Playing with a small DB and storage

Setup visualization libraries

In [1]:
```
def displaySectionCaption(caption, color='coral'):
  html_string = f'<hr><strong><p style="color:{color};font-size:16px;">{caption}</p.<
  display(HTML(html_string))
```

We study a simplified IO model for HDDs and SSDs in CMPE-138. The model will work well in practice, for our query optimzation and data layout problems.

In [2]:
```
import math
from math import ceil, log

# We'll use MBs -- for basic i to MBs
(MB, GB, TB, KB, Bytes) = (1.0, 1024.0, 1024.0*1024.0,
                           1.0/1024.0, 1.0/(1024.0*1024))

# 64 MB-Blocks (default)
PageSizeMB = 64.0*MB
size_of_types = {'int64': 8, 'int32': 4, 'double': 8, 'char': 1} # in bytes

class IOdevice:
  def __init__(self, accessTime, scanSpeed, C_w):
    self.C_r = 1.0  # Cost of reads
    self.C_w = C_w  # Cost of writes relative to reads
    self.accessTime = accessTime
    self.scanSpeed = scanSpeed

  # Read costs: Simple IOcost model using Access time + Scan speeds
  def read_pages_cost(self, numPages):
    # Assume you need to read full pages. (i.e., no partial pages)
    numPages = math.ceil(numPages)
    tsecs = numPages*self.accessTime  # time to access
    tsecs += numPages*PageSizeMB/self.scanSpeed # time to scan
    return (tsecs)

  def write_pages_cost(self, numPages):
    return self.C_w*self.read_pages_cost(numPages)

# Example IO devices in 2024
# Access and Scan speeds in [seconds, MBps], Cw cost of write vs reads.
ram1 = IOdevice(100*pow(10, -9), 100.0*1024, 1.0)
ssd1 = IOdevice(10*pow(10, -6), 5.0*1024, 1.0) # 10 microsecs, 5GBps
hdd1 = IOdevice(10*pow(10, -3), 100.0, 1.0) # 10 millissecs, 100 MBps
# machine to machine over network (modeling a network as an IO device)
m2m1 = IOdevice(10*pow(10, -6), 5.0*1024, 1.0) # 1 micro, 5 GBps

IOdevices1 = {'HDD': hdd1, 'SSD': ssd1, 'RAM': ram1}
```

In [3]:
```
"""
Basic physical table
"""
class Table:
  def __init__(self, sizeInMBs, rowSize):
    self.sizeInMBs = sizeInMBs
    self.rowSize = rowSize
    self.numRows = ceil(self.sizeInMBs/self.rowSize)

    # self.numTuples = numTuples
```

```
    self.isSorted = False
    self.isHPed = False

  # P(R) -- number of Pages for table
  def P(self):
    P = ceil(self.sizeInMBs/PageSizeMB)
    return P
  def RowSize(self):
    return self.rowSize
  def T(self):
    return self.numRows
  def SizeInMBs(self):
    return self.sizeInMBs

  # Keeping track of is table sorted, HPed, or neither (default)
  def Sort(self):
    self.isSorted = True
    self.isHPed = False
  def HP(self):
    self.isSorted = False
    self.isHPed = True
  def Reset(self):
    self.isSorted = False
    self.isHPed = False
```

Exercises:

In [4]:
```
# Spotify Songs Table [songid: int64, title: text, name: text, genre: text]
#    -- Size of row = 8 bytes (int64) + avg size of title+name+genre.
#    -- Assume avg row size = 1024 Bytes
songs_rowSize = 1024.0*Bytes
songs_numRows = 500000000.0 # 500 million songs

"""Problem 1:
Calculate the size (MBs) of SongsTable, and num pages."""



"""Problem 2: Read costs
Compute the cost in seconds to read 100 pages from the SongsTable"""




"""Problem 3: Effect of caching
Read 200 pages. 1st check RAM.
- Cache hit of 90% in RAM.
- For RAM cache misses (the other 10%), 75% are in SSD and 25% are in HDD."""



"""Problem 4:
Suppose you need to read 30 pages and write 10 blocks to the SongsTable.
Calculate the total cost (secs) if the table is on different IO devices."""
```

Out[4]:
```
'Problem 4:\nSuppose you need to read 30 pages and write 10 blocks to the SongsTabl
e.\nCalculate the total cost (secs) if the table is on different IO devices.'
```

How to store DbFiles and execute JOINs

In [5]:
```
from IPython.display import display, HTML
import pandas as pd
import numpy as np
```

```
import math, matplotlib.axes as axes, matplotlib.pyplot as plt, matplotlib.colors as
from enum import Enum
import pickle, os

class Verbose(Enum):
  SILENT = 0
  NORMAL = 1
  VERBOSE = 2
  LISTPAGES = 3

# adjust verbose mode for more debug information
verbosity = Verbose.LISTPAGES

try:
  os.mkdir('join')
  os.mkdir('tmp')
except:
  pass
##
# DbFile: Mimic how to store data on disk, managed by a RAM buffer.
class DbFile:
    def __init__(self, values=None, pages=None, k=None, filepfx = ''):
      self.filepfx = filepfx
      self._pad = (None, None)
      if pages: # Construct new pages based on copying pages
        self.pages = pages
        self.k = len(pages[0])
        self.P = len(pages)
        self.n = 0
        for i, p in enumerate(pages):
          self.n += len(p)
          while len(p) < self.k:
            self.pages[i].append(self._pad)
      elif not values: # Empty page
        self.pages = []
        self.P = 1
        self.k = k
        self.n = 0
      else: # New page based on copying values
        self.k = k
        self.n = len(values)
        self.pages = [[self._pad] * k for i in range(math.ceil(len(values)/self.k))]
        self.P = math.ceil(len(values)/k)
        for i, value in enumerate(values):
          page_num, index = divmod(i, k)
          self.pages[page_num][index] = value
      self.save_to_disk()

    # Mimics reading num-th page.
    #    If the page is in RAM, return the same page. If not, read from disk.
    def read_page(self, page_num):
      if self.pages == []:
        return []
      return self.pages[page_num]
    # Mimics writing num-th page with page_data.
    #    If page in RAM, change the values in page.
    #    If not, read page from disk, and update values.
    def write_page(self, page_num, page_data):
        self.pages[page_num] = page_data

    # Update a specific page value
    def update_page(self, idx, new_val):
      pnum, pidx = divmod(idx, self.k)
      page_data = self.pages[pnum]
      page_data[idx] = new_val
      self.write_page(pnum, page_Data)
```

```
  # Read i-th row. I.e., find (page, location in page) and return row
  def read_ith(self, idx):
    pnum, pidx = divmod(idx, self.k)
    return self.read_page(pnum)[pidx]

  def is_row_valid(self, row):
    return row != self._pad

  # Append values to the end of the DbFile
  def append_page(self, value):
    # add to last page. Or create new page
    def new_page(value):
      # need new page
      newpage = [(None, None)] * self.k
      newpage[0] = value
      self.pages.append([value] + [(None, None)]*(self.k-1))
      self.P = len(self.pages)
    self.n += 1
    if len(self.pages) > 0:
      last_page = self.pages[self.P -1]
      for i in range(len(last_page)):
        if last_page[i] == (None, None):
          last_page[i] = value
          return
    new_page(value)
    self.save_to_disk()

  # Splits files into smaller files
  def split_files(self, num_pages_per_split):
    split_files = []
    for i in range(0, self.P, num_pages_per_split):
      end = min(i+num_pages_per_split, self.P)
      tmpp = self.pages[i:end]
      tmpfdb = DbFile(pages=tmpp,
                k=self.k, filepfx = "tmp/" + self.filepfx+str(i))
      split_files.append(tmpfdb)
    return split_files

  # Store as file
  def save_to_disk(self):
      """Save the current DbFile object to disk."""
      file_name = self.filepfx + '.db'
      with open(file_name, 'wb') as file:
          pickle.dump(self, file)
      #print(f"DbFile saved to {file_name}")

  @classmethod
  def load_from_disk(cls, filepfx):
      """Load a DbFile object from disk."""
      file_name = filepfx + '.db'
      if not os.path.exists(file_name):
          raise FileNotFoundError(f"No such file: {file_name}")
      with open(file_name, 'rb') as file:
          db_file = pickle.load(file)
      print(f"DbFile loaded from {file_name}")
      return db_file

  ## Functions for visuals
  # Below is visualization code [optional read]
  def print_pages(self):
    for p in range(self.P):
      print("Page", p, ":", self.read_page(p))
  def read_all_pages(self):
    return [self.read_page(p) for p in range(self.P)]
    ## Helper functions for us to see pages
```

```
    def format_cell(self, series):
      return ', '.join(str(val) for val in series)

    def print_file(self):
      tdf = self.read_all_pages()
      dfList = [[self.filepfx, len(tdf), self.format_cell(tdf)]]
      df = pd.DataFrame(dfList, columns=["DbFiles", 'Num Pages', "Pages"])
      display(HTML(
          df.to_html(index=False, classes=['table', 'table-bordered'],
                     header=['<th style="word-wrap: break-word; max-width: 30px;">Pag
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [6]:
```
import random, heapq, string

