

PCY Example

The **PCY algorithm** is an improvement over Apriori, particularly in its handling of the first pass over the dataset. Key steps are as follows:

Pass 1: Counting Items and Hashing Pairs

- For each transaction, count the occurrences of individual items.
- Hash each pair of items in the transaction and increment the corresponding bucket count in the hash table.

Determine Frequent Items and Buckets

- After the first pass, identify frequent items based on the support threshold.
- Identify frequent buckets in the hash table.

Pass 2: Counting Candidate Pairs

- Generate candidate pairs only from frequent items that hash to frequent buckets.
- Count the occurrences of these candidate pairs in the dataset.

Prune and Generate Frequent Pairs

- Eliminate infrequent candidate pairs based on the support threshold to get the frequent pairs.

Transaction Dataset:

- T1: {A, B, C}
- T2: {A, C, D}
- T3: {B, C, D}
- T4: {A, D}

Support Threshold: 3

Assume a simple hash function $H(x, y) = (x \times y) \bmod 4$, where x and y are item IDs.
For simplicity, let's map: A=1, B=2, C=3, D=4.

Pass 1: Counting Items and Hashing Pairs

Step 1: Count Item Frequencies

- $A = 3, B = 2, C = 3, D = 3$

Frequent items (support threshold = 3):

- $\{A, C, D\}$

Step 2: Hash Each Pair into Buckets

Hash function:

$$H(x, y) = (x \times y) \bmod 4$$

Hashing pairs (only pairs of frequent items):

1. $T1 = \{A, B, C\}$: Pair (A, C)
 - $H(A, C) = (1 \times 3) \bmod 4 = 3 \rightarrow \text{Bucket 3}$
2. $T2 = \{A, C, D\}$: Pairs $(A, C), (A, D), (C, D)$
 - $H(A, C) = (1 \times 3) \bmod 4 = 3 \rightarrow \text{Bucket 3}$
 - $H(A, D) = (1 \times 4) \bmod 4 = 0 \rightarrow \text{Bucket 0}$
 - $H(C, D) = (3 \times 4) \bmod 4 = 0 \rightarrow \text{Bucket 0}$
3. $T3 = \{B, C, D\}$: Pair (C, D)
 - $H(C, D) = (3 \times 4) \bmod 4 = 0 \rightarrow \text{Bucket 0}$
4. $T4 = \{A, D\}$: Pair (A, D)
 - $H(A, D) = (1 \times 4) \bmod 4 = 0 \rightarrow \text{Bucket 0}$

Determine Frequent Items and Buckets

Step 3: Bucket Counts

- Bucket 0: 4
- Bucket 1: 0
- Bucket 2: 0
- Bucket 3: 2

Step 4: Identify Frequent Buckets

Buckets with counts ≥ 3 (support threshold):

- Frequent buckets: $\{0\}$

Pass 2: Counting Candidate Pairs

Step 1: Generate Candidate Pairs

From frequent items $\{A, C, D\}$:

$$\{(A, C), (A, D), (C, D)\}$$

Pairs must hash into frequent buckets $\{0\}$:

1. $(A, C): H(A, C) = 3 \rightarrow \text{Bucket 3 (not frequent)} \rightarrow \text{Pruned}$
2. $(A, D): H(A, D) = 0 \rightarrow \text{Bucket 0 (frequent)}$
3. $(C, D): H(C, D) = 0 \rightarrow \text{Bucket 0 (frequent)}$

Only (A, D) and (C, D) remain as candidates.

Step 2: Count Remaining Candidate Pairs in Transactions

Count how many times the remaining candidate pairs appear in the transactions:

- (A, D) : Appears in $T2, T4 \rightarrow \text{Count} = 2$
- (C, D) : Appears in $T2, T3 \rightarrow \text{Count} = 2$

Prune and Generate Frequent Pairs

Step 3: Filter Frequent Pairs

Apply the support threshold = 3:

- Frequent pairs: None (both (A, D) and (C, D) have counts below the threshold).

FINAL RESULTS

- Frequent items: $\{A, C, D\}$
- Frequent Pairs: None

This demonstrates how the PCY algorithm avoids unnecessary counting by pruning candidate pairs early:

- The pair (A, C) hashed into Bucket 3, which is not frequent. Therefore, (A, C) was pruned and never counted in Pass 2.
- Only pairs that hashed into frequent buckets ($\{0\}$) were considered as candidates.