https://vvtesh.sarahah.com/

Information Retrieval

Venkatesh Vinayakarao

Term: Aug – Sep, 2019
Chennai Mathematical Institute

My brain is like the Bermuda Triangle.

Information goes in and is never found again.

Source – Unknown.

Phonetic Correction

Sound Alikes!

- Homophones (sound alike but have different spellings and different meaning)
  - pair, pear
  - break, brake
  - cell, sell
  - cent, scent
  - knight, night

Sound Alikes!

- Homophones (sound alike but have different spellings and different meaning)
  - pair, pear
  - break, brake
  - cell, sell
  - cent, scent
  - knight, night

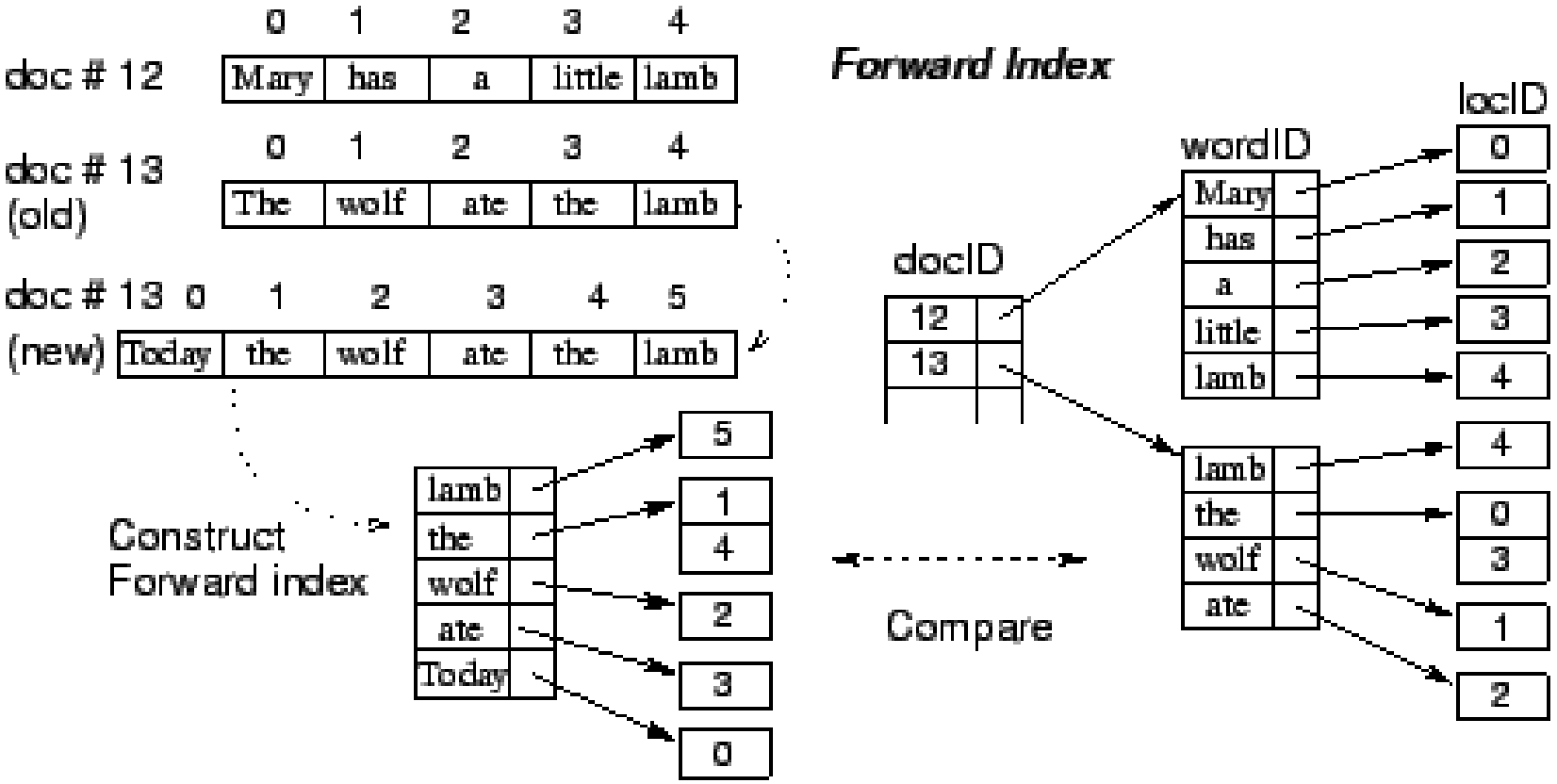
Soundex algorithm became more popular after it was discussed in “The Art of Programming”!
Find a bug and take home 100_{16} (or 0x00000100) cents!

Quiz

- What is the soundex map of Hermann? Is it same as the soundex map of Herman?

Index Types

Forward Index



Inverted Index

Doc 1
I did enact Julius Caesar: I was killed
i' the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble B
hath told you Caesar was ambitious:

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1

Dictionary

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2

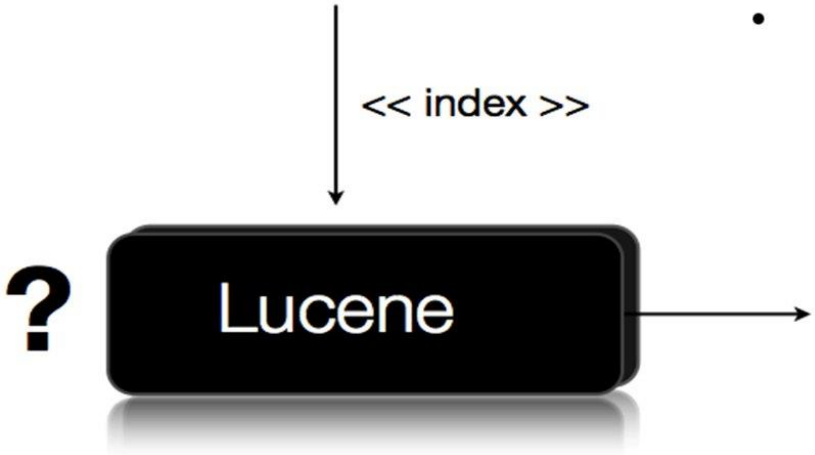
Inverted Positional Index

- Query: *"to$_1$ be$_2$ or$_3$ not$_4$ to$_5$ be$_6$"*
  - TO, 993427:
    - ‹ 1: ‹7, 18, 33, 72, 86, 231›;
    - 2: ‹1, 17, 74, 222, 255›;
    - 4: ‹8, 16, 190, 429, 433›;
    - 5: ‹363, 367›;
    - 7: ‹13, 23, 191›; . . . ›
  - BE, 178239:
    - ‹ 1: ‹17, 25›;
    - 4: ‹17, 191, 291, 430, 434›;
    - 5: ‹14, 19, 101›; . . . ›
- Document 4 is a match!

