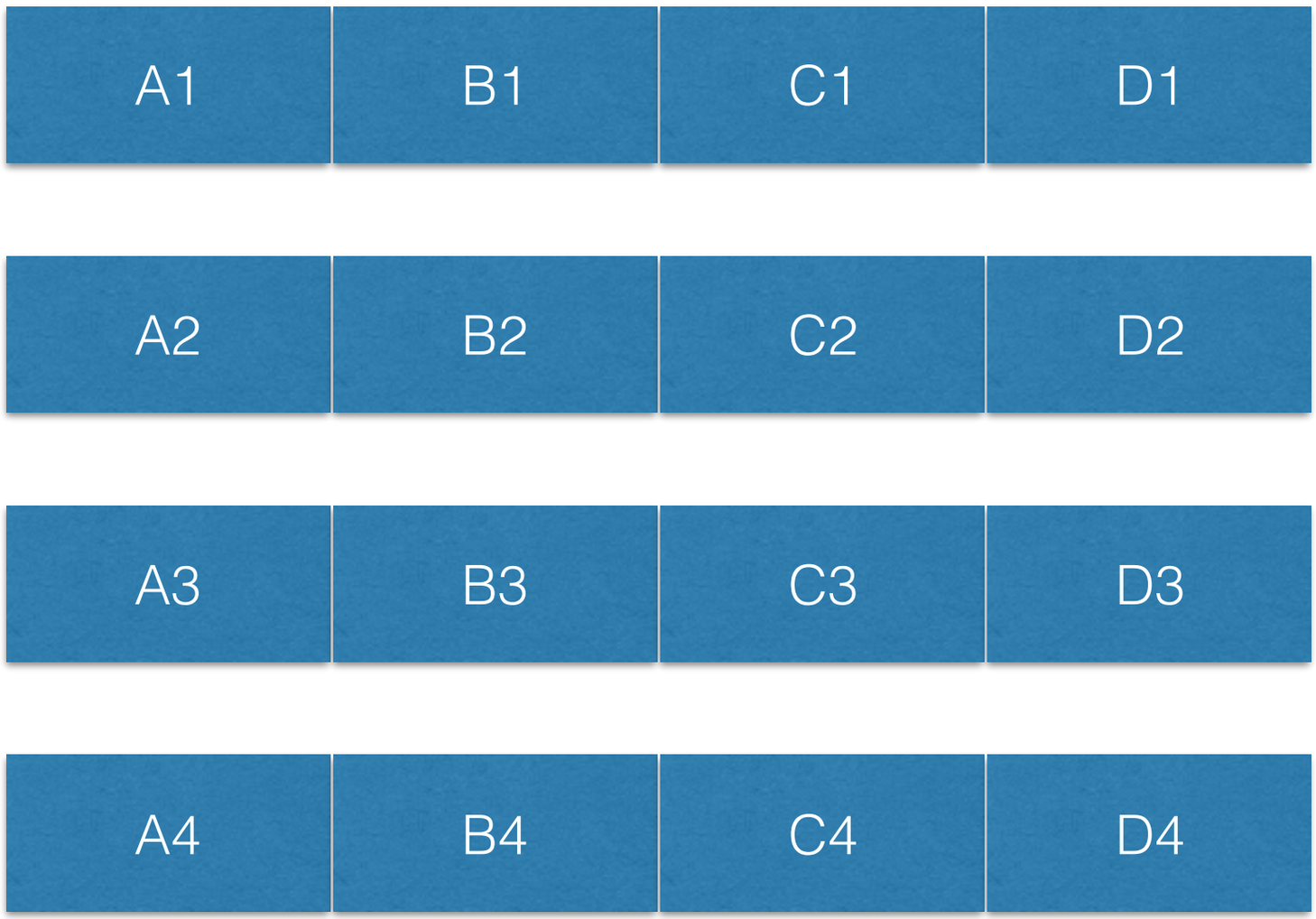
Database Cracking

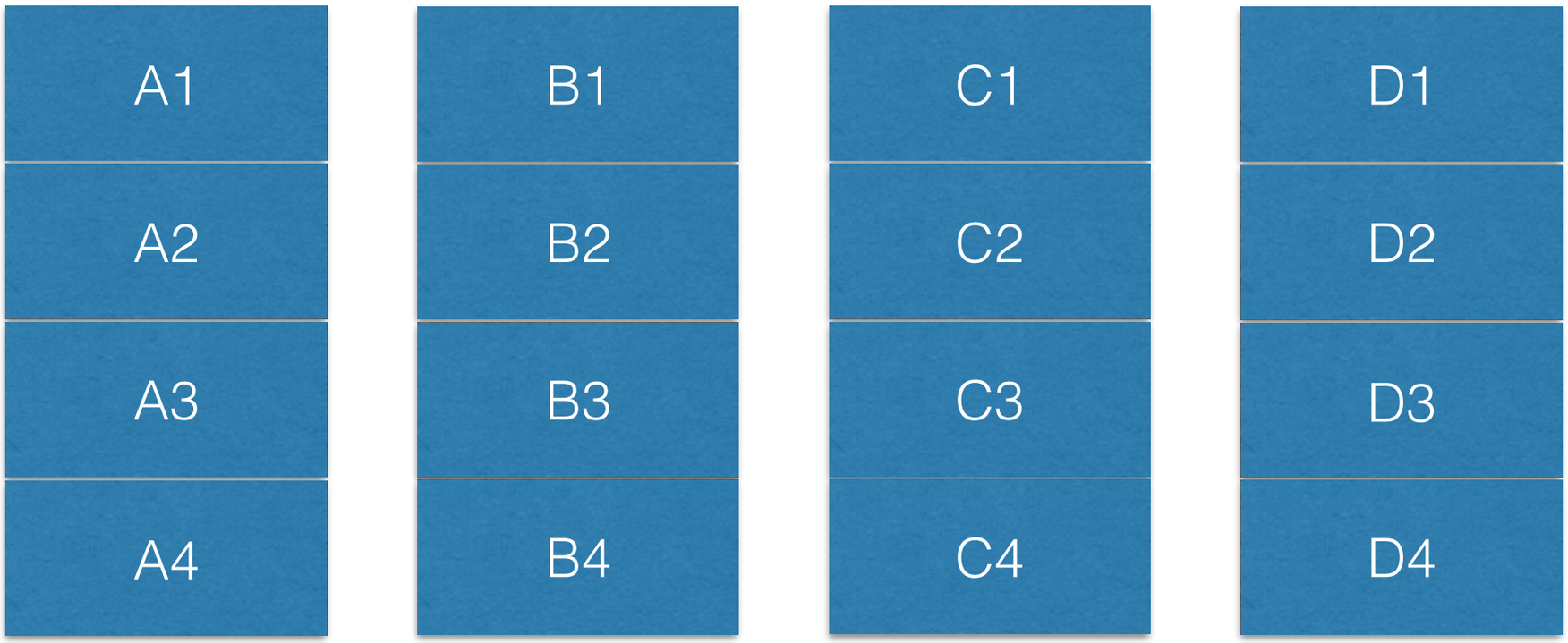
Languages and Runtimes for Big Data

Row Stores



Traditional DB: Lay out data on disk in rows

Column Stores



Columnar DB: Lay out data on disk in columns

Column Stores

A1	Row1
A2	Row2
A3	Row3
A4	Row4

B1	Row1
B2	Row2
B3	Row3
B4	Row4

C1	Row1
C2	Row2
C3	Row3
C4	Row4

D1	Row1
D2	Row2
D3	Row3
D4	Row4

Store with Row ID to recover original table

Why use a Column Store?

