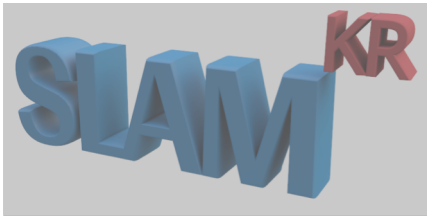
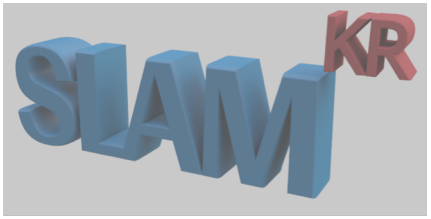


Voxel Hashing

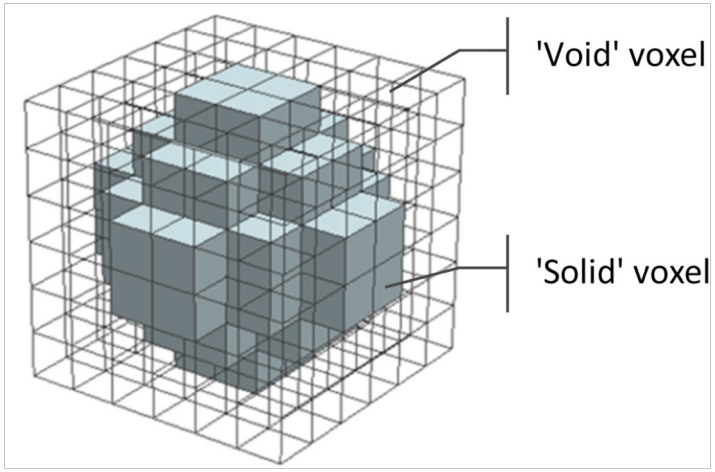
April 14, 2019

Dong-Won Shin

Volumetric Representation

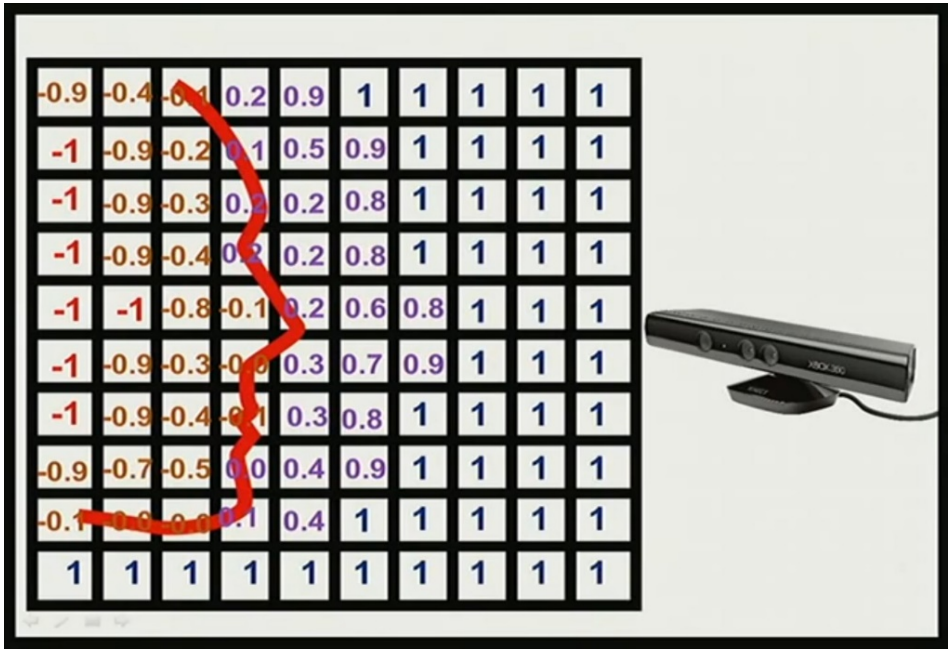


- Volumetric representation using a hash lookup
- Data structures and operations
  - Voxel
  - Voxel block hash
  - Hash table and hashing function
  - Hash table operations
- Voxel
  - A value on a regular grid in 3D space, the extended concept of 2D pixel
  - Widely used for realistic rendering 3D object in computer graphics
  - Data
    - Truncated signed distance function (TSDF) value
    - TSDF Weight
    - Color value
    - Color weight

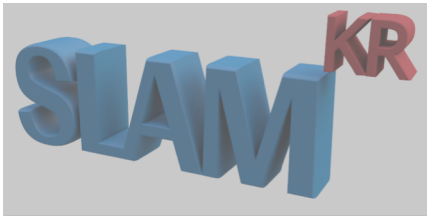


Volumetric Representation

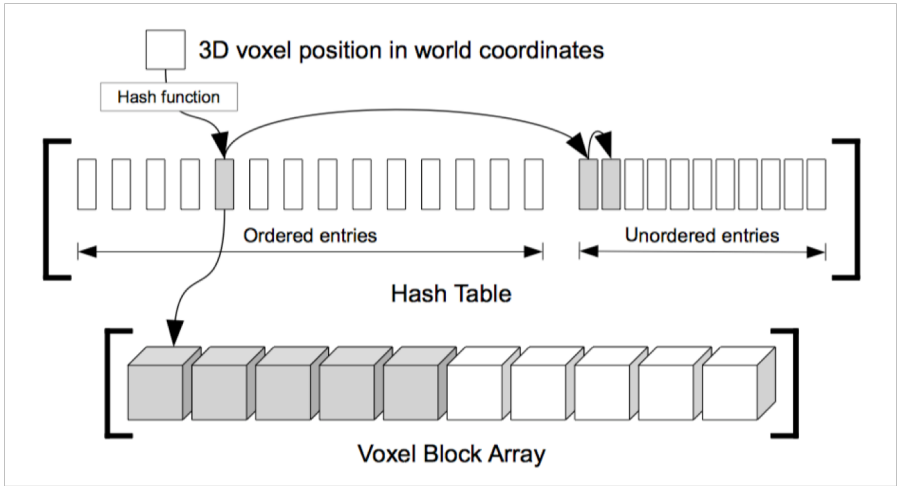
- Truncated signed distance function (TSDF)
  - Predefined 3D volume is subdivided uniformly into a 3D grid of voxels
  - These values are positive in-front of the surface and negative behind
  - Zero-crossing point means the surface of the object



Volumetric Representation

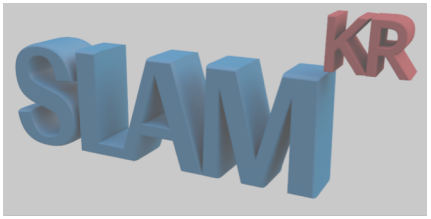


- Voxel block array
  - Majority of data stored in the regular voxel grid is marked either as
    - Free space
    - Unobserved space
  - Only store the surface data by efficient hashing scheme
  - Grouped voxels in blocks of predefined size (ex. 8x8x8)
  - Data
    - Positions of the corner of the 8x8x8 voxel block
    - Offset in the excess list
    - Pointer to the voxel block array