def gen_random_string(string_length):
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(string_length))

# create sample tables with n-rows of [<int, string of 'l' chars>]
def gen_random_rows(l, n):
  return [(random.randint(1, 500), gen_random_string(l), '..') for i in range(n)]

# Generate n rows with integer keys in descending order and random strings
def gen_reverse_rows(l, n):
  return [(100 - i, gen_random_string(l), '..') for i in range(n)]
random.seed(3141)
```

In [7]:
```
""" Generate a Songs table with 100 tuples, with k=3 rows per page"""
Songs = DbFile(values=gen_random_rows(2, 100), k=3, filepfx='Songs')
Songs.print_file()
```

DbFiles	Num Pages	Pages
Songs	34	[(188, 'kg', '..'), (492, 'oa', '..'), (359, 'cl', '..')], [(163, 'kn', '..'), (458, 'dx', '..'), (210, 'wl', '..')], [(123, 'cm', '..'), (494, 'bd', '..'), (486, 'vb', '..')], [(474, 'nt', '..'), (358, 'vu', '..'), (35, 'ez', '..')], [(21, 'lk', '..'), (288, 'nk', '..'), (229, 'sq', '..')], [(450, 'gm', '..'), (102, 'ch', '..'), (214, 'nx', '..')], [(126, 'my', '..'), (203, 'kz', '..'), (169, 'ak', '..')], [(234, 'yc', '..'), (357, 'in', '..'), (396, 'at', '..')], [(167, 'uw', '..'), (232, 'pm', '..'), (230, 'oa', '..')], [(131, 'fz', '..'), (477, 'gi', '..'), (327, 'zs', '..')], [(29, 'uh', '..'), (305, 'hp', '..'), (124, 'xi', '..')], [(168, 'sq', '..'), (489, 'uc', '..'), (279, 'fz', '..')], [(363, 'ma', '..'), (80, 'br', '..'), (318, 'oa', '..')], [(320, 'ut', '..'), (101, 'ya', '..'), (197, 'rm', '..')], [(402, 'mn', '..'), (255, 'uf', '..'), (246, 'ir', '..')], [(14, 'nc', '..'), (484, 'xd', '..'), (301, 'cp', '..')], [(243, 'hx', '..'), (18, 'lx', '..'), (328, 'jf', '..')], [(446, 'ew', '..'), (178, 'qf', '..'), (192, 'pc', '..')], [(432, 'it', '..'), (179, 'dg', '..'), (77, 'fq', '..')], [(41, 'na', '..'), (379, 'oj', '..'), (232, 'pk', '..')], [(194, 'ro', '..'), (110, 'um', '..'), (494, 'jf', '..')], [(375, 'ur', '..'), (177, 'fg', '..'), (453, 'lq', '..')], [(364, 'mq', '..'), (80, 'zy', '..'), (359, 'mj', '..')], [(292, 'xe', '..'), (207, 'hh', '..'), (134, 'mp', '..')], [(440, 'fi', '..'), (419, 'qp', '..'), (428, 've', '..')], [(383, 'rx', '..'), (166, 'aa', '..'), (266, 'bd', '..')], [(242, 'oc', '..'), (137, 'rd', '..'), (25, 'me', '..')], [(194, 'iq', '..'), (66, 'at', '..'), (261, 'dr', '..')], [(310, 'un', '..'), (491, 'uu', '..'), (416, 'nl', '..')], [(299, 'ha', '..'), (353, 'fr', '..'), (85, 'gk', '..')], [(141, 'yb', '..'), (96, 'yo', '..'), (23, 'ba', '..')], [(167, 'zp', '..'), (70, 'vt', '..'), (499, 'dh', '..')], [(63, 'iv', '..'), (429, 'al', '..'), (170, 'hy', '..')], [(17, 'oa', '..'), (None, None), (None, None)]

In [8]:
```
""" Generate a Listens table with 100 tuples, with k=7 rows per page"""
Listens = DbFile(values=gen_random_rows(2, 100), k=7, filepfx='Listens')
Listens.print_file()
```

DbFiles	Num Pages	Pages
Listens	15	[(188, 'mj', '..'), (484, 'vv', '..'), (37, 'ko', '..'), (464, 'xh', '..'), (52, 'iw', '..'), (219, 'qx', '..'), (328, 'ka', '..')], [(286, 'tu', '..'), (405, 'lg', '..'), (468, 'yh', '..'), (277, 'wm', '..'), (333, 'zp', '..'), (401, 'hg', '..'), (316, 'hl', '..')], [(348, 'nq', '..'), (136, 'ne', '..'), (354, 'vs', '..'), (88, 'wf', '..'), (369, 'mq', '..'), (96, 'lc', '..'), (251, 'kx', '..')], [(303, 'pc', '..'), (159, 'hb', '..'), (281, 'rk', '..'), (184, 'au', '..'), (96, 'tz', '..'), (203, 'mf', '..'), (405, 'ug', '..')], [(411, 'yc', '..'), (412, 'zs', '..'), (294, 'ei', '..'), (95, 'qz', '..'), (473, 'ug', '..'), (130, 'qx', '..'), (248, 'ra', '..')], [(170, 'dv', '..'), (181, 'nw', '..'), (399, 'mp', '..'), (404, 'bw', '..'), (175, 'fy', '..'), (279, 'tu', '..'), (238, 'aj', '..')], [(365, 'gr', '..'), (146, 'kr', '..'), (46, 'ke', '..'), (344, 'va', '..'), (281, 'uj', '..'), (26, 'yn', '..'), (138, 'ky', '..')], [(429, 'gb', '..'), (10, 'ct', '..'), (215, 'yl', '..'), (351, 'pd', '..'), (301, 'xv', '..'), (243, 'bh', '..'), (488, 'km', '..')], [(307, 'ei', '..'), (472, 'ni', '..'), (226, 'sw', '..'), (176, 'kb', '..'), (353, 'ht', '..'), (338, 'ab', '..'), (422, 'px', '..')], [(487, 'yl', '..'), (307, 'tt', '..'), (234, 'vi', '..'), (417, 'be', '..'), (242, 'il', '..'), (22, 'ir', '..'), (266, 'ox', '..')], [(117, 'ft', '..'), (201, 'js', '..'), (208, 'wr', '..'), (311, 'nx', '..'), (213, 'tc', '..'), (433, 'jv', '..'), (310, 'ba', '..')], [(347, 'ws', '..'), (180, 'dv', '..'), (43, 'dg', '..'), (304, 'lf', '..'), (484, 'kc', '..'), (261, 'zr', '..'), (97, 'tx', '..')], [(266, 'cl', '..'), (235, 'fx', '..'), (66, 'vy', '..'), (326, 'qe', '..'), (78, 'dl', '..'), (385, 'mw', '..'), (464, 'kk', '..')], [(218, 'ib', '..'), (150, 'xs', '..'), (411, 'ur', '..'), (67, 'xt', '..'), (373, 'df', '..'), (44, 'wp', '..'), (500, 'ya', '..')], [(446, 'ub', '..'), (114, 'ft', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

In [9]:
```
""" Generate a Listens table with 1000 tuples, with k=22 rows per page"""
Listens = DbFile(values=gen_random_rows(2, 1000), k=22, filepfx='Listens')
Listens.print_file()
```

DbFiles	Num Pages	Pages
Listens	46	[(356, 'ny', '..'), (153, 'ft', '..'), (479, 'ug', '..'), (6, 'qc', '..'), (71, 'fz', '..'), (120, 'id', '..'), (180, 'va', '..'), (83, 'tz', '..'), (215, 'xu', '..'), (78, 'zh', '..'), (442, 'ak', '..'), (422, 'uj', '..'), (181, 'ed', '..'), (69, 'hz', '..'), (406, 'jb', '..'), (417, 'xi', '..'), (372, 'kh', '..'), (297, 'ya', '..'), (250, 'bg', '..'), (448, 'ao', '..'), (363, 'ra', '..'), (150, 'mg', '..')], [(79, 'rb', '..'), (328, 'tu', '..'), (340, 'st', '..'), (264, 'wq', '..'), (305, 'vq', '..'), (238, 'uv', '..'), (266, 'wx', '..'), (232, 'yz', '..'), (144, 'vj', '..'), (393, 'je', '..'), (359, 'cr', '..'), (371, 'jc', '..'), (187, 'nh', '..'), (146, 'dp', '..'), (86, 'yd', '..'), (428, 'lo', '..'), (124, 'pl', '..'), (296, 'qq', '..'), (439, 'cg', '..'), (311, 'cu', '..'), (473, 'dw', '..'), (329, 'te', '..')], [(272, 'hv', '..'), (60, 'gr', '..'), (486, 'rz', '..'), (460, 'du', '..'), (400, 'ng', '..'), (274, 'oj', '..'), (189, 'nw', '..'), (269, 