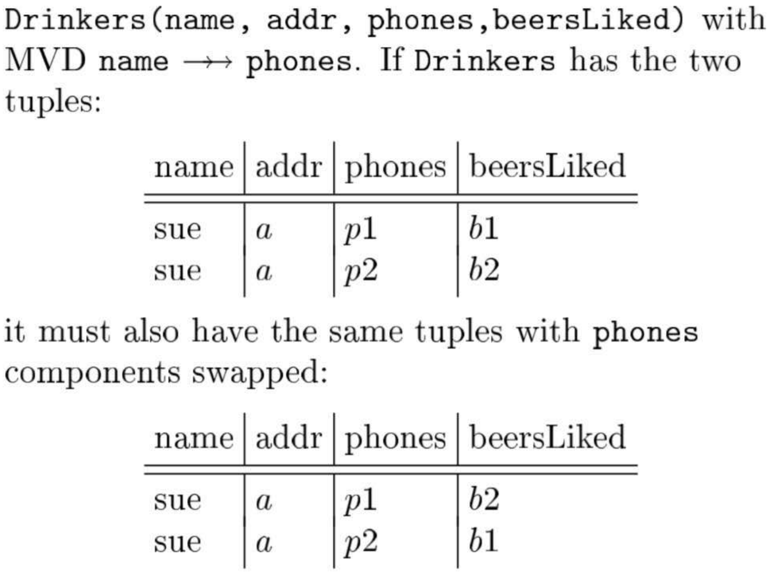
DBS Assignment V

1. **How to check whether α →→ β holds on R? Also, give 4NF decomposition algorithm.**

The multi-valued dependency α →→ β holds in a relation R whenever we have two tuples of R that agree in all attributes of α, then we can swap their β components and get two new tuples that are also in R.

For example:



`Drinkers(name, addr, phones,beersLiked)` with MVD `name` $\twoheadrightarrow$ `phones`. If `Drinkers` has the two tuples:

name	addr	phones	beersLiked
sue	a	$p1$	$b1$
sue	a	$p2$	$b2$

it must also have the same tuples with `phones` components swapped:

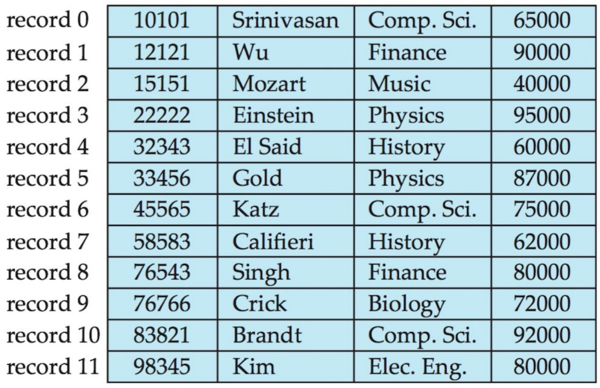
name	addr	phones	beersLiked
sue	a	$p1$	$b2$
sue	a	$p2$	$b1$

4NF decomposition algorithm:

```
result = {R}
done = NO

compute $D^+$; given Schema $R_i$, let $D_i$ denote the restriction of $D^+$ to $R_i$

while (not done):

    if (there is a schema $R_i$ in result that is not in 4NF wrt $D_i$):

        let $\alpha \twoheadrightarrow \beta$ be a nontrivial MVD that holds on $R_i$ such that
        $\alpha \rightarrow R_i$ is not in $D_i$, and $\alpha \cap \beta = \phi$

        result = (result − $R_i$) ∪ ($R_i$ − β) ∪ (α, β)

    else:
        done = YES
```

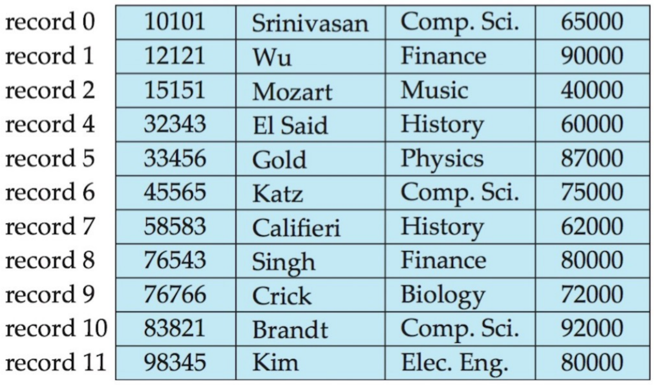
2. **Illustrate the different options available to delete a record from a file of fixed-length records with an example.**

The options available for deleting records from fixed length file records:



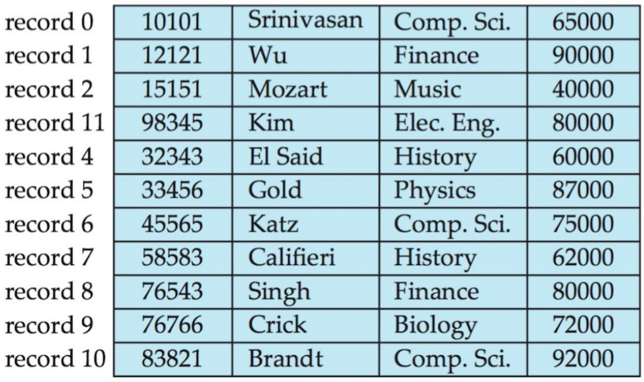
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

- **Shift records**



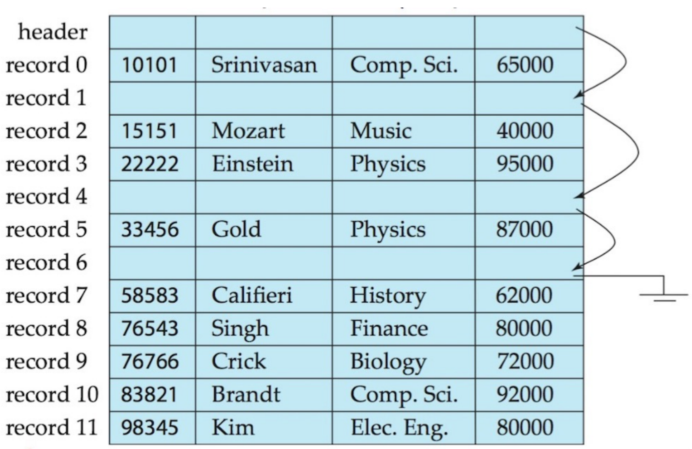
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

– Move last to the empty space



record 0	10101	Srinivasan	Comp. Sci.	65000
record 1	12121	Wu	Finance	90000
record 2	15151	Mozart	Music	40000
record 11	98345	Kim	Elec. Eng.	80000
record 4	32343	El Said	History	60000
record 5	33456	Gold	Physics	87000
record 6	45565	Katz	Comp. Sci.	75000
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000

– Keep links to all the free nodes



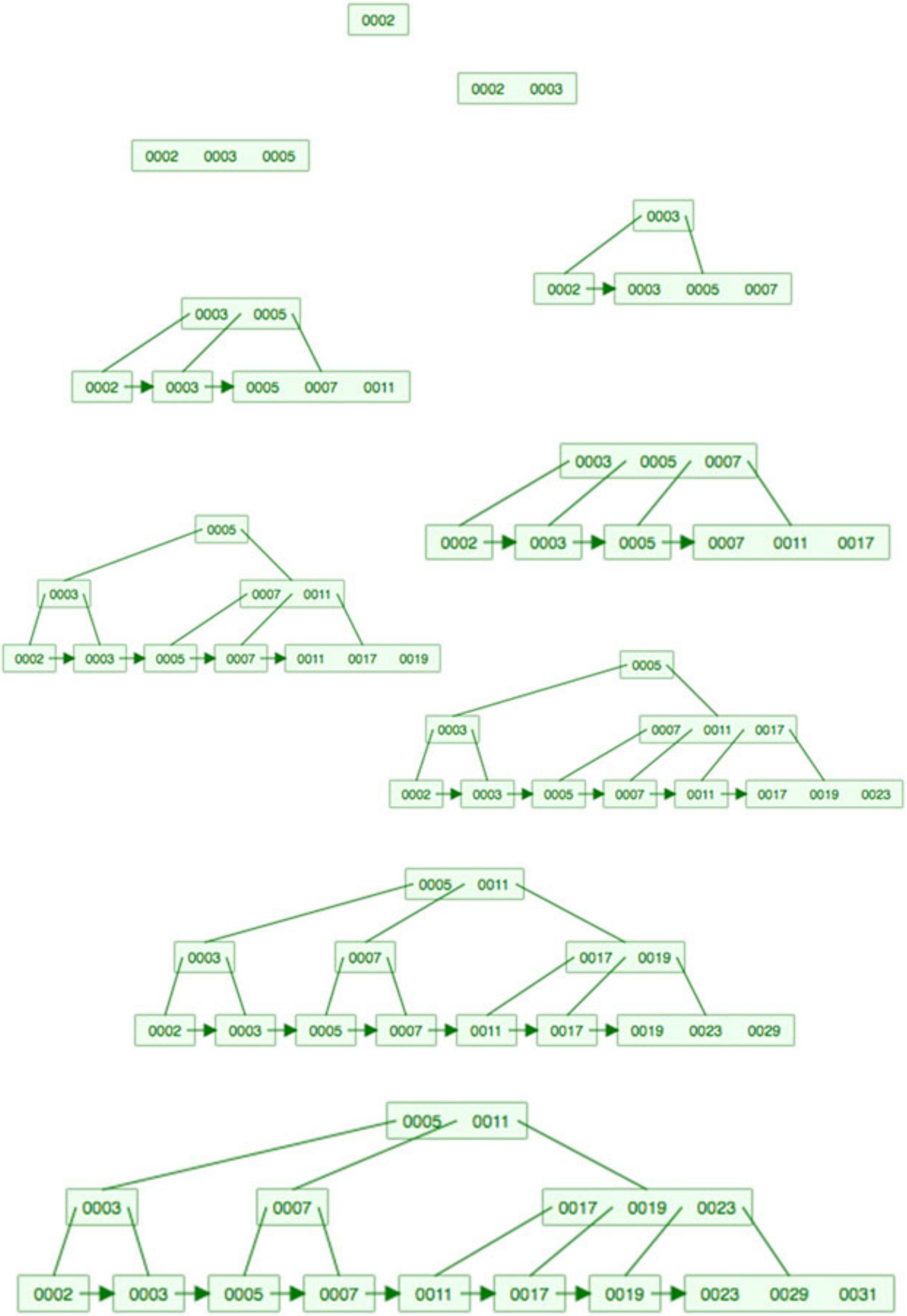
header				
record 0	10101	Srinivasan	Comp. Sci.	65000
record 1				
record 2	15151	Mozart	Music	40000
record 3	22222	Einstein	Physics	95000
record 4				
record 5	33456	Gold	Physics	87000
record 6				
record 7	58583	Califieri	History	62000
record 8	76543	Singh	Finance	80000
record 9	76766	Crick	Biology	72000
record 10	83821	Brandt	Comp. Sci.	92000
record 11	98345	Kim	Elec. Eng.	80000