M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

Volumetric Representation

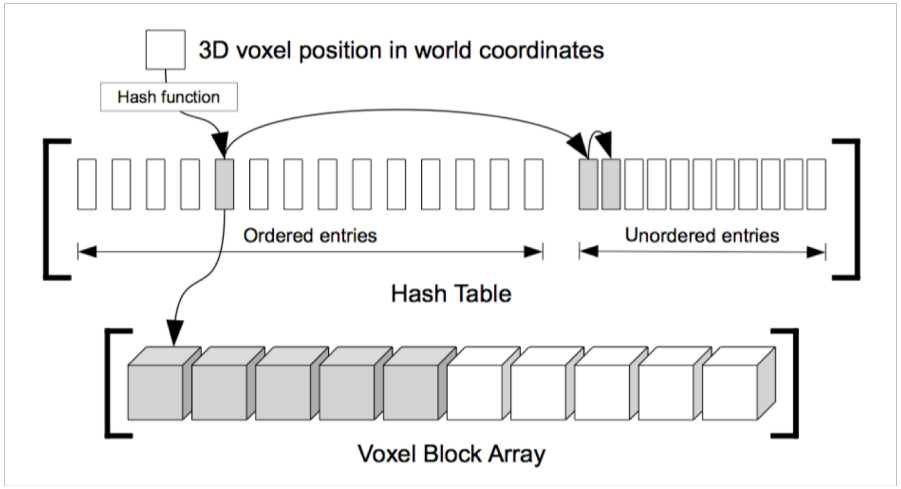


- Hash table
  - To quickly and efficiently find the position of a certain voxel block in the voxel block array

- Hashing function
  - For locating entries of the hash table takes the corner coordinates of a 3D voxel block

```
7  template<typename T> _CPU_AND_GPU_CODE_ inline int hashIndex(const THREADPTR(T) & blockPos) {
8      return (((uint)blockPos.x * 73856093u) ^ ((uint)blockPos.y * 19349669u) ^ ((uint)blockPos.z * 83492791u)) & (uint)SDF_HASH_MASK;
```

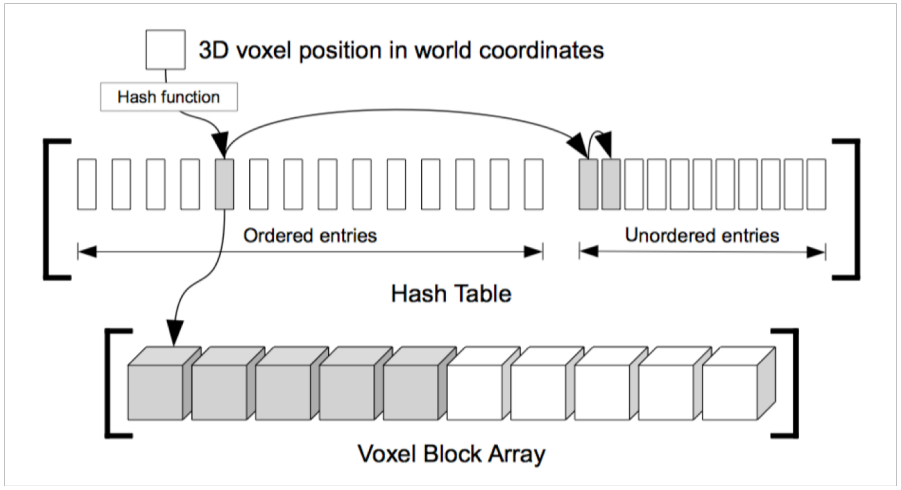
  - Hash collision case
    - Use the additional unordered excess list
    - Store an offset in the voxel block array



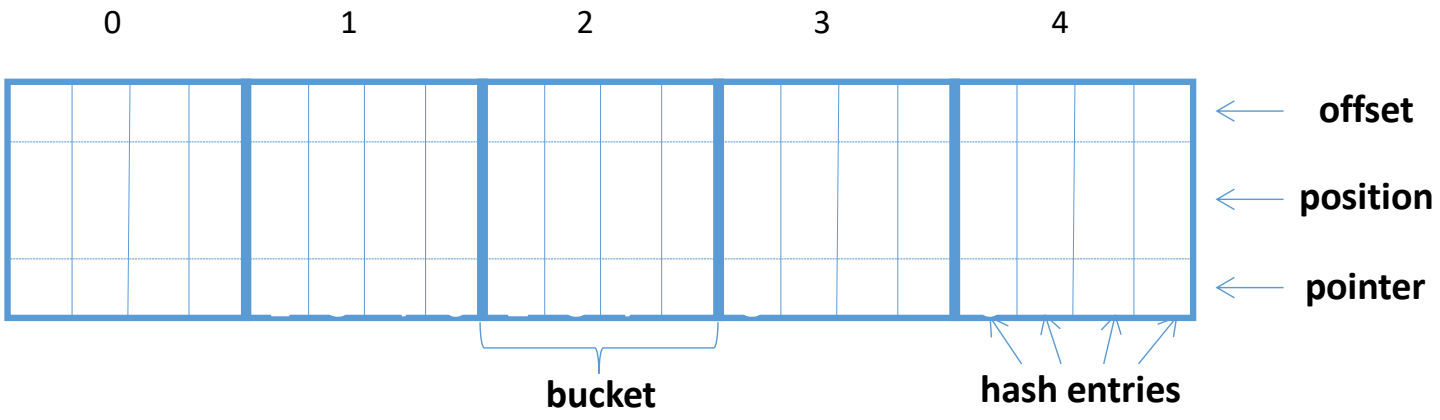
Volumetric Representation

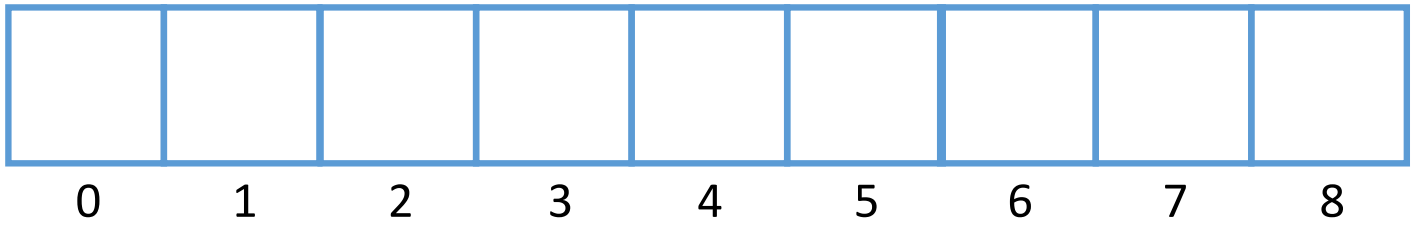
- Hash table operations
  - Given a target 3D voxel location in world coordinates
  - Compute its corresponding voxel block location by dividing the voxel location by the size of the voxel block array
  - Call the hashing function to compute the index of the bucket from the ordered part of th e hash table

  - Retrieval
    - Returns the voxel stored at the target location within the block addressed by the hash entry
  - Insertion
    - Reserves a block inside the voxel block array



