# Big Data Analytics

**Homework 1**

**Instructor:** **Ruoming Jin**

**Name:** Shiva Kumar Peddapuram

**KSU ID:** 811235874 [speddapu@kent.edu](mailto:speddapu@kent.edu)

1. What are the 3V features of big data? Please briefly describe them. [15 points]

**Answer**: The 3 Vs of big data are ***Velocity, Variety and Volume***.

Volume = amount of the data

Velocity = Speed by which the data is processed,

Variety = different forms of the data available.

***Velocity****:*

* Initially, a batch system was used to process the data.
* The processing of the data has to be done first and then the next packet has to be sent. This was a terribly slow and time-consuming process.
* These days, we have real-time data from different forms of sources such as sensors, social media, mobile etc. To process this data very fast we are using servers so they can stream the data easily.
* For example, the number of people watching a cricket match around the world or around a country in a particular website/application varies and it is very high. So, in order to manipulate these kinds of data the processing speed is very important.

***Variety:***

* Velocity is measured as the how fast the data is coming in.
* In these real world there are so many types of data and the different types of data exists in the different forms such as in text, images, audio, video, and in many more formats.
* Each type of data has its own methods/ways of storing its data. So, the input would be of so many different data formats. The quality of the data is vastly different, and it requires different techniques to process the different formats of the data and nowadays different companies are using variety of applications to process the data.
* Velocity is defined as how fast the data gets processed and stores that data.

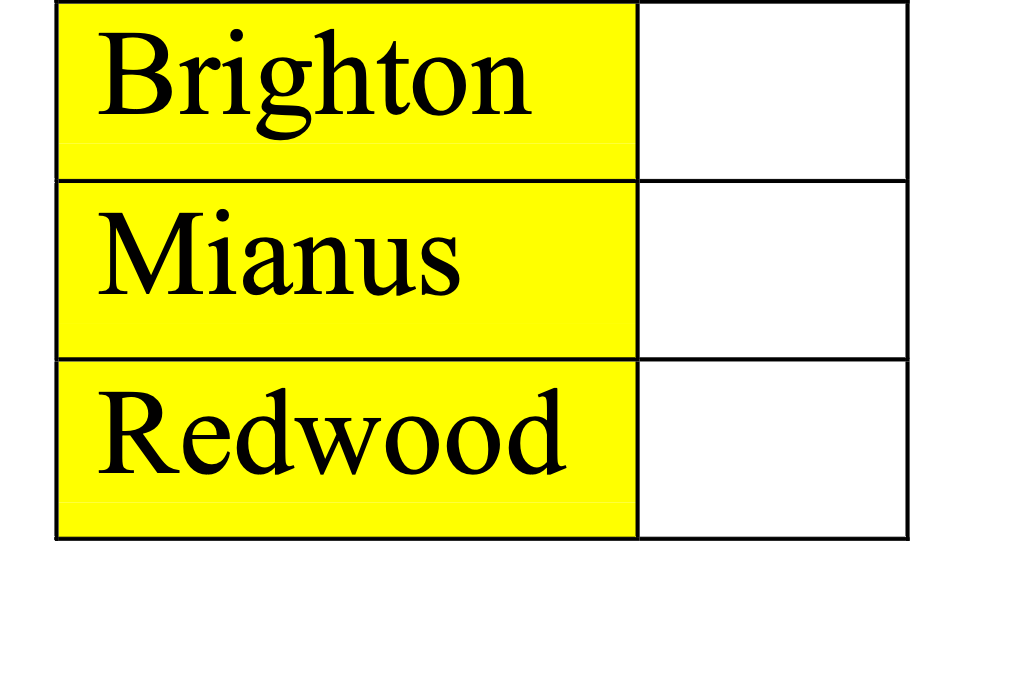
***Volume:***

* As technology is booming nowadays the data us also increasing widely in such a way that the collected data are stored in petabytes and exabytes.
* The data is arriving from different platforms like social media, Reservations, hospitals, sensors, educational institutions data,
* Initially it was done in megabytes and gigabytes. If we consider this scenario, even 2% of the data is also not used efficiently.

2. In the following figure, which is sparse index? Which is dense index? [20 points]

**Index A:**

**Index B:**



**Answer:**

From the above figure we can define them as:

Index A **is Sparse Index** becauseevery entry of Index A is being mapped or pointed to a page (a group of tuples) of data file in the table and are clustered.

Index B **is Dense Index** because records are pointed to actual records within multiple records in the table (every entry of Index B is pointing to every record of data file) and are unclustered.

1. Please tell the differences between B-tree and B+-tree. (Hint: please use Google to find the structure of B-tree index) [20 points]

**Answer:**

***B tree:***

B tree is also known as Binary Search tree in data structures. In B tree the tree can have a one key which can act as parent key and more than two children based upon the scenarios which goes in left and right direction..

