EECS 484 Homework 3 (80 points)
Due June 5th, 2024 at 11:55pm (EST)

Please read the following instructions before starting the homework.

Deliverables

You need to submit all your solutions in **a single PDF on Gradescope**. Your solutions can be either handwritten or created electronically, as long as they are clear.

For Question 1, 3 and 4, there are templates in the file *hw3_helper_templates.pdf* to help you draw your diagrams. You may choose to use templates, or develop your diagrams by yourself.

This homework is to be done in a group of 2.

No late days for homework! *If you miss the due date, you get 0 points. No exceptions on this.*

Honor Code

By submitting this homework, you are agreeing to abide by the Honor Code:

I have neither given nor received unauthorized aid on this assignment, nor have I concealed any violations of the Honor Code.

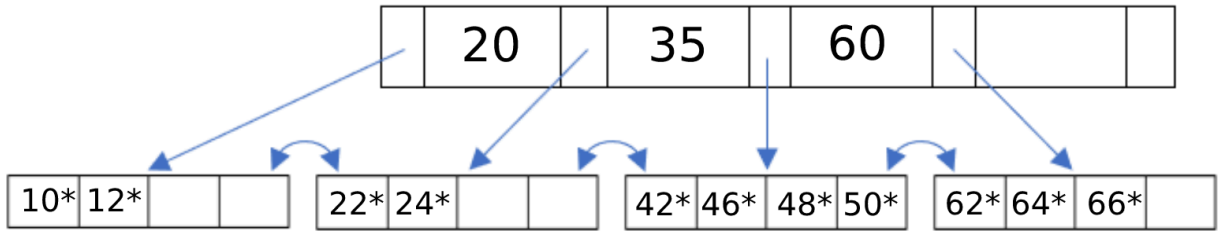
The file *HW4_templates.docx* includes several B+ tree structures that may be helpful for drawing the diagram. Whether you use them or not won't affect your HW score as long as your diagrams are clear and legible.

Q1: B+ Tree Operations (22 points)

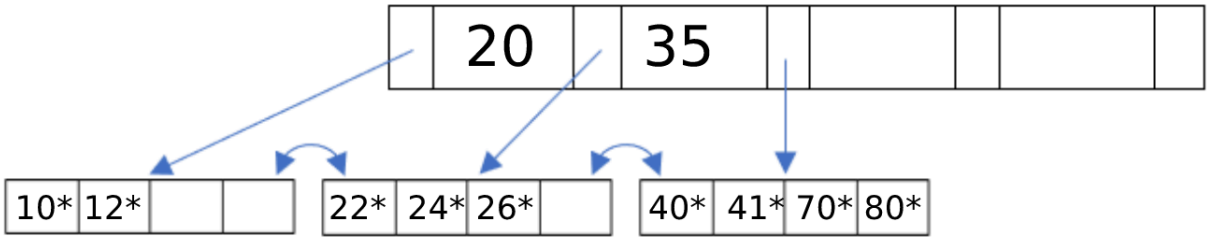
Assume these rules for B+ tree operations in this question (Q1):

- The order of both inner nodes and leaf nodes is 2.
- The left pointer points to values that are strictly less than (<) the key value.
- During redistribution, shift elements to/from the right node over the left when both are possible redistribution partners.
- During the splitting of a node, the right node will have one more value than the left node.
- For insertions, favor redistribution over splitting.
- For deletions, favor redistribution over merging.
- Only redistribute one entry at a time.

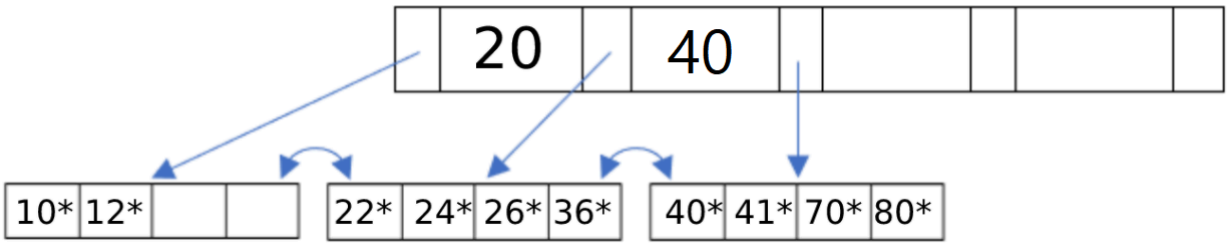
Q1.1 Given the tree below, draw the tree after inserting 58*