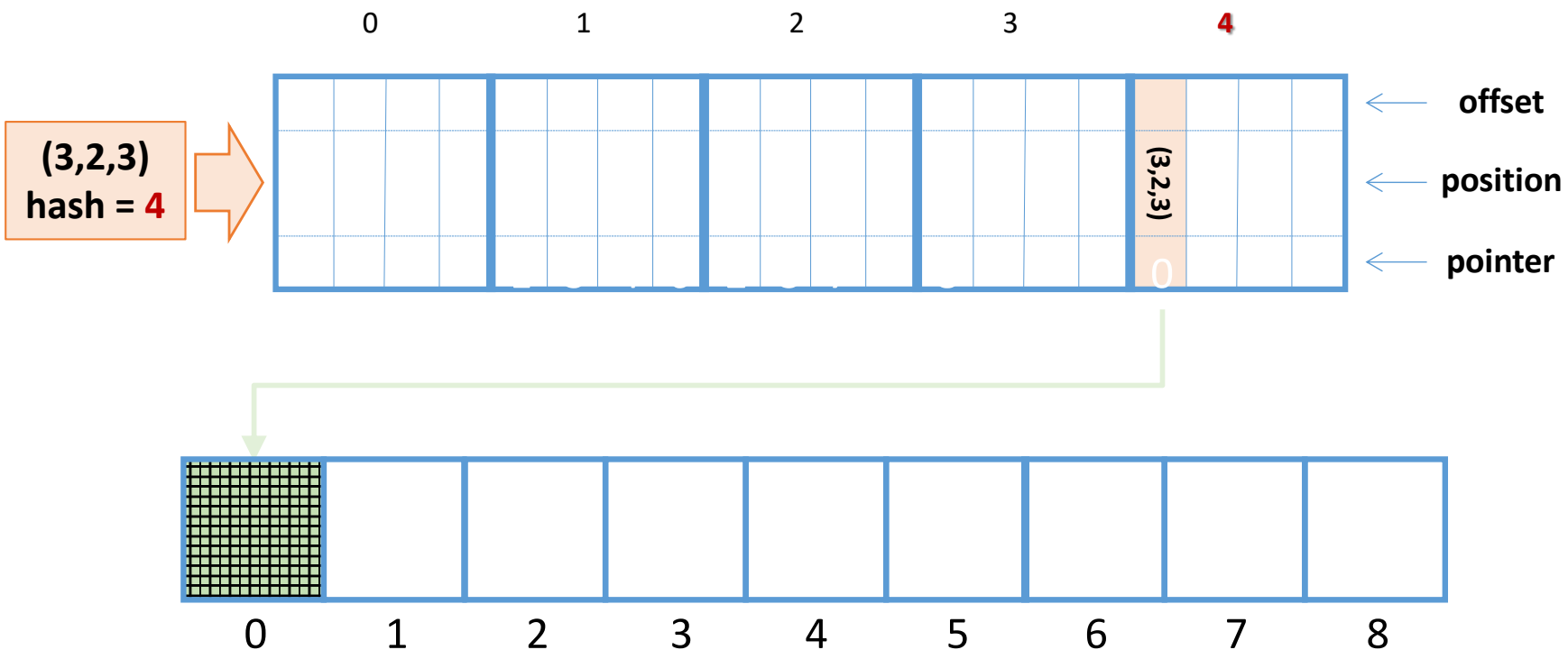
Hashing Operation



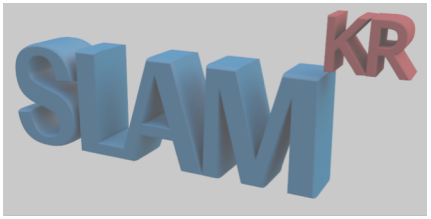


Hashing Operation

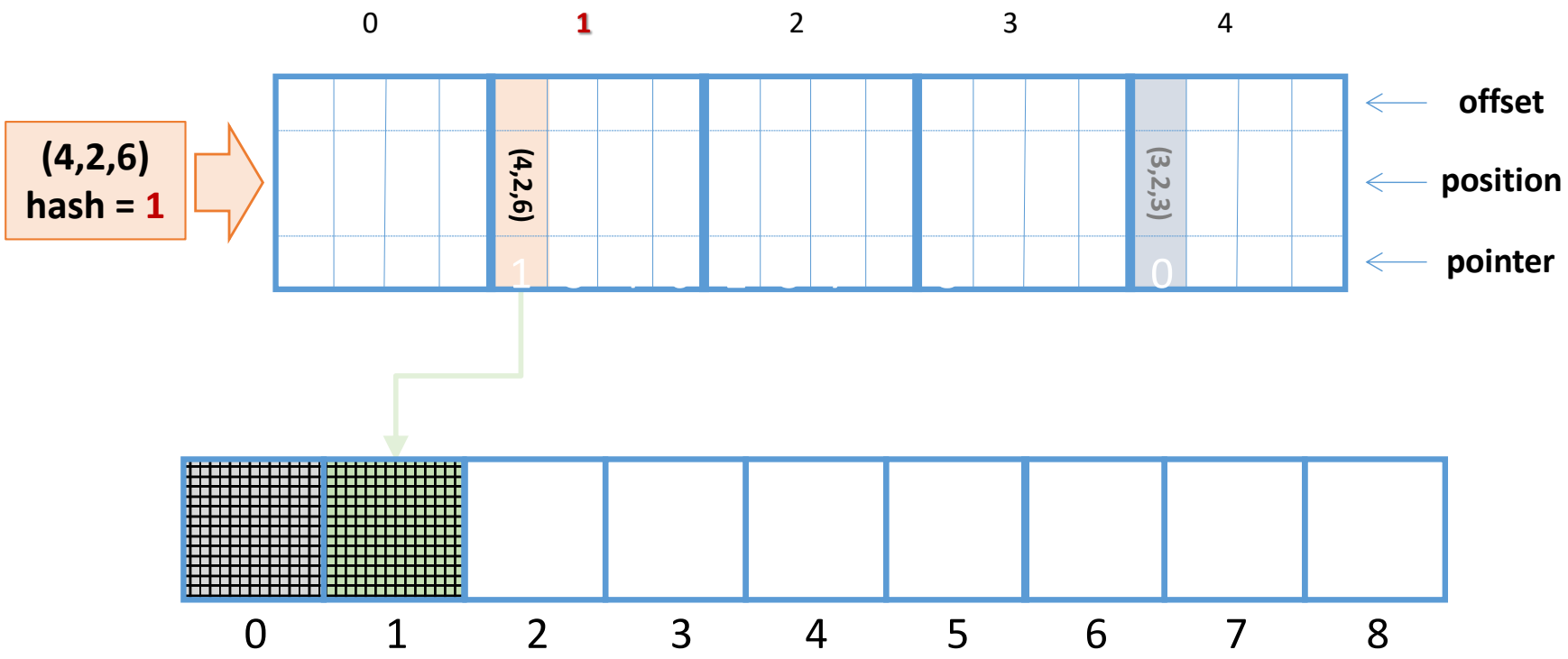
- Insert



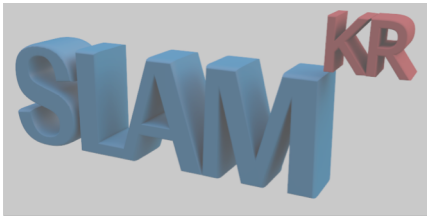