In the B tree, the data is sorted in such a way where the left side of the subtree has all the lower values and the right side has the higher values.

***B+ tree:***

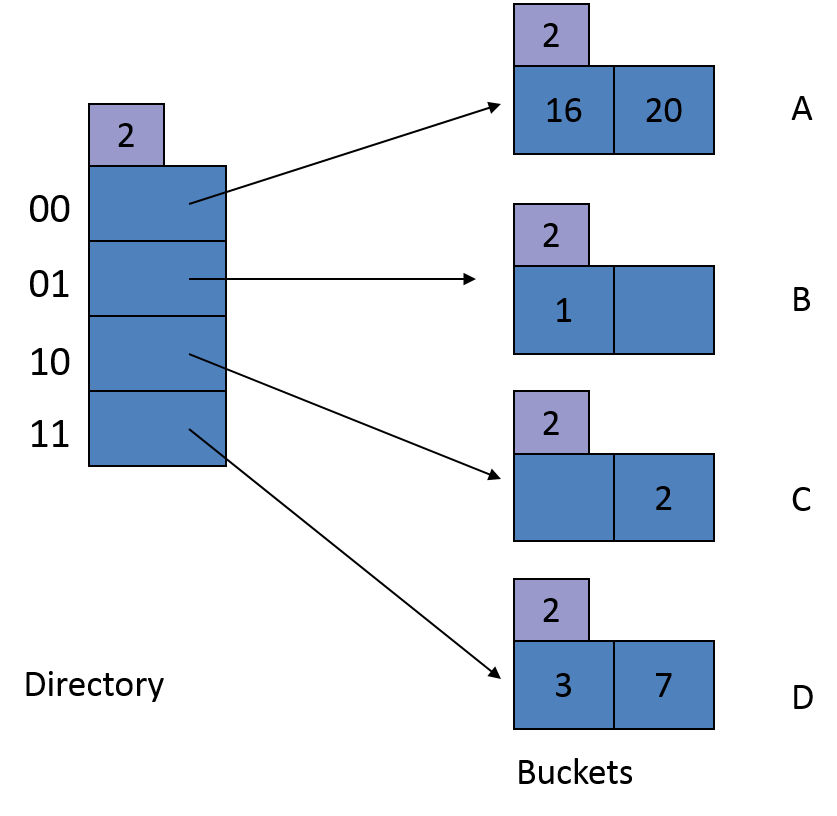
The B+ tree is also known as an advanced self-balanced tree because every path from the root of the tree to the leaf of the tree has the same length.

If we have the same length in a binary tree that means we have all the leaf nodes at the same level.

The difference between B-tree and B+ tree index:

|  |  |
| --- | --- |
| **B-tree** | **B+- tree** |
| 1)The data and the keys can be stored in internal nodes and leaf nodes. | The data and keys can be stored only in the leaf nodes. |
| 2) In this tree, the leaf nodes cannot store data using linked list. | Here, we use sequential linked list to order the data in a leaf node. |
| 3) the data is difficult to search if the data is not found in the leaf node. | As we know all the data is present in the leaf node, the searching becomes easy. |
| 4) no redundant keys are stored in the tree. | Redundant search keys can be stored. |
| 5)There is a lot of space wastage | There is no wastage of space. |
| 6)Insertion operation is difficult. | Insertion in this tree is very easy as the data is present in the leaf |
| 7)The complexity is O(log n) | Here, it is O(logb n) so it is not so complex. |
| 8.) These occupies lot of data. | Compared to B tree this tree occupies less data. |

4. In the extensible hash index (each bucket can hold at most 2 items) below, if we want to insert 19, what will be the structure of the hash index after the insertion? [20 points]



