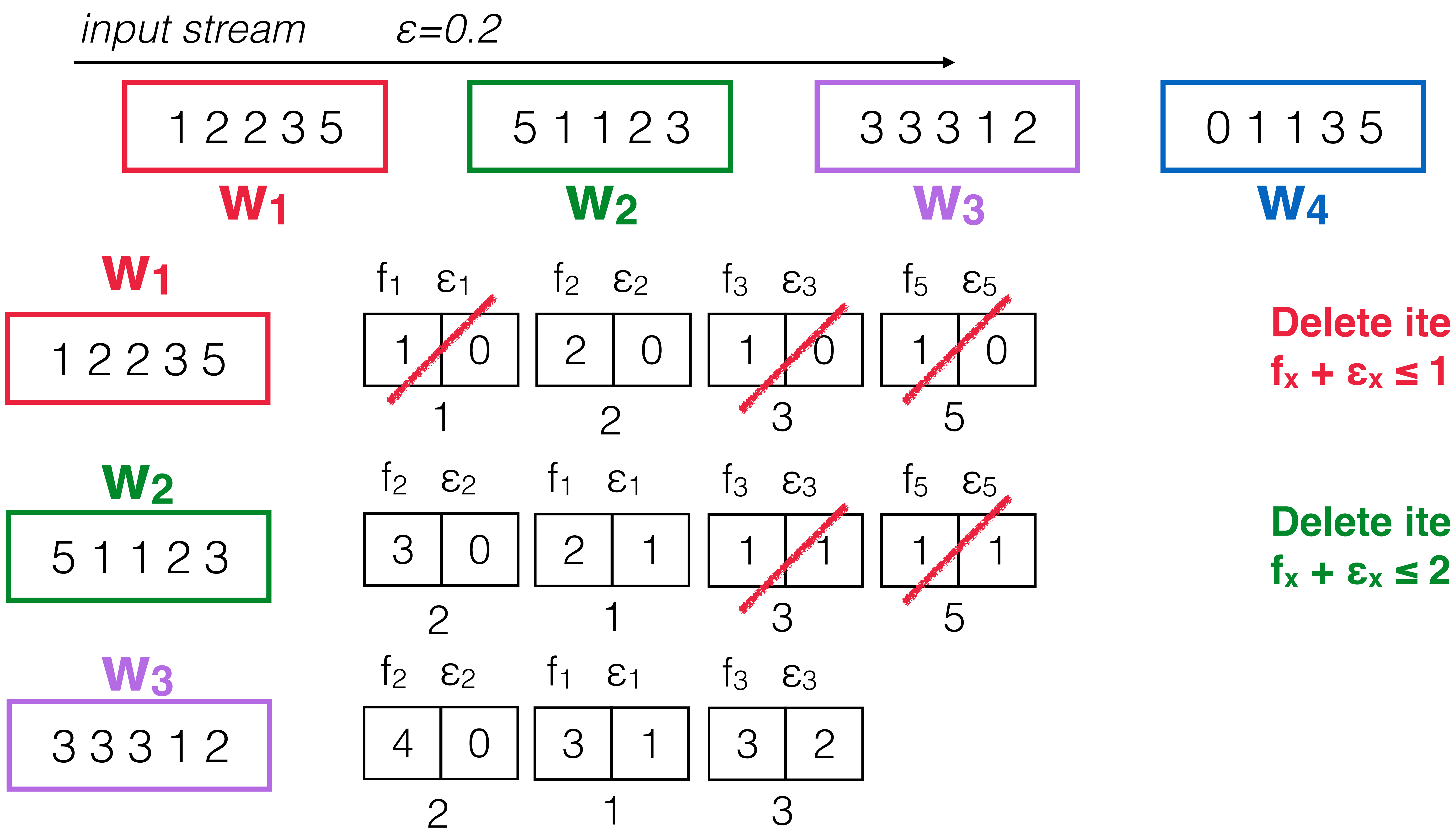
Delete items with $f_x + \varepsilon_x \leq 2$

