PRINCIPLES OF DATA ORGANISATION

External Hashing

MOTIVATION

- Key/pointer pairs ~ index
  - Key space and address space
- Hashed file organization

EXTERNAL HASHING

- Hashing structure does not fit into the main memory
  - We need to use the secondary memory
- A bucket can contain multiple records
- Efficiency counted in number of accessed blocks ~ each block can accommodate a given number of records
- Algorithms
  - Static : Cormack, Larson & Kajla
  - Dynamic : Fagin, Linear hashing (Litwin)

EXTERNAL HASHING

Static methods

- Hash function maps keys into a "fixed" number of addresses/pages
- Allows to add records, but not to extend the address space without the need to rebuild the whole index
- Good for more or less static databases
  - If we run out of space, we have to re-build the whole structure

Dynamic methods

- Allow dynamic growth of the address space based on the size of the database
- Allow the hash function to be modified dynamically
- Good for databases that grow and shrink in size

EXTERNAL HASHING

Directory Schemes

- Directory – a structure in the main memory
  - May grow large
- There is a level of indirection
  - Pages can be scattered in the address space
- High utilization
  - We can store some meta-data
- Can allow overflows, but do not have to

Directoryless Schemes

- No directory
  - The address of the page is determined by the hash function directly
- No level of indirection – we get a number denoting a block – we need a **continuous number of blocks** to use
- Can show poor utilisation
- We cannot store additional metadata
- Pages split in a predefined order
  - A page can be full earlier than it is its turn to split
  - Must allow overflows

OVERFLOW HANDLING POLICY

- Splitting control has a direct effect on how much overflow will be tolerated
- Delayed splitting improves space utilization
  - E.g., 1 full page split into 2 $\rightarrow$ 50% utilization
    - Wasting space, we want about 80/90%
  - Instead of splitting a page as soon as it overflows, an overflow page is utilized
  - The size of an overflow page can be different from the size of the primary file page
- Deferring overflow can be applied to directory schemes
  - Especially helpful when the overflow causes directory doubling

DEFERRED SPLITTING

Sharing overflow pages

- Space utilization can be increased by sharing overflow pages
- Multiple pages share one overflow page
  - Even one overflow page for the entire structures
    - Can fill quickly
    - Can be kept in the main memory

Buddy pages

- Logical pairing of pages
- If a page overflows, the overflown records are inserted into the buddy page
- If the buddy page needs its space or too many overflows occur, the original page overflows
- Good when we do not insert too many records

DYNAMIC (EXTENDIBLE) HASHING

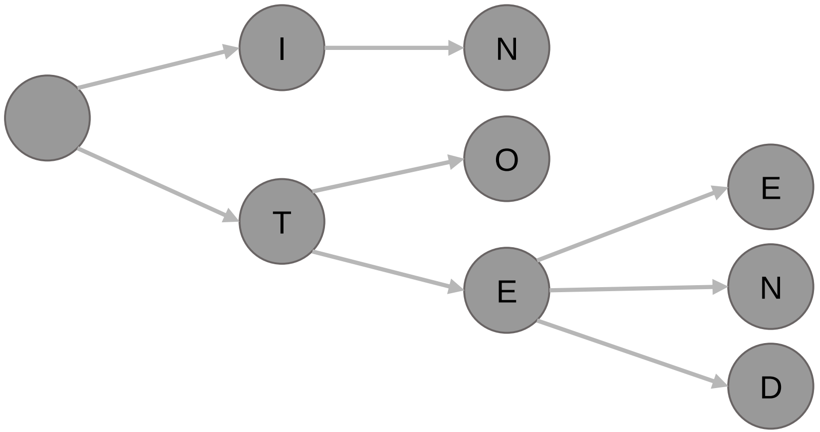
Motivation:

- Static hashing structures or a standard hashing table structure have a fixed maximum size
- Chaining methods lose the expected constant-time operations
  - Some operations are slower
- Maximum size limitations should be avoided while retaining the advantages of constant-time find, insert, and delete operations
- Hash function needs to grow/shrink its domain according to the data
  - Not a simple function but an algorithm

