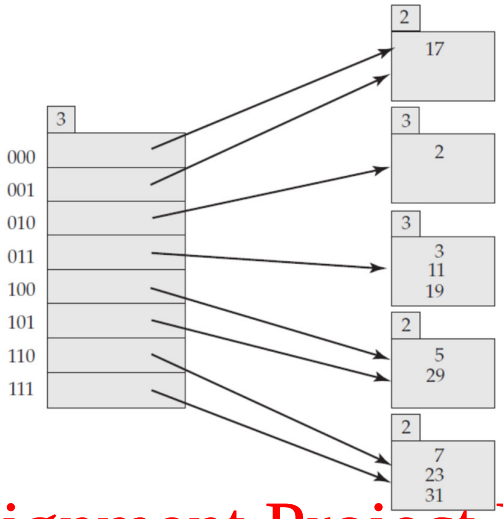
1. Suppose that we are using extendable hashing on a file that contains records with the following search-key values:

2, 3, 5, 7, 11, 17, 19, 23, 29, 31

Show the extendable hash structure for this file if the hash function is $h(x) = x \bmod 8$ and buckets can hold three records.



2.

(a) Consider a record to be deleted from an extendable hash structure.

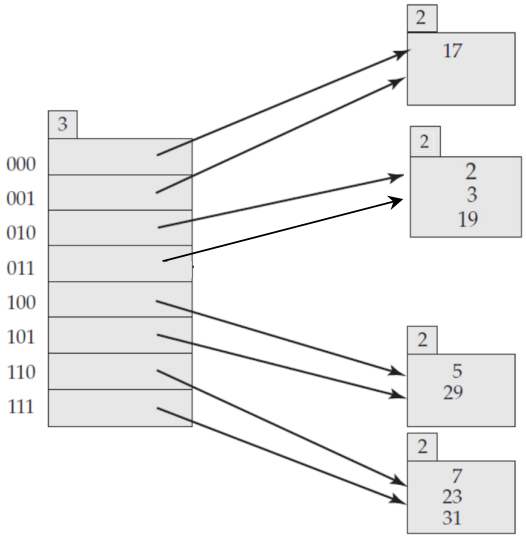
(i) With which bucket can this bucket be coalesced?

(ii) Under which conditions the buckets can be coalesced?

(i) The bucket can be coalesced with its buddy bucket differing from it only at the last bit of the common hash prefix. Suppose the record is deleted from bucket j. Its buddy bucket k has the first $(i_j - 1)$ bits same as that of bucket j while the bit i_j is reversed.

(ii) The conditions are (1) $i_j = i_k$, i.e., the common hash prefix of its buddy bucket is i_j, and (2) the total number of records can be stored in one bucket.

(b) Show how the extendable hash structure in Question 1 changes as the result of deleting 11. Coalesce buckets if possible.



3. Suppose that a secondary B^+-tree index on *building* is available on relation *department*, and that no other index is available. Discuss different ways to process the following selections.

(a) $\sigma_{\neg (building = \text{“Watson”})}(department)$

For this query, the index serves no purpose. We can scan the file sequentially and select all tuples whose *building* field is anything other than “Watson”.

(b) $\sigma_{\neg (building < \text{“Watson”})}(department)$

Use the index to locate the first tuple whose *building* field has value greater than or equal to “Watson”. From this tuple, follow the pointer chains till the end, retrieving all the tuples.

(c) $\sigma_{\neg (building < \text{“Watson”} \vee budget < 5000)}(department)$

This query is equivalent to the query:

$$\sigma_{\text{building} \geq \text{“Watson”} \wedge budget \geq 5000}(department)$$

Using the *building* index, we can retrieve all tuples with *building* value greater than or equal to “Watson” by following the pointer chains from the first “Watson” tuple. We also apply the additional criteria of *budget* ≥ 5000 on every tuple.

4. Suppose that an *employee* file has 10,000 records stored in 2,000 contiguous disk blocks and the following indices.

- A 3-level B^+-tree primary index on *salary*
- A 2-level B^+-tree secondary index on *dept_no*

(a) Explain your choice of algorithm in evaluating the following selection, assuming that $t_S = 4$ms and $t_T = 0.1$ms.

$$\sigma_{salary > 9{,}000}\,(employee)$$

Using linear search, cost estimate
$= t_S + 2{,}000\ t_T$
Using primary index, cost estimate
$= 3 \times (t_S + t_T) + t_S + \lceil (s \times 10{,}000)/5 \rceil \times t_T$, where s is the percentage of matching records.
So, linear search should be chosen if
$3 \times (t_S + t_T) + t_S + \lceil (s \times 10{,}000)/5 \rceil \times t_T > t_S + 2{,}000\ t_T$
$s > 0.9385$, i.e., number of matching records > 9,385

(b) Suppose 80% of the employee has salary over 9,000 and the department number of 1% of the employees is greater than 198. Explain your choice of algorithm in evaluating the following selection, assuming that $t_S = 4$ms and $t_T = 0.1$ms.

$$\sigma_{salary > 9{,}000\ \text{AND}\ dept\_no > 198}\,(employee)$$

Using linear search, cost estimate
$= t_S + 2{,}000\ t_T = 204$ms
Using the condition (*salary* > 9000) first, cost estimate
$= 3 \times (t_S + t_T) + t_S + (2000 \times 0.8) \times t_T$
$= 176.3$ ms
Using the condition (*dept_no* > 198) first, cost estimate
$= (2 + (10{,}000 \times 0.01)) \times (t_S + t_T)$
$= 418.2$ ms

So, we should choose to use the primary index on *salary* because it has the lowest cost estimate. The condition (*salary* > 9000) is used to retrieve the records, the remaining part of the conjunctive condition (*dept_no* > 198) is checked for each selected record after it is retrieved into memory. Only the records that satisfy the additional condition are included in the result of the query.