'nk', '..'), (406, 'az', '..'), (437, 'ly', '..'), (6, 'ig', '..'), (428, 'mw', '..'), (456, 'nm', '..'), (178, 'dd', '..'), (136, 'wk', '..'), (199, 'xf', '..'), (8, 'ur', '..'), (500, 'xr', '..'), (190, 'ug', '..'), (303, 'gh', '..'), (490, 'pv', '..'), (367, 'ka', '..')], [(133, 'kd', '..'), (451, 'hj', '..'), (395, 'wu', '..'), (416, 'dd', '..'), (95, 'tk', '..'), (281, 'dg', '..'), (194, 'oq', '..'), (222, 'cr', '..'), (79, 'ch', '..'), (370, 'ud', '..'), (286, 'ux', '..'), (61, 'li', '..'), (4, 'jz', '..'), (455, 'nw', '..'), (275, 'po', '..'), (358, 'ba', '..'), (321, 'nz', '..'), (449, 'dz', '..'), (346, 'vn', '..'), (286, 'jr', '..'), (86, 'fa', '..'), (273, 'fx', '..')], [(415, 'cb', '..'), (164, 'mt', '..'), (415, 'ik', '..'), (62, 'ee', '..'), (161, 'mv', '..'), (77, 'yn', '..'), (313, 'ib', '..'), (5, 'kk', '..'), (46, 'wl', '..'), (54, 'sj', '..'), (420, 'ln', '..'), (340, 'fj', '..'), (320, 'rw', '..'), (272, 'lo', '..'), (166, 'qa', '..'), (71, 'co', '..'), (186, 'hr', '..'), (104, 'on', '..'), (120, 'dy', '..'), (17, 'ok', '..'), (82, 'sf', '..'), (75, 'zp', '..')], [(307, 'mc', '..'), (150, 'ie', '..'), (77, 'in', '..'), (451, 'is', '..'), (148, 'fx', '..'), (416, 'tg', '..'), (4, 'ig', '..'), (77, 'pz', '..'), (231, 'fc', '..'), (69, 'aj', '..'), (351, 'vs', '..'), (163, 'pn', '..'), (312, 'tm', '..'), (394, 'yq', '..'), (32, 'ad', '..'), (249, 'dv', '..'), (59, 'my', '..'), (271, 've', '..'), (97, 'qu', '..'), (157, 'xb', '..'), (458, 'cm', '..'), (464, 'aa', '..')], [(161, 'xe', '..'), (340, 'ba', '..'), (337, 'qr', '..'), (202, 'eo', '..'), (380, 'vl', '..'), (43, 'fn', '..'), (216, 'he', '..'), (465, 'rl', '..'), (49, 'hl', '..'), (451, 'an', '..'), (324, 'mk', '..'), (62, 'aq', '..'), (189, 'pq', '..'), (478, 'al', '..'), (169, 'ls', '..'), (19, 'mc', '..'), (69, 'xv', '..'), (254, 'fy', '..'), (305, 'yw', '..'), (315, 'ey', '..'), (195, 'lk', '..'), (410, 'xz', '..')], [(377, 'oq', '..'), (339, 'kq', '..'), (269, 'jp', '..'), (53, 'jw', '..'), (329, 'oc', '..'), (150, 'zk', '..'), (430, 'sj', '..'), (438, 'fd', '..'), (301, 'lk', '..'), (132, 'if', '..'), (230, 'gb', '..'), (482, 'vg', '..'), (254, 'ui', '..'), (144, 'xd', '..'), (438, 'xf', '..'), (200, 'xb', '..'), (188, 'zl', '..'), (188, 'qe', '..'), (207, 'dn', '..'), (489, 'vv', '..'), (373, 'jm', '..'), (399, 'hn', '..')], [(184, 'iy', '..'), (314, 'qb', '..'), (449, 'uj', '..'), (159, 'ot', '..'), (63, 'yc', '..'), (303, 'fa', '..'), (194, 'cc', '..'), (230, 'hf', '..'), (30, 'an', '..'), (157, 'nd', '..'), (92, 'cu', '..'), (434, 'ma', '..'), (287, 'jb', '..'), (89, 'nu', '..'), (165, 'wt', '..'), (273, 're', '..'), (104, 'vk', '..'), (14, 'rf', '..'), (217, 'cx', '..'), (94, 'nv', '..'), (256, 'vt', '..'), (493, 'pd', '..')], [(221, 'ku', '..'), (154, 'fx', '..'), (36, 'zc', '..'), (374, 'ba', '..'), (232, 'nl', '..'), (438, 'ug', '..'), (138, 'he', '..'), (370, 'hv', '..'), (174, 'ip', '..'), (429, 'pk', '..'), (295, 'ux', '..'), (31, 'gm', '..'), (285, 'nq', '..'), (180, 'uv', '..'), (493, 'nm', '..'), (357, 'xr', '..'), (167, 'gc', '..'), (239, 'du', '..'), (291, 'hr', '..'), (324, 'eg', '..'), (410, 'tu', '..'), (114, 'fv', '..')], [(363, 'si', '..'), (85, 'vb', '..'), (277, 'ce', '..'), (406, 'ob', '..'), (354, 'ei', '..'), (481, 'qn', '..'), (48, 'vd', '..'), (470, 'vf', '..'), (105, 'fy', '..'), (36, 'pu', '..'), (448, 'in', '..'), (376, 'ki', '..'), (408, 'kw', '..'), (273, 'cc', '..'), (270, 'ii', '..'), (444, 'hz', '..'), (359, 'nc', '..'), (86, 'mx', '..'), (84, 'wq', '..'), (427, 'pu', '..'), (427, 'lz', '..'), (147, 'dl', '..')], [(230, 'sy', '..'), (90, 'ub', '..'), (174, 'dp', '..'), (68, 'wt', '..'), (31, 'cs', '..'), (38, 'qg', '..'), (448, 'sd', '..'), (373, 'ek', '..'), (429, 'it', '..'), (190, 'hr', '..'), (405, 'xx', '..'), (78, 'vy', '..'), (324, 'sf', '..'), (250, 'rl', '..'), (102, 'ie', '..'), (436, 'ej', '..'), (114, 'wl', '..'), (222, 'qv', '..'), (472, 'ju', '..'), (229, 'df', '..'), (27, 'hp', '..'), (172, 'av', '..')], [(100, 'ol', '..'), (274, 'ds', '..'), (389, 'li', '..'), (499, 'zt', '..'), (500, 'eo', '..'), (224, 'om', '..'), (113, 'xe', '..'), (322, 'ig', '..'), (122, 'jy', '..'), (467, 'ls', '..'), (471, 'ps', '..'), (158, 'bw', '..'), (259, 'yr', '..'), (60, 'fb', '..'), (371, 'fc', '..'), (114, 'gf', '..'), (467, 'lr', '..'), (346, 'ss', '..'), (279, 'cp', '..'), (474, 'jd', '..'), (497, 'qr', '..'), (257, 'qi', '..')], [(403, 'cg', '..'), (479, 'ww', '..'), (418, 'va', '..'), (408, 'nj', '..'), (227, 'jb', '..'), (38, 'ak', '..'), (5, 'ej', '..'), (432, 'fh', '..'), (398, 'xn', '..'), (360, 'cq', '..'), (355, 'io', '..'), (494, 'dp', '..'), (164, 'yy', '..'), (442, 'jf', '..'), (460, 'tu', '..'), (102, 'ss', '..'), (162, 'wh', '..'), (441, 'qk', '..'), (278, 'lc', '..'), (130, 'cn', '..'), (471, 'av', '..'), (375, 'eg', '..')], [(406, 'fm', '..'), (10, 'lv', '..'), (285, 'rl', '..'), (330, 'xv', '..'), (437, 'es', '..'), (322, 'tm', '..'), (188, 'er', '..'), (209, 'gu', '..'), (23, 'yw', '..'), (394, 'te', '..'), (275, 'pt', '..'), (434, 'ao', '..'), (494, 'tv', '..'), (259, 'am', '..'), (458, 'zh', '..'), (16, 'bb', '..'), (370, 'ce', '..'), (44, 'pm', '..'), (178, 'nf', '..'), (238, 'vp', '..'), (143, 'cu', '..'), (405, 'wl', '..')], [(51, 'sm', '..'), (393, 'ct', '..'), (9, 'tf', '..'), (119, 'ao', '..'), (10, 'lp', '..'), (26, 'ps', '..'), (111, 'bo', '..'), (363, 'ik', '..'), (349, 'rv', '..'), (84, 'zo', '..'), (151, 'ou', '..'), (399, 'gv', '..'), (248, 'vs', '..'), (439, 'ul', '..'), (326, 'gd', '..'), (148, 'zg', '..'), (46, 'cu', '..'), (66, 'xf', '..'), (387, 'ha', '..'), (377, 'ii', '..'), (384, 'dq', '..'), (343, 'ce', '..')], [(443, 'va', '..'), (315, 'ao', '..'), (21, 'ua', '..'), (40, 'df', '..'), (481, 'ct', '..'), (403, 'ni', '..'), (452, 'aw', '..'), (18, 'hc', '..'), (41, 'hg', '..'), (270, 'zd', '..'), (72, 'bs', '..'), (423, 'rl', '..'), (385, 'tl', '..'), (268, 'pf', '..'), (48, 'mk', '..'), (384, 'cj', '..'), (225, 'rx', '..'), (262, 'mq', '..'), (216, 'sl', '..'), (428, 'ip', '..'), (119, 'ig', '..'), (285, 'pe', '..')], [(308, 'gr', '..'), (349, 'kr', '..'), (131, 'vs', '..'), (414, 'wm', '..'), (397, 'ke', '..'), (456, 'sv', '..'), (405, 'vk', '..'), (55, 'vf', '..'), (8, 'di', '..'), (286, 'ei', '..'), (348, 'ch', '..'), (287, 'dd', '..'), (463, 'hv', '..'), (113, 'ez', '..'), (482, 'bz', '..'), (285, 'id', '..'), (429, 'xc', '..'), (124, 'sz', '..'), (247, 'mk', '..'), (126, 'll', '..'), (268, 'tg', '..'), (209, 'oc', '..')], [(68, 'cd', '..'), (257, 'mj', '..'), (486, 'dv', '..'), (154, 'dg', '..'), (278, 'vt', '..'), (111, 'hi', '..'), (84, 'sg', '..'), (310, 'hd', '..'), (79,