w_3

3 3 3 1 2

f_2	ε_2	f_1	ε_1
4	0	3	1
2		1	

Example



input stream $\varepsilon=0.2$

W1	**W2**	**W3**	**W4**
1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5

W1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

W2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

W3

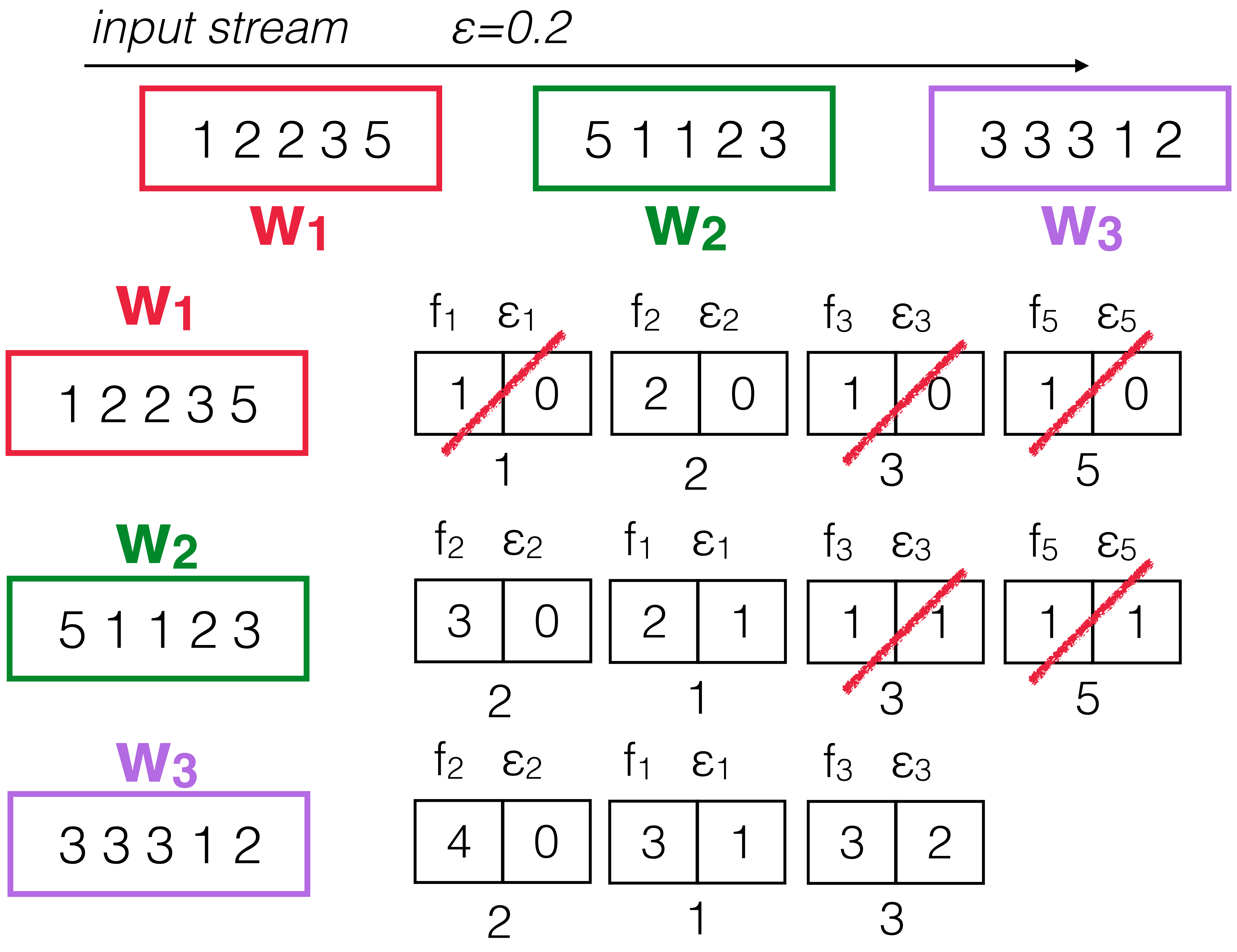
3 3 3 1 2

f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	3	1	3	2
2		1		3	

Delete items with
$f_x + \varepsilon_x \leq 1$

Delete items with
$f_x + \varepsilon_x \leq 2$

Example



Delete items with $f_x + \varepsilon_x \leq 1$