Immediate Data Access

Problem: Data is initially unsorted

Query: Find all rows where $100 < A \leq 200$

What is the fastest way to answer this query?

Immediate Data Access

Problem: Data is initially unsorted

What if you get 2 queries?

… 3 queries?

… 100 queries?

Immediate Data Access

Problem: Data is initially unsorted

Strategy 1: Index the data then run queries

First few queries are much slower (upfront indexing cost)

Strategy 2: Linear scans over the data

Last few queries are much slower (no indexing!)

Immediate Data Access

Problem: Data is initially unsorted

Strategy 3: Index **while** you run queries!

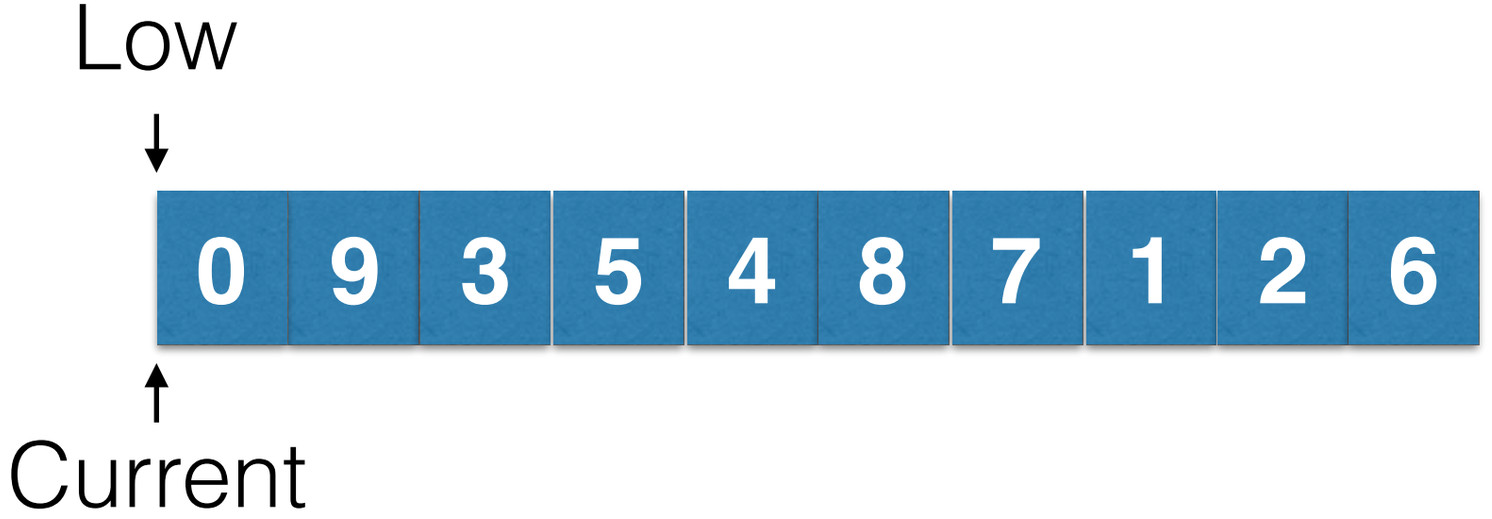
Re-use compute effort of scans.

Cracking

Query 1: Find $4 < X \leq 7$

Cracking

Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

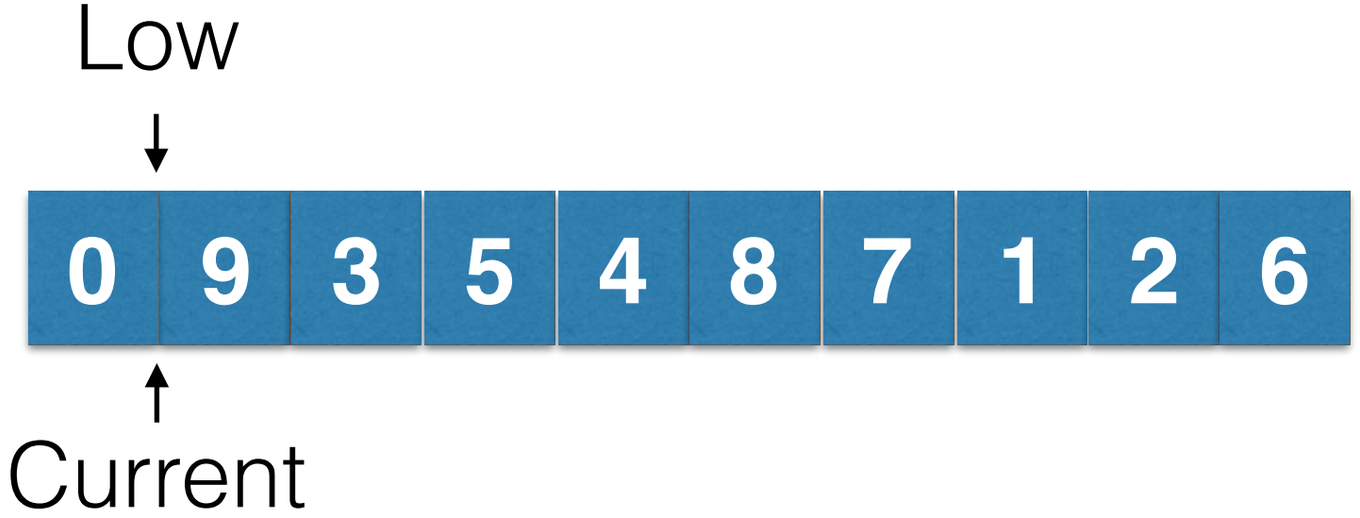
Cracking

Step 1: Split into 2 bins: > 4 and ≤ 4

Query 1: Find 4 < X ≤ 7

Cracking

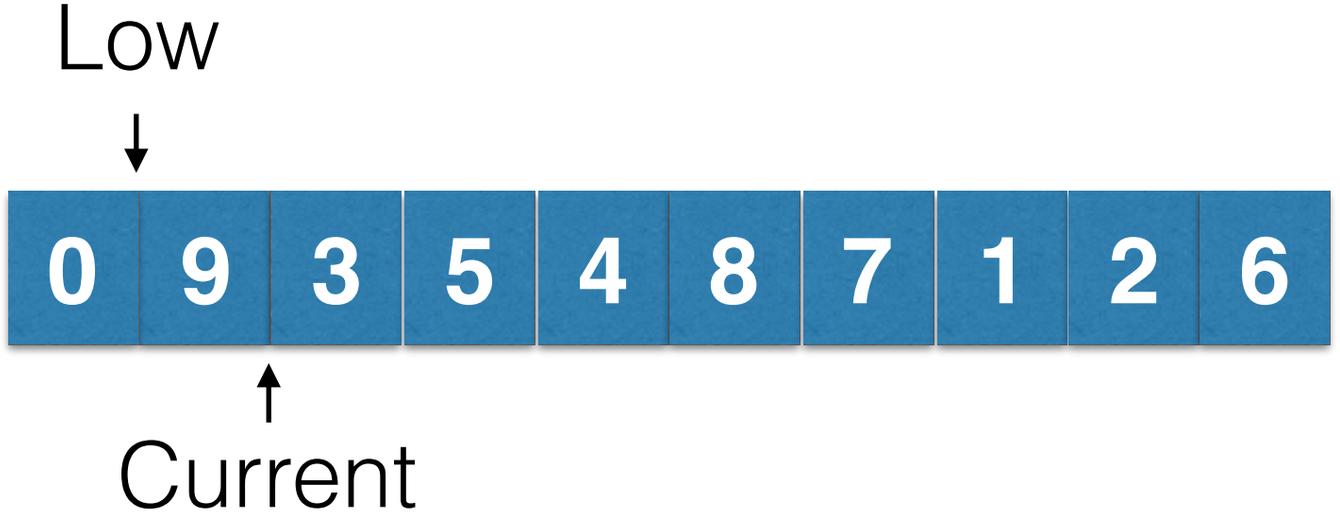
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