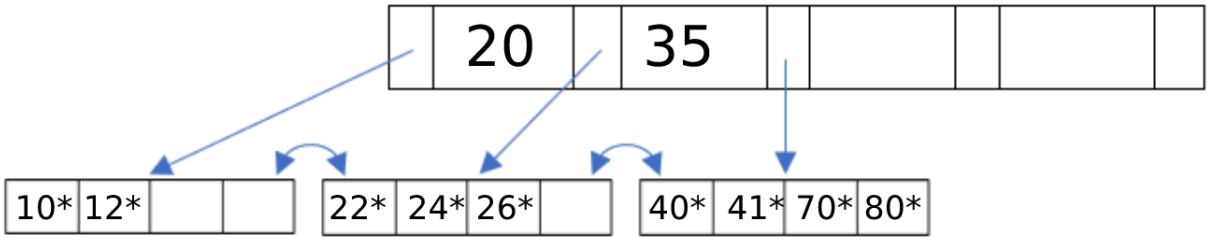
Q1.2 Given the tree below, draw the tree after inserting 25*, 30*



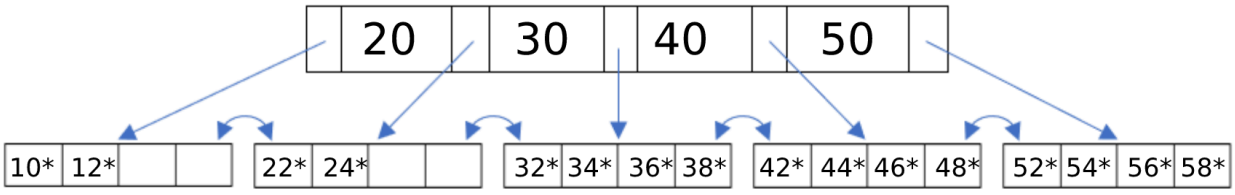
Q1.3 Given the tree below, draw the tree after inserting 60*



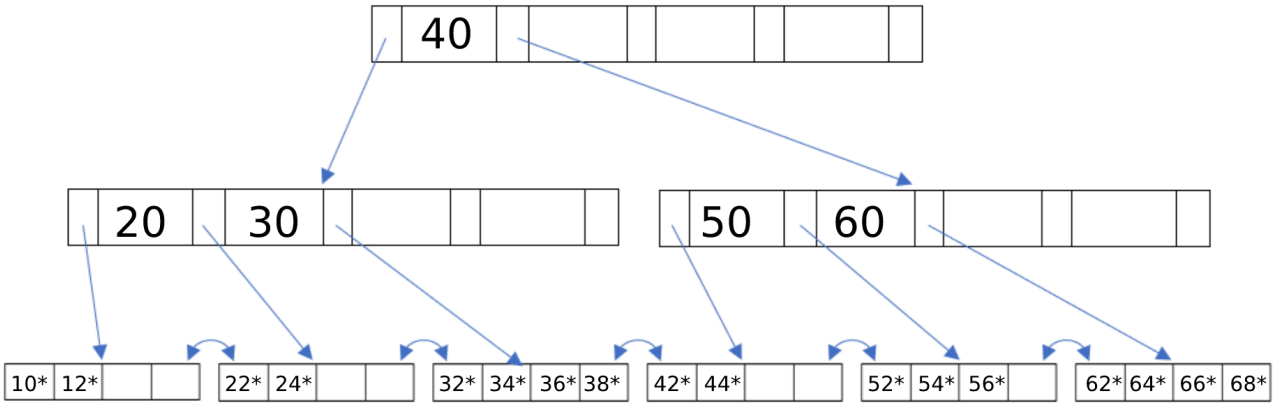
Q1.4 Given the tree below, draw the tree after deleting 12*



Q1.5 Given the tree below, draw the tree after inserting 47*



Q1.6 Given the tree below, draw the tree after deleting 44*, 52*



Q2. Clustering and IO cost (20 points)

Consider a table Sailors, where there is a B+ tree index on attribute Age, and the following query:

```
SELECT * FROM Sailors ORDER BY Age;
```

Furthermore, assume that the first level (root) of the B+ tree is kept in memory at all times, and that the B+ tree has the following properties:

- total size of records = 12MB
- size of record = 4KB
- size of page = 64KB
- both inner nodes and leaf nodes have order of 4
- the fill factor of B+ tree nodes is 75%
- the data record pages are kept full

Assume 1MB = 1024KB. Assume only the root node is in memory and nothing else is saved in memory. Assume that we try to fit all records in the smallest number of pages possible and each node is in its own page. Assume **Alternate 2** (as described in lecture) is used.

