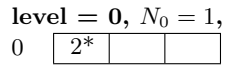
CS3223 Tutorial 2: Hash-based Indexing

Wk 4, Sem 2, 2022/23

Questions to be discussed: 1, 2, 3 & 4

1. Consider the Linear Hashing index shown below



Linear Hashing index with one data bucket

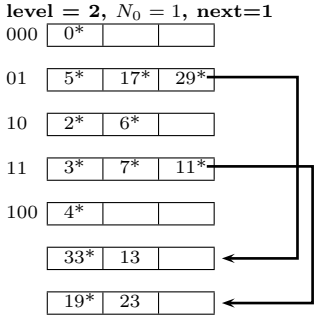
where each data bucket can store 3 data entries. Show the Linear Hashing index after inserting the following sequence of data entries:

$$3*,\ 5*,\ 7*,\ 11*,\ 17*,\ 19*,\ 23*,\ 0*,\ 4*,\ 29*,\ 33*,\ 13*,\ 6*$$

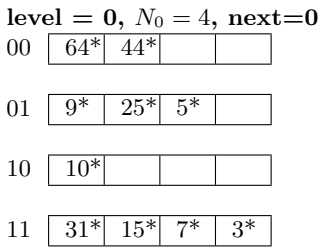
Assume that a bucket split is triggered whenever some bucket overflows.

Solution: Note that 2= 10, 3=11, 4 = 100, 5=101, 6=110, 7=111, 11=1011, 13=1101, 17=10001, 19=10011, 23=10111, 29=11101, 33=100001.

1. Insert 3*: $B_0 = (2*, 3*)$
2. Insert 5*: $B_0 = (2*, 3*, 5*)$
3. Insert 7*: B_0 overflows and splits into $B_0 = (2*)$ and $B_1 = (3*, 5*, 7*)$. *level* is incremented to 1. *next* is reset to 0.
4. Insert 11*: B_1 overflows and an overflow page $P_1 = (11*)$ is created. B_0 is split into $B_{00} = ()$ and $B_{10} = (2*)$. *next* is incremented to 1.
5. Insert 17*: $P_1 = (11*, 17*)$.
6. Insert 19*: $P_1 = (11*, 17*, 19*)$.
7. Insert 23*: P_1 overflows and B_1 (and its overflow page P_1) is split into $B_{01} = (5*, 17*)$ and $B_{11} = (3*, 7*, 11*)$ (with overflow page $P_1 = (19*, 23*)$). *level* is incremented to 2. *next* is reset to 0.
8. Insert 0*: $B_{00} = (0*)$.
9. Insert 4*: $B_{00} = (0*, 4*)$.
10. Insert 29*: $B_{01} = (5*, 17*, 29*)$.
11. Insert 33*: B_{01} overflows and an overflow page $P_2 = (33*)$ is created. B_{00} is split into $B_{000} = (0*)$ and $B_{100} = (4*)$. *next* is incremented to 1.
12. Insert 13*: $P_2 = (33*, 13*)$.
13. Insert 6*: $B_{10} = (2*, 6*)$.



2. (Exercise 11.9, R&G) Consider the snapshot of the Linear Hashing index shown below.



Assume that a bucket split is triggered whenever some bucket overflows.

(a) What is the maximum number of data entries that can be inserted (given the best possible distribution of keys) before you have to split a bucket? Explain very briefly.

(b) Show the file after inserting a *single* record whose insertion causes a bucket split.

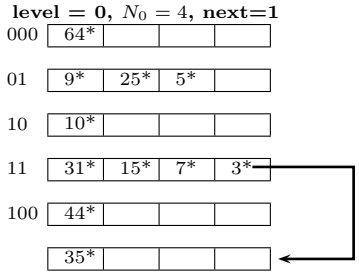
(c) Consider the original Linear Hashing index shown.
(i) What is the minimum number of record insertions that will cause a split of all four buckets? Explain very briefly.
(ii) What is the value of `next` after making these insertions?
(iii) What can you say about the number of pages in the fourth bucket shown after this series of record insertions?