3. **Construct a B+-tree for the following set of key values:**

`(2, 3, 5, 7, 11, 17, 19, 23, 29, 31 )`

Assume that the tree is initially empty and values are added in ascending order.

Assuming B value to be 4. (max leaf nodes = 4, max keys = 3)



Source

4.

A. **Compare B-Tree and B+ Tree with respect to their structure, advantages and disadvantages.**

B Tree	B+ Tree
Store data pointers on all the nodes (including interior ones).	Store data pointers on only the leaf nodes. Internal nodes only have keys.
Leaf nodes don't have links to their siblings.	Leaf nodes are linked to their siblings.
An scan of all keys requires a sort of a tree traversal through the entire tree.	A scan of all keys just requires just one pass of all the leaf nodes, hence efficient searching.
Occupies less size, cause no key is repeated.	Occupies more space, primary keys are repeated, more number of leaf nodes.
Possible to find the value (sometimes) for a given key before reaching the leaf.	Since all the values are on the leaf level, have to reach leaf nodes to find a value.
Non leaf nodes are larger, so fan-out is reduced.	Non leaf nodes only have keys, so fan-out is increased.
Implementation is much more difficult than B+ trees. Insertion and deletion is a complex process.	Much easier implementation, Insertion and deletion is simpler.

B. **What type of queries could be retrieved using multiple key access indices? Discuss why such queries cannot be accessed using single index.**

Queries where there are more than one predicates on non-primary keys can be retrieved using multiple key indices.

For example, say we've to find the IDs of all the TVShows with channel = 'HBO' and rating ≥ 9.0, primary key bring ID.

It can be done by single index queries on channel = 'HBO' and check for rating ≥ 9.0, or on rating ≥ 9.0 and then checking for channel = 'HBO'. These types of queries are slow, cause the index has to go through every record.

Or by using both of them independently as indices and taking the intersection, which uses multiple key access.

To speed up multiple search key queries, special structures can be maintained. Such queries will still work if there are multiple shows airing on 'HBO' and lots of shows with rating ≥ 9.0.

5.

A. **Explain the three choices when using Static Hashing for a database.**

In a static hashing, we obtain the bucket of a record directly from its search-key value using a hash function. Hash function *h* is a function from the set of all search-key values *K* to the set of all bucket addresses *B*. Hash function is used to locate addresses for access, insertion and deletion.

Records with different search key values may be mapped to the same bucket, thus the entire bucket has to be searched sequentially to locate a record.

