**Algorithm 1:** Space Saving Algorithm

1 Table *T* has *m* slots, either containing *(keyj, valj)*

at slot *j* ∈{1,…,*m*}, or empty. Incoming packet

has key *iKey*

2 **if** ∃ *slot j in T with iKey = keyj* **then**

3 *valj* ← *valj* + 1

4 **else**

5 **if** ∃ *empty slot j in T* **then**

6 (*keyj, valj*) ← (*iKey*, 1)

7 **else**

8 *r ← argminj ∈ {1,…,m}*(*valj)*

9 (*keyr, valr)* ← *(iKey, valr + 1)*

10 **end**

11 **end**

**Algorithm 2:** HashPipe: Pipeline of *d* hash tables [6]

1 > Insert in the first stage

2 *l1* ← *h1(iKey)*

3 **if** *keyl1* = *iKey* **then**

4 *vall1 ← vall1* + 1

5 end processing

6 **end**

7  **else if** *l1 is an empty slot* ***then***

8 (*keyl1, vall1)* ← *(iKey,1)*

9 end processing

10 **end**

11 **else**

12 (c*Key, cValj*) ← (*keyl1*, *vall1*)

13 (*keyl1, vall1*) ← (*iKey*, 1)

14 **end**

15 > Track a rolling minimum

16 **for** *i ← 2* ***to*** *d* ***do***

17 *l ← hi(cKey)*

18 **if** *keyl* = *cKey* ***then***

19 *val1 ← vall + cVal*

20 end processing

21 **end**

22 **else if** *l is an empty slot* **then**

23 (*keyl, vall*) ← (*cKey*, *cVal)*

24 end processing

25 **end**

26 **else if** *vall* < *cVal* **then**

27 swap *(cKey,cVal)* with *(keyl,vall)*

28 **end**

29 **end**

**Algorithm 3:** HashPipe Bouncer Admission

1 > Randomly position in first stage

2 *l1* ← *rand1(iKey)*

3 **if** *keyl1* = *iKey* **then**

4 *vall1 ← vall1* + 1

5 end processing

6 **end**

7  **else if** *l1 is an empty slot* ***then***

8 (*keyl1, vall1)* ← *(iKey,1)*

9 end processing

10 **end**

11 **else**

12 > Randomized Admission Policy

13 *m ← vall1*

14 **if** *random()* < **then**

15 (c*Key, cValj*) ← (*keyl1*, *vall1*)

16 (*keyl1, vall1*) ← (*iKey*, 1)

17 **end**

18 **else**

19 end processing

20 **end**

21 > Track a rolling minimum

22 **for** *i ← 2* ***to*** *d* ***do***

23 *l ← hi(cKey)*

24 **if** *keyl* = *cKey* ***then***

25 *val1 ← vall + cVal*

26 end processing

27 **end**

28 **else if** *l is an empty slot* **then**

29 (*keyl, vall*) ← (*cKey*, *cVal)*

30 end processing

31 **end**

32 **else if** *vall* < *cVal* **then**

33 swap *(cKey,cVal)* with *(keyl,vall)*

34 **end**

35 **end**