TRIES

Trie = prefix tree

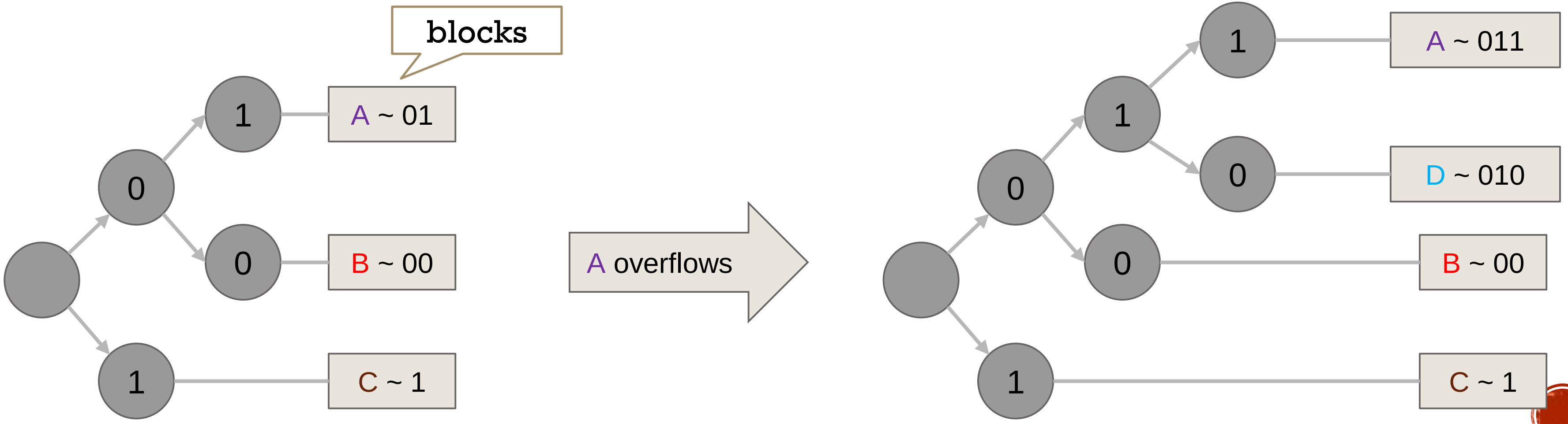
- Branching pattern determined not by the entire key but only by part of it
- All the descendants of a node have a common prefix
  - Other than string types can be converted to bit strings



