Searching

- Basic Points
  - Search returns
    - found
    - not found
  - Related terms
    - table lookup
    - find operation
  - Performance
    - may vary with size of collection
- Search Methods
  - internal (memory)
  - external (file/disk)
  - static (content fixed)
  - dynamic (changeable)
  - key comparison
  - key transformation
- **Searching is closely linked to sorting**

Searching: Examples

- Sequence lookup
  - hash tables
  - compiler symbol tables
  - database systems
  - key searching

    (info = key + data)
- Sequence
  - array / file / table / database
  - linked list / strings (patterns)

- **Sequential Searching**
- **O(n)**

  collection of e where

  e = key + data (+ ref)

ref = first(C)

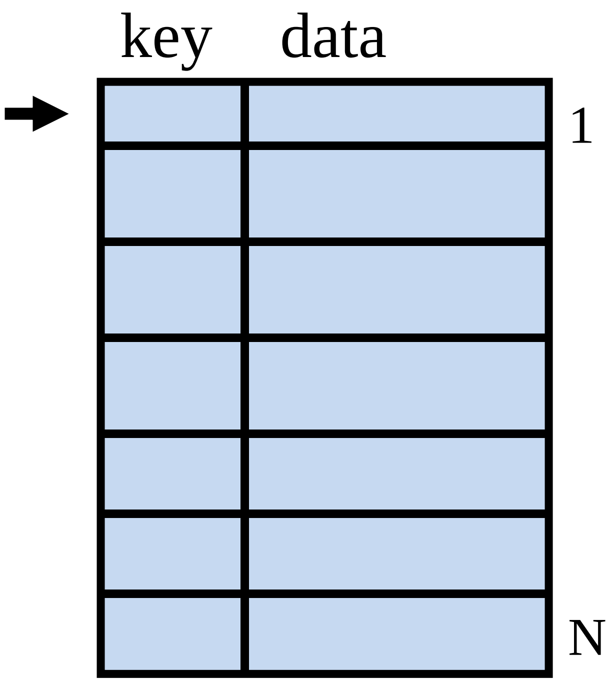
while (not eoc(C))

if (V = value(e)) return T

else ref = next(ref)

Abstract Model

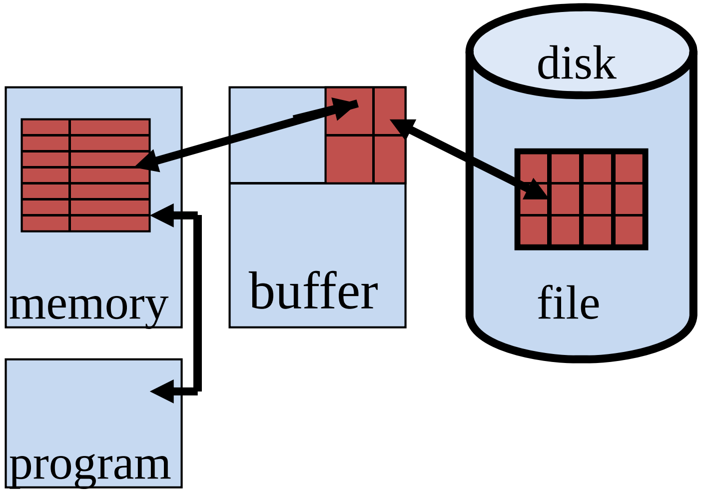
- Collection



- Operations
  - size: C → int
  - is_empty: C → Bool
  - get_key: ref → key
  - get_data: ref → data
  - find: C x key → ref
  - add: C x e → C
  - remove: C x ref → C
  - display: C → C