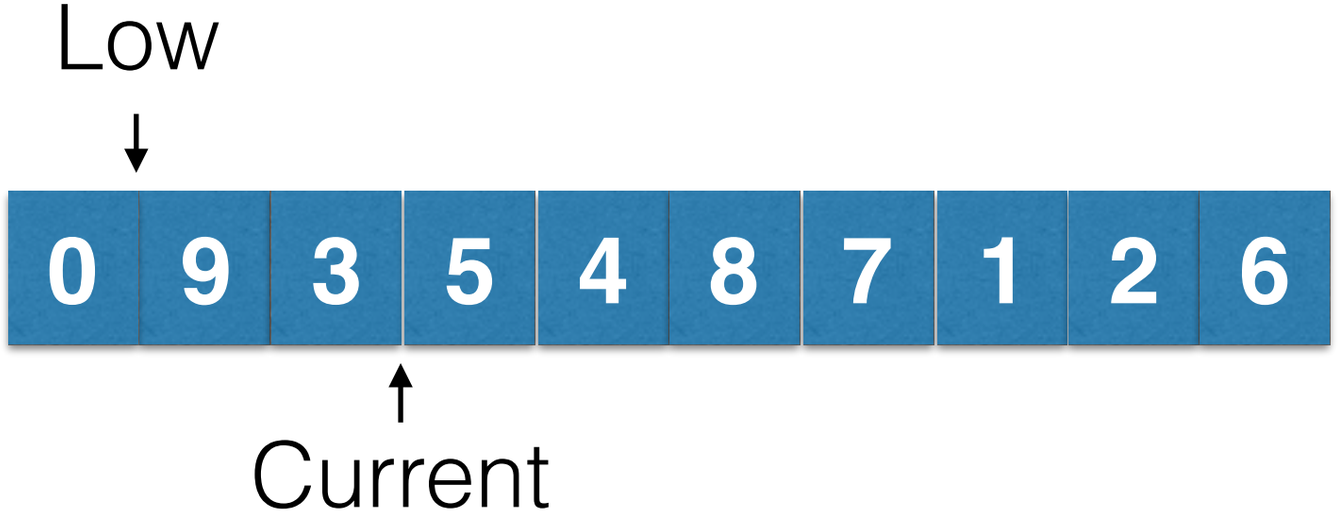
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

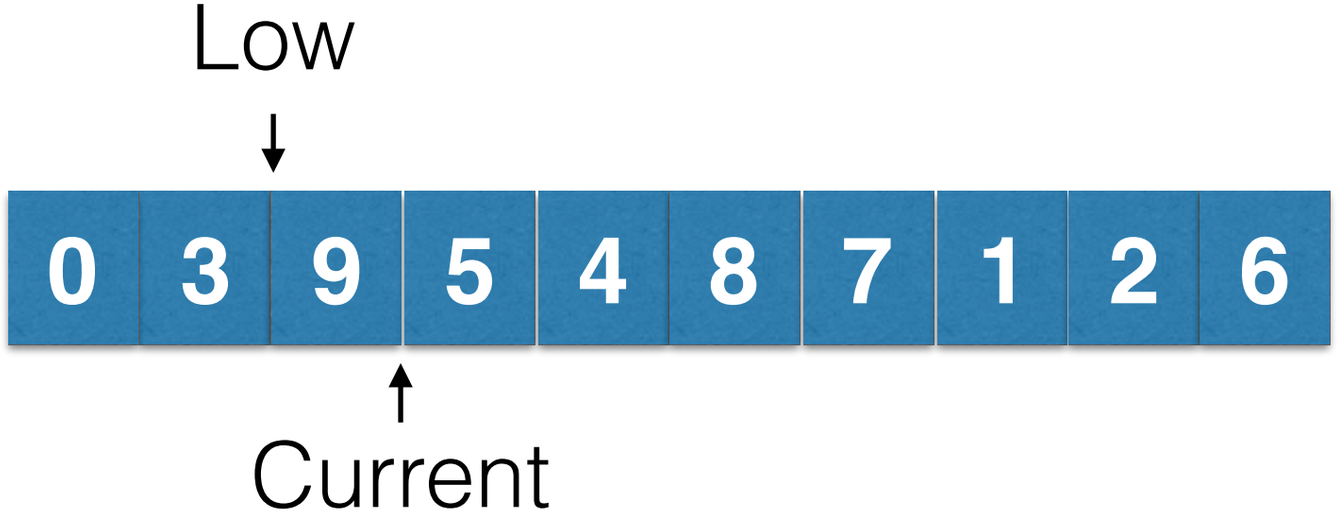
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

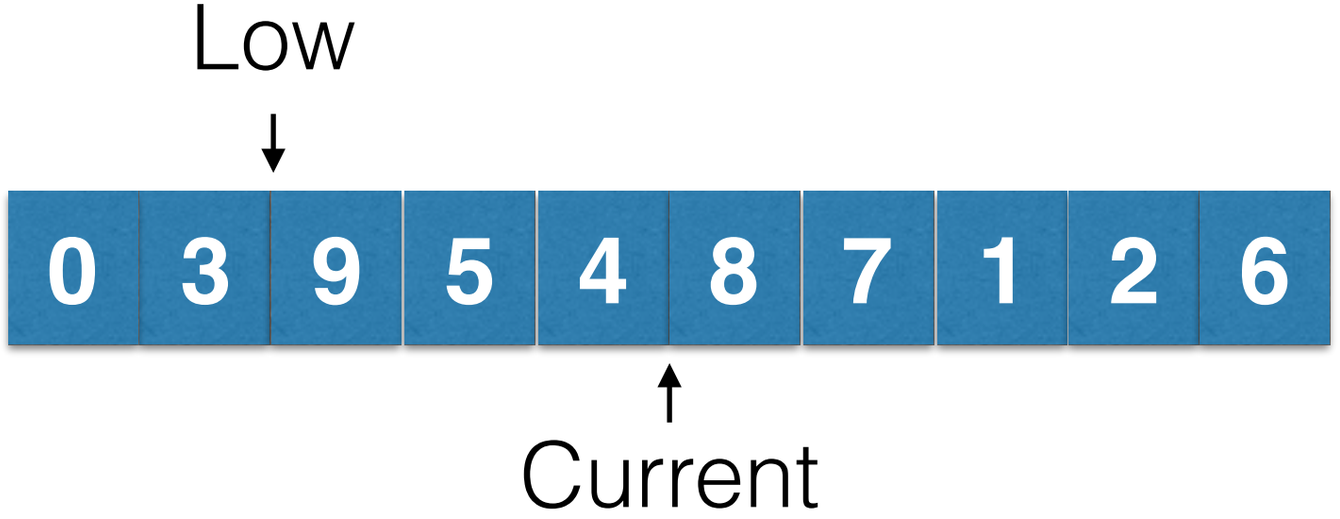
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