Answer the following questions:

2.1) What is the number of record pages and leaf nodes? What is the height of the B+ tree?

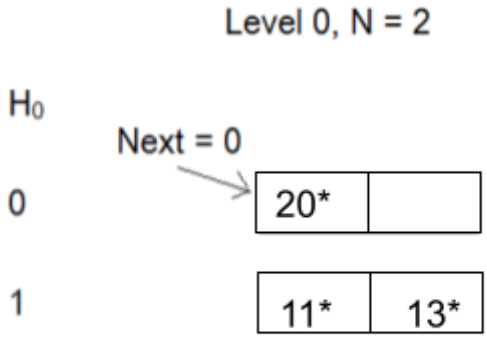
2.2) How many IOs (in the worst case) will it take to run this query if our index was clustered?

2.3) How many IOs (in the worst case) will it take to run this query if our index was unclustered?

The file *HW4_templates.docx* contains hashing bucket templates for Q3 and Q4. You may either use templates or draw completely by yourself.

Q3. Linear Hashing (20 points)

For Question 3, assume the split policy used is splitting when inserting into an overflow page. Assume we initialize the Linear Hashing index as follows:



The hash function that will be used:

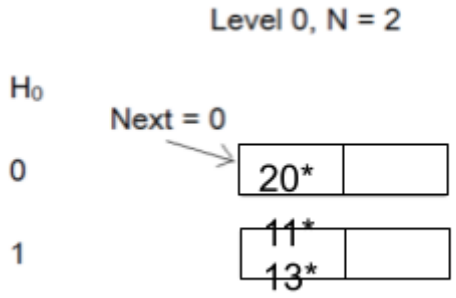
$$h_i(value) = value \bmod (2^i N)$$

Assume that we use the same algorithm discussed in the lecture.

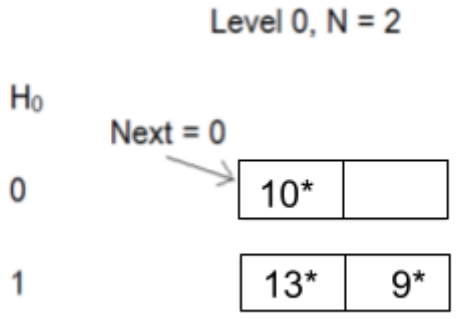
3.1) What is the minimum number of insertions needed such that the index structure has 4 buckets and is at Level = 1? Give an example insertion sequence for your answer.

3.2) For each subquestion, draw what the given index structure looks like after the listed insertion. Remember to show the “next” pointer and level post-insertion.

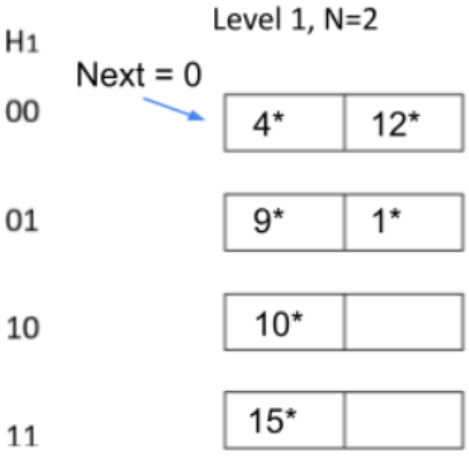
3.2.1) Insert 22 = $(10110)_2$



3.2.2) Insert 23 = $(10111)_2$

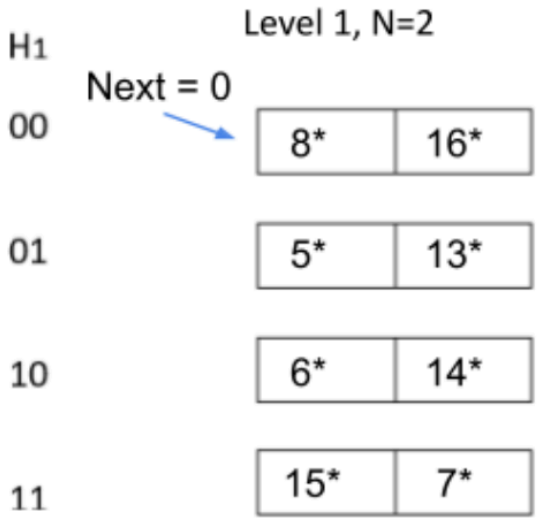


3.2.3) Insert 19 = $(10011)_2$



3.2.4) Insert 12 = $(01100)_2$

3.2.5) Insert 20 = $(10100)_2$



Q4. Extendible Hashing (18 points)