Implementation Issues

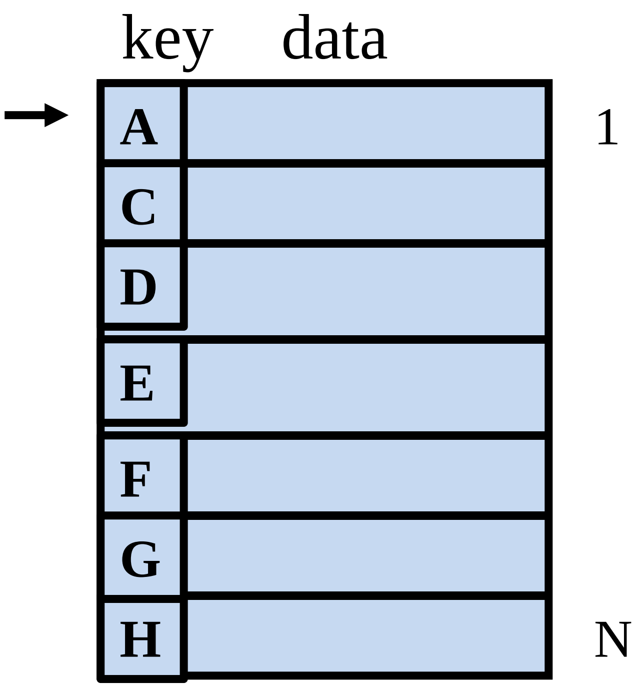
- Example



- Performance
  - size of C
  - disk swapping/buffering
  - search methods
    - hashing / B-trees
  - key / data ratio
  - number of key collections
    - primary (one)
    - secondary (>= 0)

Sorted Sequence

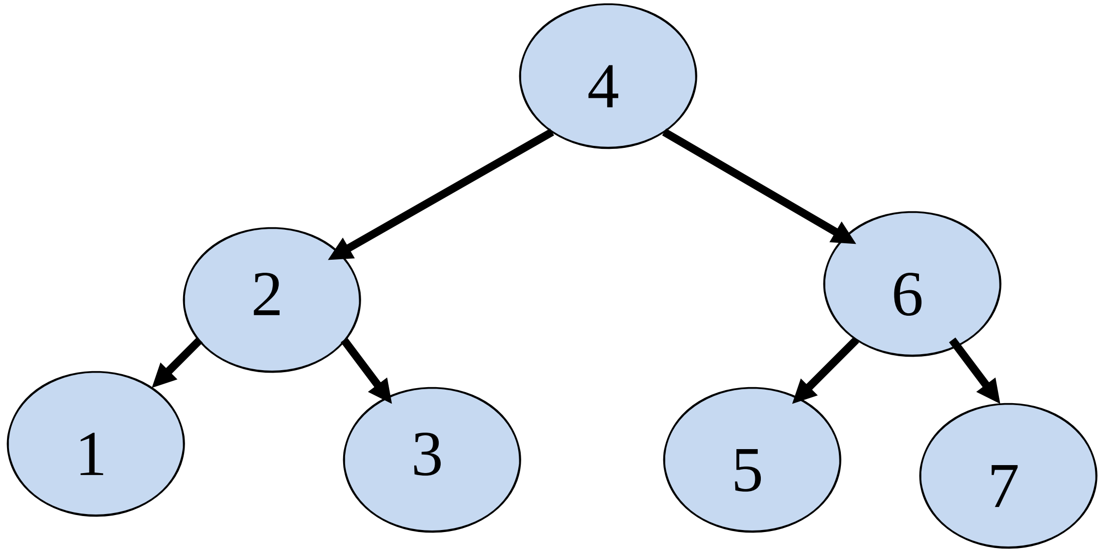
- Collection (sorted)