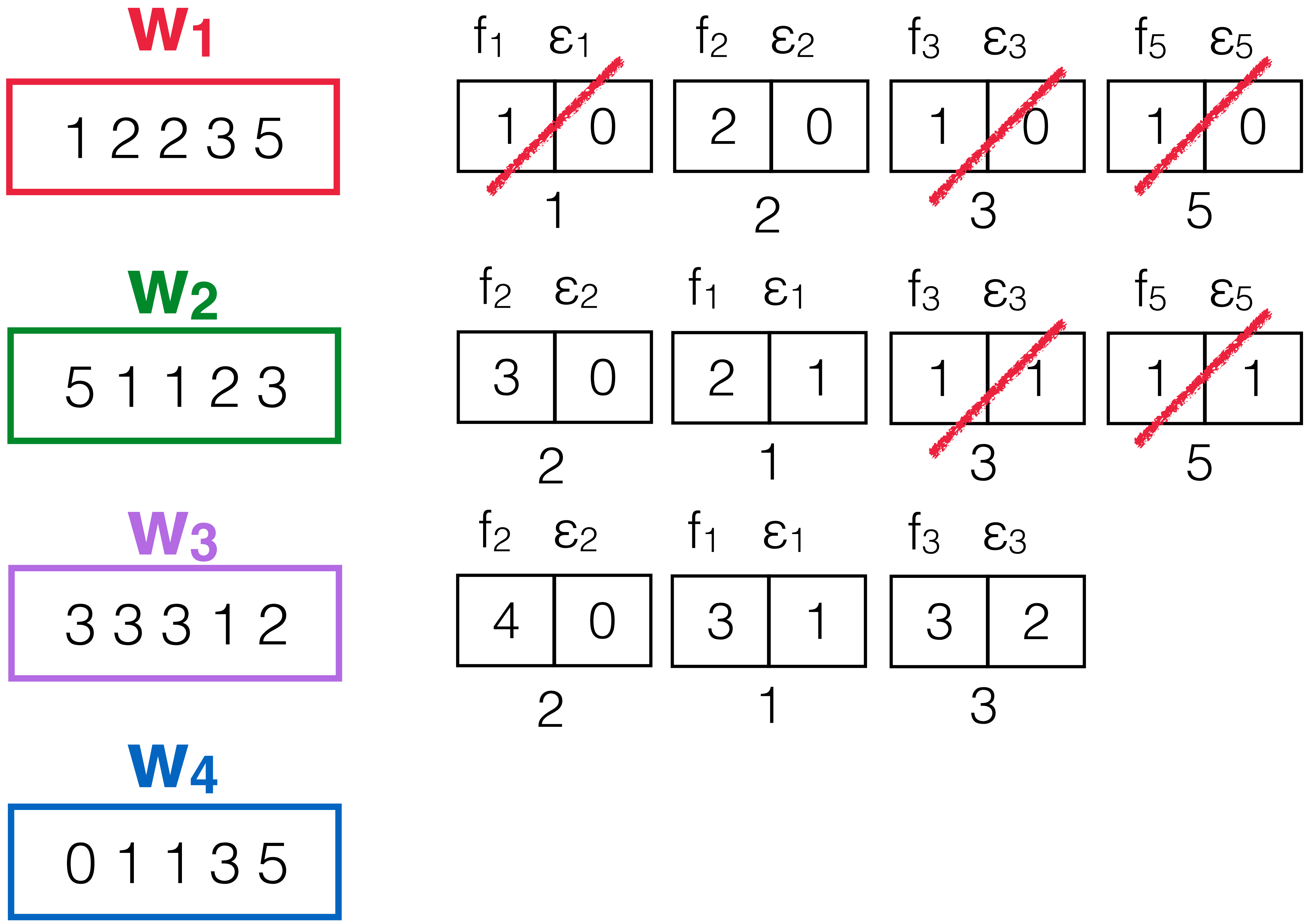
Delete items with $f_x + \varepsilon_x \leq 2$

Delete items with $f_x + \varepsilon_x \leq 3$

Example

input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	**w_2**	**w_3**	**w_4**



w_1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

w_2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

Delete items with $f_x + \varepsilon_x \leq 2$

w_3

3 3 3 1 2

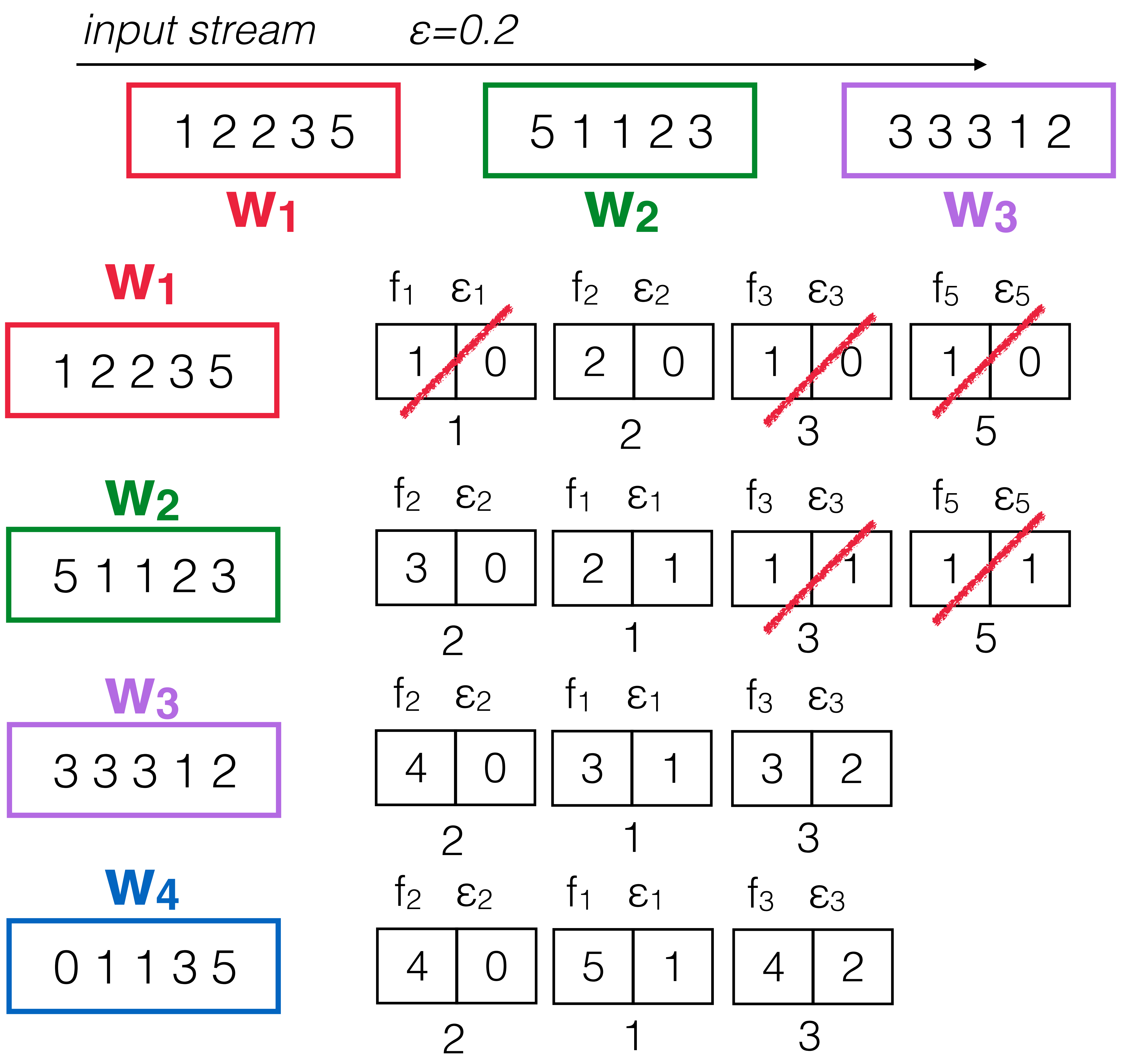
f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	3	1	3	2
2		1		3	