Inverted Positional Index

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

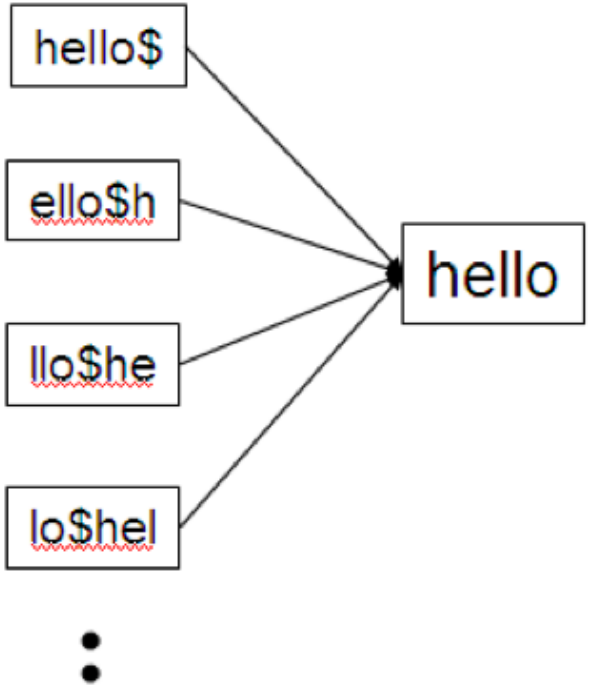
Table with 6 documents



Term	Freq	Postings & Positions
and	1	(6,1) (6,6)
big	2	(2,3) (2,8) (3,8)
dark	1	(6,5)
did	1	(4,7)
gown	1	(2, 10)
had	1	(3,6)
house	2	(2,5) (3,2)
in	5	<(1,8)> <(2,1)> <(2,6)> <(3,3)> <(5,7)> <(6,3)> <(6, 8)>
keep	3	(1,7) (3,10) (5,6)
keeper	3	(1, 4) (4, 5) (5, 3)
keeps	3	(1, 5) (5, 4) (6, 2)
light	1	(6, 10)
never	1	(4, 6)
night	3	(1, 3) (4, 4) (5, 2) (5, 9)
old	4	(1, 2) (2, 4) (2,9) (3, 9) (4, 3)
sleep	1	(4, 8)
sleeps	1	(6, 7)
the	6	<(1, 1)> <(1,6)> <(2, 2)> <(2,7)> <(3, 1)> <(3,4)> <(3,7)> <(4, 2)> <(5, 1)> <(5, 5)> <(5, 8)> <(6, 4)> <(6, 9)>
town	2	(1, 10) (3, 5)
where	1	(4,1)

Permuterm index

- For HELLO, we've stored: *hello$, ello$h, llohe, lohel, and o$hell*



K-gram Index

- Enumerate all character *k*-grams (sequence of *k* characters) occurring in a term
  - Example: from "April is the cruelest month" we get the bigrams: $a ap pr ri il l$ $i is s$ $t th he e$ $c cr ru ue el le es st t$ $m mo on nt h$
- $ is a special word boundary symbol, as before.