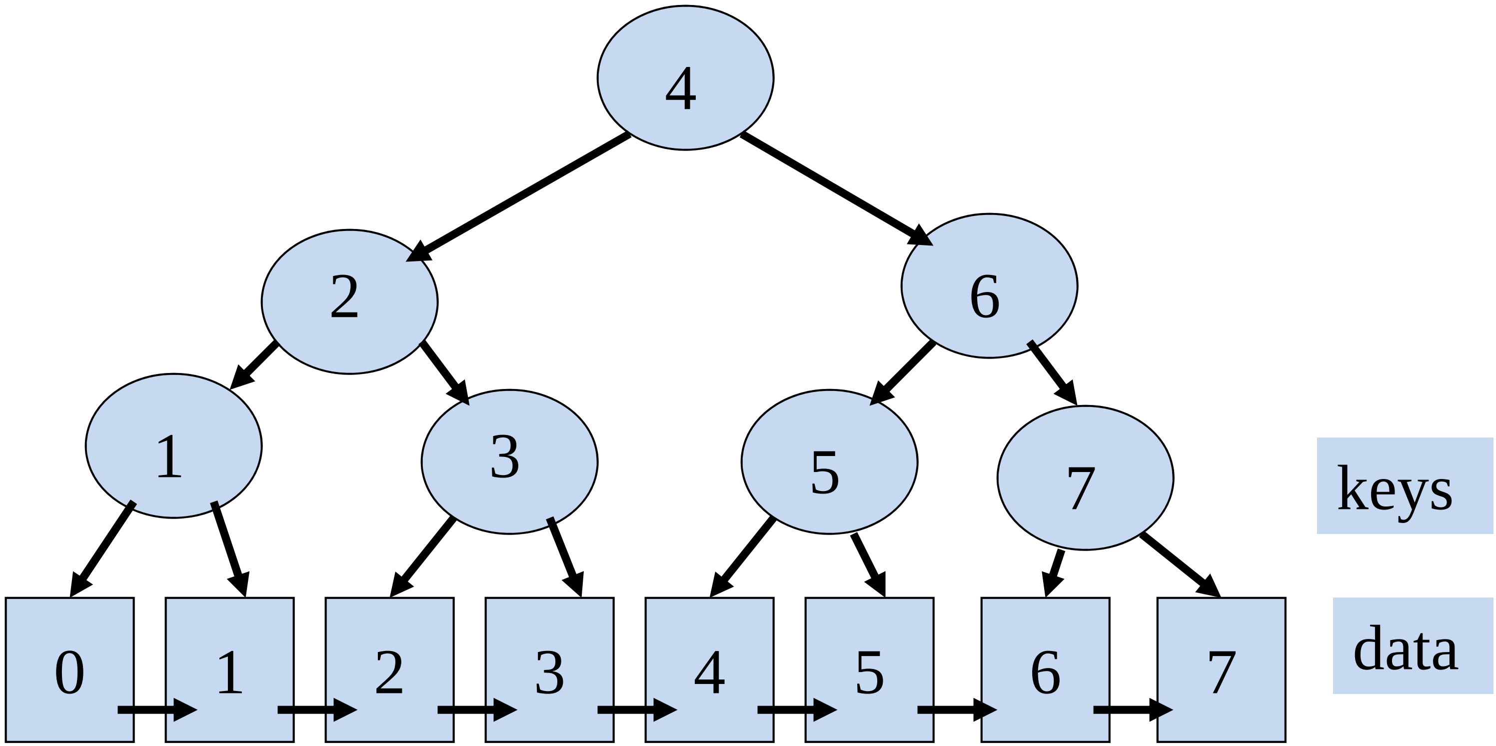
- Keys sorted
  - binary search
    - initialise lo_ref / hi_ref

      K_{lo}<= K<= K_{hi}
    - get mid-point K_{mid}
    - compare
      - **K = K_{mid} => found**
      - **> search upper**
      - **< search lower**
    - repeat process

