# CIS 263 Introduction to Data Structures and Algorithms

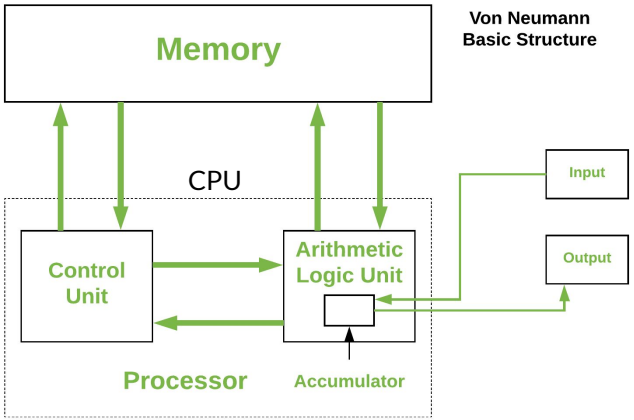
B-Trees

# B-Tree

- A **B-tree** is a self-balancing tree.
- The B-tree generalizes BST, allowing more than two children.
- Unlike other self balancing (AVL Tree, RBT), the B-Tree is well suited for **storage systems** that read and write relatively large blocks of data, such as databases and file systems.

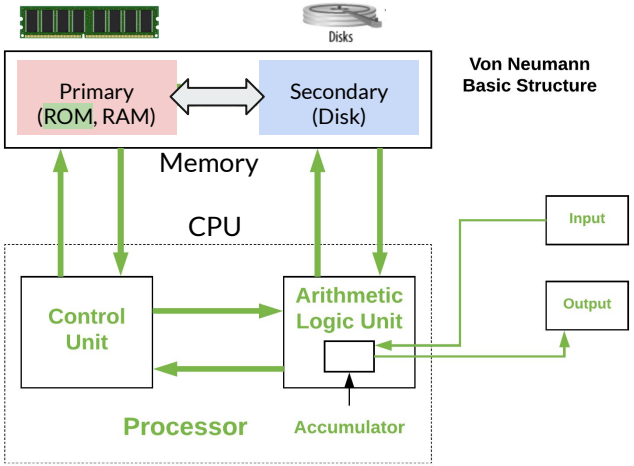
https://en.wikipedia.org/wiki/B-tree

# Computer Architecture basics




https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/

# Computer Architecture basics

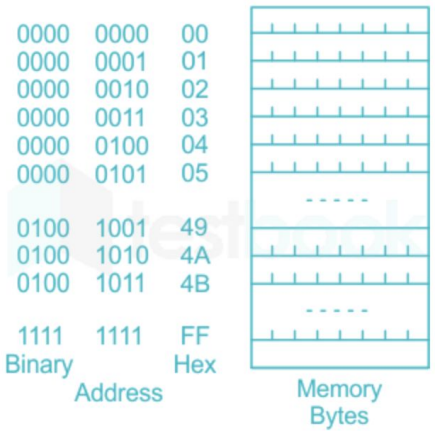



Von Neumann
Basic Structure

- In this course, our main focus will be mainly focused on the **memory management** part; more importantly the **primary memory**
- And **number of disk accesses** (latency delay)
- Through
  - Using appropriate data structures, and
  - Algorithms
- We are not doing any improvements over architecture etc.

https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/

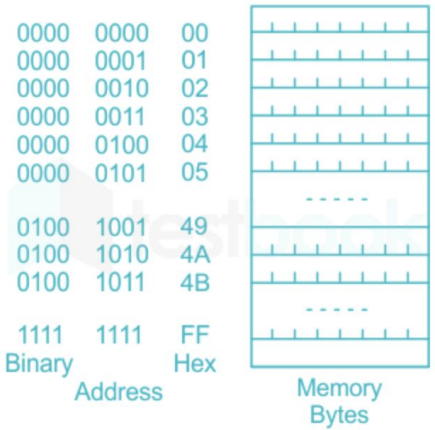
# Memory, the concept of Array




- 8 bit memory (a simple case)