DbFiles	Num Pages	Pages
		'wa', '..'), (245, 'cz', '..'), (230, 'cm', '..'), (432, 'ol', '..'), (96, 'ue', '..'), (104, 'lz', '..'), (388, 'xp', '..'), (154, 'gv', '..'), (166, 'en', '..'), (167, 'gm', '..'), (168, 'hz', '..'), (258, 'rf', '..'), (391, 'vl', '..'), (154, 'as', '..')], [(380, 'in', '..'), (171, 'lr', '..'), (49, 'xk', '..'), (148, 'vp', '..'), (72, 'vr', '..'), (149, 'ui', '..'), (103, 'yw', '..'), (459, 'sd', '..'), (254, 'iw', '..'), (68, 'nf', '..'), (411, 'su', '..'), (207, 'fz', '..'), (45, 'ps', '..'), (430, 'ch', '..'), (232, 'is', '..'), (344, 'ep', '..'), (394, 'vs', '..'), (20, 'ud', '..'), (12, 'ry', '..'), (315, 'qj', '..'), (160, 'wv', '..'), (392, 'ju', '..')], [(346, 'wc', '..'), (301, 'xe', '..'), (350, 'sj', '..'), (106, 'cy', '..'), (384, 'gf', '..'), (480, 'bt', '..'), (477, 'zn', '..'), (262, 'ws', '..'), (251, 'vs', '..'), (431, 'qb', '..'), (379, 'pe', '..'), (67, 'ls', '..'), (61, 'vv', '..'), (472, 'eq', '..'), (400, 'uz', '..'), (373, 'ej', '..'), (68, 'an', '..'), (113, 'sr', '..'), (89, 'ms', '..'), (354, 'no', '..'), (60, 'xu', '..'), (69, 'vo', '..')], [(307, 'wp', '..'), (473, 'le', '..'), (365, 'fy', '..'), (176, 'qj', '..'), (500, 'oi', '..'), (9, 'xr', '..'), (481, 'wt', '..'), (459, 'jx', '..'), (167, 'vw', '..'), (328, 'zm', '..'), (203, 'bw', '..'), (479, 'xn', '..'), (438, 'uq', '..'), (47, 'vo', '..'), (346, 'no', '..'), (198, 'hm', '..'), (127, 'ni', '..'), (494, 'dk', '..'), (384, 'ht', '..'), (432, 'tg', '..'), (96, 'fl', '..'), (65, 'xc', '..')], [(456, 'tf', '..'), (271, 'wr', '..'), (22, 'oi', '..'), (428, 'tm', '..'), (437, 'tb', '..'), (383, 'nx', '..'), (439, 'ro', '..'), (24, 'pr', '..'), (214, 'qn', '..'), (326, 'on', '..'), (463, 'kc', '..'), (219, 'cz', '..'), (120, 'sl', '..'), (155, 'nd', '..'), (373, 'ag', '..'), (440, 'gq', '..'), (215, 'ce', '..'), (433, 'tn', '..'), (16, 're', '..'), (158, 'bz', '..'), (400, 'ra', '..'), (219, 'zb', '..')], [(84, 'kt', '..'), (354, 'qz', '..'), (320, 'gg', '..'), (157, 'zm', '..'), (328, 'vx', '..'), (196, 'ie', '..'), (283, 'kk', '..'), (184, 'ca', '..'), (440, 'xh', '..'), (37, 'py', '..'), (242, 'lh', '..'), (41, 'xg', '..'), (50, 'df', '..'), (287, 'st', '..'), (171, 'ci', '..'), (284, 'op', '..'), (22, 'no', '..'), (132, 'yf', '..'), (35, 'wc', '..'), (147, 'ia', '..'), (435, 'ws', '..'), (435, 'cc', '..')], [(117, 'ov', '..'), (62, 'mh', '..'), (116, 'ts', '..'), (60, 'jr', '..'), (115, 'rg', '..'), (354, 'pp', '..'), (493, 'bs', '..'), (312, 'tl', '..'), (421, 'ah', '..'), (442, 'zv', '..'), (148, 'ds', '..'), (334, 'eg', '..'), (251, 'vf', '..'), (302, 'es', '..'), (483, 'ko', '..'), (440, 'qh', '..'), (240, 'tr', '..'), (130, 'zp', '..'), (405, 'ej', '..'), (495, 'co', '..'), (233, 'wd', '..'), (14, 'sf', '..')], [(3, 'yf', '..'), (416, 'yf', '..'), (405, 'dg', '..'), (218, 'ua', '..'), (436, 'av', '..'), (89, 'gf', '..'), (341, 'pv', '..'), (343, 'uo', '..'), (466, 'nq', '..'), (192, 'hj', '..'), (450, 'oz', '..'), (39, 'cs', '..'), (259, 'ty', '..'), (92, 'kd', '..'), (66, 'xr', '..'), (144, 'nr', '..'), (265, 'yi', '..'), (67, 'of', '..'), (498, 'pp', '..'), (314, 'ql', '..'), (336, 'kf', '..'), (176, 'dp', '..')], [(31, 'rc', '..'), (222, 'qv', '..'), (259, 'ev', '..'), (394, 'rn', '..'), (417, 'oz', '..'), (293, 'vp', '..'), (492, 'co', '..'), (147, 'wq', '..'), (388, 'jj', '..'), (474, 'cz', '..'), (150, 'hv', '..'), (83, 'bz', '..'), (109, 'sf', '..'), (416, 'lr', '..'), (300, 'jr', '..'), (154, 'qr', '..'), (492, 'ek', '..'), (272, 'pe', '..'), (43, 'ly', '..'), (18, 'hx', '..'), (193, 'gn', '..'), (457, 'ge', '..')], [(83, 'lg', '..'), (288, 'sm', '..'), (35, 'al', '..'), (60, 'zh', '..'), (457, 'cv', '..'), (79, 'ui', '..'), (46, 'fq', '..'), (42, 'mt', '..'), (387, 'za', '..'), (272, 'hf', '..'), (112, 'on', '..'), (308, 'lg', '..'), (313, 'so', '..'), (98, 'lu', '..'), (477, 'mw', '..'), (95, 'pe', '..'), (343, 'yj', '..'), (27, 'yn', '..'), (285, 'mc', '..'), (169, 'at', '..'), (244, 'st', '..'), (10, 'qa', '..')], [(437, 'tw', '..'), (306, 'pj', '..'), (203, 'sq', '..'), (244, 'dl', '..'), (422, 'ir', '..'), (249, 'ry', '..'), (382, 'tt', '..'), (477, 'fj', '..'), (362, 'cq', '..'), (362, 'sd', '..'), (319, 'oc', '..'), (494, 'cr', '..'), (107, 'lq', '..'), (483, 'dx', '..'), (337, 'gg', '..'), (53, 'zy', '..'), (355, 'qi', '..'), (356, 'ph', '..'), (485, 'ws', '..'), (165, 'jw', '..'), (184, 'ci', '..'), (23, 'wh', '..')], [(295, 'ls', '..'), (202, 'uo', '..'), (116, 'wi', '..'), (376, 'ss', '..'), (48, 'mr', '..'), (374, 'vh', '..'), (240, 'ah', '..'), (82, 'dn', '..'), (176, 'nj', '..'), (93, 'nq', '..'), (371, 'ts', '..'), (115, 'rj', '..'), (17, 'oy', '..'), (152, 'da', '..'), (268, 'yv', '..'), (55, 'io', '..'), (334, 'xz', '..'), (241, 'ro', '..'), (154, 'cv', '..'), (254, 'ln', '..'), (466, 'lp', '..'), (180, 'hh', '..')], [(222, 'dt', '..'), (380, 'ri', '..'), (132, 'xd', '..'), (122, 'lw', '..'), (191, 'wd', '..'), (185, 'vh', '..'), (23, 'ja', '..'), (101, 'uf', '..'), (412, 'fg', '..'), (189, 'fh', '..'), (92, 'wy', '..'), (9, 'ke', '..'), (18, 'sx', '..'), (87, 'ln', '..'), (189, 'hr', '..'), (316, 'uv', '..'), (343, 'wj', '..'), (278, 'kp', '..'), (122, 'qk', '..'), (137, 'cg', '..'), (319, 'bx', '..'), (356, 'fe', '..')], [(13, 'gp', '..'), (311, 'qn', '..'), (27, 'xs', '..'), (415, 'oq', '..'), (271, 'hs', '..'), (317, 'dq', '..'), (96, 'ao', '..'), (13, 'bz', '..'), (255, 'mc', '..'), (337, 