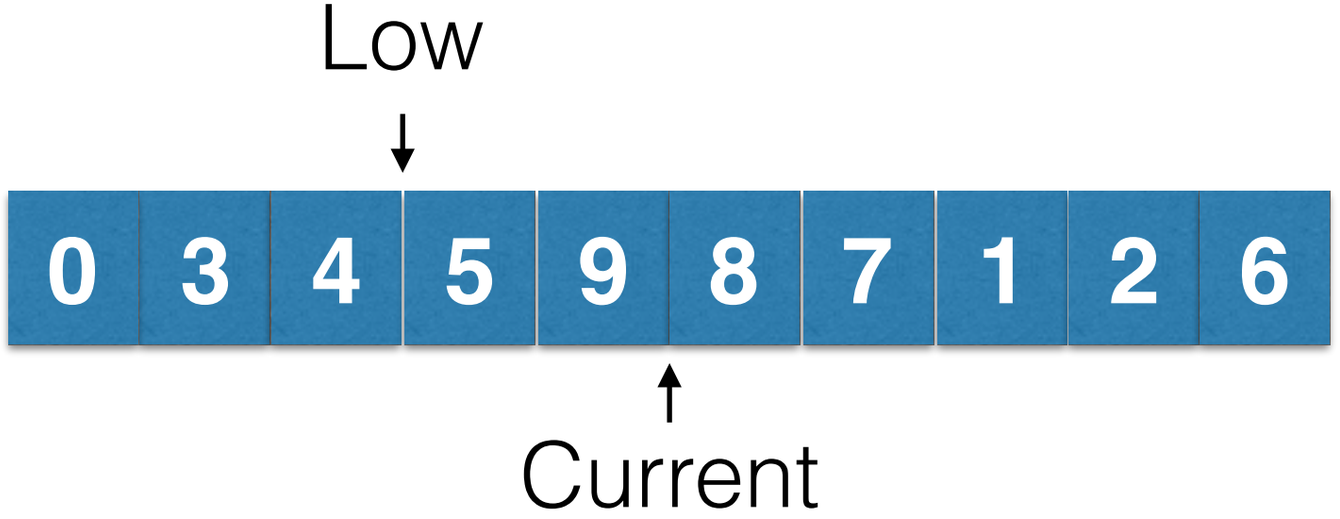
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

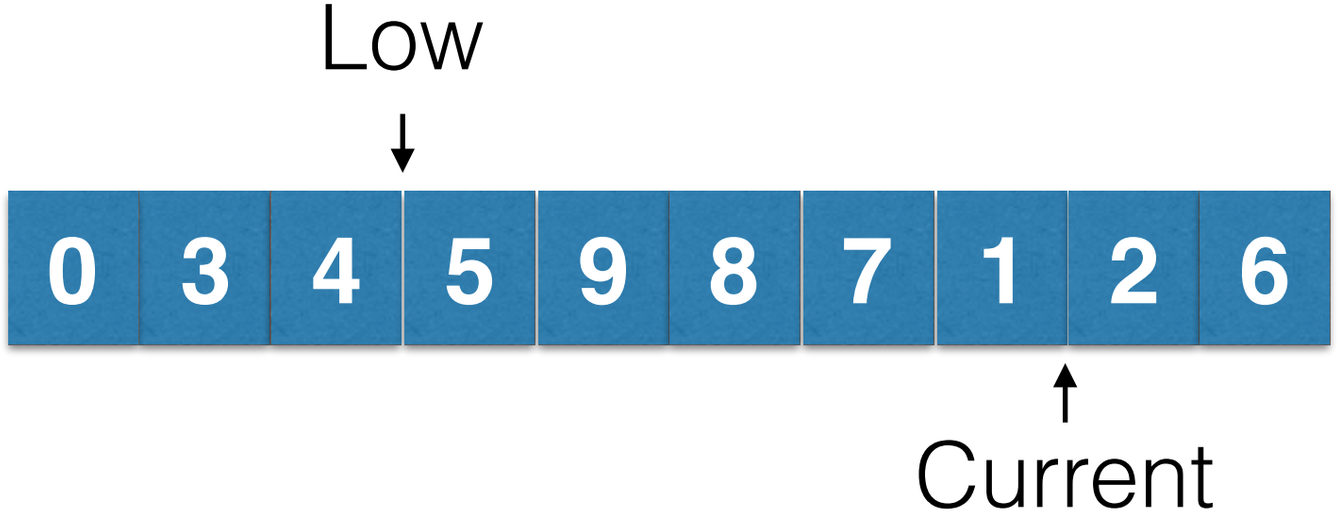
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

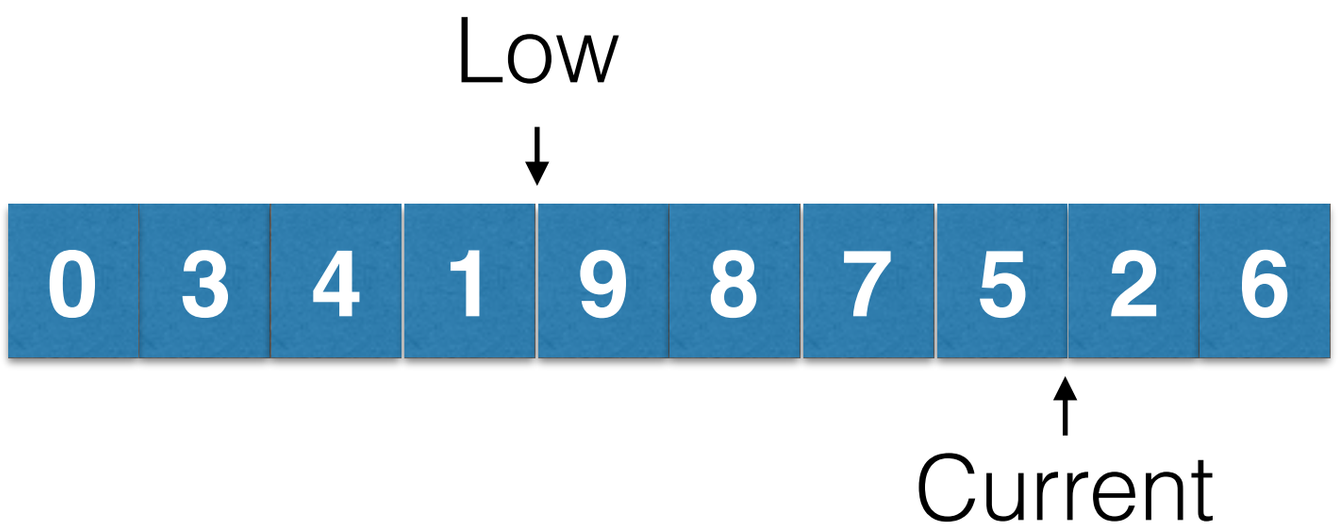
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