*h*(1) = 000**01**

*h*(2) = 000**10**

*h*(3) = 000**11**

*h*(7) = 001**11**

*h*(16) = 100**00**

*h*(18) = 100**10**

*h*(19) = 100**11**

*h*(20) = 101**00**

**Ans 4.**

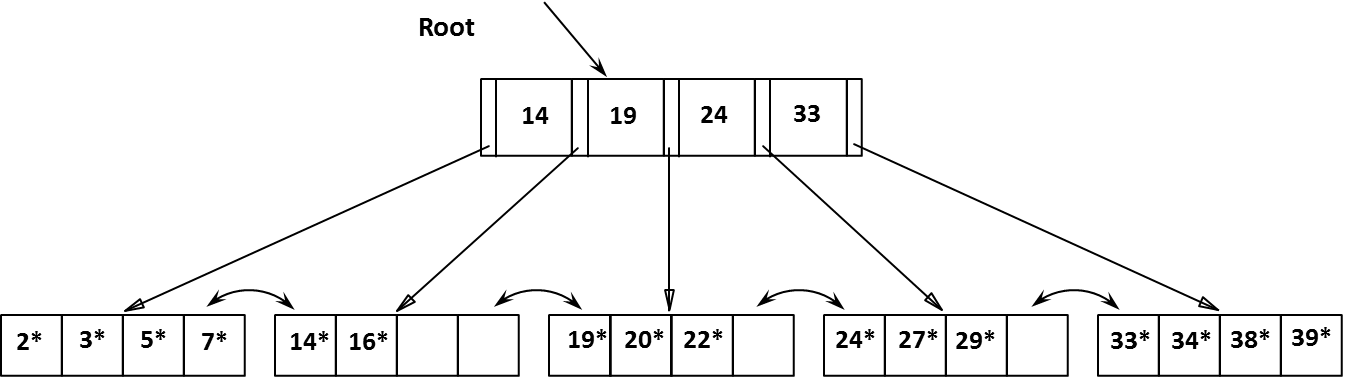
* There are **4 buckets,** and each bucket can store 2 items. Hence, the page **capacity is 2.**
* Now, we want to insert 19 so we will look at last two bits of 19 it is **h(19)=11.**
* As we see in the figure above, 11 index position is already filled with 2 items that is 3 and 7 already. There is No space available to enter item in the bucket. So, we need to check by extending the hash values to h3 that is 3 bits.
* To solve this overflow problem, we will switch into the hash function h3, considering the last 3 bits for positioning the number.
* So, we will have **23=8 indexing positions.**

A picture containing text, whiteboard

Description automatically generated

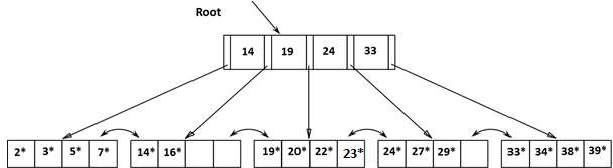
5. Assuming that the node capacity of B+-trees is 4, please draw the updated B+-trees upon insertions. [25 points]

5(a). Insert tuples with key 23\* into the following B+-tree. [10 points]



**Ans 5(a):**

* As we can see from the figure the **19<23<24.** As we can see each node has the **capacity 4**
* When we traverse through 19. We can see there is one node empty where we can insert 23 in the B+ tree.
* We can Transverse as 19,20,22 and easily **insert 23 in the leaf node** because of the space at the position.
* Hence, the updated tree structure:

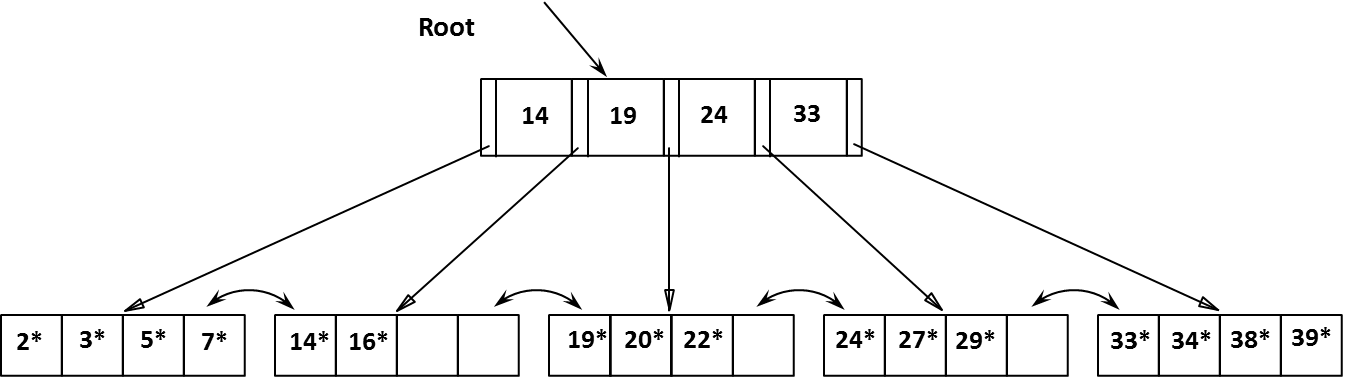


5 answer: representation in written format.