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

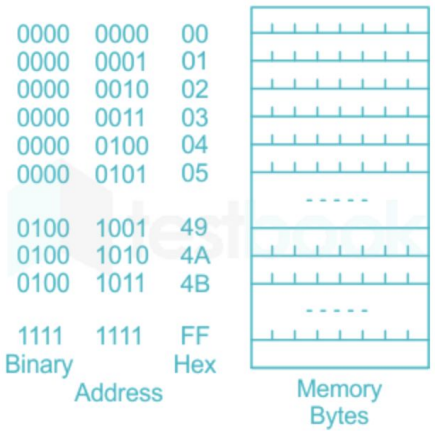
# Memory, the concept of Array




- 8 bit memory (a simple case)
- **What's the total memory size?**

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

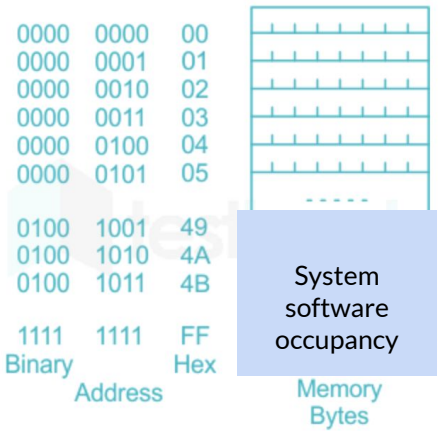
# Memory, the concept of Array




- 8 bit memory (a simple case)
- **What's the total memory size?**
  - **$2^8$ or 1024 byte**

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

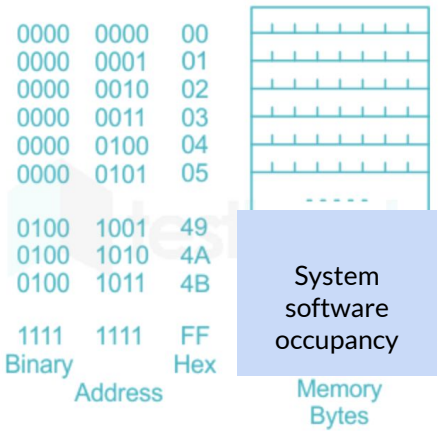
# Memory, the concept of Array




- 8 bit memory (a simple case)
- What's the total memory size?
  - $2^8$ or 1024 byte
- **Addressing is inherent (consecutive, a sequence)**

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

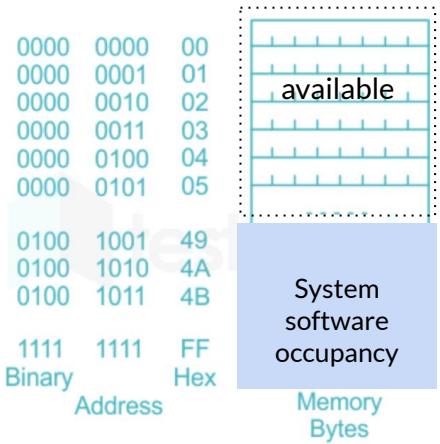
# Memory, the concept of Array




- 8 bit memory (a simple case)
- What's the total memory size?
  - $2^8$ or 1024 byte
- **Addressing is inherent (consecutive, a sequence)**
- **Let's assume our system softwares reserved part of the memory like**