Hashing Operation



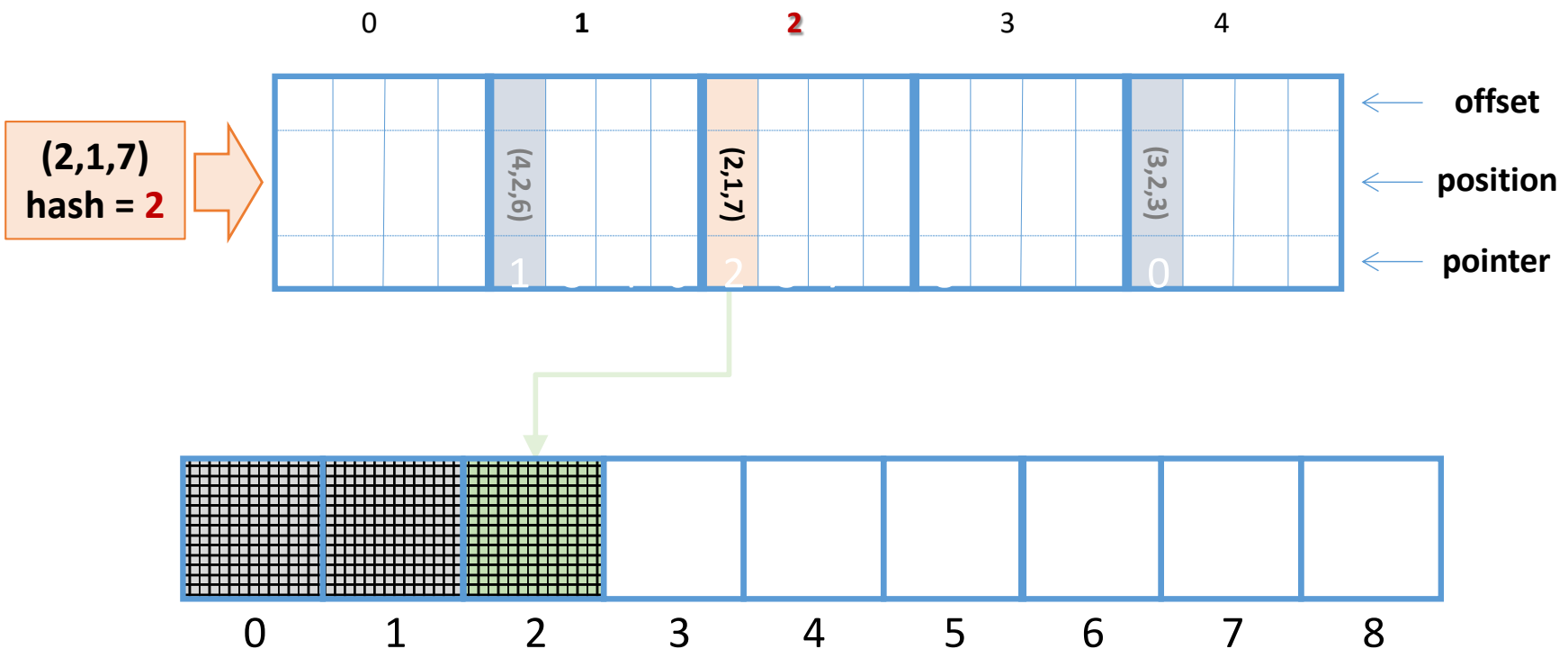
- Insert



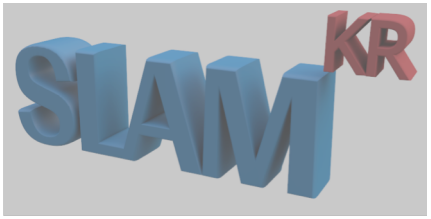
Hashing Operation



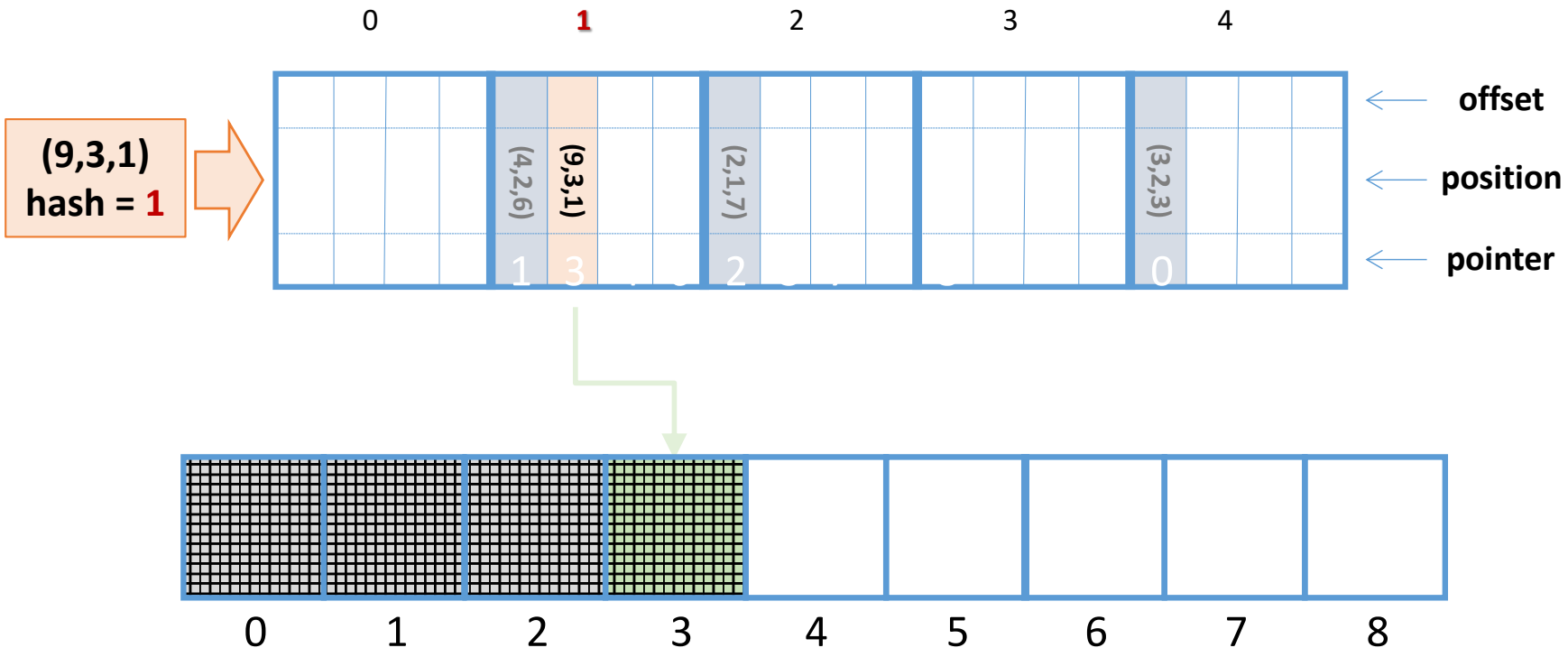
- Insert



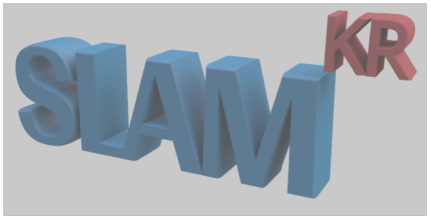