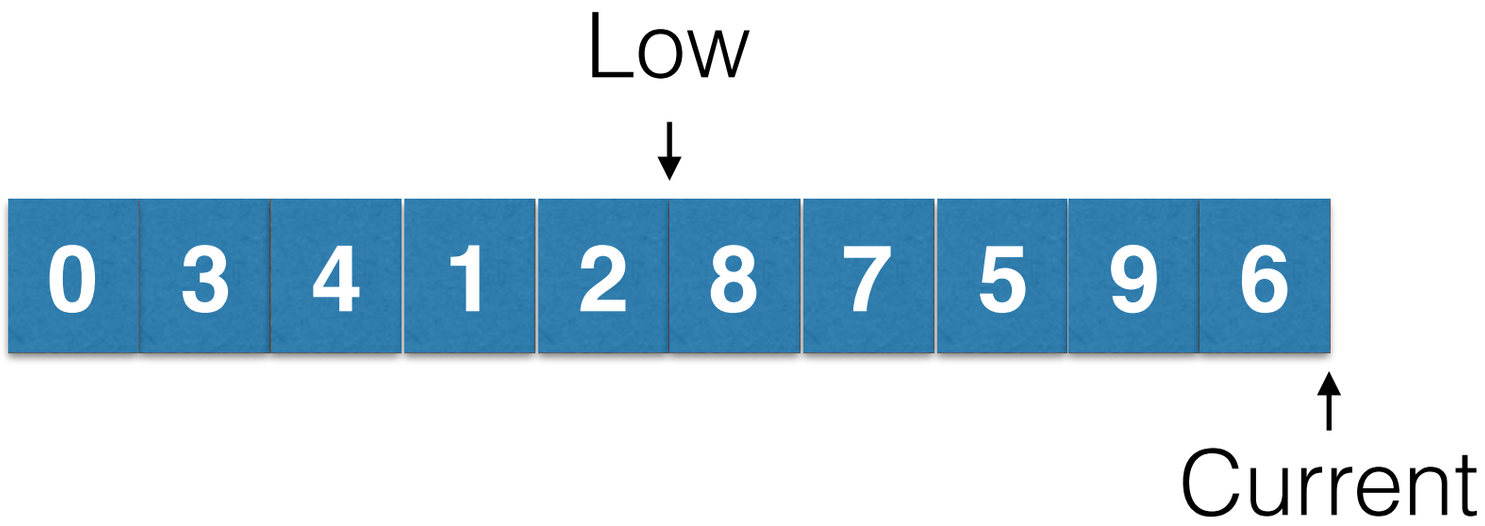
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