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

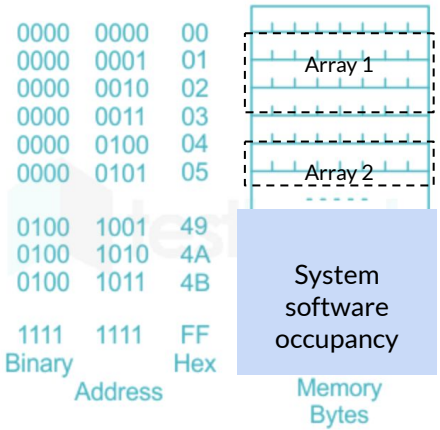
# Memory, the concept of Array




- Addressing is inherent (consecutive, a sequence)
- Let's assume our system softwares reserved part of the memory like

https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

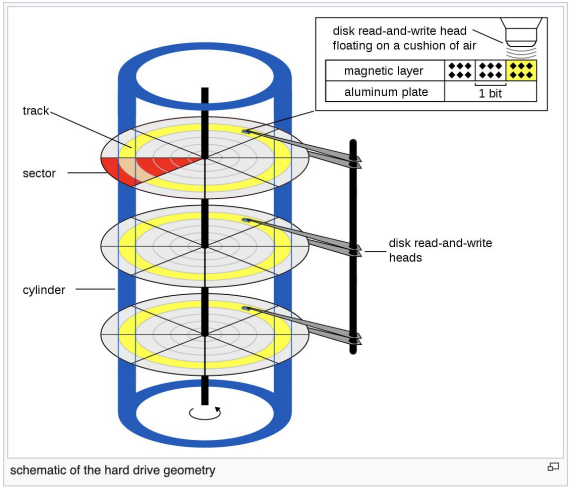
# Memory, the concept of Array




- Addressing is inherent (consecutive, a sequence)
- Let's assume our system softwares reserved part of the memory like
- **Array** (our first data structure) meaning holding a fixed block whether you use it or not.

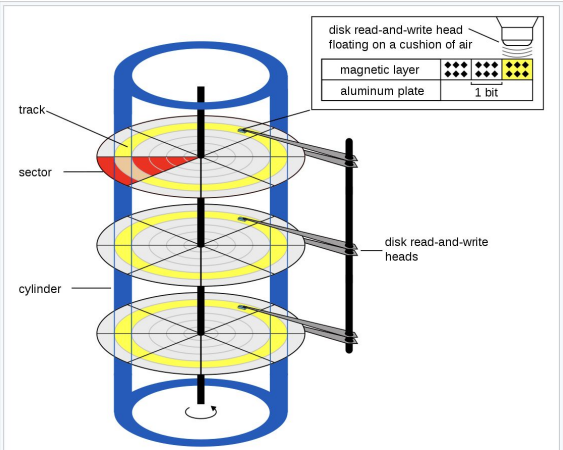
https://testbook.com/question-answer/a-memory-with-8-bit-data-bus-and-8-bit-address-bus--5e6e029ff60d5d4a3b0c5ce9

# Disk: track, sector, block



schematic of the hard drive geometry

# Disk: track, sector, block, offset



schematic of the hard drive geometry

# It’s all about how to reduce number of disk accesses

# Disk: track, sector, block, offset

An exercise:

- One student record requires 128 Byte
- **How many such records a block can store at max ?**

One student record requires 128 Byte

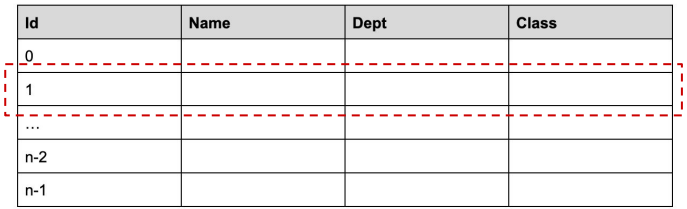
| Id | Name | Dept | Class |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| … | | | |
| n-2 | | | |
| n-1 | | | |

# Disk: track, sector, block, offset

An exercise:

- One student record requires 128 Byte
- How many such records a block can store at max ?
- **512 / 128 = 4**

One student record requires 128 Byte



| Id | Name | Dept | Class |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| … | | | |
| n-2 | | | |
| n-1 | | | |

# Disk: track, sector, block, offset

An exercise:

- One student record requires 128 Byte
- How many such records a block can store at max ?
- 512 / 128 = 4
- **How many blocks you will require to store n=100 student records?**

One student record requires 128 Byte

| Id | Name | Dept | Class |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| … | | | |
| n-2 | | | |
| n-1 | | | |

# Disk: track, sector, block, offset

An exercise:

- One student record requires 128 Byte
- How many such records a block can store at max ?
- 512 / 128 = 4
- How many blocks you will require to store n=100 student records?
- **100 / 4 = 25 blocks; right?**

One student record requires 128 Byte

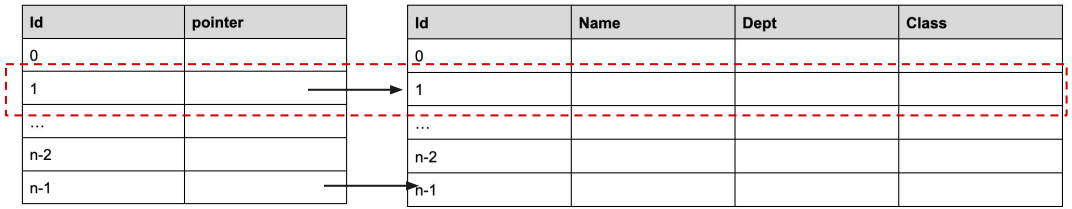
| Id | Name | Dept | Class |
|---|---|---|---|
| 0 | | | |
| 1 | | | |
| … | | | |
| n-2 | | | |
| n-1 | | | |

# How about adding more content

An exercise:

- One student record requires 128 Byte
- 16 additional bytes for the pointer record
- **How many pointer records a Block can store?**

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes

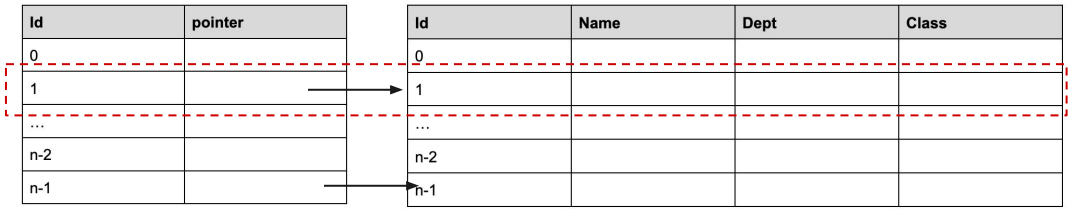


# How about adding more content

An exercise:

- One student record requires 128 Byte
- 16 additional bytes for the pointer record
- How many pointer records a Block can store?
- **512 / 16 = 32**

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes

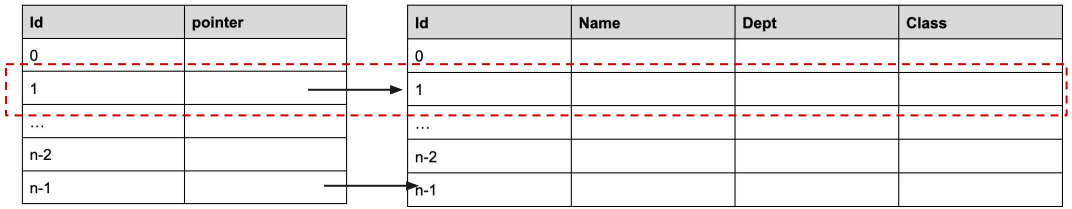


# How about adding more content

An exercise:

- One student record requires 128 Byte
- 16 additional bytes for the pointer record
- How many pointer records a Block can store?
- 512 / 16 = 32
- **How many block you will require to query 100 records ?**