A partial snapshot of a sample 3-gram index

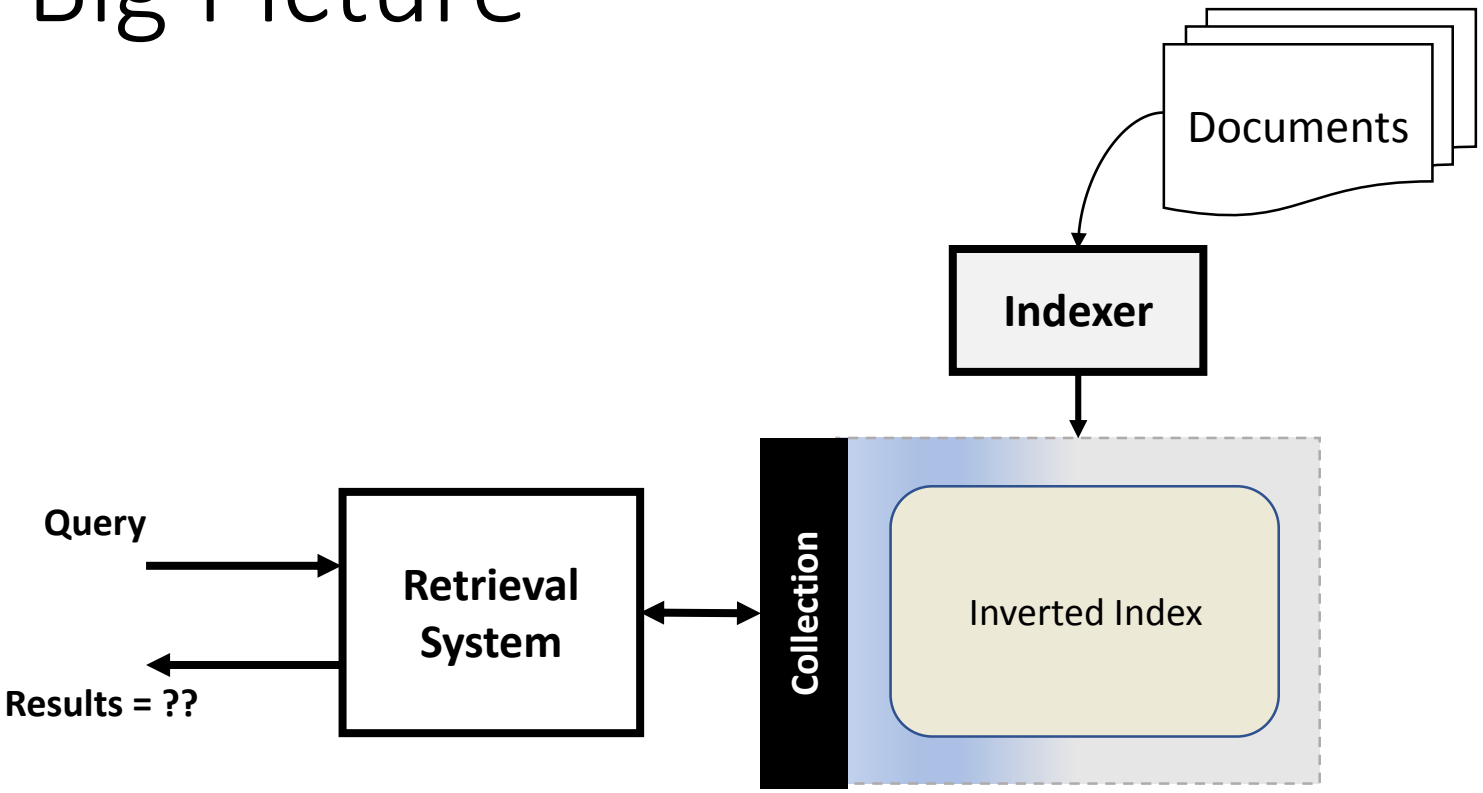
Quiz

- How does the bi-gram inverted non-positional index look like?
  - Assume:
    - Collection: d1: INDIA, d2: ASIA

Index Construction

How to construct a non-positional inverted index?

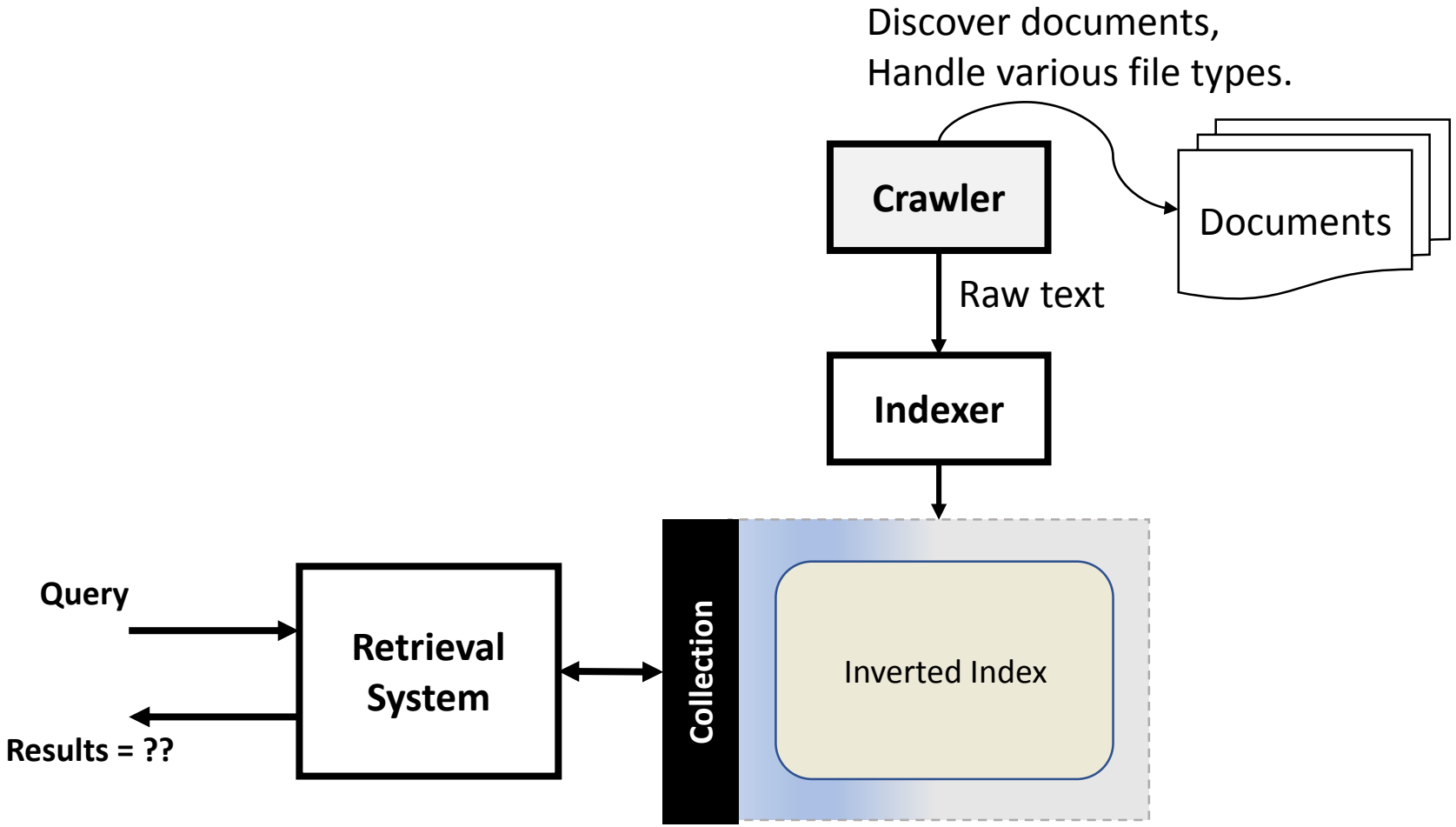
The Big Picture



Agenda

- Hardware Basics
- Indexing static collections
  - Single machine indexing
    - Blocked Sort-based Indexing
    - Single-pass in-memory indexing
  - Distributed Indexing
- Dynamic Indexing
- Index Security

The Big Picture



Hardware Basics

- In-memory data access is faster than access to data on disk.
  - Keep as much data as possible in memory.
    - Caching!
- Seek-time reduces the data transfer rate.
  - Transfer data in chunks.
    - Store related data contiguously.
  - Block sizes of 64 MB are common.
- During I/O, processor may be used for decompression.