Delete items with $f_x + \varepsilon_x \leq 3$

w_4

0 1 1 3 5

Example



input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w1	**w2**	**w3**	**w4**

w1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

w2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

Delete items with $f_x + \varepsilon_x \leq 2$

w3

3 3 3 1 2

f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	3	1	3	2
2		1		3	

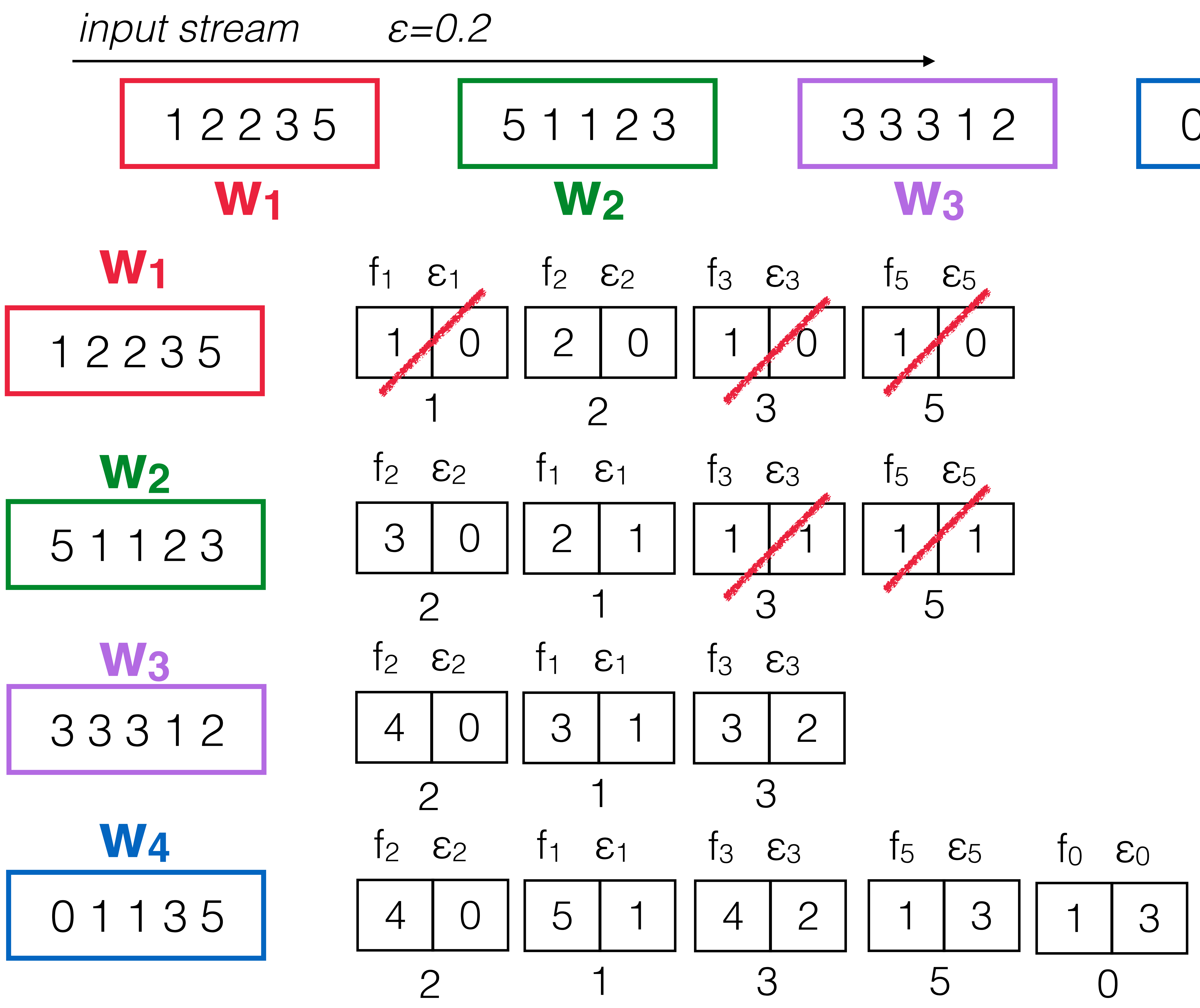
Delete items with $f_x + \varepsilon_x \leq 3$