A piece of paper with writing on it

Description automatically generated

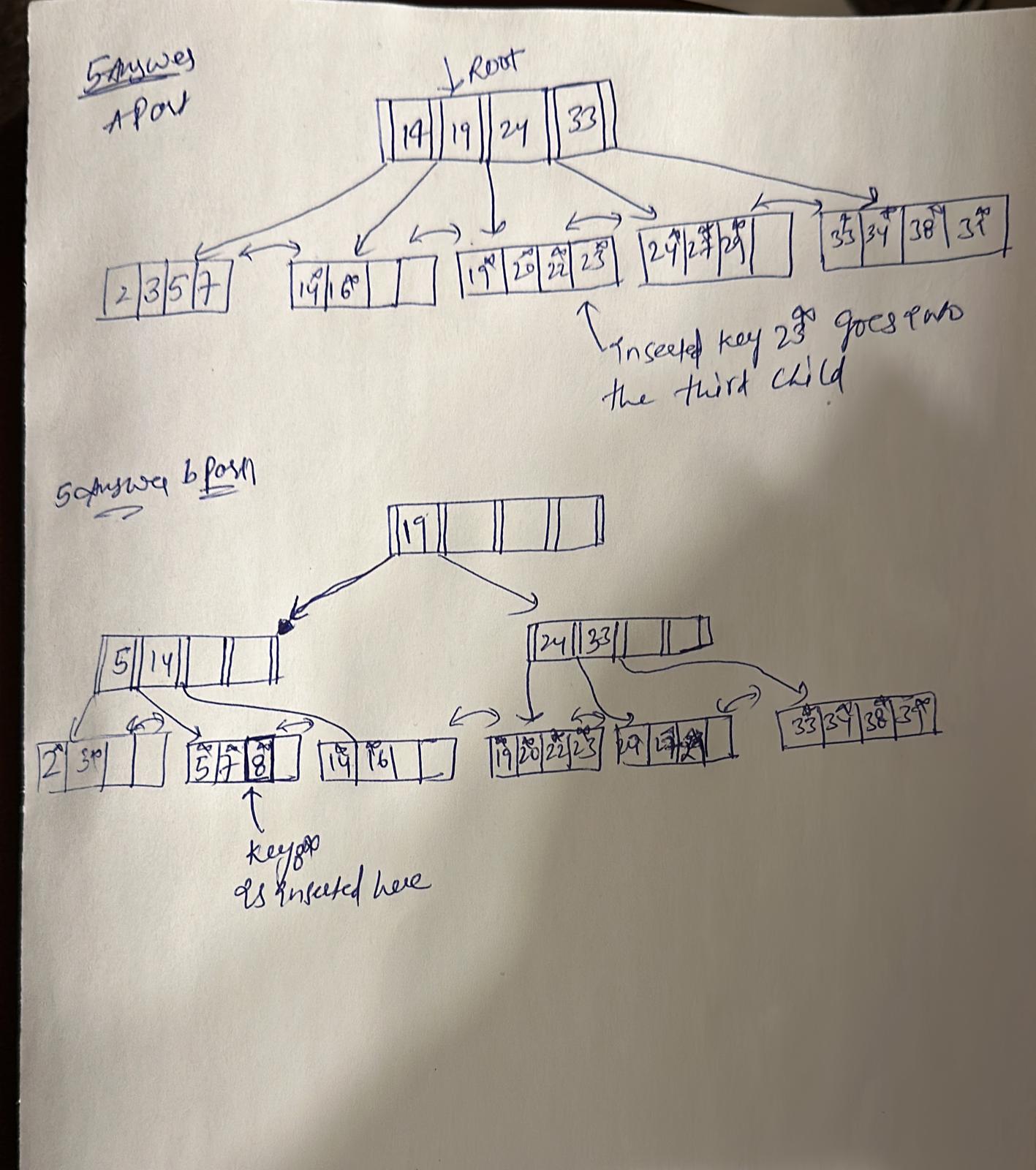
5(b). Insert tuples with key 8\* into the following B+-tree. [15 points]

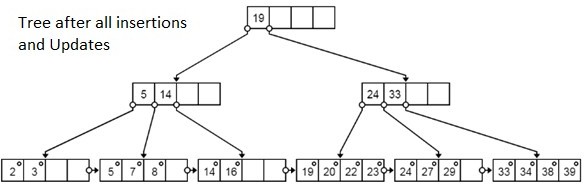


**Ans 5(b)**

* We want to insert 8 so we can clearly observe that 8<14 so it will automatically gets inserted/moved towards the leftmost leaf node.
* As we can observe that the leftmost leaf node is not empty and we can see that all the four nodes in the left side are filled by the values 2,3,5,7.
* We need to insert 8 after 7. So we need to automatically extend the node.
* To insert a new node we need to create new node and divide the indexes, but we need to keep all in a sorted order/ Ascending order. We will take middle value as a root.
* So, (2,3) should be on one side, and the others are {5,7,8} The middle value will be 5. 5 will become as a root.
* So, we need to add root 5 to the parent node. But, Parent node is already filled with 4 numbers 14,19,24,33.
* Now to add 5, we will divide the parent node as {5,14} and other {19,24,33}. 19 is the middle value. So, 19 will become a parent node and others as a child node.

**Updated Tree**

****

****