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes

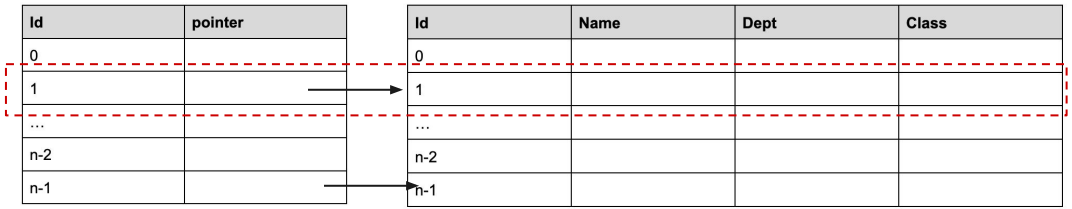


# How about adding more content

An exercise:

- One student record requires 128 Byte
- 16 additional bytes for the pointer record
- How many pointer records a Block can store?
- 512 / 16 = 32
- How many block you will require to query 100 records ?
- **100 / 32 = ~ 4 (max)**

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes

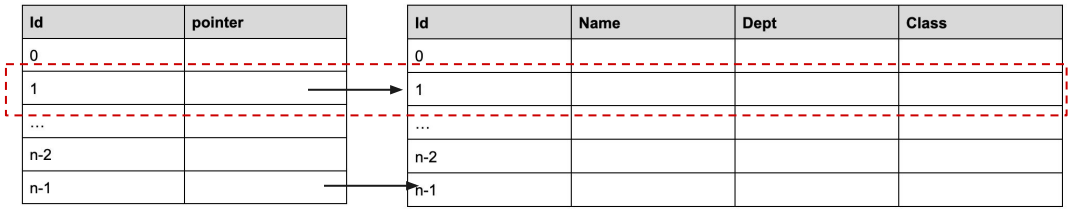


# How about adding more content

An exercise:

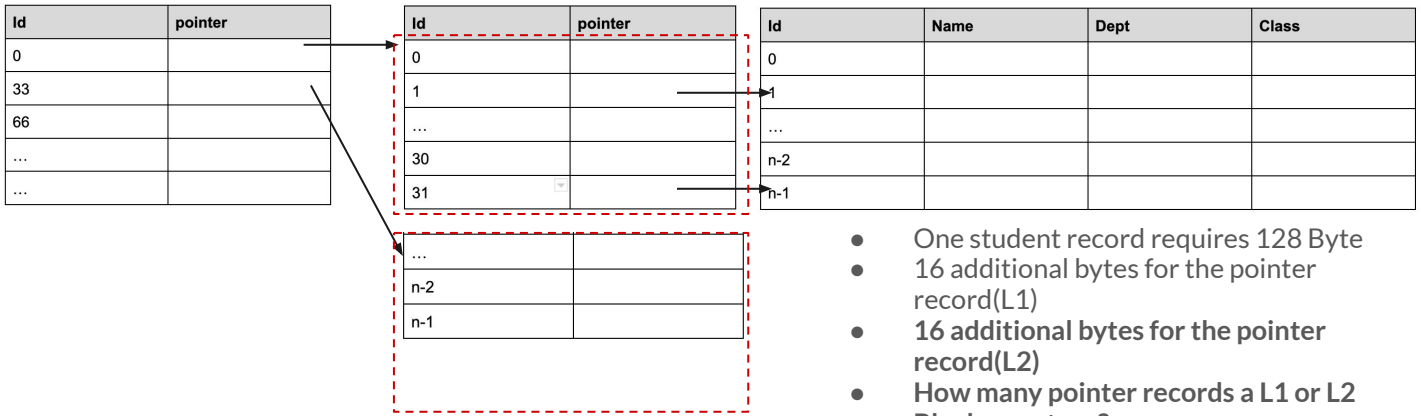
- One student record requires 128 Byte
- 16 additional bytes for the pointer record
- How many pointer records a Block can store?
- 512 / 16 = 32
- How many block you will require to query 100 records ?
- 100 / 32 = ~ 4 (max)
- **So we reduced from 25 - > 4 Block accesses; right?**
- **This is the general idea of database indexing.**