w4

0 1 1 3 5

f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	5	1	4	2
2		1		3	

Example



input stream *ε=0.2*

1 2 2 3 5	5 1 1 2 3	3 3 3 1 2	0 1 1 3 5
w_1	**w_2**	**w_3**	**w_4**

w_1 1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with $f_x + \varepsilon_x \leq 1$

w_2 5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

Delete items with $f_x + \varepsilon_x \leq 2$

w_3 3 3 3 1 2

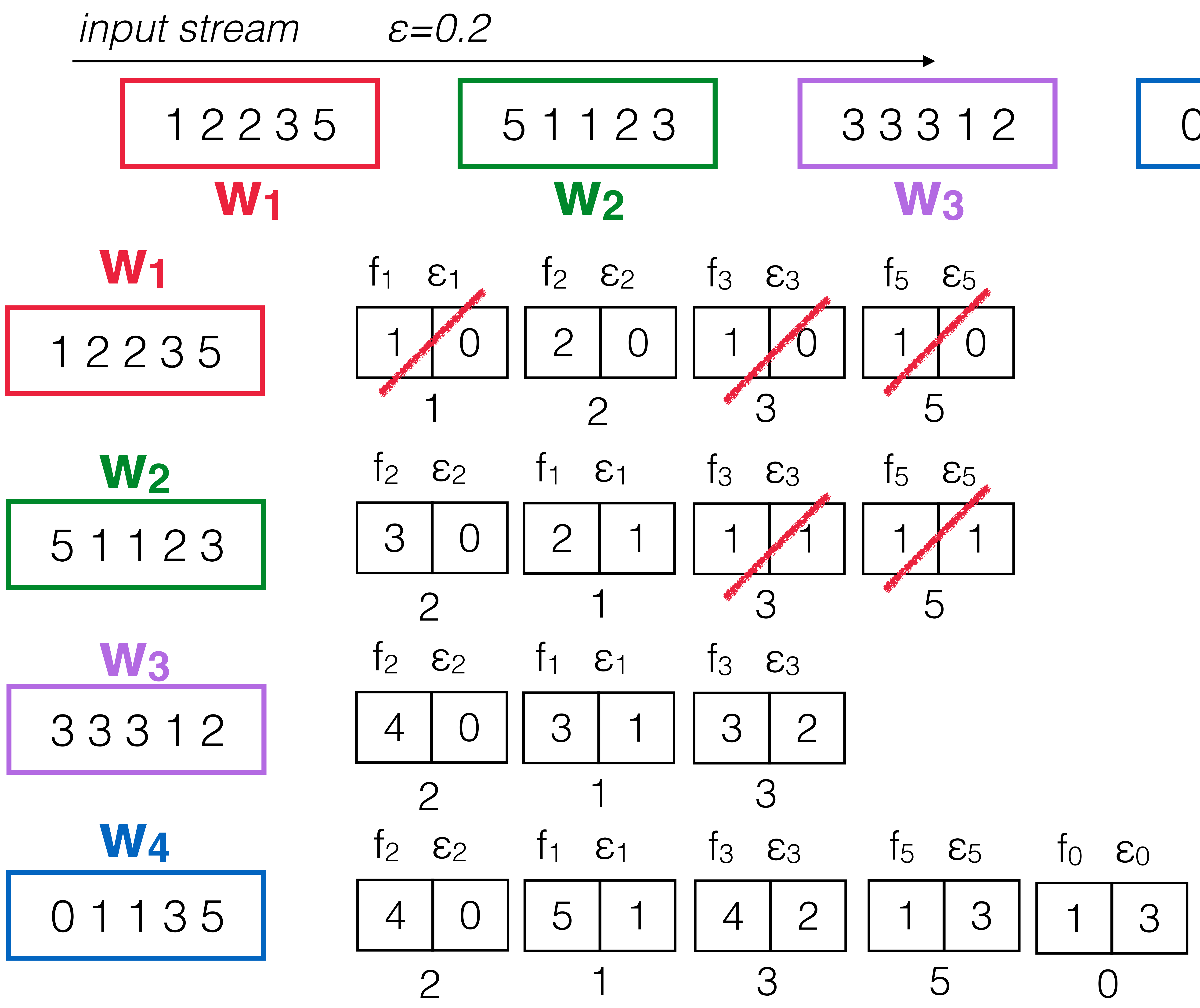
f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	3	1	3	2
2		1		3	