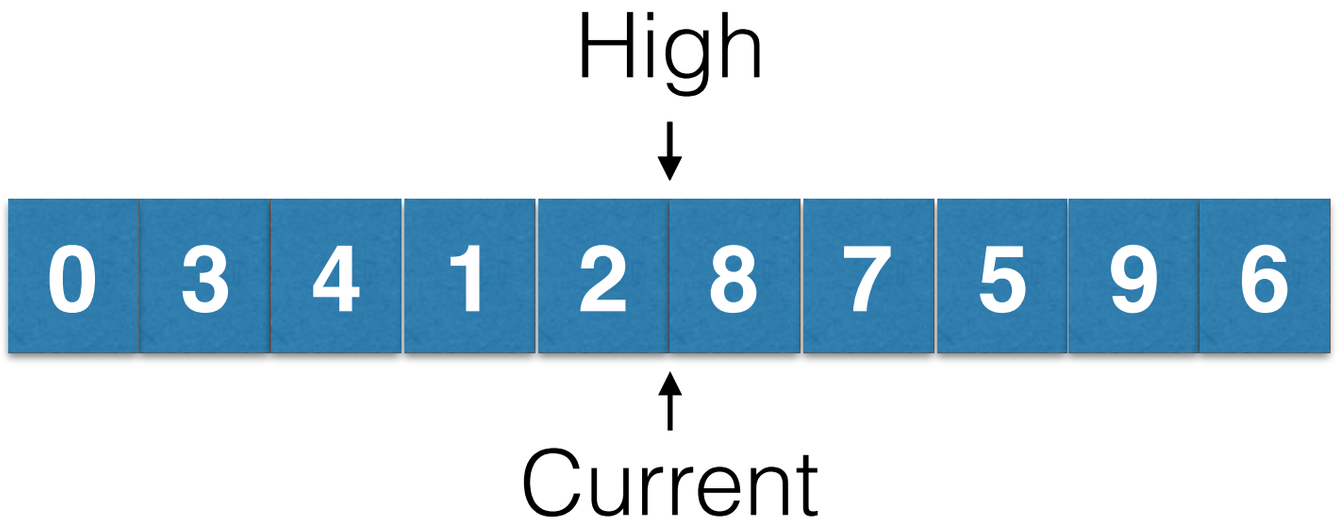
Step 1: Split into 2 bins: > 4 and ≤ 4



Query 1: Find 4 < X ≤ 7

Cracking

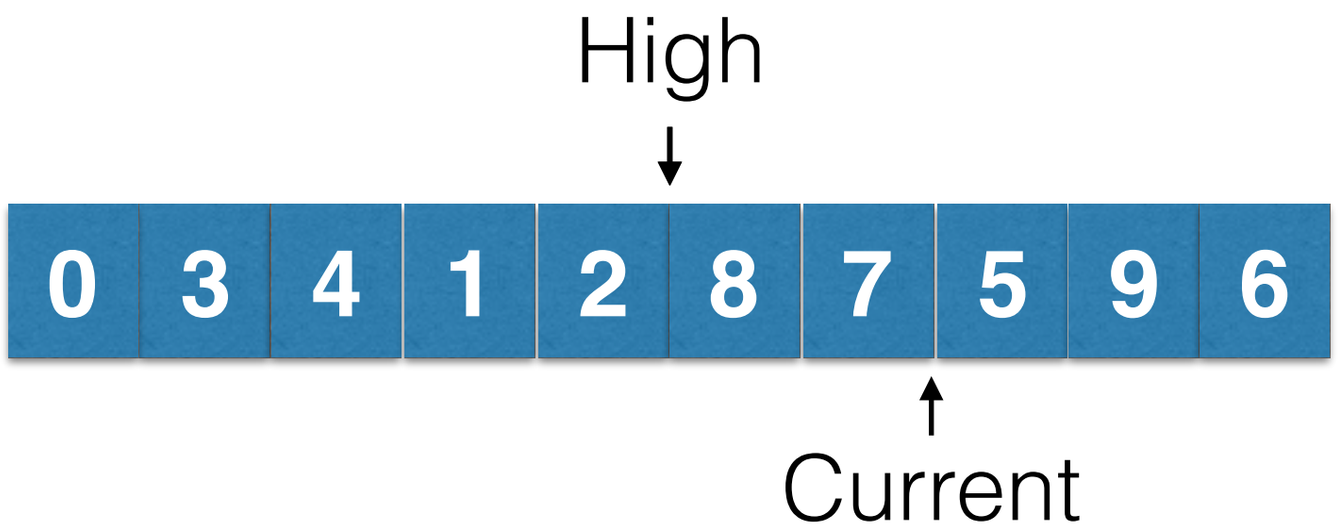
Step 2: Split into 2 bins: > 7 and ≤ 7



Query 1: Find 4 < X ≤ 7

Cracking

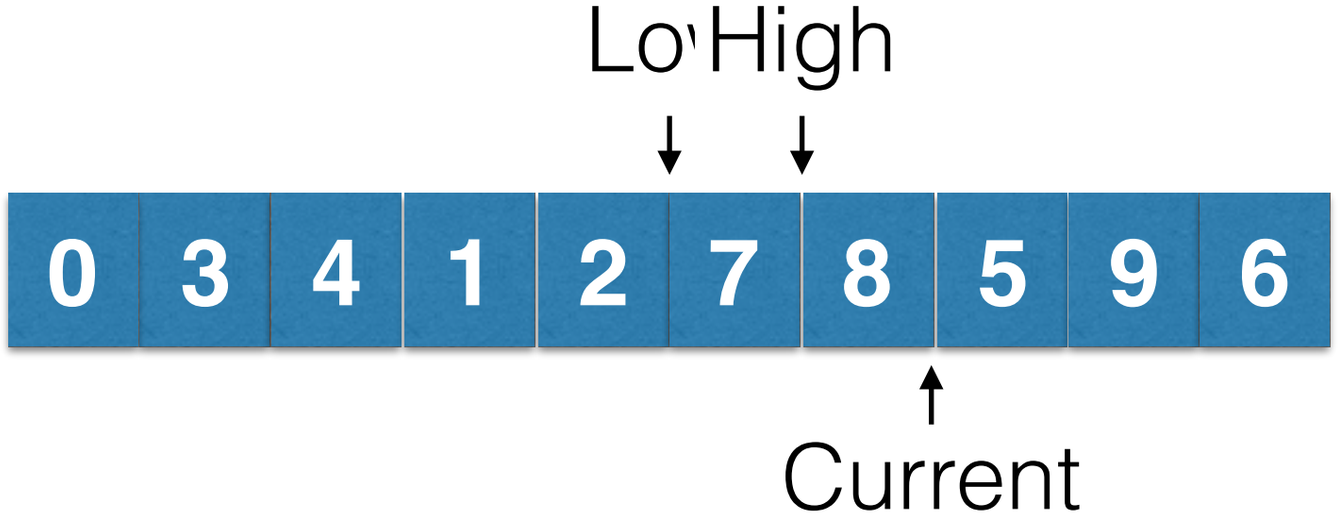
Step 2: Split into 2 bins: > 7 and ≤ 7



Query 1: Find 4 < X ≤ 7

Cracking

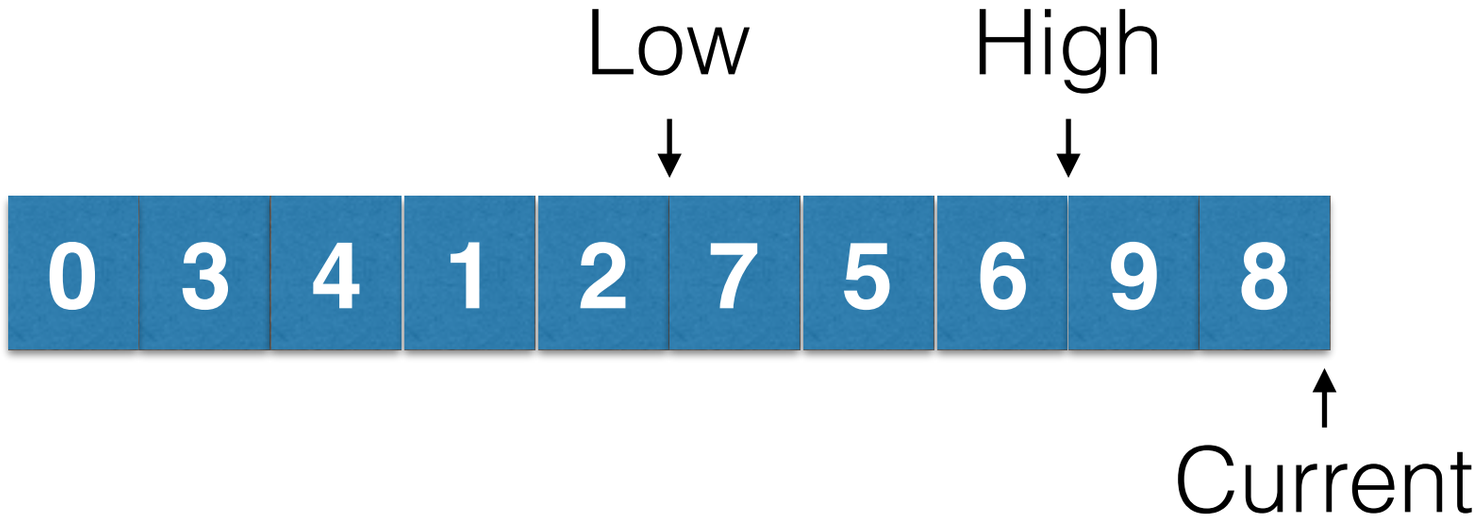
Step 2: Split into 2 bins: > 7 and ≤ 7



Query 1: Find 4 < X ≤ 7

Cracking

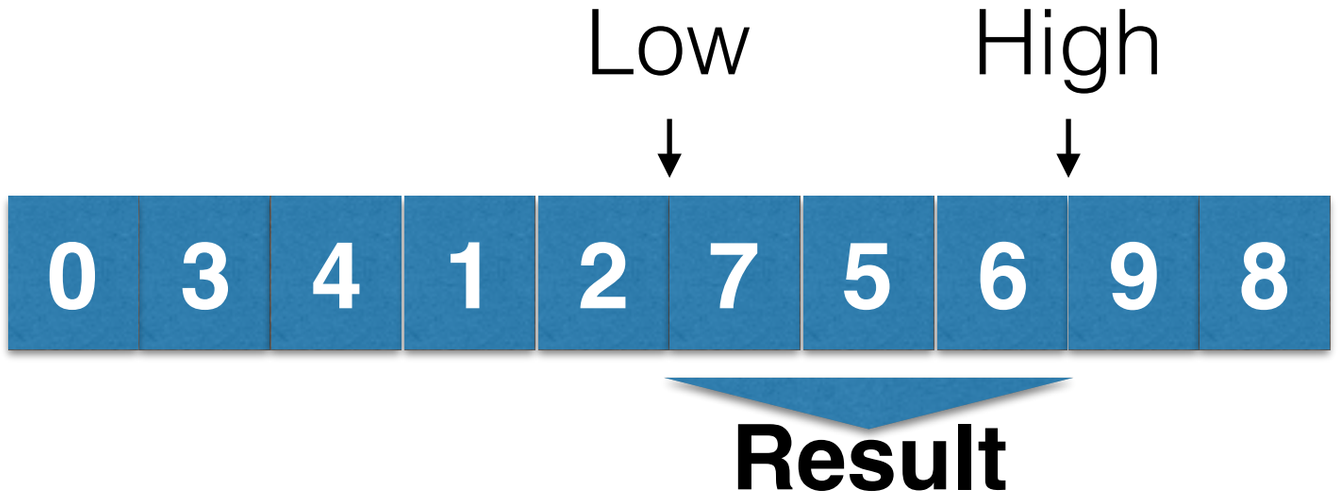
Step 2: Split into 2 bins: > 7 and ≤ 7



Query 1: Find 4 < X ≤ 7

Cracking

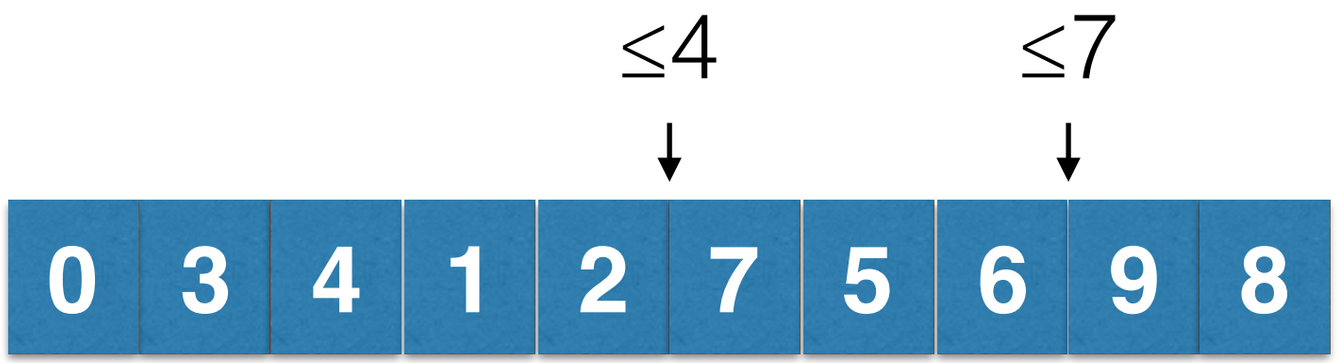