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes



# How about adding more content

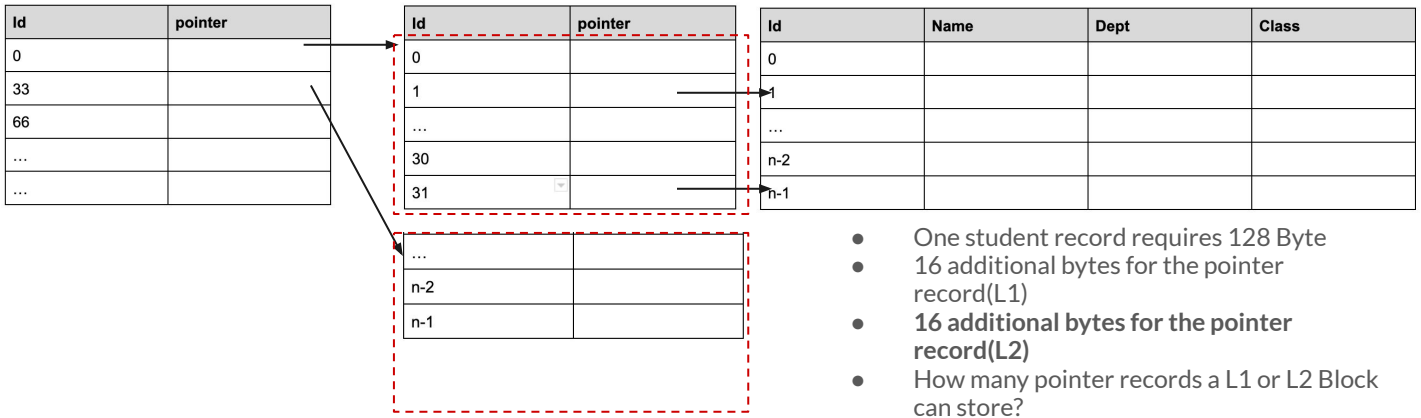
One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes




- One student record requires 128 Byte
- 16 additional bytes for the pointer record(L1)
- **16 additional bytes for the pointer record(L2)**
- **How many pointer records a L1 or L2 Block can store?**

# How about adding more content

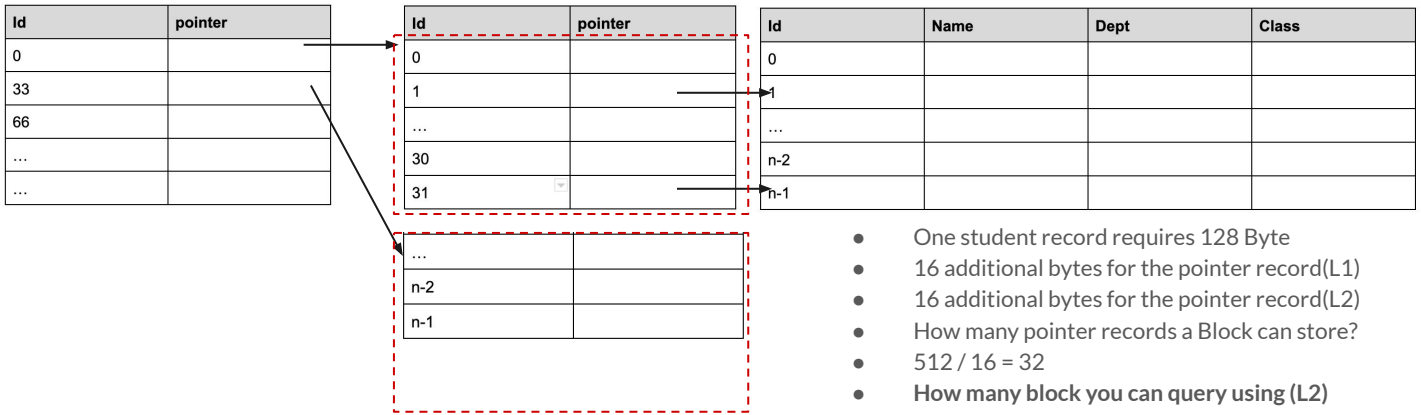
One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes




- One student record requires 128 Byte
- 16 additional bytes for the pointer record(L1)
- **16 additional bytes for the pointer record(L2)**
- How many pointer records a L1 or L2 Block can store?
- **512**

# How about adding more content

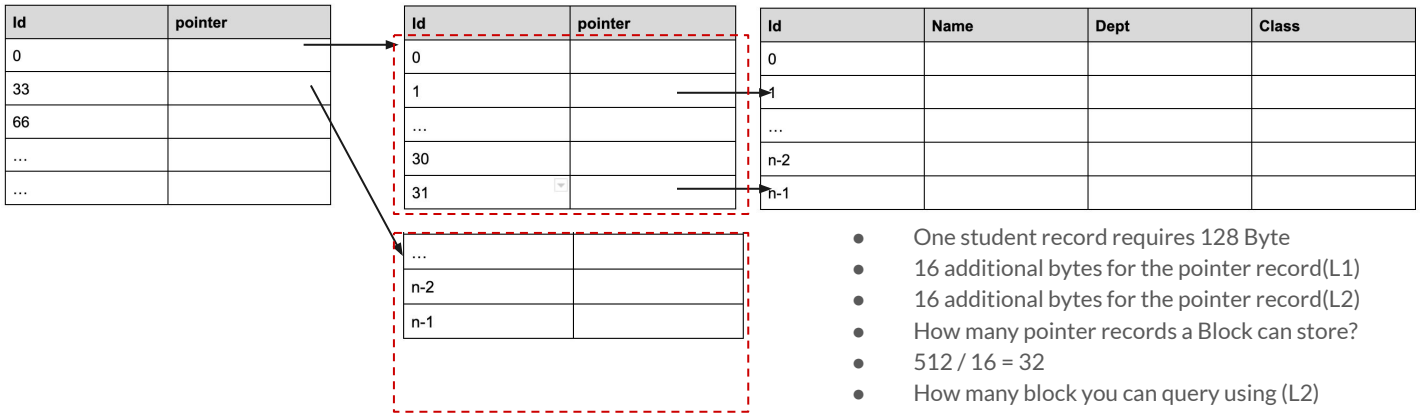
One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes




- One student record requires 128 Byte
- 16 additional bytes for the pointer record(L1)
- 16 additional bytes for the pointer record(L2)
- How many pointer records a Block can store?
- 512 / 16 = 32
- **How many block you can query using (L2)**

# How about adding more content

One student record requires 128 Bytes
One (id + pointer takes): 16 Bytes




- One student record requires 128 Byte
- 16 additional bytes for the pointer record(L1)
- 16 additional bytes for the pointer record(L2)
- How many pointer records a Block can store?
- 512 / 16 = 32
- How many block you can query using (L2)
- **32 * 32 = 1024**

# m-way tree

- ***m-1*** keys
- ***m*** children

# B-tree

- Every node has ***m/2 (ceil)*** children at least
- Root has minimum ***2 children***
- All leaves at same level
- Bottom up method

# Generating a m(3)-way B-Tree

- M = 3
- Max 3-1 = 2 keys per node

45

**Input sequence:** 45

# Generating a m(3)-way B-Tree

- M = 3
- Max 3-1 = 2 keys per node

35, 45

**Input sequence:** 45, 35

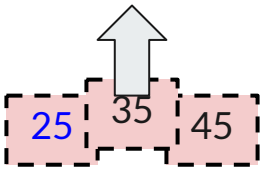
# Generating a m(3)-way B-Tree

- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds

25, 35, 45

**Input sequence:** 45, 35, 25

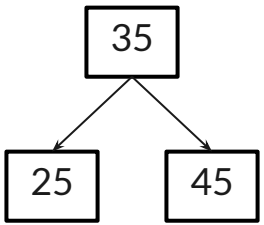
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25