Delete items with $f_x + \varepsilon_x \leq 3$

w_4 0 1 1 3 5

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5	f_0	ε_0
4	0	5	1	4	2	1	3	1	3
2		1		3		5		0	

Example



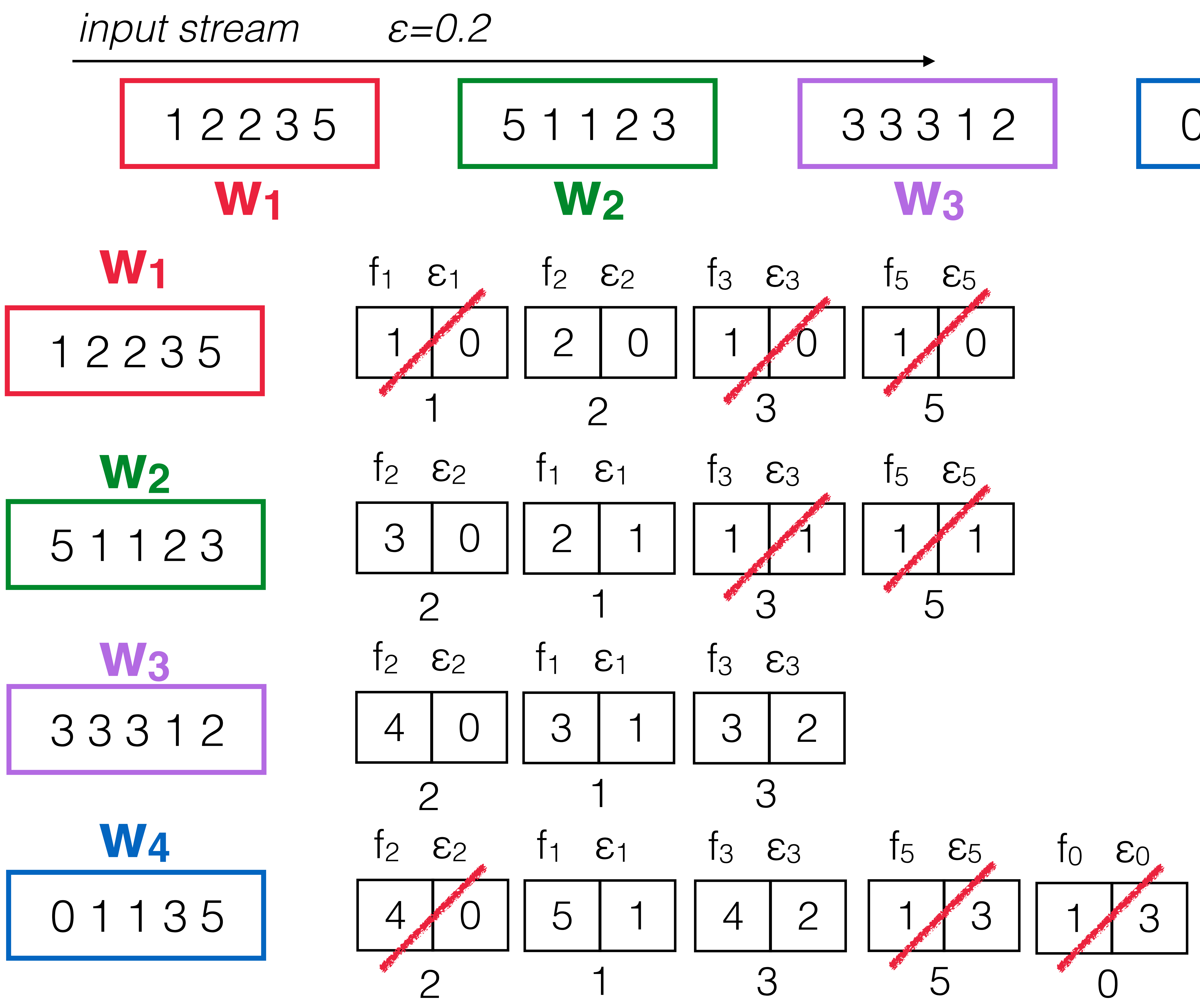
Delete items with $f_x + ε_x ≤ 1$

Delete items with $f_x + ε_x ≤ 2$

Delete items with $f_x + ε_x ≤ 3$

Delete items with $f_x + ε_x ≤ 4$

Example



input stream ε=0.2

W1: 1 2 2 3 5

W2: 5 1 1 2 3

W3: 3 3 3 1 2

W4: 0 1 1 3 5

W1

1 2 2 3 5

f_1	ε_1	f_2	ε_2	f_3	ε_3	f_5	ε_5