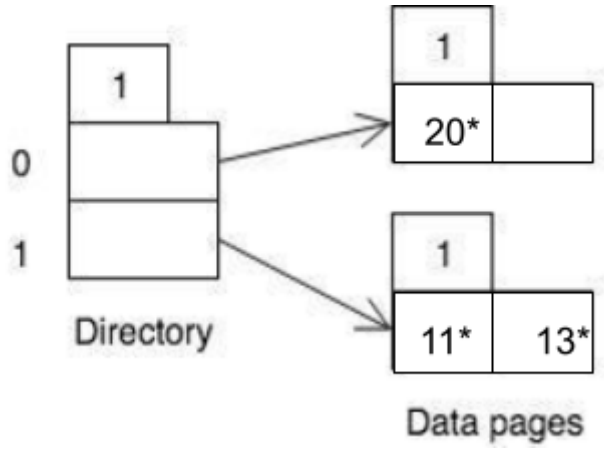
In Q3, we looked at linear hashing. What about extendible hashing? Suppose we have an extendible hash index that uses the hash function:

H(x) = last *d* bits of the binary representation of x, where *d* is the global depth.

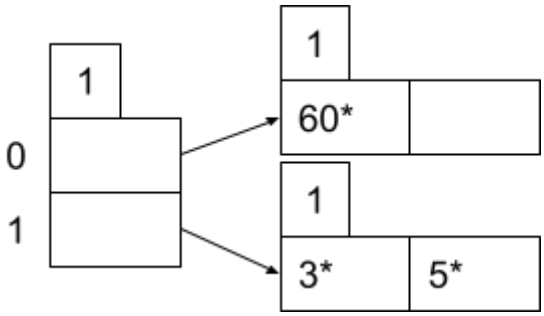
For the following subquestions, draw what the index structure looks like after each respective insertion. Be sure to clearly show the global and local depths and all bucket pointers.

Assume that we use the same algorithm discussed in the lecture.

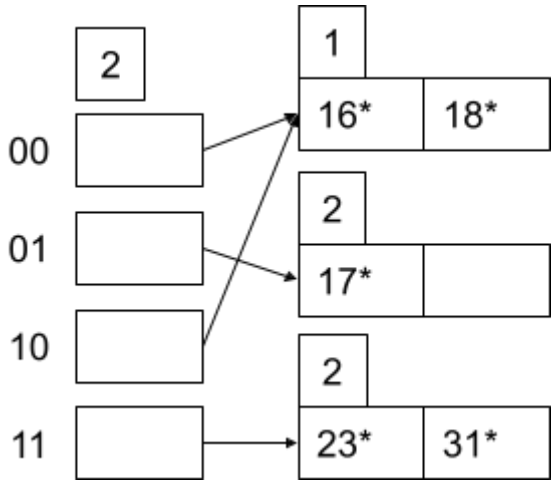
4.1) Insert 22 = $(10110)_2$



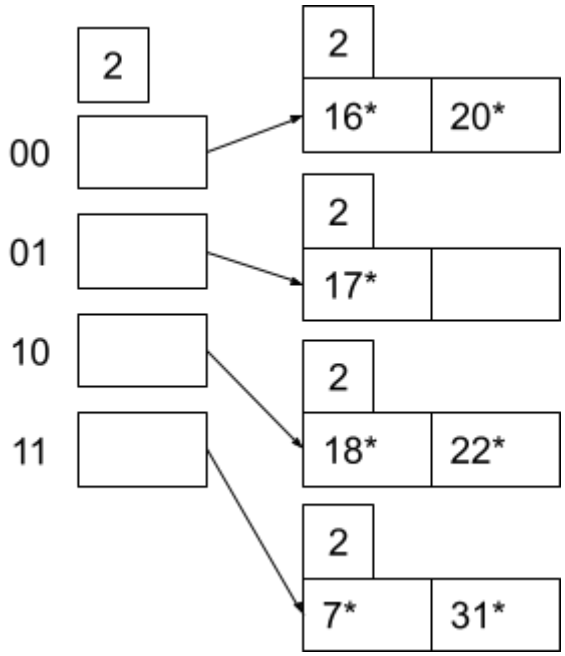
4.2) Insert 61 = $(111101)_2$



4.3) Insert 26 = $(11010)_2$



4.4) Insert 1 = $(00001)_2$



4.5) Insert 39 = $(100111)_2$