Hashing Operation



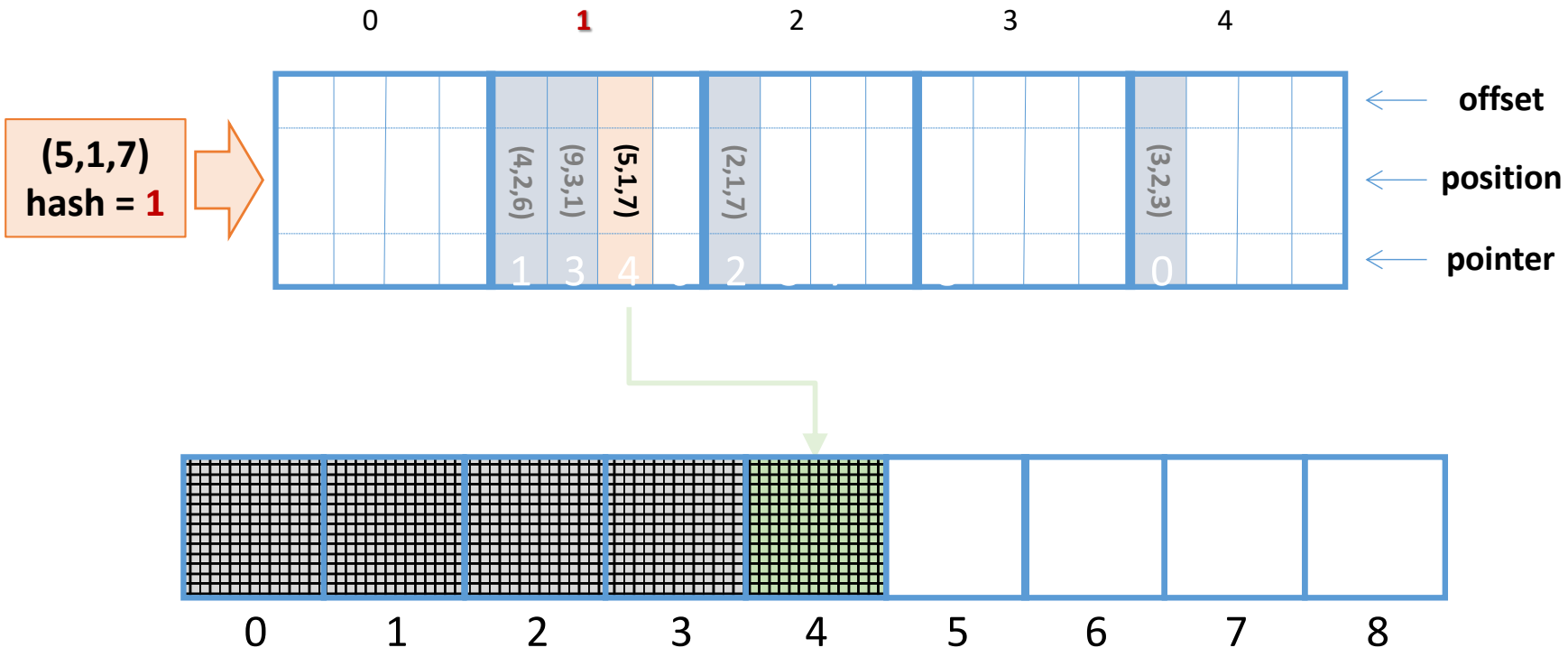
- Insert



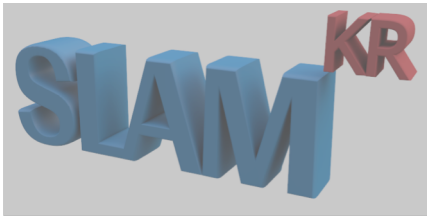
Hashing Operation



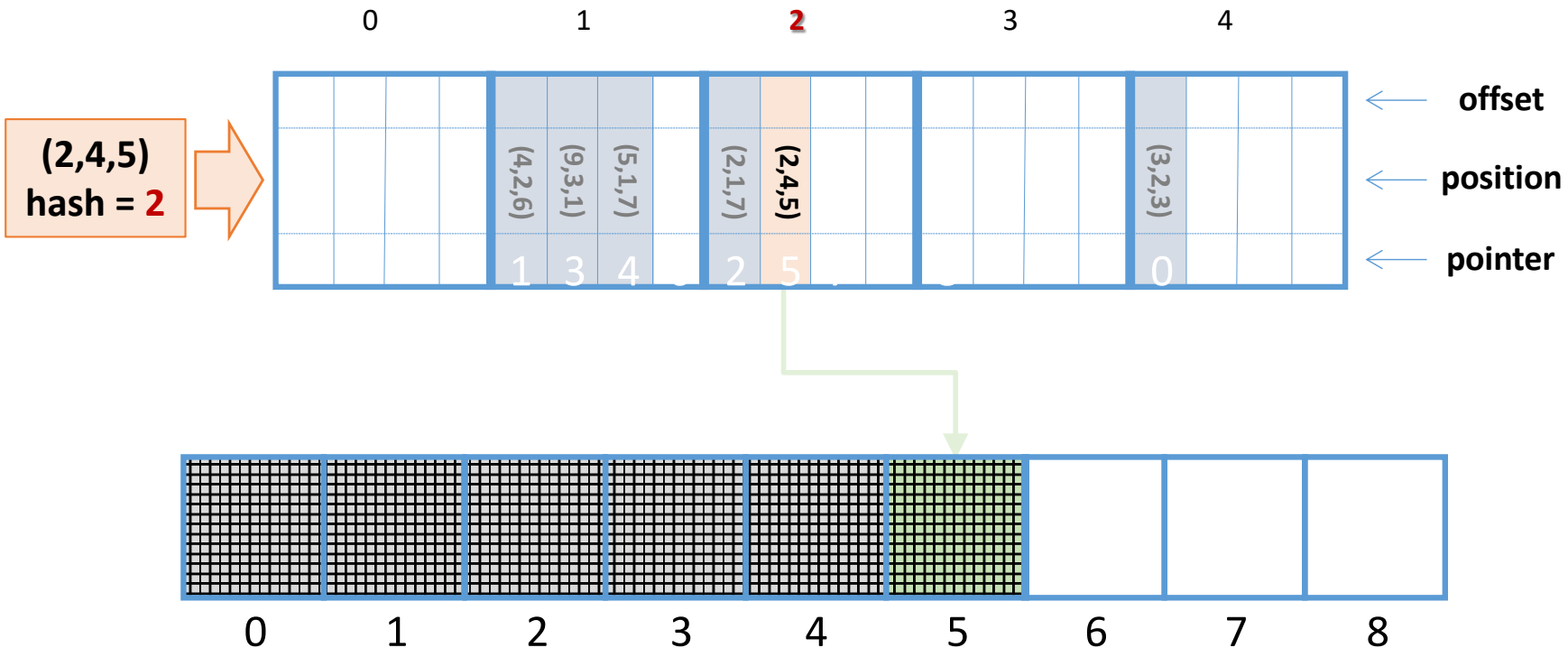
- Insert



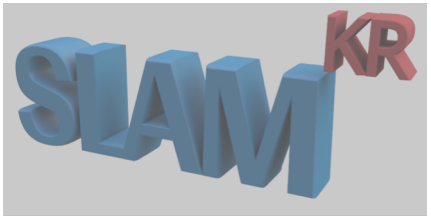