Indexing for Small Collections

- Single pass, in-memory
  - Collect term-docID pairs.
  - Sort by term (first), (and then) docID.
  - Organize docIDs as a list.
  - Compute statistics like term frequency and document frequency.
- For large collections, we need efficient ways for the same.

How can we efficiently store a large bunch of strings?

Map them to numbers!

termID → term

1. abacus
2. academia
3. …

Indexing Big Data!

REUTERS

You are here: Home > News > Science > Article

Go to a Section: U.S. International Business Markets Politics Entertainment Technology Sports Oddly Enoug

Extreme conditions create rare Antarctic clouds

Tue Aug 1, 2006 3:20am ET

Email This Article | Print This Article | Reprints

[-] Text [+]

SYDNEY (Reuters) - Rare, mother-of-pearl colored clouds caused by extreme weather conditions above Antarctica are a possible indication of global warming, Australian scientists said on Tuesday.

Known as nacreous clouds, the spectacular formations showing delicate wisps of colors were photographed in the sky over an Australian meteorological base at Mawson Station on July 25.

Reuters RCV1 Corpus

symbol	statistic	value
N	documents	800,000
L	avg. # tokens per doc	200
M	terms (= word types)	400,000
	avg. # bytes per token (incl. spaces/punct.)	6
	avg. # bytes per token (without spaces/punct.)	4.5
	avg. # bytes per term	7.5
	non-positional postings	100,000,000

How can you assign an ID (sequential number) to each term in your collection?

Two cases:

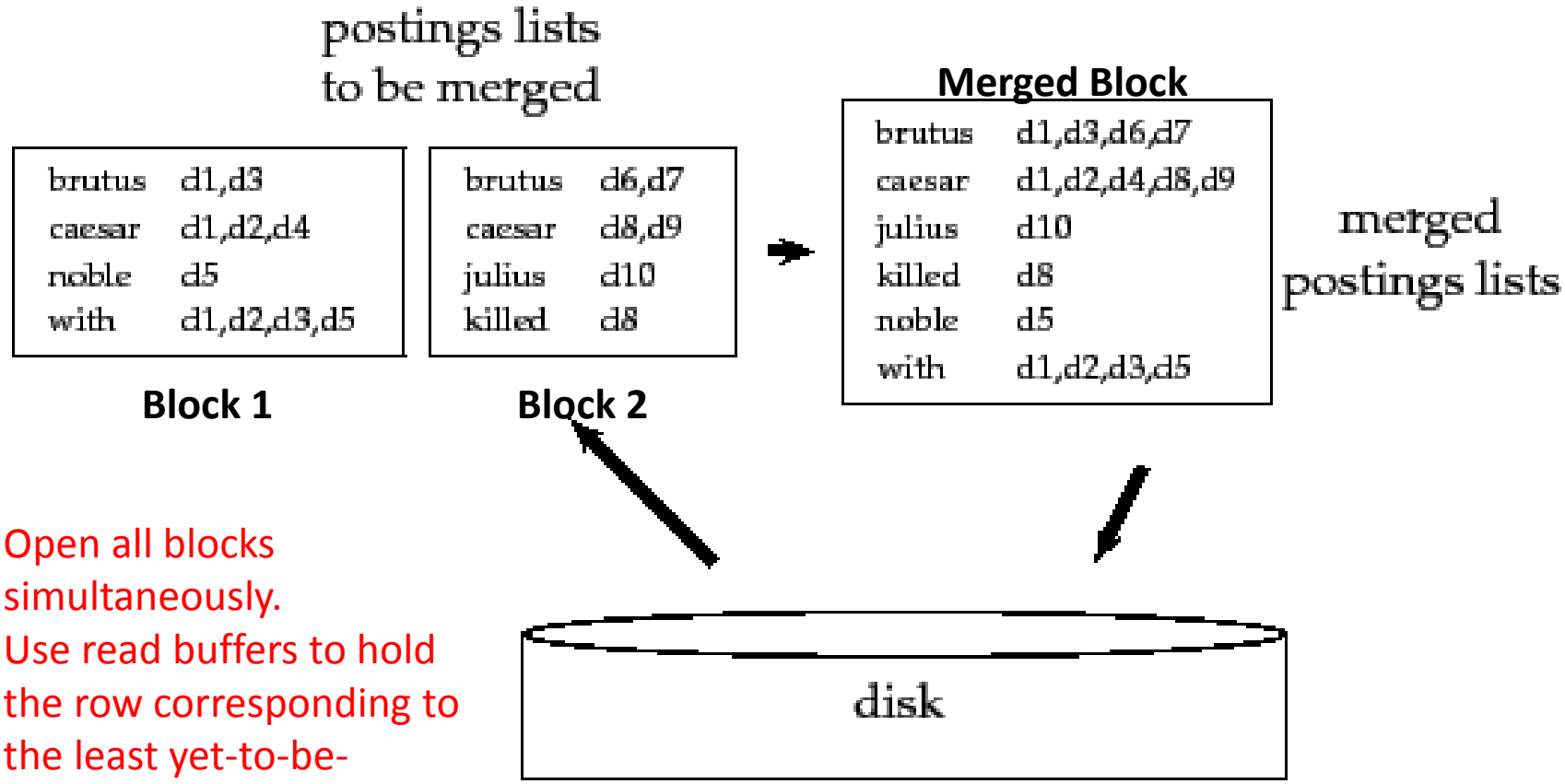
1. **{(termID, term) … } fits into main memory.**
2. **Our collection is too large that the term mappings do not fit into main memory.**

Blocked Sort-Based Indexing

- Steps
  1. Write Blocks of Inverted Index
     1. Parse documents (only as many as can be contained in a block).
     1. Collect termID – docID pairs
     2. Accumulate in a block (till it is full).
     3. Invert the block.
        1. Sort termID – docID pairs.
        2. Merge entries with same termID to create a postings list.
     4. Write to disk.
  2. Merge the blocks.

How will you merge these blocks?

Blocked Sort-Based Indexing



1. Open all blocks simultaneously.
2. Use read buffers to hold the row corresponding to the least yet-to-be-processed termID.
3. Merge and write to disk.
4. Repeat.