Binary Search Tree

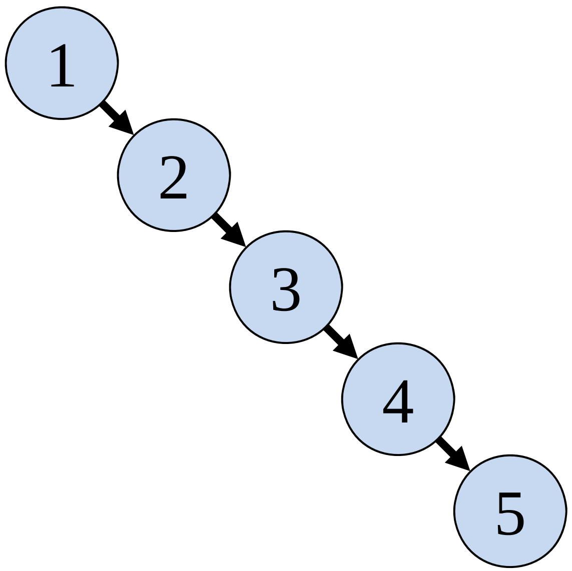


Binary Search Tree + Sequence

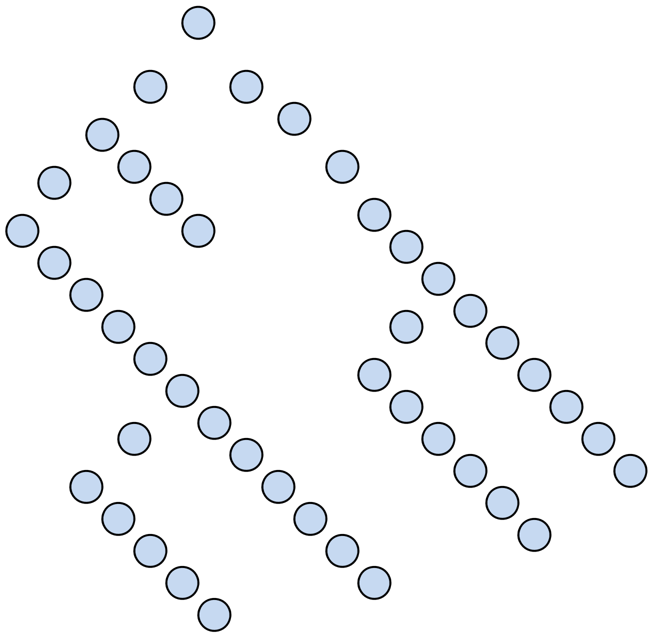


Problem (pathological) cases

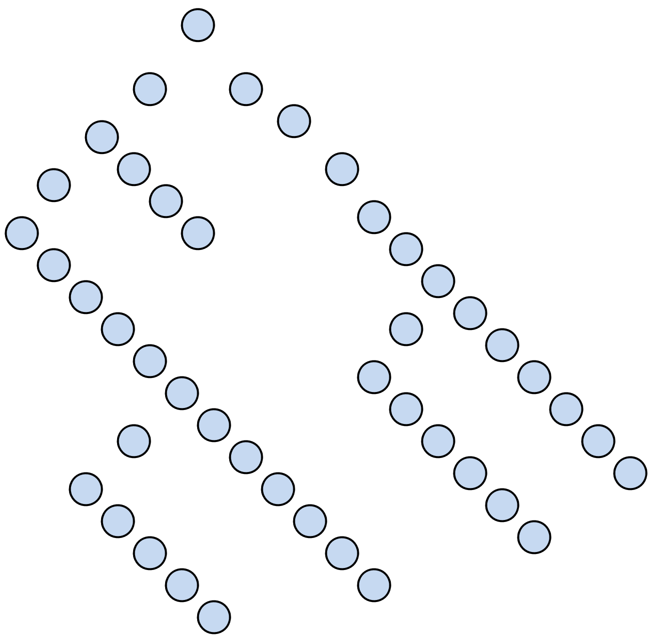
- bst created from 1 2 3 4 5 (sorted)