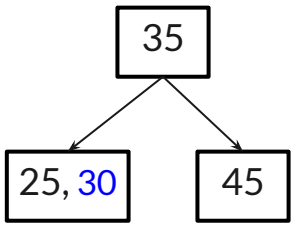
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25

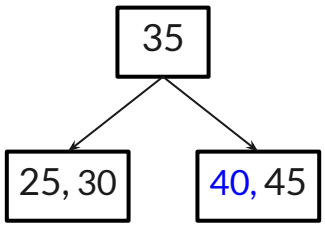
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node

**Input sequence:** 45, 35, 25, 30

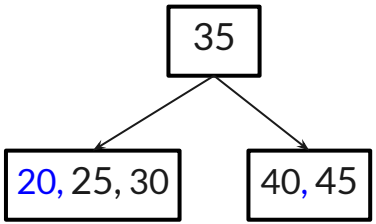
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node

**Input sequence:** 45, 35, 25, 30, 40

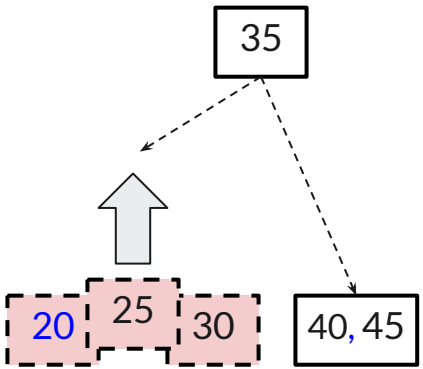
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25, 30, 40, 20

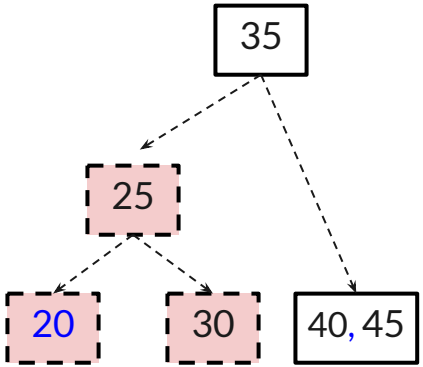
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25, 30, 40, 20

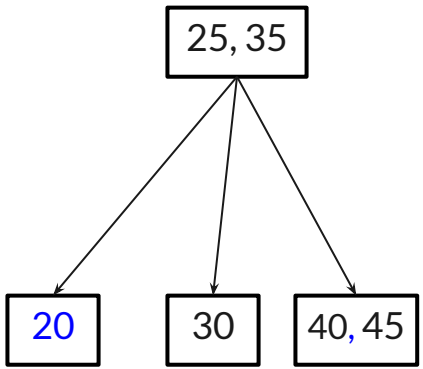
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25, 30, 40, 20

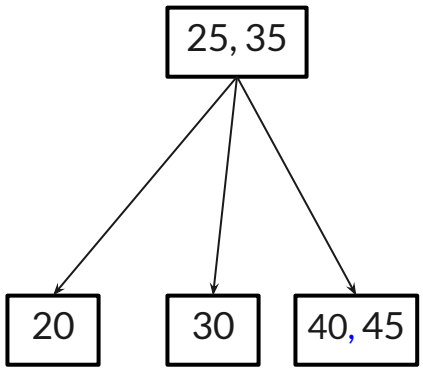
# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node
- Limit exceeds
- Split and adjust levels

**Input sequence:** 45, 35, 25, 30, 40, 20

# Generating a m(3)-way B-Tree




- M = 3
- Max 3-1 = 2 keys per node

**Input sequence:** 45, 35, 25, 30, 40, 20