Data Structures

From a large set of numbers, if you need to remove the min in each iteration (in O(1) efficiency!), which data structure will you use?

Heaps and Priority Queues

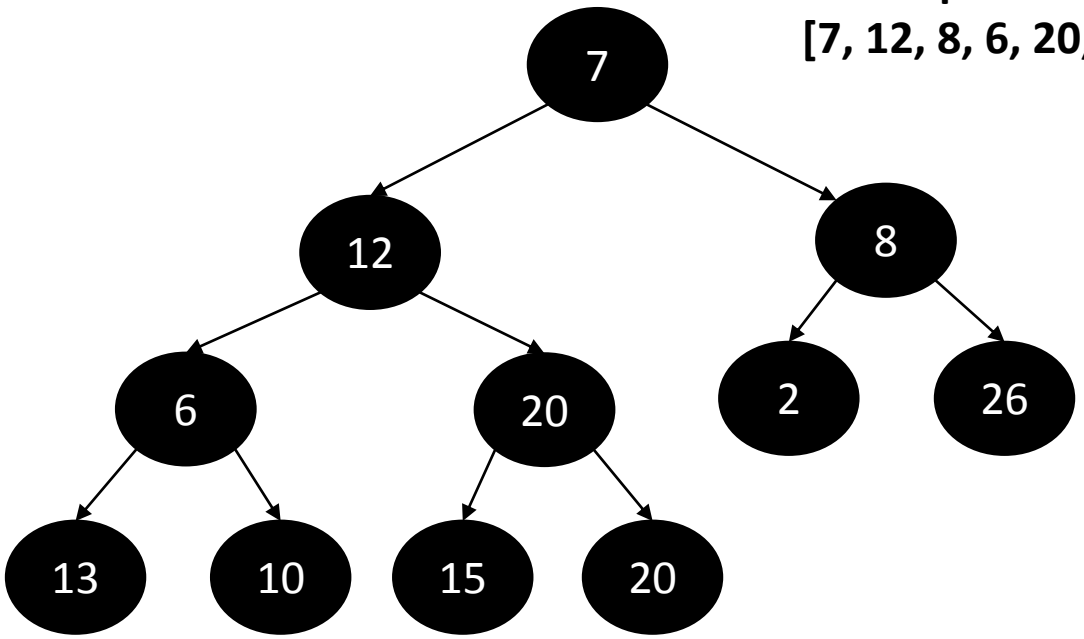
- Priority queue is an abstract data type where
  - each element has a priority
  - element with highest priority is at the top of the queue
- Priority queues are often implemented with heaps.

Heap Example

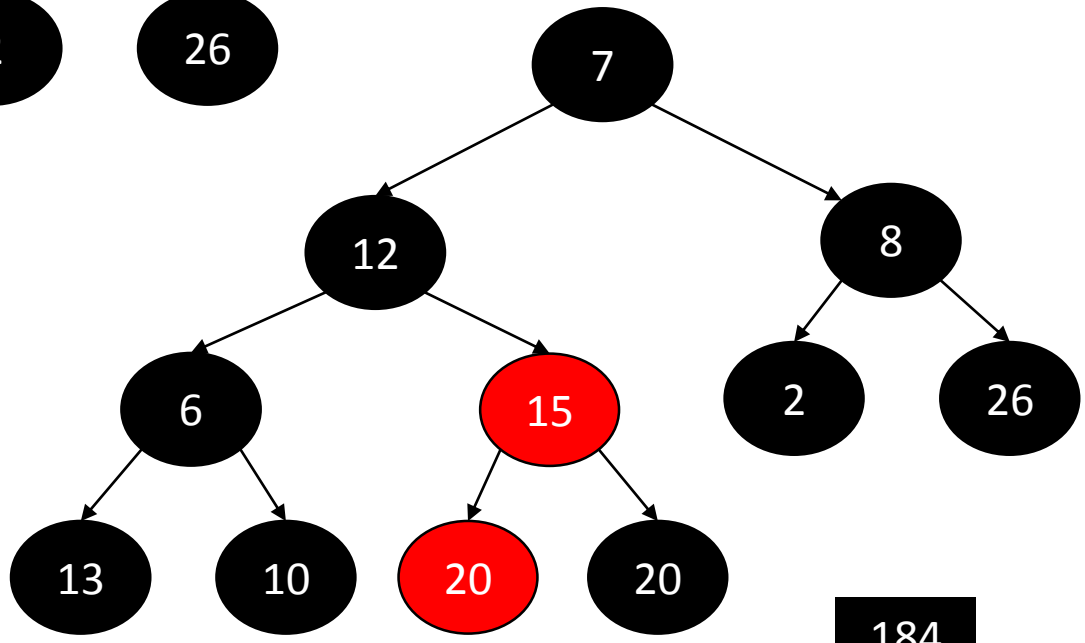
- Heap is an almost complete tree satisfying the heap property.
- Heap Property
  - In a min-heap, the parent node has a value lesser than or equal to the values of child node. [So, root carries the ____ value].

Heapify Example

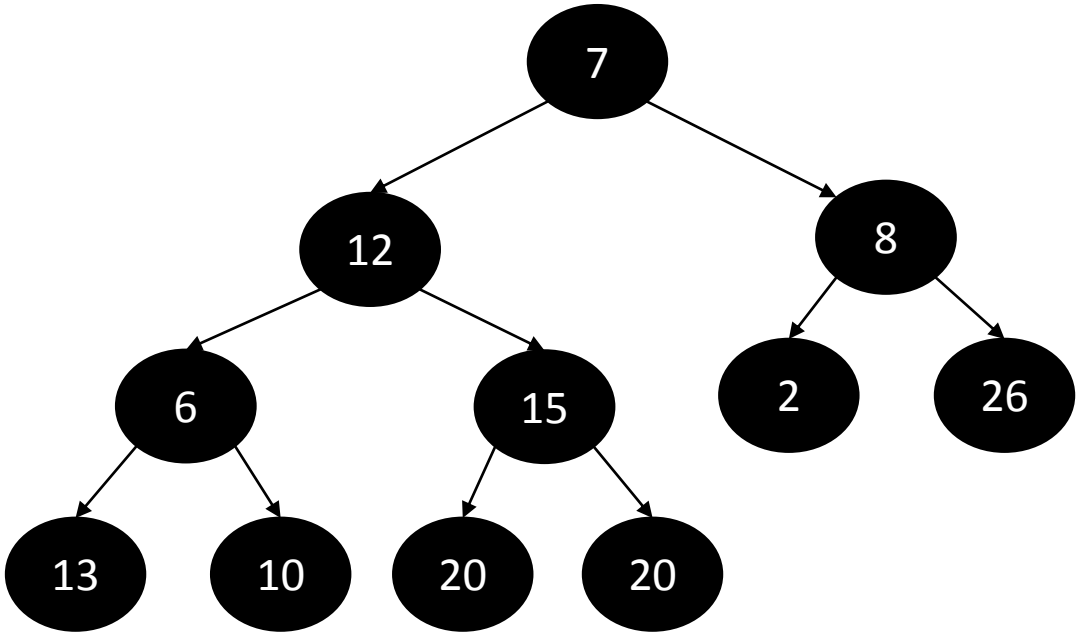
Tree representation of an arbitrary array
[7, 12, 8, 6, 20, 2, 26, 13, 10, 15, 20]