~~1~~	~~0~~	2	0	~~1~~	~~0~~	~~1~~	~~0~~
1		2		3		5	

Delete items with
$f_x + \varepsilon_x \le 1$

W2

5 1 1 2 3

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5
3	0	2	1	~~1~~	~~1~~	~~1~~	~~1~~
2		1		3		5	

Delete items with
$f_x + \varepsilon_x \le 2$

W3

3 3 3 1 2

f_2	ε_2	f_1	ε_1	f_3	ε_3
4	0	3	1	3	2
2		1		3	

Delete items with
$f_x + \varepsilon_x \le 3$

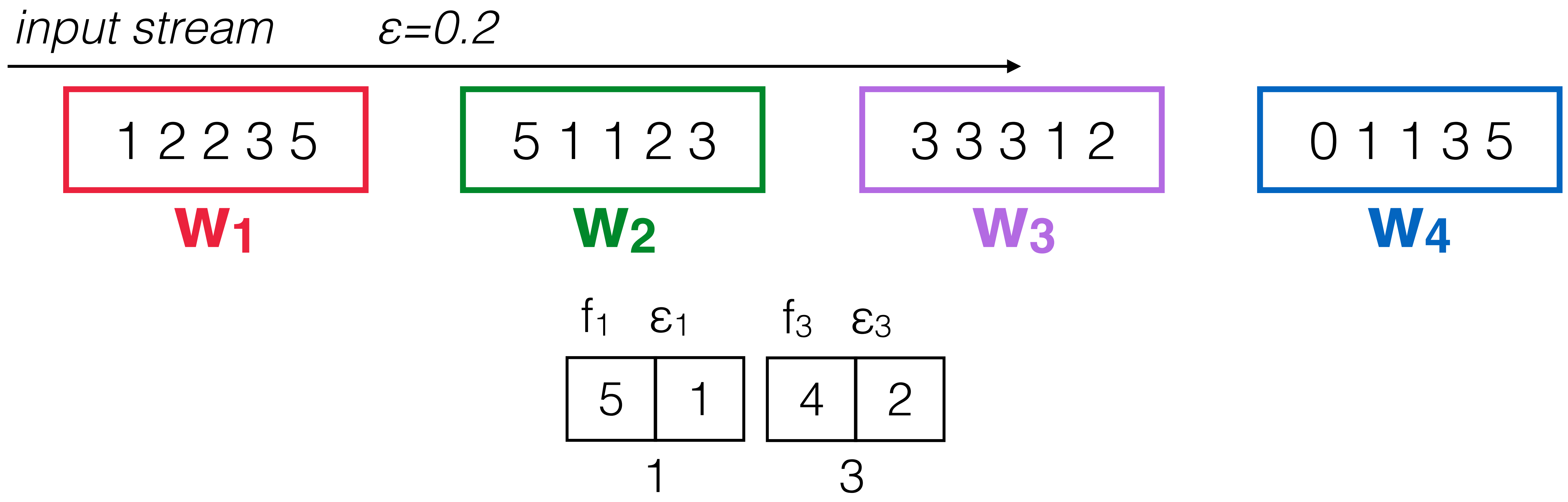
W4

0 1 1 3 5

f_2	ε_2	f_1	ε_1	f_3	ε_3	f_5	ε_5	f_0	ε_0
~~4~~	~~0~~	5	1	4	2	~~1~~	~~3~~	~~1~~	~~3~~
2		1		3		5		0	