'iy', '..'), (440, 'gl', '..'), (454, 'ye', '..'), (79, 'gt', '..'), (223, 'lk', '..'), (185, 'tn', '..'), (184, 'xb', '..'), (36, 'az', '..'), (220, 'bg', '..'), (87, 'nt', '..'), (238, 'go', '..'), (53, 'ue', '..'), (100, 'zl', '..')], [(487, 'kz', '..'), (129, 'tt', '..'), (409, 'zv', '..'), (359, 'mu', '..'), (373, 'eb', '..'), (149, 'vo', '..'), (331, 'gr', '..'), (460, 'te', '..'), (371, 'nr', '..'), (293, 'ls', '..'), (129, 'du', '..'), (130, 'dd', '..'), (451, 'mr', '..'), (84, 'or', '..'), (177, 'zv', '..'), (498, 'ue', '..'), (65, 'wl', '..'), (26, 'wv', '..'), (171, 'br', '..'), (497, 'zt', '..'), (351, 'qx', '..'), (15, 'kg', '..')], [(378, 'pv', '..'), (443, 'ey', '..'), (257, 'qt', '..'), (351, 'jf', '..'), (478, 'sb', '..'), (361, 'oz', '..'), (147, 'uo', '..'), (157, 'xj', '..'), (251, 'vk', '..'), (355, 'iw', '..'), (450, 'zu', '..'), (274, 'wn', '..'), (436, 'rk', '..'), (222, 'yq', '..'), (52, 'ge', '..'), (465, 'so', '..'), (114, 'ds', '..'), (272, 'bh', '..'), (10, 'ri', '..'), (436, 'vc', '..'), (224, 'yh', '..'), (220, 'ua', '..')], [(80, 'au', '..'), (397, 'sa', '..'), (345, 'xy', '..'), (273, 'kz', '..'), (150, 'qj', '..'), (419, 'zj', '..'), (303, 'mu', '..'), (131, 'jd', '..'), (88, 'dm', '..'), (218, 'gx', '..'), (252, 'll', '..'), (60, 'rg', '..'), (135, 'vs', '..'), (325, 'yp', '..'), (35, 'mg', '..'), (129, 'ho', '..'), (126, 'nf', '..'), (231, 'um', '..'), (77, 'gv', '..'), (34, 'xr', '..'), (349, 'fh', '..'), (189, 'yw', '..')], [(65, 'iq', '..'), (344, 'jy', '..'), (475, 'kn', '..'), (138, 'qj', '..'), (172, 'tg', '..'), (6, 'oh', '..'), (320, 'dn', '..'), (449, 'xj', '..'), (138, 'ru', '..'), (471, 'ya', '..'), (452, 'ir', '..'), (179, 'yt', '..'), (248, 'po', '..'), (46, 'aj', '..'), (143, 'vh', '..'), (484, 'co', '..'), (22, 'om', '..'), (61, 'io', '..'), (351, 'ar', '..'), (497, 'gf', '..'), (275, 'vk', '..'), (296, 'sg', '..')], [(446, 'sh', '..'), (166, 'iq', '..'), (417, 'nn', '..'), (215, 'uh', '..'), (258, 'ov', '..'), (412, 'gh', '..'), (52, 'em', '..'), (366, 'iv', '..'), (130, 'pl', '..'), (459, 'zy', '..'), (123, 'lz', '..'), (403, 'eq', '..'), (469, 'sp', '..'), (108, 'ok', '..'), (188, 'lw', '..'), (416, 'id', '..'), (250, 'kw', '..'), (214, 'jo', '..'), (244, 'gf', '..'), (95, 'ck', '..'), (437, 'lw', '..'), (134, 'ak', '..')], [(220, 'gs', '..'),

DbFiles	Num Pages	Pages
		(51, 'pv', '..'), (281, 'kg', '..'), (494, 'ae', '..'), (24, 'bh', '..'), (32, 'jl', '..'), (325, 'nl', '..'), (321, 'od', '..'), (84, 'li', '..'), (226, 'tb', '..'), (367, 'iy', '..'), (72, 'iy', '..'), (377, 'qx', '..'), (366, 'jp', '..'), (90, 'vp', '..'), (381, 'zc', '..'), (62, 'sq', '..'), (186, 'ou', '..'), (320, 'is', '..'), (475, 'vz', '..'), (123, 'wl', '..'), (103, 'wn', '..')], [(26, 'rl', '..'), (398, 'lf', '..'), (102, 'hm', '..'), (267, 'fo', '..'), (319, 'dw', '..'), (456, 'hn', '..'), (326, 'xq', '..'), (117, 'kl', '..'), (125, 'dd', '..'), (33, 'oi', '..'), (457, 'bn', '..'), (266, 'ea', '..'), (277, 'pq', '..'), (42, 'wk', '..'), (334, 'xy', '..'), (54, 'ls', '..'), (415, 'aj', '..'), (272, 'oi', '..'), (12, 'zb', '..'), (177, 'zv', '..'), (175, 'ui', '..'), (148, 'un', '..')], [(85, 'xt', '..'), (300, 'uf', '..'), (360, 'ix', '..'), (124, 'fj', '..'), (328, 'dq', '..'), (218, 'jl', '..'), (346, 'dp', '..'), (284, 'gb', '..'), (287, 'um', '..'), (299, 'qa', '..'), (224, 'dn', '..'), (16, 'df', '..'), (291, 'sq', '..'), (432, 'iv', '..'), (159, 'mm', '..'), (462, 'cc', '..'), (199, 'bs', '..'), (10, 'yq', '..'), (141, 'es', '..'), (431, 'gp', '..'), (32, 'lb', '..'), (341, 'np', '..')], [(267, 'lj', '..'), (469, 'cg', '..'), (90, 'si', '..'), (8, 'sm', '..'), (122, 'ef', '..'), (416, 'pp', '..'), (344, 'al', '..'), (462, 'yb', '..'), (143, 'tl', '..'), (47, 'co', '..'), (154, 'sr', '..'), (255, 'yv', '..'), (389, 'fe', '..'), (120, 'sh', '..'), (307, 'lo', '..'), (99, 'hy', '..'), (315, 'ik', '..'), (41, 'gu', '..'), (188, 'bv', '..'), (170, 'fh', '..'), (279, 'op', '..'), (461, 'za', '..')], [(262, 'rr', '..'), (496, 'vh', '..'), (59, 'wq', '..'), (445, 'bg', '..'), (105, 'tt', '..'), (467, 'ys', '..'), (53, 'uj', '..'), (107, 'fn', '..'), (389, 'mp', '..'), (211, 'vt', '..'), (288, 'tr', '..'), (420, 'gf', '..'), (248, 'pi', '..'), (490, 'vl', '..'), (203, 'wo', '..'), (443, 'zj', '..'), (435, 'jq', '..'), (8, 'rt', '..'), (297, 'vm', '..'), (410, 'ch', '..'), (80, 'vb', '..'), (55, 'ih', '..')], [(395, 'kf', '..'), (479, 'rp', '..'), (259, 'tt', '..'), (191, 'zy', '..'), (12, 'fr', '..'), (148, 'ph', '..'), (40, 'er', '..'), (411, 'dh', '..'), (492, 'li', '..'), (131, 'kt', '..'), (460, 'iu', '..'), (9, 'vm', '..'), (25, 'ke', '..'), (168, 'af', '..'), (278, 'dt', '..'), (361, 'px', '..'), (491, 'yj', '..'), (334, 'wx', '..'), (262, 'fo', '..'), (189, 've', '..'), (339, 'ge', '..'), (245, 'sa', '..')], [(148, 'uk', '..'), (489, 'dw', '..'), (375, 'jv', '..'), (237, 'lf', '..'), (19, 'km', '..'), (207, 'iv', '..'), (282, 'oz', '..'), (471, 'fk', '..'), (283, 'jd', '..'), (210, 'rx', '..'), (18, 'tq', '..'), (254, 'bc', '..'), (262, 'yy', '..'), (102, 'gc', '..'), (284, 'qt', '..'), (369, 'll', '..'), (500, 'tw', '..'), (260, 'hh', '..'), (467, 'ef', '..'), (336, 'ha', '..'), (466, 'se', '..'), (412, 'wr', '..')], [(438, 'sz', '..'), (131, 'at', '..'), (469, 'te', '..'), (224, 'nx', '..'), (478, 'uw', '..'), (6, 'mh', '..'), (158, 'ct', '..'), (167, 'hf', '..'), (373, 'hg', '..'), (66, 'ga', '..'), (212, 'kv', '..'), (39, 'wn', '..'), (324, 'br', '..'), (498, 'os', '..'), (115, 'fc', '..'), (112, 'bc', '..'), (375, 'ad', '..'), (89, 'si', '..'), (264, 'ek', '..'), (276, 'yv', '..'), (206, 'up', '..'), (387, 'jx', '..')], [(85, 'yj', '..'), (403, 'ku', '..'), (266, 'zl', '..'), (183, 'oi', '..'), (191, 'rv', '..'), (274, 'hw', '..'), (267, 'zq', '..'), (411, 'rk', '..'), (417, 'yq', '..'), (322, 'hl', '..'), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None)]

In [10]:
```
# Read the DbFile into Songs2 -- same data
Songs2 = Songs.load_from_disk('Songs')
Songs2.print_file()
```