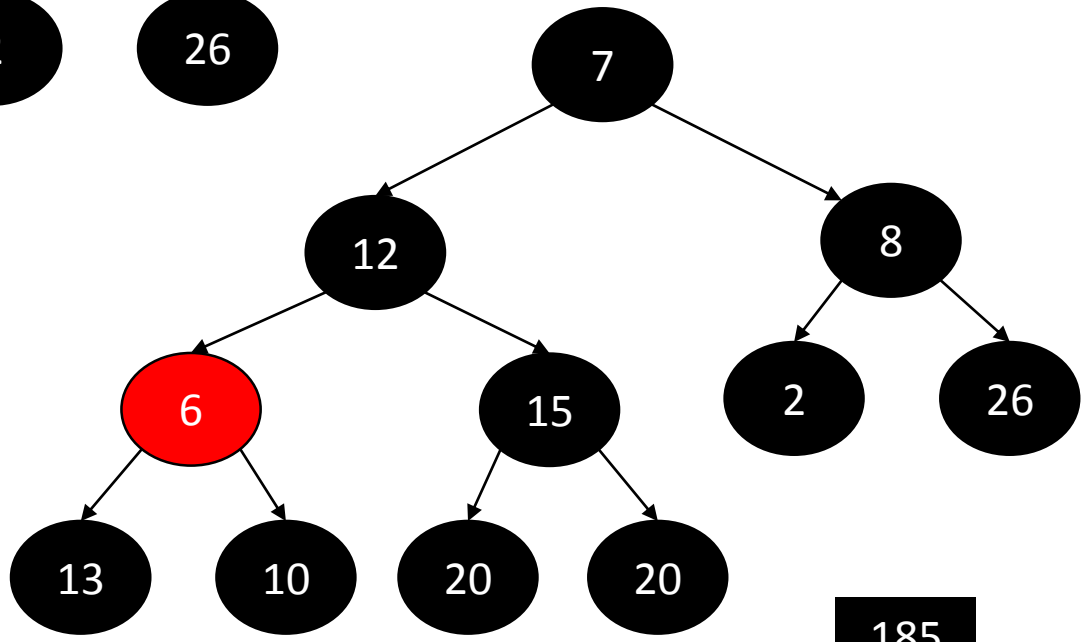
Heapify – Step 1



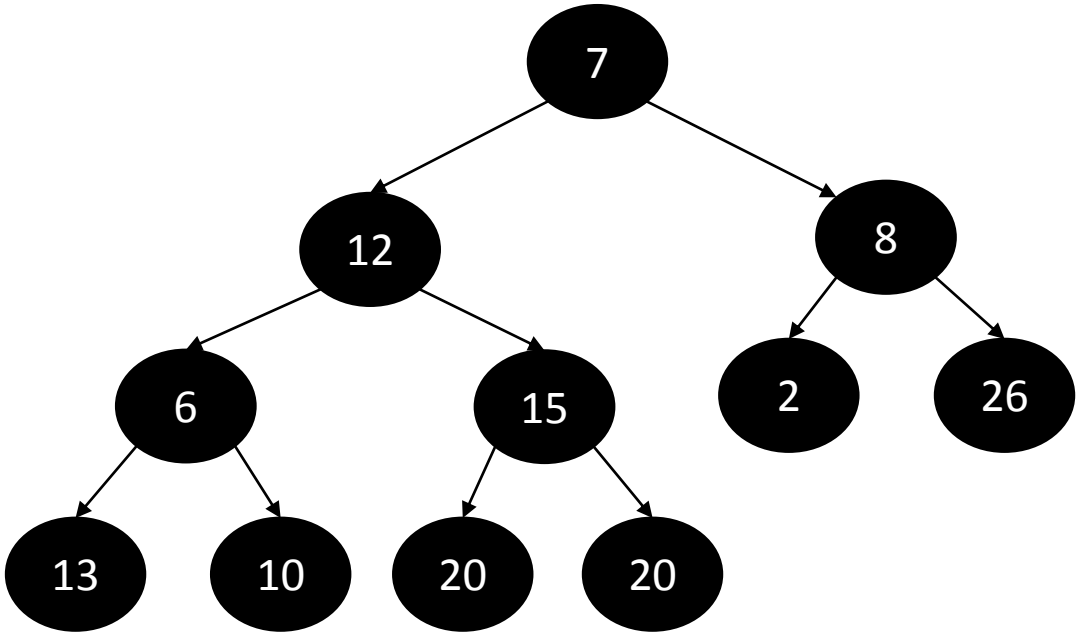
Heapify Example



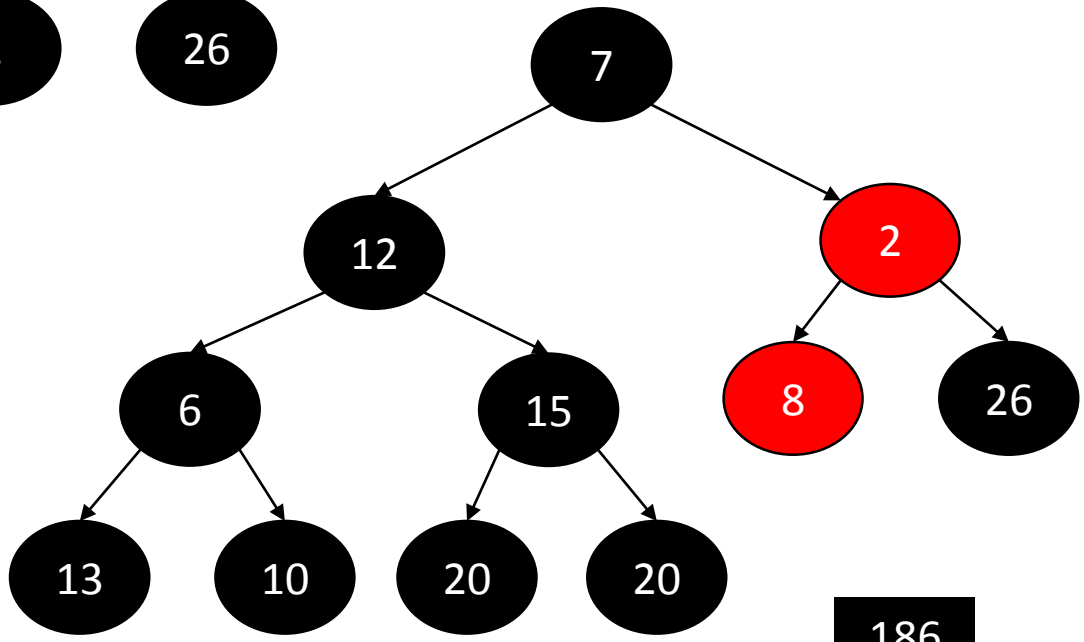
Heapify – Step 2



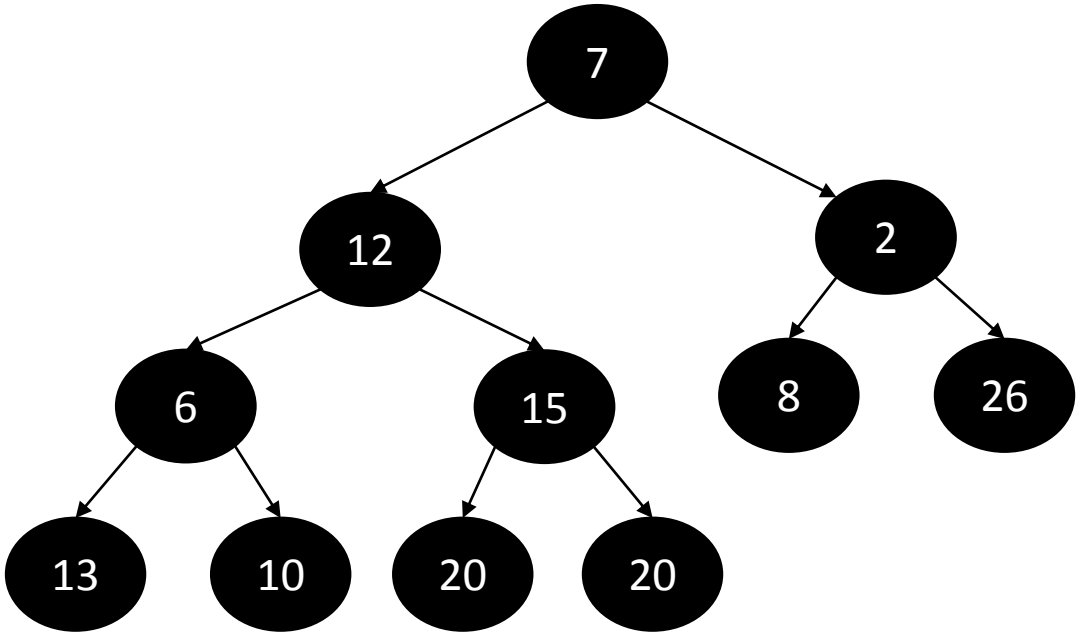
Heapify Example



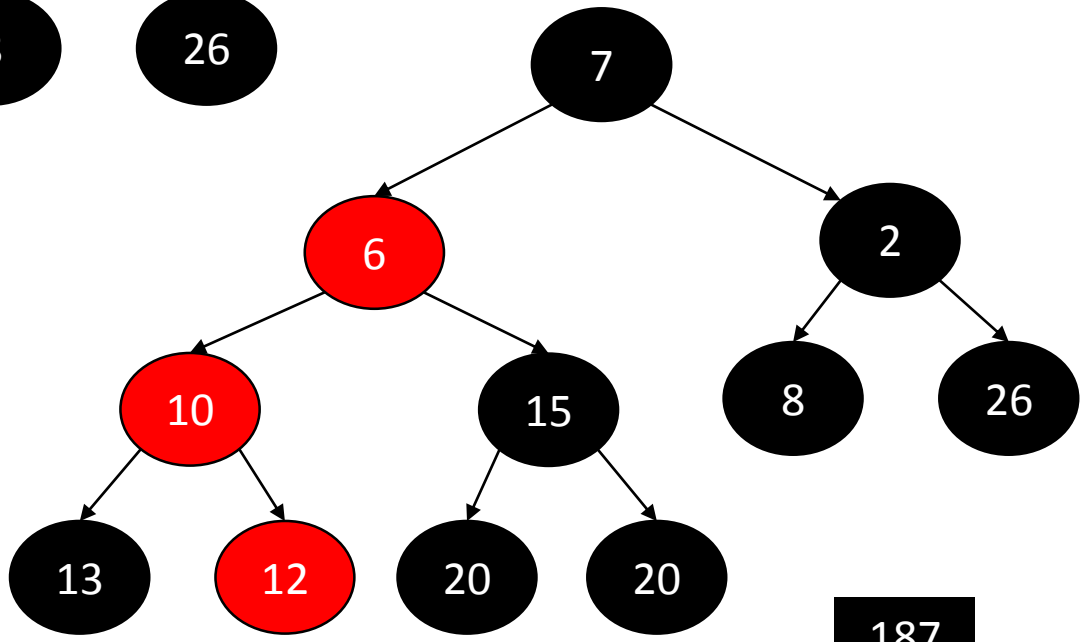
Heapify – Step 3



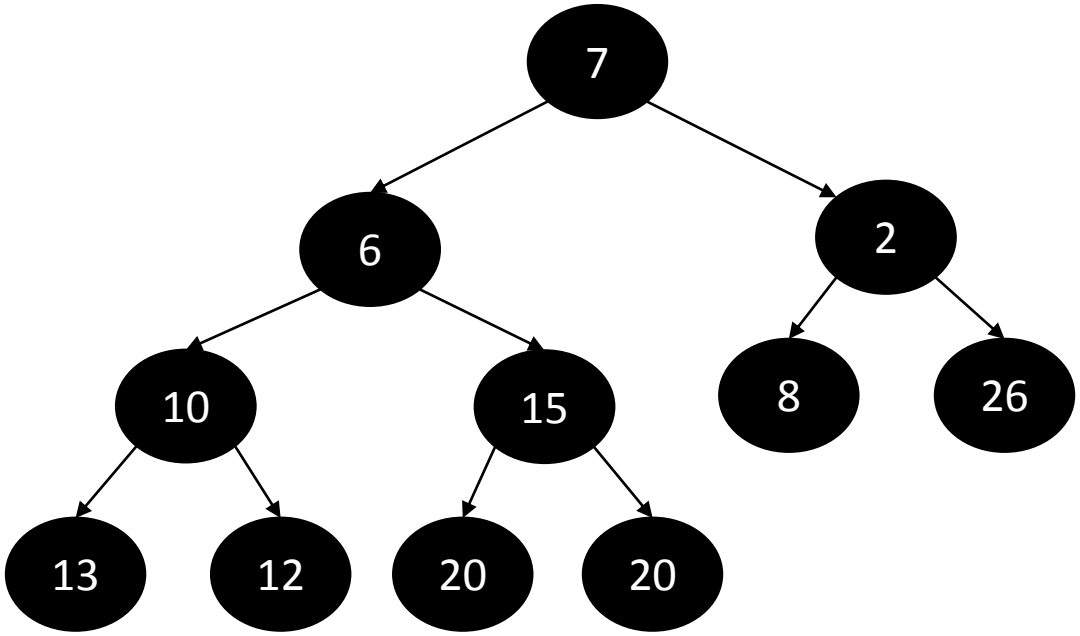
Heapify Example



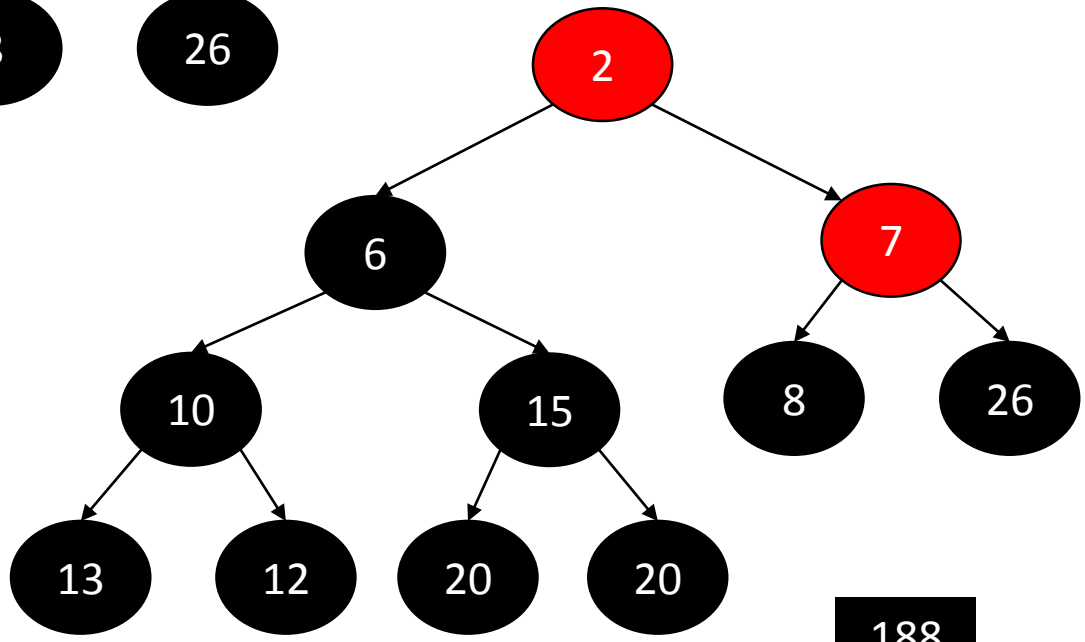
Heapify – Step 4



Heapify Example



Heapify – Step 5



Single Pass In-Memory Indexing

- No upfront creation of termID-DocIDs pairs.
- Uses terms instead of termIDs.
- Repeat if block size exhausted