- Deep trees

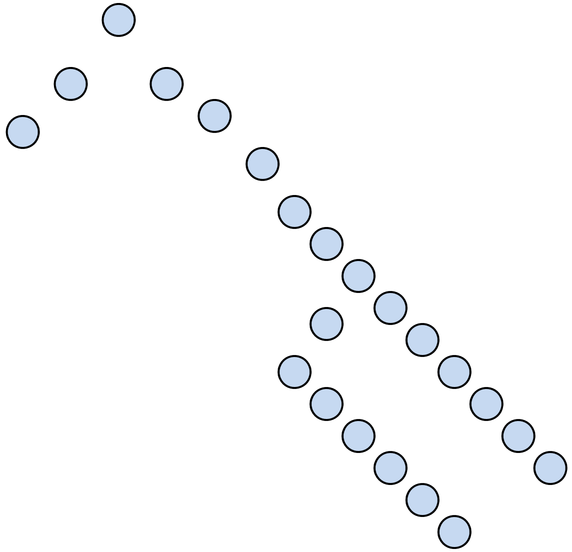


Deep Trees



- Disadvantages
  - path length (# accesses)
  - if node = disk page => high cost for swapping
  - BTs can easily become deep as N increases
  - N (max) = $2^{d+1} - 1$

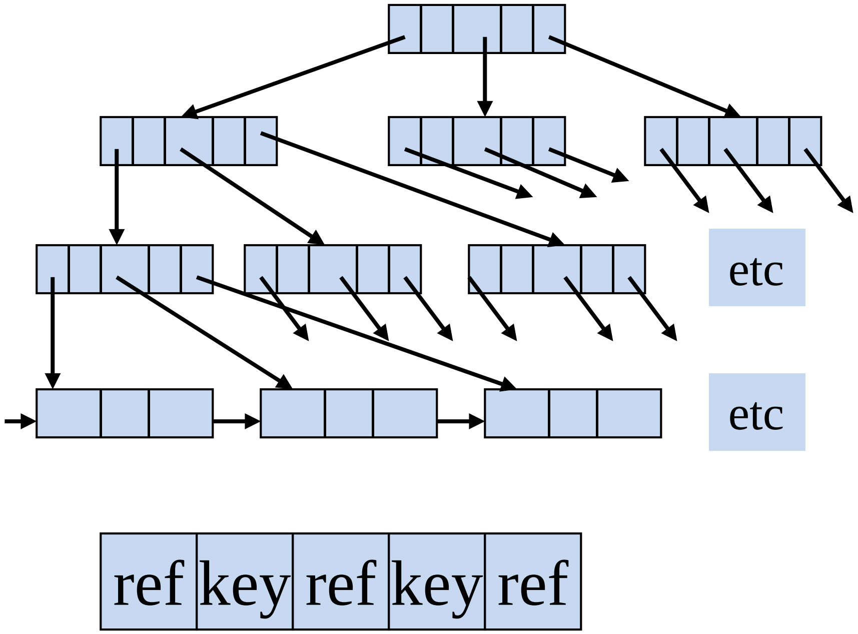
Unbalanced Trees



- Disadvantages
  - path length (# accesses)
  - if node = disk page => high cost for swapping
  - BTs can easily become deep as N increases
  - depends on distribution of the input data
  - static/dynamic analysis

Solution: Balanced Shallow Trees

Knuth's B+ variant of a B-tree



- Advantages
  - shorter search paths
  - index + sequence set
  - index = p_1 k_1 p_2 k_2 …
  - sequence = d_1 d_2 d_3 …
  - sequence set may still be accessed linearly
- Issues
  - add/remove algorithms

Summary

- Search ➔ found / not found
- performance depends on size(C) - O(N), O(logN), …
- internal / external searching
- source data distribution
- unsorted => linear search - O(n)
- sorted => binary search - O(logN)
- -ve : deep / unbalanced trees
- +ve : shallow / balanced trees
- b-tree family of trees
- based on <u>key</u> comparison

- Applications
  - file / DB searching
  - table lookup
    - symbol / function tables
    - primary / secondary key indexes
  - lexicographical search
    - natural language
    - pattern matching
  - fundamental to CS