```
DbFile loaded from Songs.db
```

DbFiles	Num Pages	Pages
Songs	34	[(188, 'kg', '..'), (492, 'oa', '..'), (359, 'cl', '..')], [(163, 'kn', '..'), (458, 'dx', '..'), (210, 'wl', '..')], [(123, 'cm', '..'), (494, 'bd', '..'), (486, 'vb', '..')], [(474, 'nt', '..'), (358, 'vu', '..'), (35, 'ez', '..')], [(21, 'lk', '..'), (288, 'nk', '..'), (229, 'sq', '..')], [(450, 'gm', '..'), (102, 'ch', '..'), (214, 'nx', '..')], [(126, 'my', '..'), (203, 'kz', '..'), (169, 'ak', '..')], [(234, 'yc', '..'), (357, 'in', '..'), (396, 'at', '..')], [(167, 'uw', '..'), (232, 'pm', '..'), (230, 'oa', '..')], [(131, 'fz', '..'), (477, 'gi', '..'), (327, 'zs', '..')], [(29, 'uh', '..'), (305, 'hp', '..'), (124, 'xi', '..')], [(168, 'sq', '..'), (489, 'uc', '..'), (279, 'fz', '..')], [(363, 'ma', '..'), (80, 'br', '..'), (318, 'oa', '..')], [(320, 'ut', '..'), (101, 'ya', '..'), (197, 'rm', '..')], [(402, 'mn', '..'), (255, 'uf', '..'), (246, 'ir', '..')], [(14, 'nc', '..'), (484, 'xd', '..'), (301, 'cp', '..')], [(243, 'hx', '..'), (18, 'lx', '..'), (328, 'jf', '..')], [(446, 'ew', '..'), (178, 'qf', '..'), (192, 'pc', '..')], [(432, 'it', '..'), (179, 'dg', '..'), (77, 'fq', '..')], [(41, 'na', '..'), (379, 'oj', '..'), (232, 'pk', '..')], [(194, 'ro', '..'), (110, 'um', '..'), (494, 'jf', '..')], [(375, 'ur', '..'), (177, 'fg', '..'), (453, 'lq', '..')], [(364, 'mq', '..'), (80, 'zy', '..'), (359, 'mj', '..')], [(292, 'xe', '..'), (207, 'hh', '..'), (134, 'mp', '..')], [(440, 'fi', '..'), (419, 'qp', '..'), (428, 've', '..')], [(383, 'rx', '..'), (166, 'aa', '..'), (266, 'bd', '..')], [(242, 'oc', '..'), (137, 'rd', '..'), (25, 'me', '..')], [(194, 'iq', '..'), (66, 'at', '..'), (261, 'dr', '..')], [(310, 'un', '..'), (491, 'uu', '..'), (416, 'nl', '..')], [(299, 'ha', '..'), (353, 'fr', '..'), (85, 'gk', '..')], [(141, 'yb', '..'), (96, 'yo', '..'), (23, 'ba', '..')], [(167, 'zp', '..'), (70, 'vt', '..'), (499, 'dh', '..')], [(63, 'iv', '..'), (429, 'al', '..'), (170, 'hy', '..')], [(17, 'oa', '..'), (None, None), (None, None)]

In [11]:
```
"""
  Algorithms supported in a DB for sorting, hashing, JOINs, etc"""
def genFilePath(str):
  return str.split('/')[-1]
```