Hashing Operation



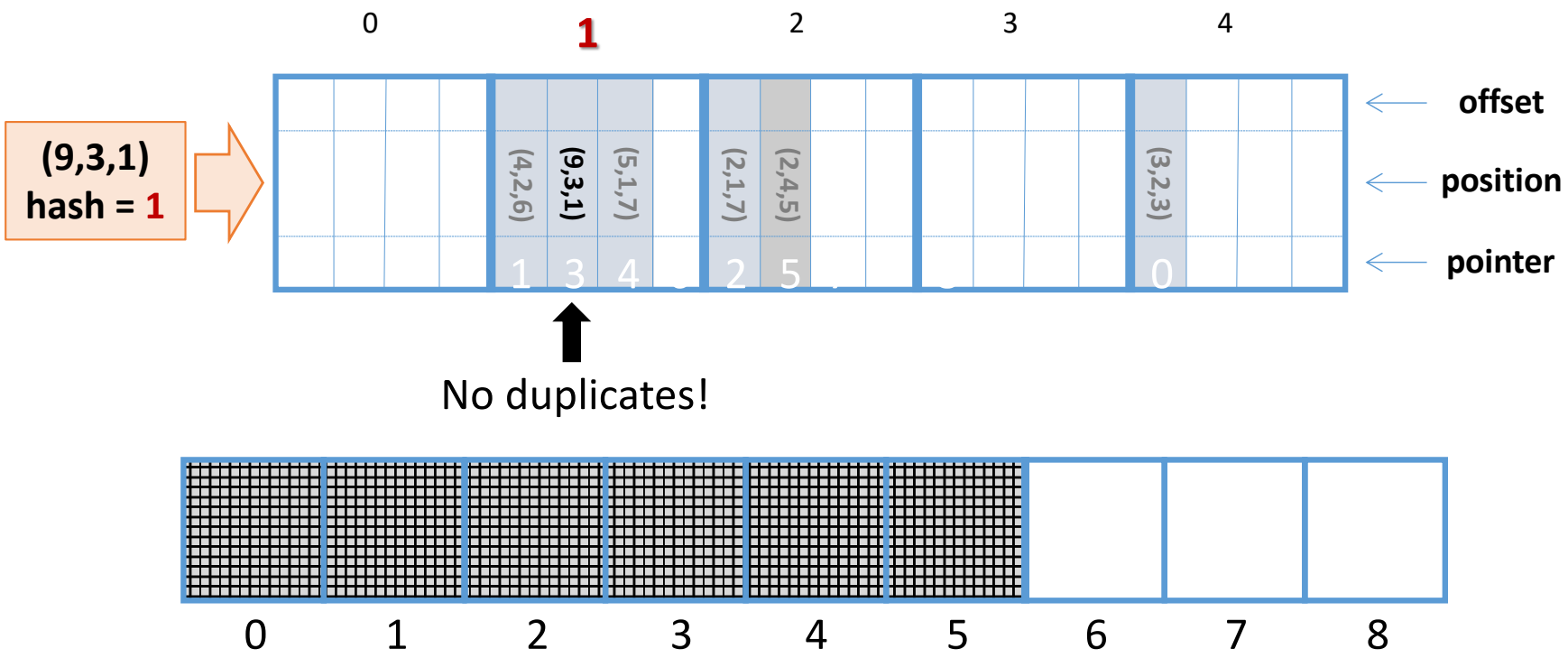
- Insert



Hashing Operation

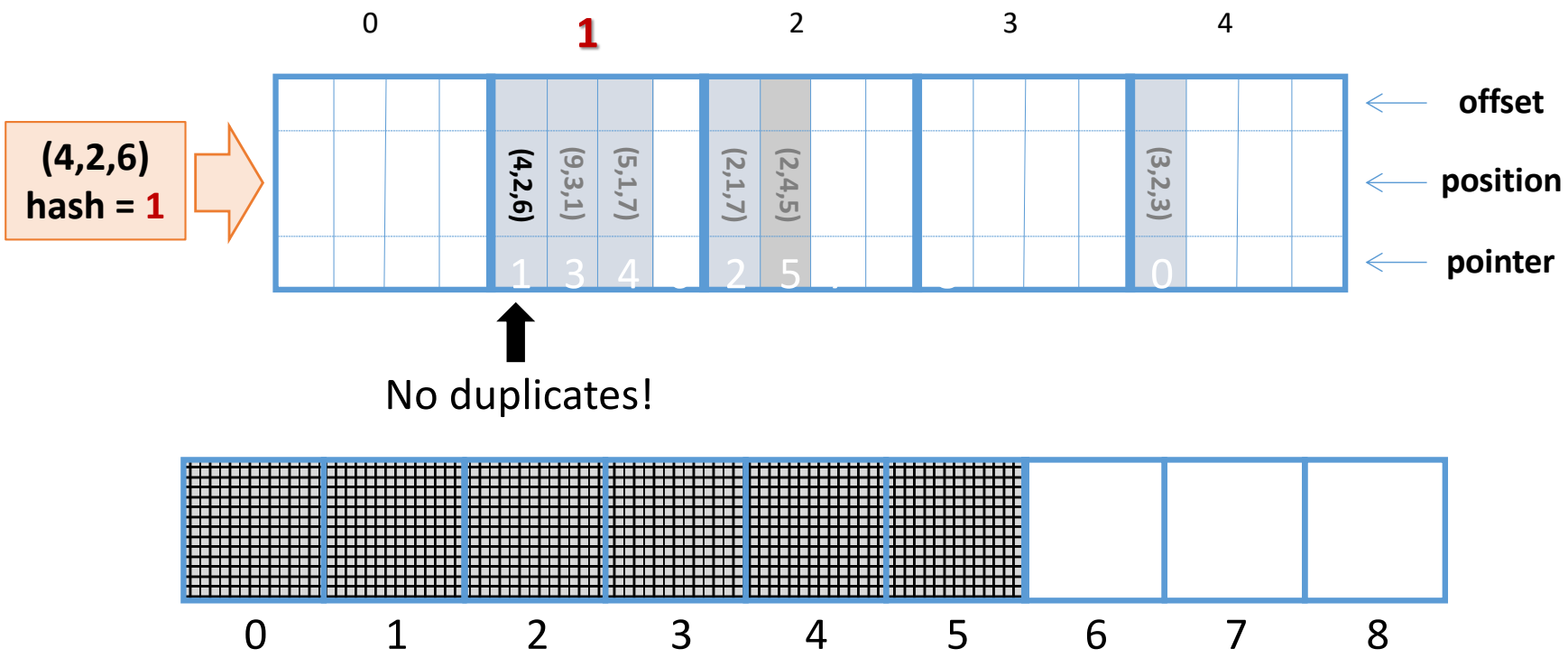


- Double insert

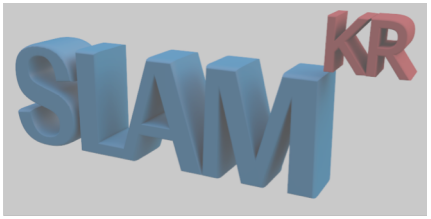


Hashing Operation