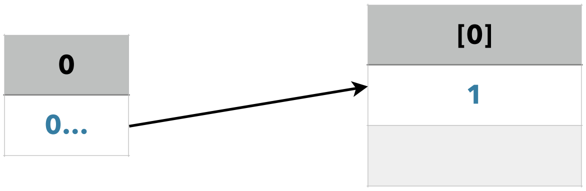
In cases when a bucket has fulfilled it's capacity, overflow buckets are used to attach more records (in a linked-list fashion) to that bucket.

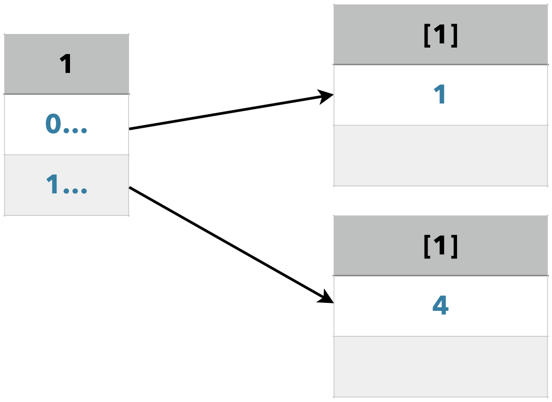
An ideal hash function is random, so each bucket will have almost the same number of records assigned to it. Bad hash functions maps major of the search keys to a single bucket, leaving most of the buckets empty.

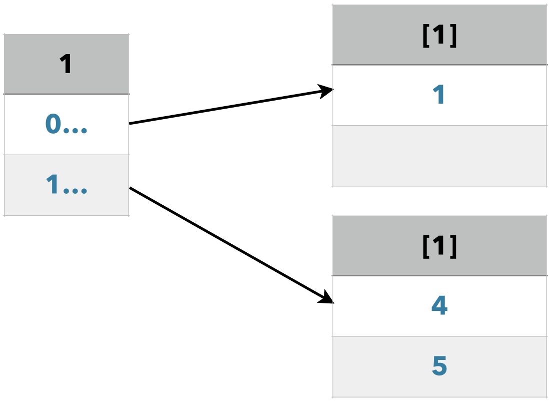
Since the hash function cannot change in static hashing, we have to choose a proper hash function for the mapping.

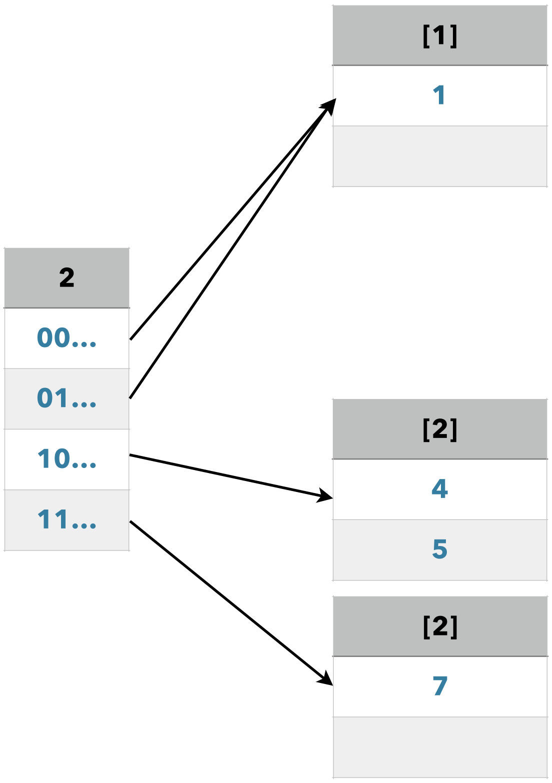
B. **Suppose the hash function is h(x) = x mod 8 and each bucket can hold at most two records. Show the form of extendable hash structure after each of the following insertion:**