```
class Algos:
  def __init__(self, B, verbose=False): # B (RAM buffer) to keep Pages
    self.B = B
    self.verbose = verbose
    return
  ##
  # Functions for Sorting
  def split(self, R):
    return R.split_files(self.B)

  # Sort a given (small) file in RAM
  def sortRAM(self, v_list, R):
    concat = []
    for l in v_list: # only use valid rows (ignore padded 'None' rows)
      l = list(filter(lambda x: R.is_row_valid(x), l))
      concat += l
    s_concat = sorted(concat, key=lambda x:x[0])
    sublists = [s_concat[i:i + R.k] for i in range(0, len(s_concat), R.k)]
    return DbFile(pages=sublists, k=R.k, filepfx=R.filepfx)

  # MergeBWay(Rlist): Merge B partially sorted files, into a bigger sorted file
    # Step 1: Read 1st page of each of the B files. Create a small heap in RAM.
    # Step 2: Repeat below steps until all data is sorted
    # Step 2a:   Select smallest value from the heap, Append to the output file
    # Step 2b:   Read the next value from the file associated with value in 2a
    #            and add it into heap
  def mergeBway(self, Rlist):
    if not Rlist:
      return []
    out = DbFile(values=None, k=Rlist[0].k, filepfx=Rlist[0].filepfx)
    heaps = []
    # Step 1: Read 1st values.
    # Also, keep a heap to track values, and which file/page they are from
    for b in range(len(Rlist)):
      page = Rlist[b].read_page(0)
      # value from (sort key, row, bth file, 0th page, 0th slot)
      heapq.heappush(heaps, (page[0][0], page[0], b, 0))

    while (heaps): # Step 2a
      cur_min, row, fnum, vidx = heapq.heappop(heaps)
      out.append_page(row)
      # Step 2b: Read the next value (after cur_min in same file)
      vidx += 1
      if vidx < Rlist[fnum].n:
        row = Rlist[fnum].read_ith(vidx)
        heapq.heappush(heaps, (row[0], row, fnum, vidx))
    return [out]

  # BigSort(R): Sort a big file (does not fit in RAM)
    # 1. Split big file into many small files of B pages each
    # 2. Sort each small file (in RAM)
    # 3. Merge B sorted files at a time into bigger files. Repeat until done
  def BigSort(self, R):
    # Step 1 and 2
    split_files = self.split(R)
    merged_files = [[self.sortRAM(fsplit.read_all_pages(), R) \
                     for fsplit in split_files]]
    numpass = 0

    # Step 3
    while (len(merged_files[numpass]) > 1):
      mfiles = []
      for start in range(0, len(merged_files[numpass]), self.B):
        end = min(start + self.B, len(merged_files[numpass]))
        mfiles += self.mergeBway(merged_files[numpass][start:end])
      merged_files.append(mfiles)
```

```
    numpass += 1
  self.print_files(merged_files, add_sfx=True,
                   caption=f'Big Sort {R.filepfx}',
                   subcaption=f'[Pass #]')
  return merged_files[numpass]

## Functions for Hashing
#
# HP(R): Hash partition a big file
  # Step 1: Set up B partitions (stored as DbFiles)
  # Step 2: Read each page of R into RAM.
  # Step 3: Hash each value h(v). Append to the h(v)th partition.
def HashP(self, R):
  # internal fn to hash a given integer or string
  import hashlib
  def hash_value(value):
    if isinstance(value, int):
      return hash(value) % self.B
    sha256 = hashlib.sha256()
    sha256.update(value.encode('utf-8'))
    return int(sha256.hexdigest(), 16) % self.B

  ## Step 1
  hash_files = [DbFile(values=[], k=R.k, filepfx="tmp/"
                       + genFilePath(R.filepfx) + f'.{i}') \
                for i in range(self.B)]
  # Steps 2 and 3
  for i in range(R.P):
    page = R.read_page(i) ## Step 2
    for j in range(len(page)): ## process values in RAM
      if R.is_row_valid(page[j]):
        h = hash_value(page[j][0])
        hash_files[h].append_page(page[j]) # Step 3
  self.print_files([hash_files],
                   caption=f'HashPartiton: {R.filepfx}')
  return hash_files