Step 2: Split into 2 bins: > 7 and ≤ 7



Query 1: Find 4 < X ≤ 7

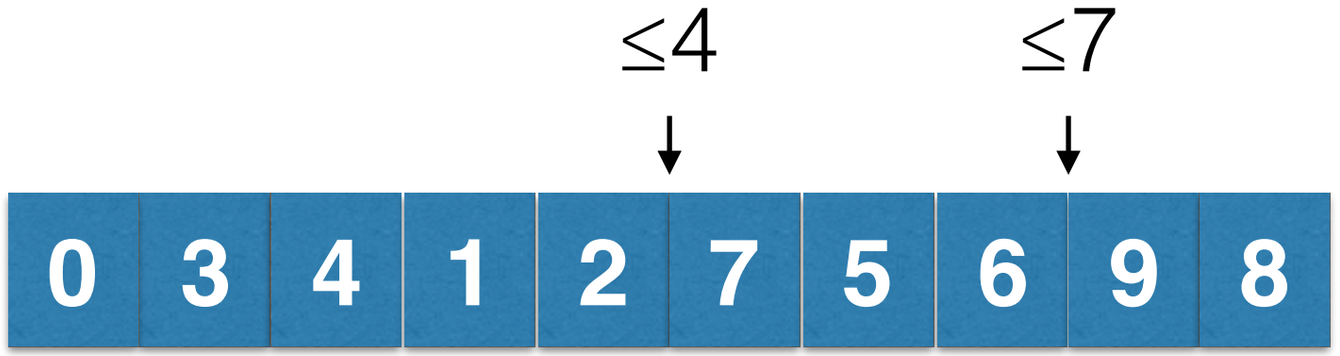
Cracking

Binary Tree



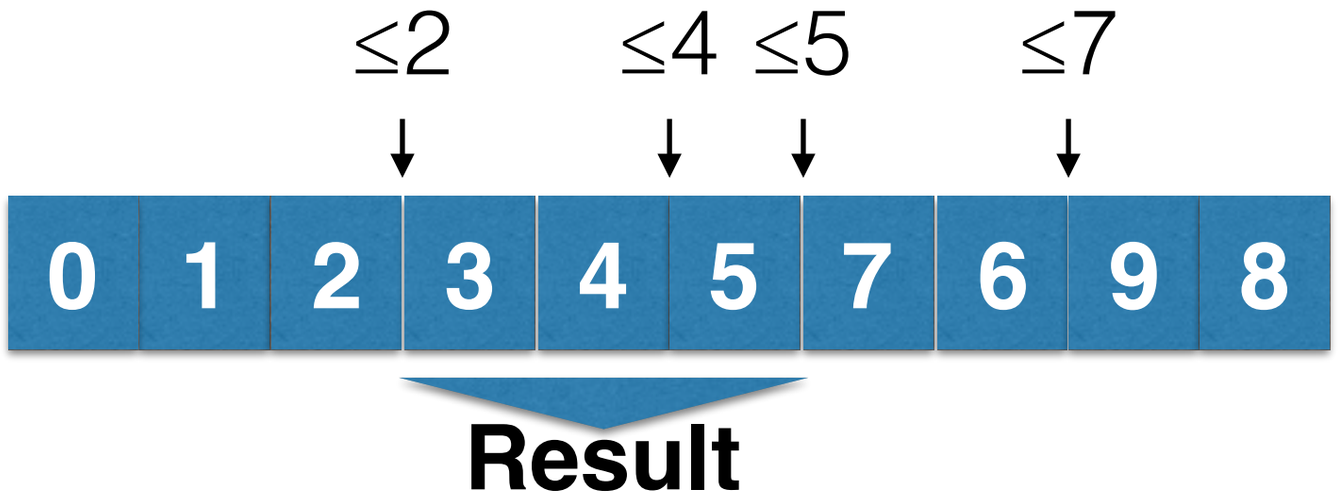
Query 1: Find $4 < X \leq 7$

Cracking



Query 2: Find $2 < X \leq 5$

Cracking



Query 2: Find $2 < X \leq 5$

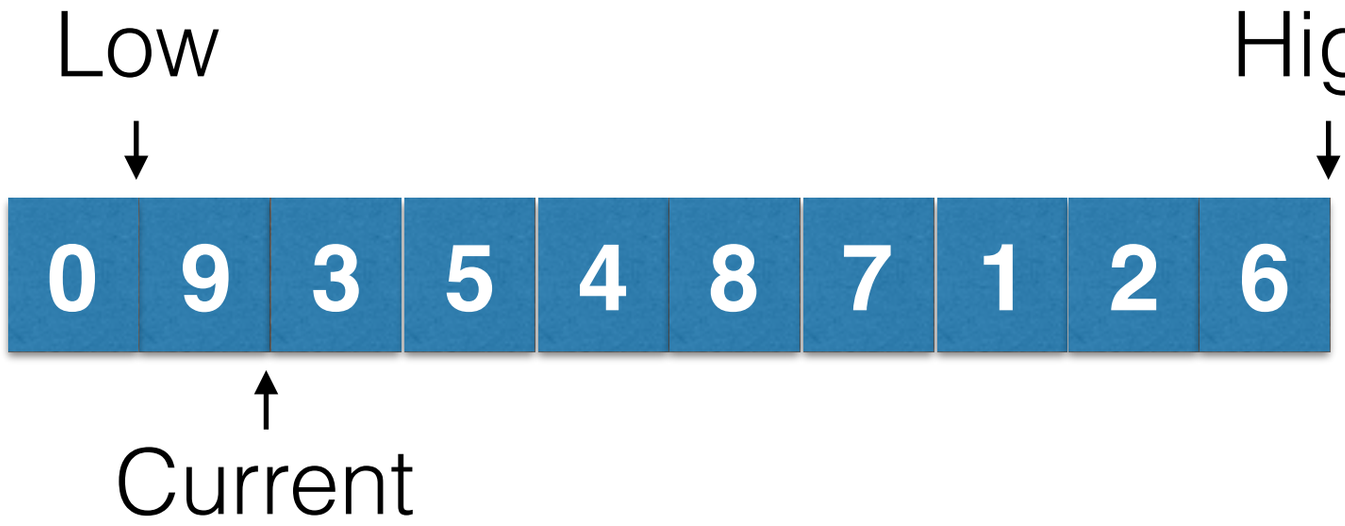
3-Way Cracking

Query 1: Find $4 < X \leq 7$

3-Way Cracking

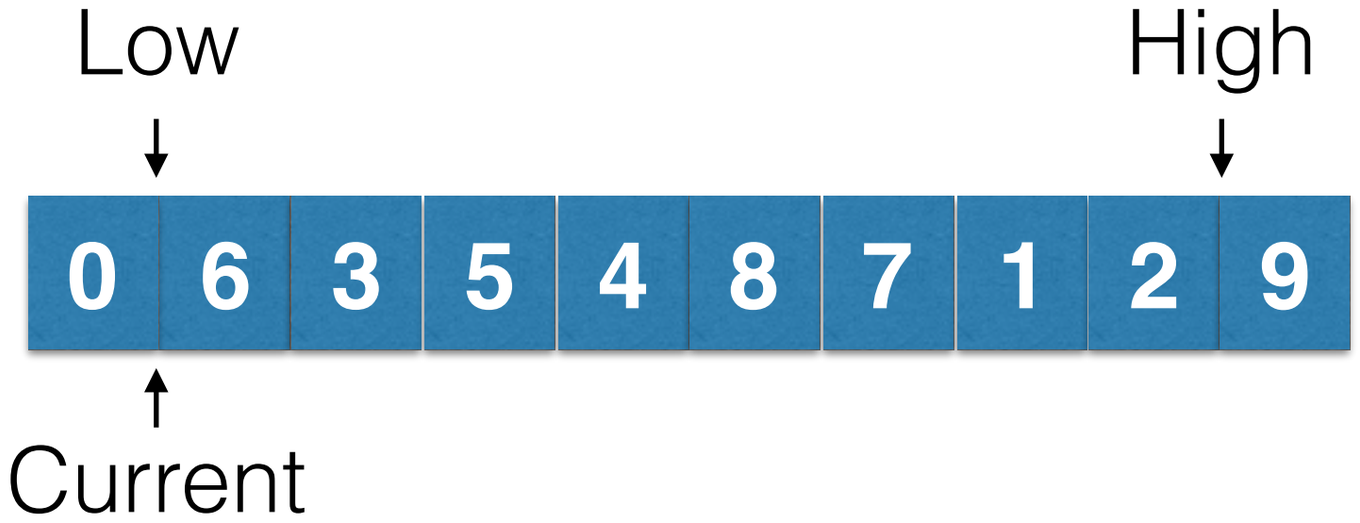
Query 1: Find $4 < X \leq 7$

3-Way Cracking



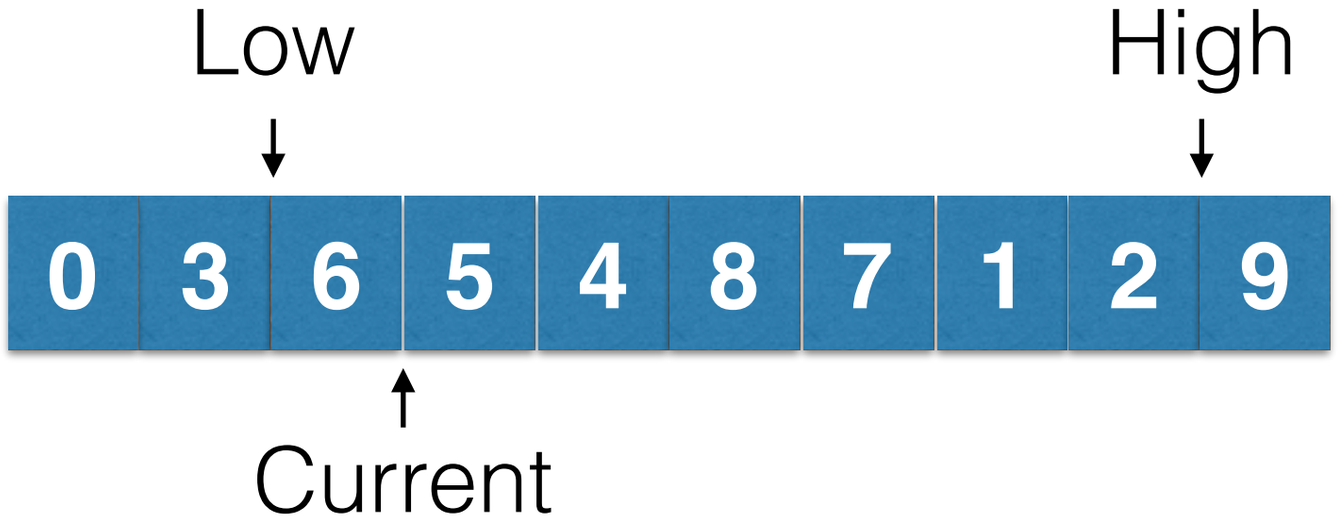
Query 1: Find $4 < X \leq 7$

3-Way Cracking



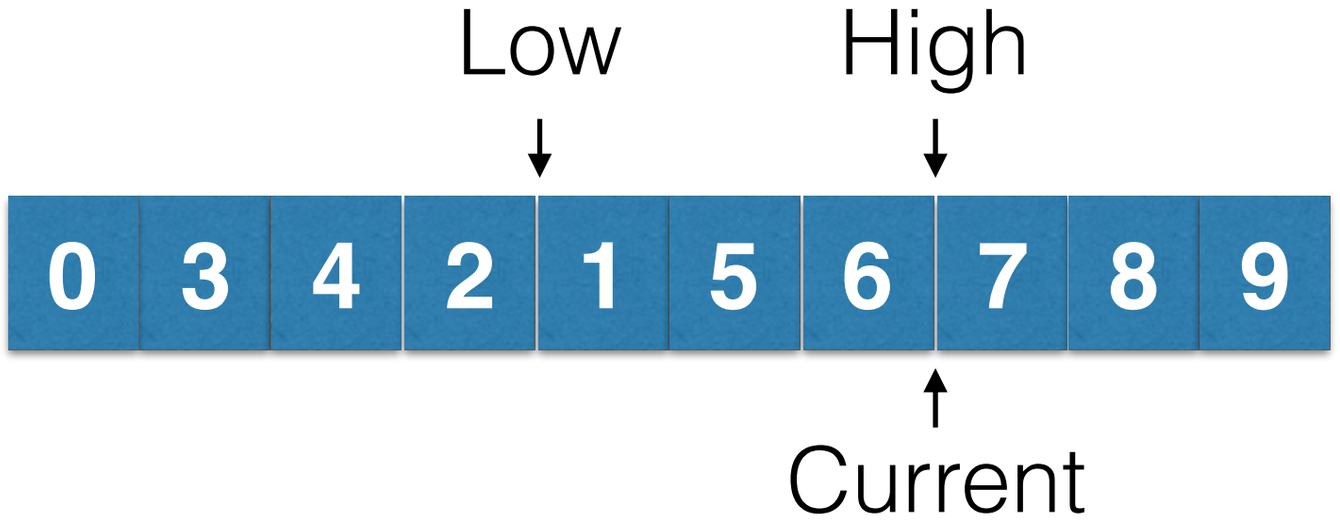
Query 1: Find $4 < X \leq 7$

3-Way Cracking



Query 1: Find $4 < X \leq 7$

3-Way Cracking



Query 1: Find $4 < X \leq 7$

Discussion Questions…

Does cracking work with a row-oriented database?

Discussion Questions…

How would one crack a multi-attribute index?

(e.g., a spatial index?)

Discussion Questions…

Can updates be performed efficiently on a cracker index?

Discussion Questions…

Can updates be performed efficiently on a cracker index?

What constraints are required?

Discussion Questions…

What applications would cracking work well on?

What applications would cracking work poorly on?

Discussion Questions…

Upfront Indexing vs Sequential Scan vs Cracking…

Where is the cutoff?