`1, 4, 5, 7, 8, 2, 20.`

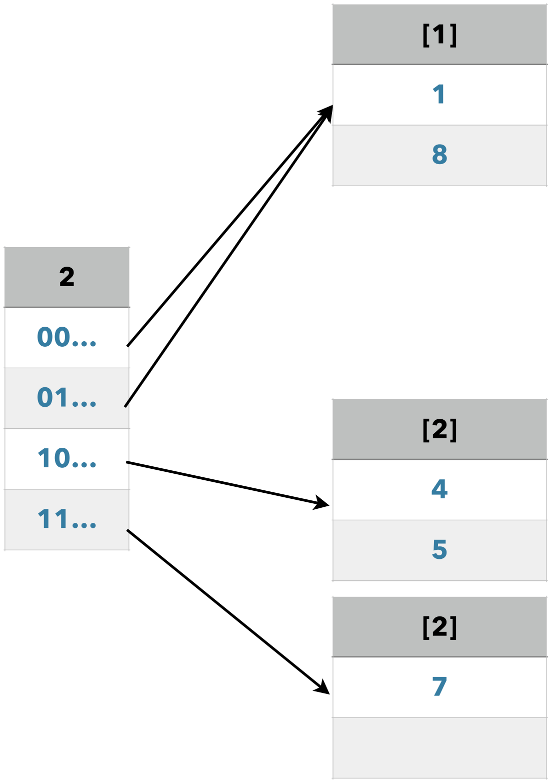




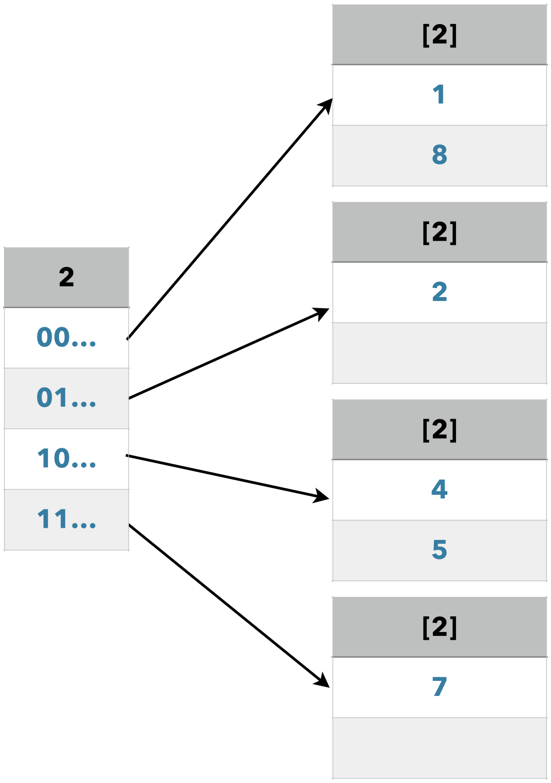




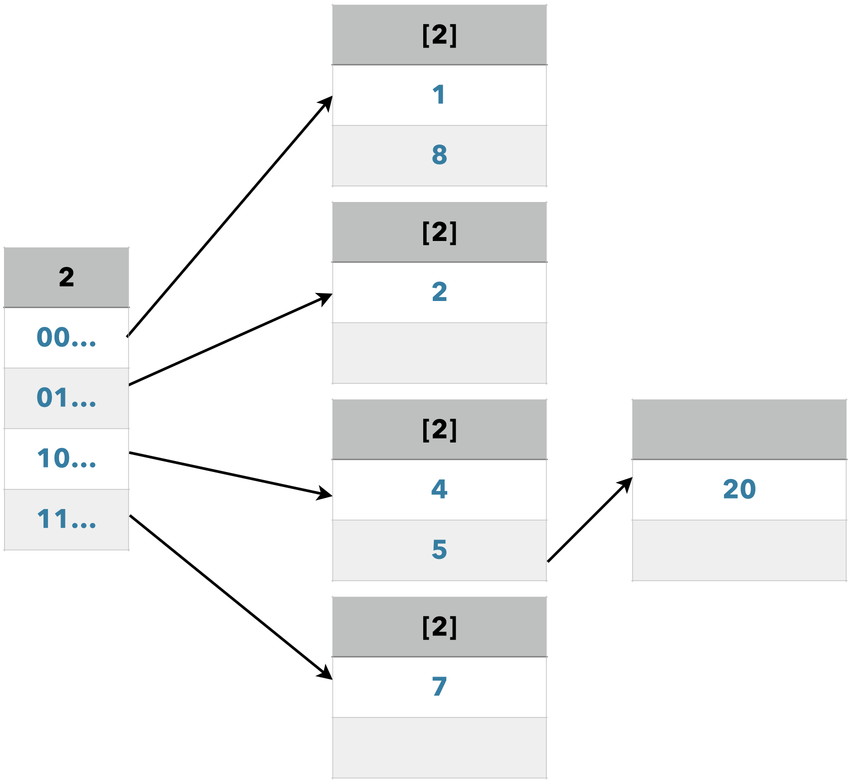
2
00...
01...
10...
11...
[1]
1
[2]
4
5
[2]
7



2
00...
01...
10...
11...
[1]
1
8
[2]
4
5
[2]
7



2
00...
01...
10...
11...
[2]
1
8
[2]
2
[2]
4
5
[2]
7



2
00...
01...
10...
11...
[2]
1
8
[2]
2
[2]
4
5
20
[2]
7