Search for word TEN

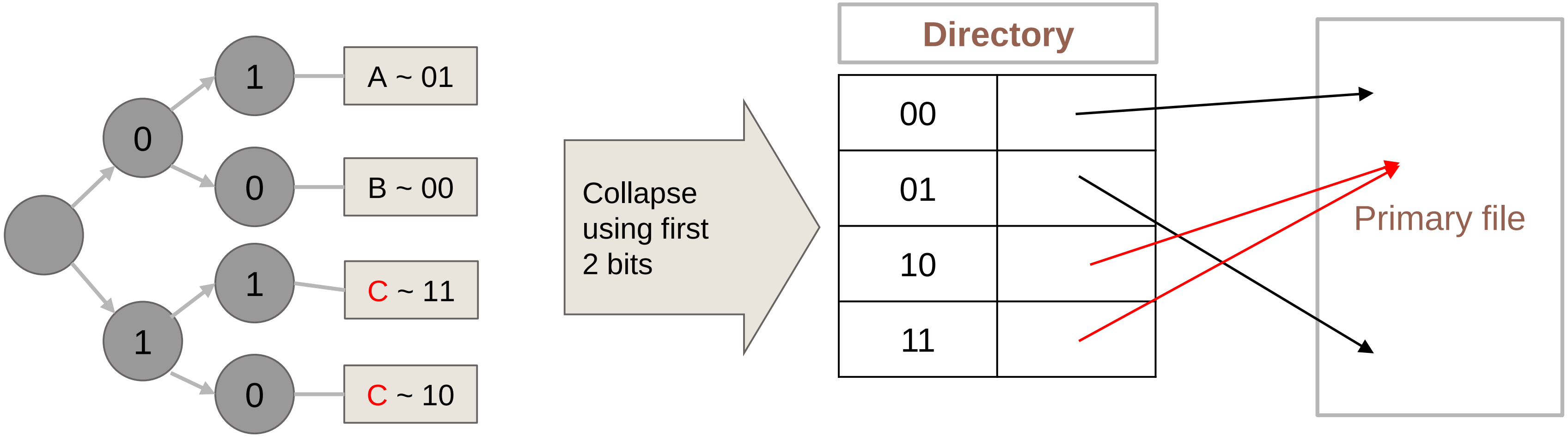
TRIES AS THE BASIS FOR DYNAMIC HASHING

- We will work with binary number/string keys
  - Longer access times in case of a skewed key distribution



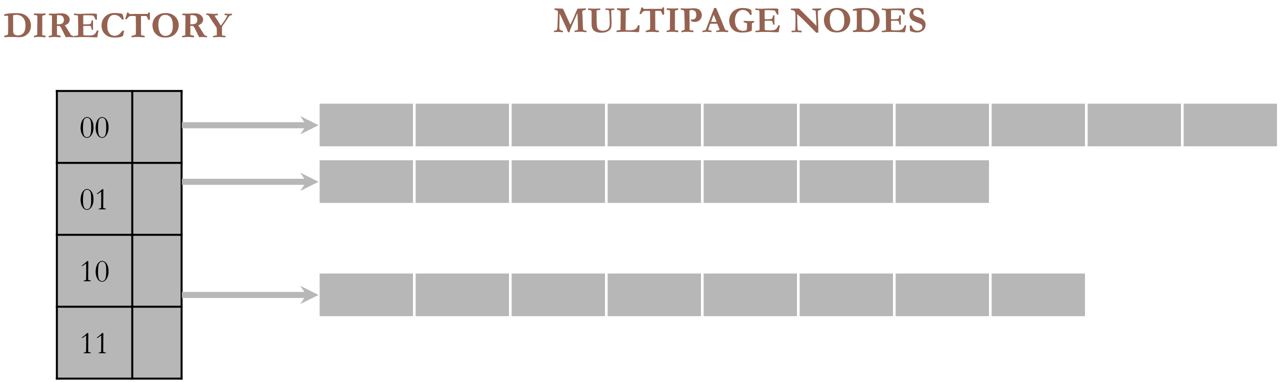
I. Collapsing a Trie into a Directory

- Shortening of a trie by collapsing it into a directory ~ decreasing search time
  - Prefix tree is made complete
- Accessing the directory using a hash function
- Uniform hash function ensures a balanced trie
- Directory introduces a level of indirection in the addressing
- Directory doubling when splitting, e.g., A



MULTIPAGE NODES

- Dealing with the possible growth of the directory in directory schemes
- Fixed upper limit placed on the size of the directory
- When the limit is reached, the nodes expand (not the directory) forming a multipage node
  - Access to the record = access to the directory + searching the multipage node



MANAGEMENT OF MULTIPAGE NODES

- Multipage nodes are <u>stored next to each other</u>; thus they can be managed using standard file organization techniques
  - Sorted sequential files
    - Records stored in the order they were inserted into the multipage node
  - Dynamically hashed file
    - The number of pages can be kept in the directory; as a result, the multipage node can be managed as a dynamically hashed file

II. Collapsing a trie without a directory

- Directory-less schemes
- Maintaining pages in a **contiguous** address space
- The search path in the trie (prefix) forms the address
- Decreases utilisation of the pages
  - We cannot store metadata
- Overflow of a page causes a doubling of the address space size and redistributing records based on the bit prefixes (suffixes) of their keys