Delete items with
$f_x + \varepsilon_x \le 4$

Example


input stream $\varepsilon=0.2$

1 2 2 3 5 — **w_1**

5 1 1 2 3 — **w_2**

3 3 3 1 2 — **w_3**

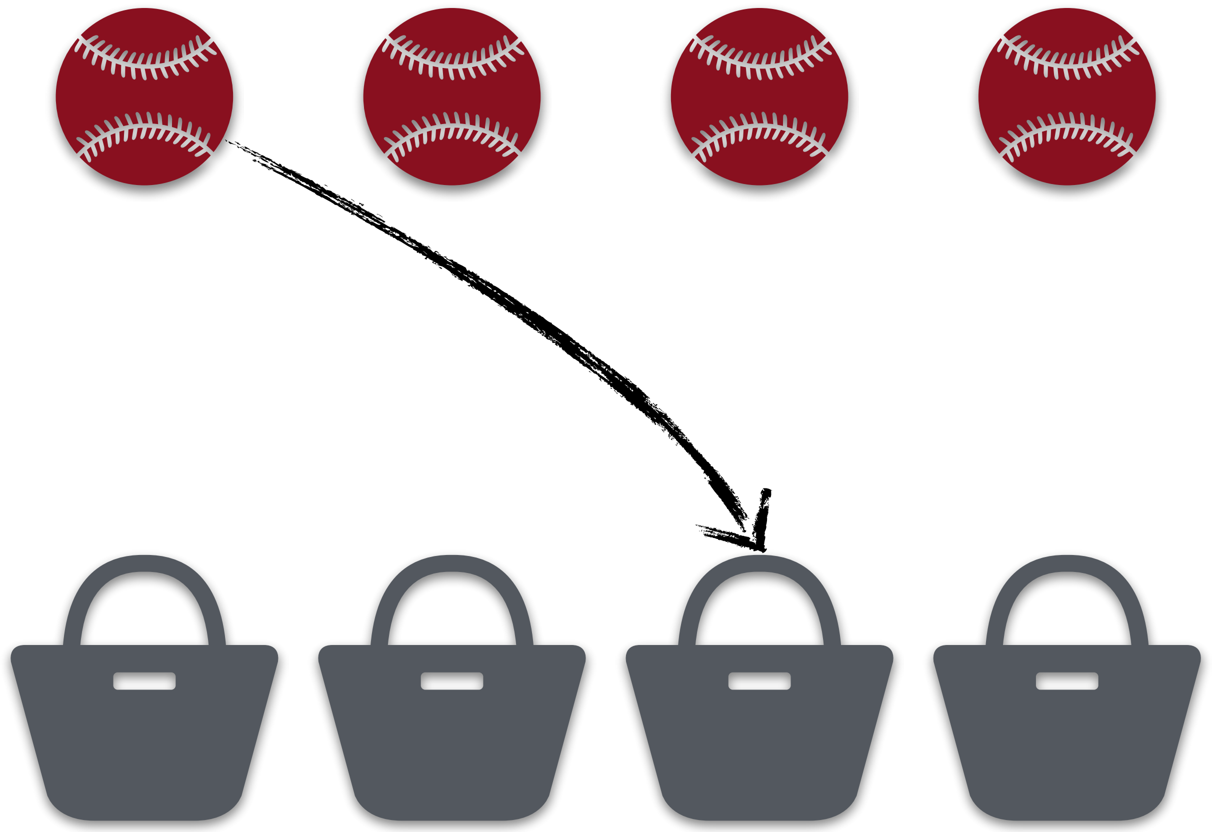
0 1 1 3 5 — **w_4**

f_1	ε_1	f_3	ε_3
5	1	4	2
1		3	

When the algorithm terminates
`D` contains an item **`x`**
if its actual frequency is $f_x > \varepsilon * N$

Worst case: $O(\frac{1}{\varepsilon} * log(\varepsilon N))$ counters

The power of two choices



- Consider the problem of throwing n balls to n bins sequentially (balls -> records, bins -> workers)
- Bins are selected uniformly at random
- At the end of the process, the maximum load is $\Theta(\ln n / \ln \ln n)$, with high probability

- Instead, we select d destination bins, each uniformly at random, and place the ball at the least full bin:
  - when d=2, the maximum load is $\ln \ln n \, / \ln 2 + O(1)$, with high probability
  - when d>2, the maximum load keeps decreasing, but only by a constant factor

Dynamic resource allocation

- Choose one among n workers
  - check the load of each worker and send the item to the least loaded one
  - load checking for every item can be expensive
- Choose *two* workers at random and send the item to the least loaded of those two
  - the system uses two hash functions, H_1 and H_2 and checks the load of the two sampled workers: $P(k) = \arg\min_i (L_i(t): H_1(k)=i \lor H_2(k)=i)$
  - provably reduces load variation exponentially as compared to the single choice
- How can we preserve the key semantics?

The power of *both* choices

- Applying the power of two choices in a streaming setting and preserving key semantics would require remembering the choices made previously
- **Partial key grouping** maps each key to both choices: the partitioner sends the item to the worker with the currently lowest load
    - no routing history required
    - state needs to be merged to produce the final result: the computation must consist of combinable functions
    - workers need to be able to compute their current load locally

Further reading

- Muhammad Anis Uddin Nasir et. al. **The power of both choices: Practical load balancing for distributed stream processing engines.** ICDE 2015.
- Mitzenmacher, Michael. **The power of two choices in randomized load balancing**. IEEE TPDS 2001.
- Manku, G.S., Motwani, R. **Approximate frequency counts over data streams**. VLDB 2002.