Solution: Note that 3 = 0000011, 5 = 0000101, 7 = 0000111, 9 = 0001001, 10 = 0001010, 15 = 0001111, 25 = 0011001, 31 = 0011111, 44 = 0101100, 64 = 1000000.

(a) Insert 6 entries to fill up all the buckets. The 7th insertion causes an overflow and a bucket split.

(b) Inserting a record into B_{11} leads to an overflow and a split.

For example, the following index assumes that an entry with hashed value 35 (100011_2) is inserted.



(c) i To cause all the buckets to split, the insertions need to result in at least four bucket overflows.

To minimize the number of insertions, there are two good strategies. The first strategy is to always insert into the most fully occupied bucket. This would suggest that the first insertion should go into bucket B_{11} to trigger the split of B_{00}. However, since B_{11} is not the next bucket to be split, its newly created overflow page would have only a single entry which means that it would require four further insertions into B_{11} before it overflows again. Thus, although the first strategy minimizes the number of insertions to overflow a bucket, it does not necessarily maximize the occupancy of the overflowed bucket after it has been split.

To maximize the occupancy of an overflowed bucket, a second strategy is to target insertions into the next bucket to be split. This would suggest inserting the first three entries into B_{00} to overflow and trigger the split of B_{00}. By choosing the newly inserted values such that they remain in B_{000} after the split, only one entry in B_{00} (i.e., 44^*) is redistributed to B_{100}. Thus, this results in B_{000} being a fully occupied

bucket which can overflow again with just a single further insertion. Assuming that the newly inserted entries into B_{00} are $\{a^*, b^*, c^*\}$, we have $B_{000} = (64^*, a^*, b^*, c^*)$ and $B_{100} = (44^*)$.

We now apply the second strategy again by inserting two entries d^* and e^* into B_{01} to overflow and trigger the split of B_{01}. By choosing d^* and e^* such that they remain in B_{001} after the split, we have $B_{001} = (9^*, 25^*, d^*, e^*)$ and $B_{101} = (5^*)$.

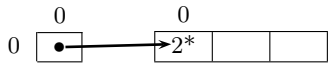
To trigger the two remaining bucket splits, we just need to insert a single entry each into any two of the three fully occupied buckets (i.e, B_{000}, B_{001}, B_{11}).

Thus, the minimum number of required insertions = 3+2+1+1=7.

ii Next will be reset to 0.

iii When B_{11} splits, B_{011} has at least 1 entry (3^*) and B_{111} has at least 3 entries ($7^*, 15^*, 31^*$). Depending on whether a new entry was inserted into bucket B_{11}, the fourth bucket B_{011} could have 1 entry or 2 entries.

3. Consider the Extendible Hashing index shown below



Extendible Hashing index with one data bucket

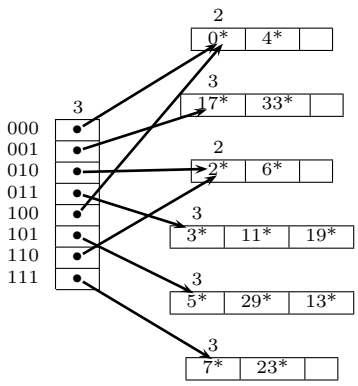
where each data bucket can store 3 data entries. Show the Extendible Hashing index after inserting the following sequence of data entries:

$$3*,\ 5*,\ 7*,\ 11*,\ 17*,\ 19*,\ 23*,\ 0*,\ 4*,\ 29*,\ 33*,\ 13*\ 6*$$

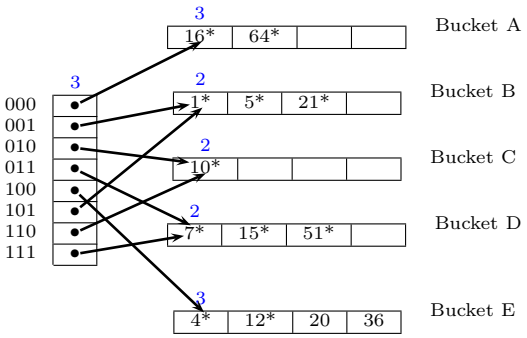
Note that k* denotes a data entry having a hashed value of k.