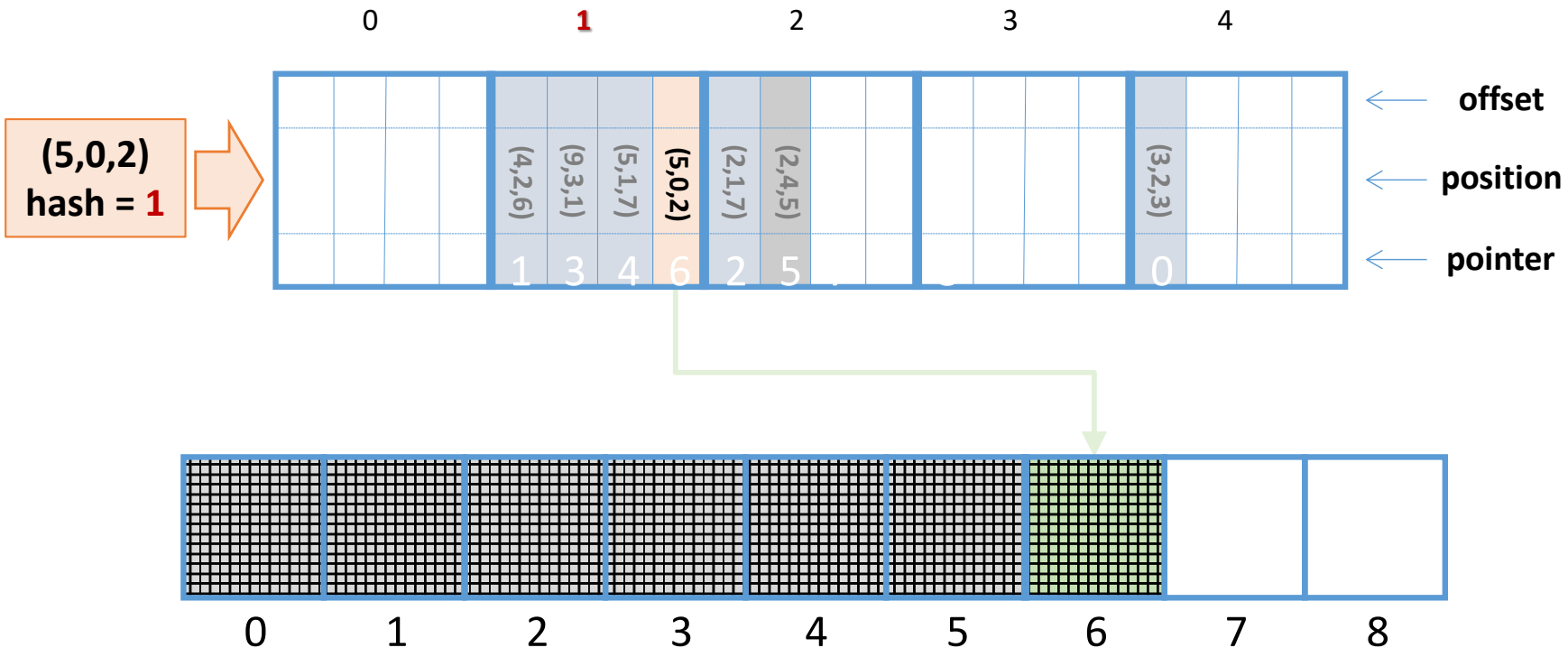
- Double insert



Hashing Operation

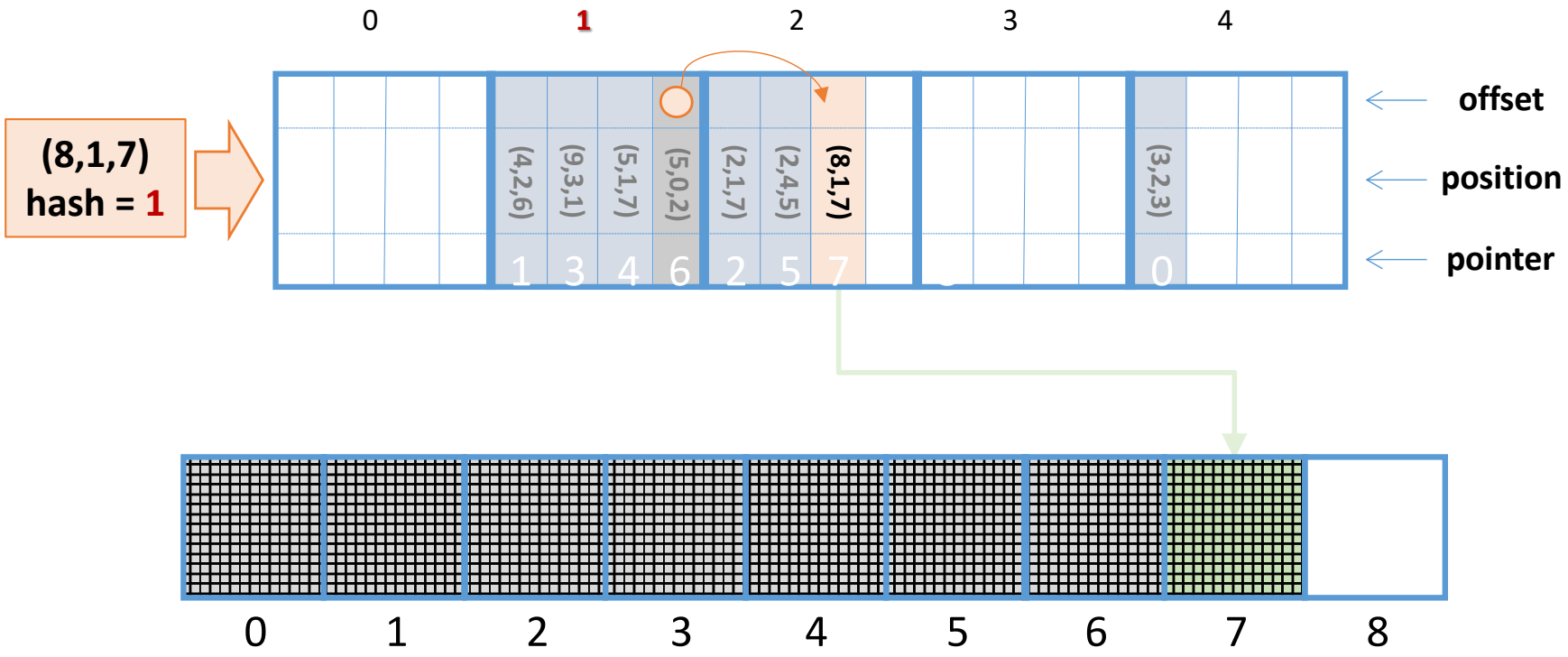


- Insert

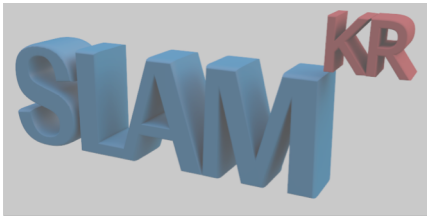


Hashing Operation

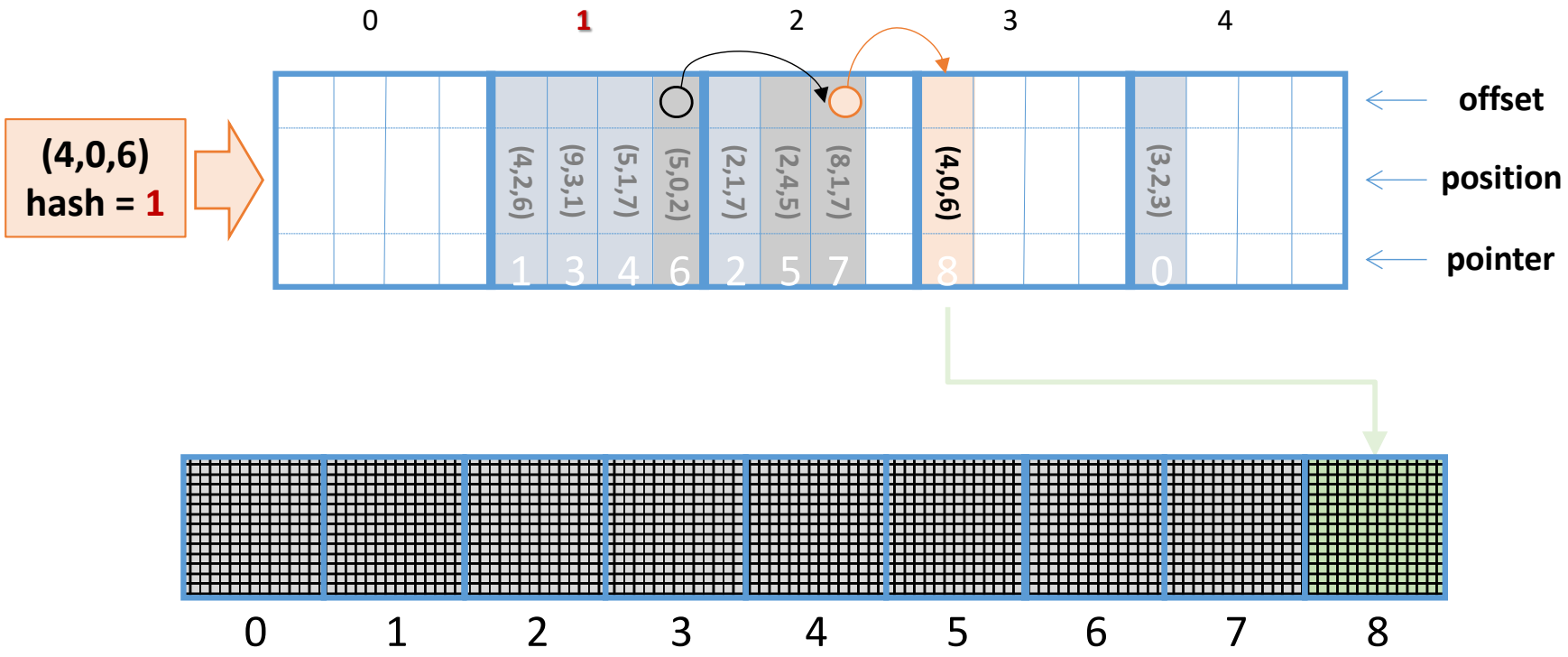
- Insert