  - Get tokens from documents one by one.
  - Add terms to dictionary (if not encountered already).
  - Create a posting list or add docID to the existing posting list.
  - Sort the dictionary.
  - Write block to disk if block-size exhausted.
- Merge blocks.

SPIM Indexing

```
SPIMI-INVERT(token_stream)
 1  output_file = NEWFILE()
 2  dictionary = NEWHASH()
 3  while (free memory available)
 4  do token ← next(token_stream)
 5     if term(token) ∉ dictionary
 6        then postings_list = ADDTODICTIONARY(dictionary, term(token))
 7        else postings_list = GETPOSTINGSLIST(dictionary, term(token))
 8     if full(postings_list)
 9        then postings_list = DOUBLEPOSTINGSLIST(dictionary, term(token))
10     ADDTOPOSTINGSLIST(postings_list, docID(token))
11  sorted_terms ← SORTTERMS(dictionary)
12  WRITEBLOCKTODISK(sorted_terms, dictionary, output_file)
13  return output_file
```

Google Data Centers

- Contain commodity machines distributed around the world.
- Estimate: 1 million servers = 3 million processors/cores (Gartner 2007)
- Estimate: Google installs 100,000 servers each quarter.
  - Based on expenditures of 200–250 million dollars per year
- This would be 10% of the computing capacity of the world!?!

Source: cecs.wright.edu/~tkprasad/