Solution: Note that 2= 10, 3=11, 4 = 100, 5=101, 6=110, 7=111, 11=1011, 13=1101, 17=10001, 19=10011, 23=10111, 29=11101, 33=100001.

1. Insert 3*: $B_0 = (2*, 3*)$
2. Insert 5*: $B_0 = (2*, 3*, 5*)$
3. Insert 7*: B_0 overflows and splits into $B_0 = (2*)$ and $B_1 = (3*, 5*, 7*)$. The local depths of B_0 and B_1 are both 1. The directory is doubled to 2 entries. The global depth is incremented to 1.
4. Insert 11*: B_1 overflows and splits into $B_{01} = (5*)$ and $B_{11} = (3*, 7*, 11*)$. The local depths of B_{01} and B_{11} are both 2. The directory is doubled to 4 entries. The global depth is incremented to 2. Directory entries 00 and 10 point to B_0.
5. Insert 17*: $B_{01} = (5*, 17*)$
6. Insert 19*: B_{11} overflows and splits into $B_{011} = (3*, 11*, 19*)$ and $B_{111} = (7*)$. The local depths of B_{011} and B_{111} are both 3. The directory is doubled to 8 entries. The global depth is incremented to 3. Directory entries 000, 010, 100, and 110 point to B_0. Directory entries 001 and 101 point to B_{01}.
7. Insert 23*: $B_{111} = (7*, 23*)$.
8. Insert 0*: $B_0 = (2*, 0*)$.
9. Insert 4*: $B_0 = (2*, 0*, 4*)$.
10. Insert 29*: $B_{01} = (5*, 17*, 29*)$
11. Insert 33*: B_{01} overflows and splits into $B_{001} = (17*, 33*)$ and $B_{101} = (5*, 29*)$. The local depths of B_{001} and B_{101} are both 3.
12. Insert 13*: $B_{101} = (5*, 29*, 13*)$.
13. Insert 6*: B_0 overflows and splits into $B_{00} = (0*, 4*)$ and $B_{10} = (2*, 6*)$. The local depths of B_{00} and B_{10} are both 2. Directory entries 000 and 100 point to B_{00}. Directory entries 010 and 110 point to B_{10}.



4. (Adapted from Exercise 11.1, R&G) Consider the Extendible Hashing index shown below which was created by inserting a sequence of 13 data entries without any deletions. Which bucket could be the last bucket that was split by the insertions?



Bucket	Base 10	Base 2
B	1	0000001
E	4	0000100
B	5	0000101
D	7	0000111
C	10	0001010
E	12	0001100
D	15	0001111
A	16	0010000
E	20	0010100
B	21	0010101
E	36	0100100
D	51	0110011
A	64	1000000

Solution: Note that if bucket X was the last bucket to be split with split image bucket Y, then (1) both X and Y must have the same local depth ℓ; (2) the last $\ell - 1$ bits of X & Y's bucket addresses must be the same; (3) the last ℓ^{th} bit of X and Y's bucket addresses are 0 and 1, respectively; and (4) the total number of entries in X & Y must be at least 5.

Therefore, there are two possibilities: bucket A could be the last bucket that was split to split image bucket E, and bucket B could be the last bucket that was split to split image bucket D.

Note that bucket C can't be the last bucket that was split since it has no feasible split image.

One could also observe from the following depiction of the creation of split images as the buckets are split (starting from an initial single bucket). In this binary tree representation, an internal node P represents a bucket before it was split, the left child node of P represents P after it was split, and the right child node of P represents the split image created by the split of P.