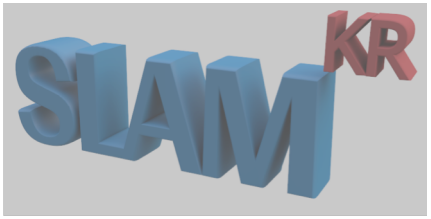
Hashing Operation



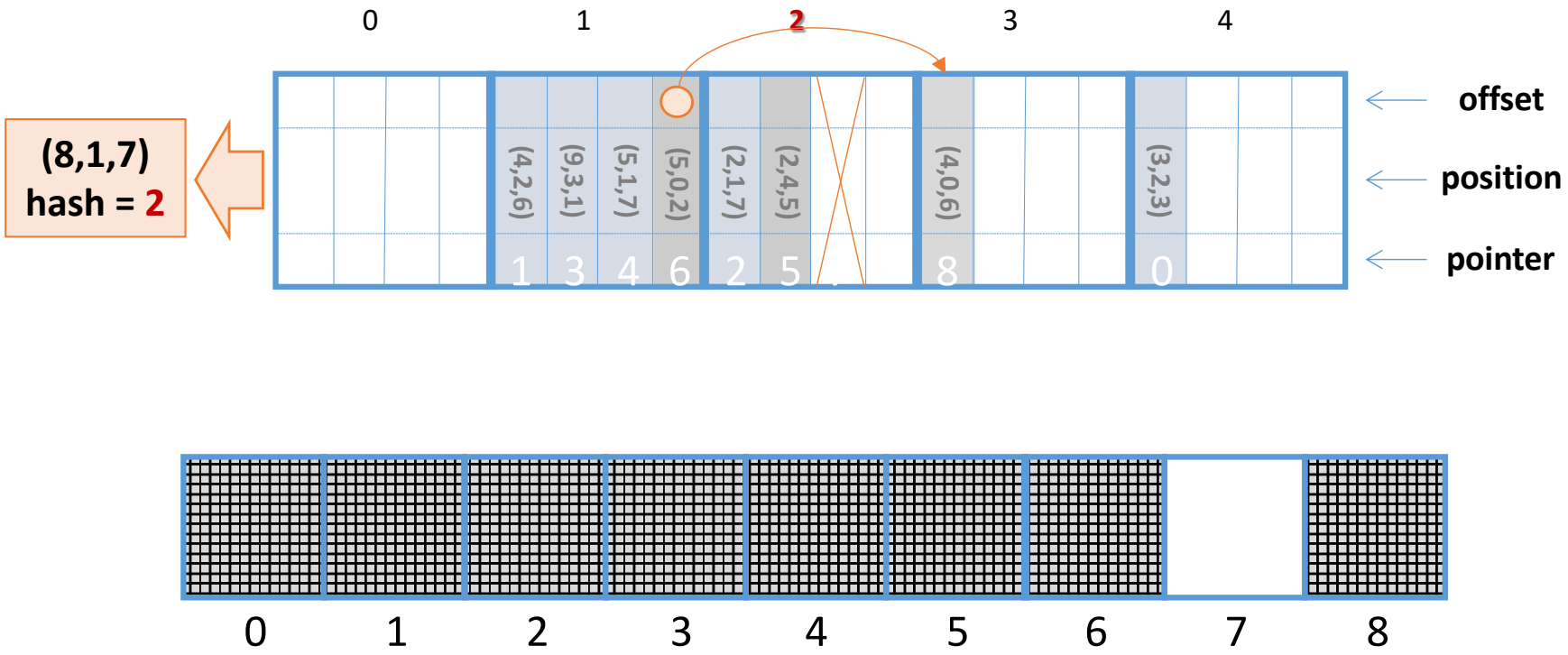
- Insert



Hashing Operation

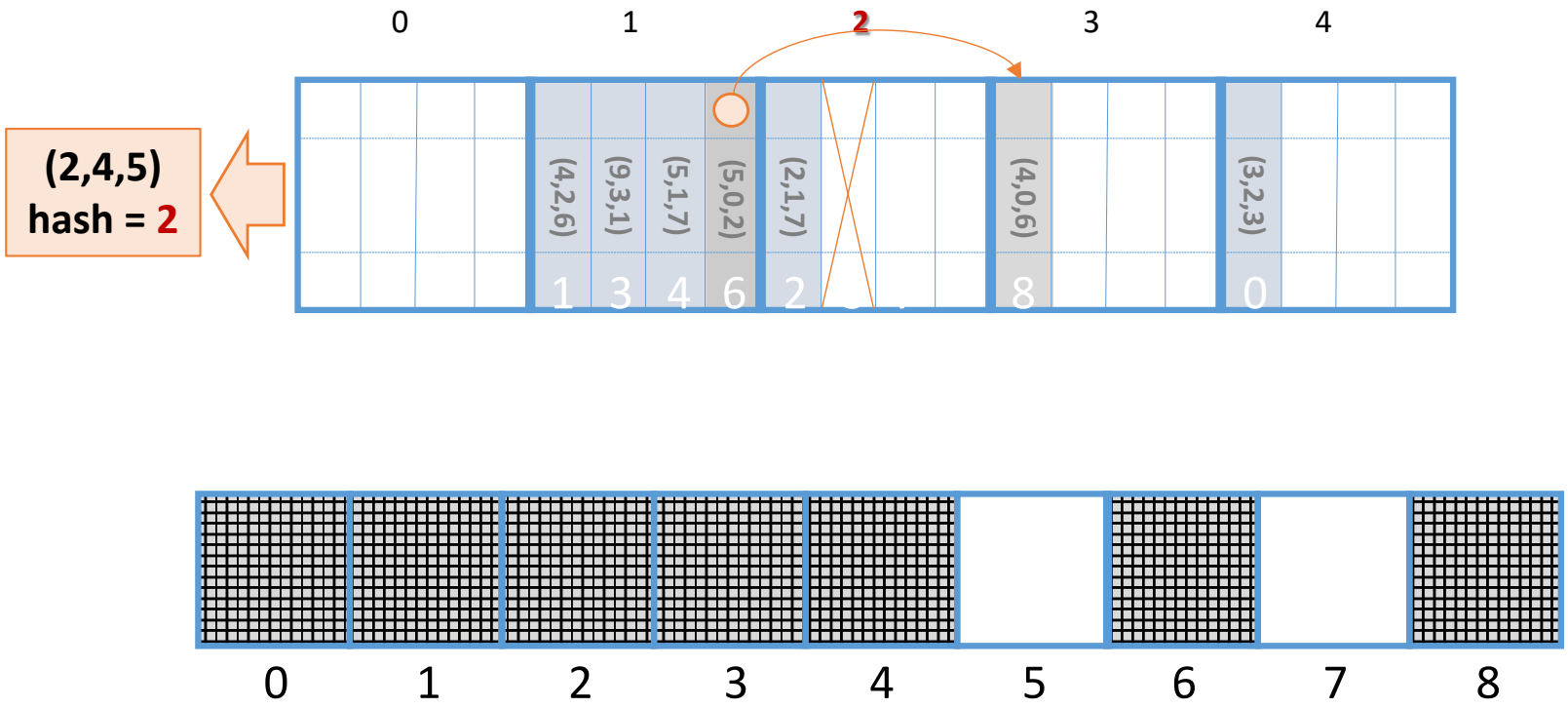


- Remove



Hashing Operation

- Remove



Thank you