## Functions for JOINs
#

# BNLJ(R, S): Joins R and S using a block-nested loop algo
  # Step 1. Read R's pages into RAM, 'B' pages at a time.
  #    (And cache these 'B' pages in RAM)
  # Step 2. Read S's pages to join with R's cached pages.
def BNLJ(self, R, S, outk):
  out = DbFile(values=[], k=outk,
               filepfx="join/" + genFilePath(R.filepfx) + \
               '-bnlj-' + genFilePath(S.filepfx))
  for i in range(0, R.P, self.B): # Step 1
    pages_R = [R.read_page(j) \
               for j in range(i, min(i + self.B, R.P))] # IO-cost
    for j in range(S.P): # Step 2
      page_S = S.read_page(j) # IO-cost
      for p in pages_R: # Below is in RAM. I.e. IO-cost = 0
        for r in p:
          if not R.is_row_valid(r):
            continue
          for s in page_S:
            if not S.is_row_valid(s) or r[0] != s[0]:
              continue
            out.append_page((r[0],) + r[1:] + s[1:]) # IO-cost to append
  self.print_files([[out]],
                   caption=f'BNLJ: {R.filepfx} JOIN {S.filepfx}')
  return out
```

```
  # HPJ(R, S): Join R and S with Hash Partition Joins
    # Step 1: HP(R), HP(S)
    # Step 2: For each partition, run BNLJ(). Finally append
  def HPJ(self, R, S, outk):
    # Step 1
    rhash = self.HashP(R)
    shash = self.HashP(S)
    hpjout = []
    # Step 2
    for i in range(self.B):
      hpjout.append(self.BNLJ(rhash[i], shash[i], outk))

  # SMJ(R): SortMerge Join R and S
    # Step 1: Sort R and S (stored as DbFiles)
    # Step 2: Iterate thro' each page of R and S, in sequence
  def SMJ(self, inR, inS, outk):
    # Step1
    R = self.BigSort(inR)[0]; S = self.BigSort(inS)[0]

    out_file = DbFile(values=[], k = outk, filepfx="join/"
                      + genFilePath(R.filepfx) +  "-smj-" + genFilePath(S.filepfx))
    ridx = 0; sidx = 0

    # Step 2: Merge 2 sorted files.
    while (ridx < R.n and sidx < S.n):
      r = R.read_ith(ridx); s = S.read_ith(sidx)
      if r[0] < s[0]:
        ridx += 1
      if r[0] > s[0]:
        sidx += 1
      if r[0] == s[0]:
        # Scan through all equal values. Backup when necessary
        # E.g., R=[... 'bb', 'bb', 'cc', ...] and S=[...'bb', 'cc', 'cc', ...]
        # Make sure to match all 'bb' and 'cc's.
        backup_sidx = sidx
        while r[0] == s[0] and sidx < S.n:
          out_file.append_page((r[0],)+r[1:]+s[1:])
          sidx += 1
          s = S.read_ith(sidx)
        ridx += 1
        sidx = backup_sidx

    self.print_files([[out_file]],
                     caption=f'SMJ: {inR.filepfx} JOIN {inS.filepfx}')
    return out_file

  def format_cell(self, series):
    return ', '.join(str(val) for val in series)

  def print_files(self, mfiles, add_sfx=False, caption="", subcaption=""):
    displaySectionCaption(caption)
    for i, mfile in enumerate(mfiles):
      pd.options.display.max_colwidth = 10
      displaySectionCaption(f'====>> {subcaption} {i}')
      for j, dfile in enumerate(mfile):
        filesfx = '-p' + str(i) + "-f" + str(j) if (add_sfx) else ""
        dfile.filepfx += filesfx
        dfile.print_file()
```

Examples for Sorting, Hashing, JOINs

In [12]:

```
import random, heapq, string
```

```
def gen_random_string(string_length):
    letters = string.ascii_lowercase
    return ''.join(random.choice(letters) for i in range(string_length))

# create sample tables with n-rows of [<int, string of 'l' chars>]
def gen_random_rows(l, n):
  return [(random.randint(1, 500), gen_random_string(l), '..') for i in range(n)]
random.seed(3141)

""" Generate a Songs table with 100 tuples, with k=3 rows per page"""
Songs = DbFile(values=gen_random_rows(2, 100), k=3, filepfx='Songs')
Songs.print_file()
```

DbFiles	Num Pages	Pages
Songs	34	[(188, 'kg', '..'), (492, 'oa', '..'), (359, 'cl', '..')], [(163, 'kn', '..'), (458, 'dx', '..'), (210, 'wl', '..')], [(123, 'cm', '..'), (494, 'bd', '..'), (486, 'vb', '..')], [(474, 'nt', '..'), (358, 'vu', '..'), (35, 'ez', '..')], [(21, 'lk', '..'), (288, 'nk', '..'), (229, 'sq', '..')], [(450, 'gm', '..'), (102, 'ch', '..'), (214, 'nx', '..')], [(126, 'my', '..'), (203, 'kz', '..'), (169, 'ak', '..')], [(234, 'yc', '..'), (357, 'in', '..'), (396, 'at', '..')], [(167, 'uw', '..'), (232, 'pm', '..'), (230, 'oa', '..')], [(131, 'fz', '..'), (477, 'gi', '..'), (327, 'zs', '..')], [(29, 'uh', '..'), (305, 'hp', '..'), (124, 'xi', '..')], [(168, 'sq', '..'), (489, 'uc', '..'), (279, 'fz', '..')], [(363, 'ma', '..'), (80, 'br', '..'), (318, 'oa', '..')], [(320, 'ut', '..'), (101, 'ya', '..'), (197, 'rm', '..')], [(402, 'mn', '..'), (255, 'uf', '..'), (246, 'ir', '..')], [(14, 'nc', '..'), (484, 'xd', '..'), (301, 'cp', '..')], [(243, 'hx', '..'), (18, 'lx', '..'), (328, 'jf', '..')], [(446, 'ew', '..'), (178, 'qf', '..'), (192, 'pc', '..')], [(432, 'it', '..'), (179, 'dg', '..'), (77, 'fq', '..')], [(41, 'na', '..'), (379, 'oj', '..'), (232, 'pk', '..')], [(194, 'ro', '..'), (110, 'um', '..'), (494, 'jf', '..')], [(375, 'ur', '..'), (177, 'fg', '..'), (453, 'lq', '..')], [(364, 'mq', '..'), (80, 'zy', '..'), (359, 'mj', '..')], [(292, 'xe', '..'), (207, 'hh', '..'), (134, 'mp', '..')], [(440, 'fi', '..'), (419, 'qp', '..'), (428, 've', '..')], [(383, 'rx', '..'), (166, 'aa', '..'), (266, 'bd', '..')], [(242, 'oc', '..'), (137, 'rd', '..'), (25, 'me', '..')], [(194, 'iq', '..'), (66, 'at', '..'), (261, 'dr', '..')], [(310, 'un', '..'), (491, 'uu', '..'), (416, 'nl', '..')], [(299, 'ha', '..'), (353, 'fr', '..'), (85, 'gk', '..')], [(141, 'yb', '..'), (96, 'yo', '..'), (23, 'ba', '..')], [(167, 'zp', '..'), (70, 'vt', '..'), (499, 'dh', '..')], [(63, 'iv', '..'), (429, 'al', '..'), (170, 'hy', '..')], [(17, 'oa', '..'), (None, None), (None, None)]

In [13]:
```
algoV = Algos(4, verbose=True) # algorithms in verbose mode
algo = Algos(4) # algorithms in non-verbose mode
SongsSorted = algoV.BigSort(Songs) # Sort Songs table
```

Big Sort Songs

====>> [Pass #] 0

DbFiles	Num Pages	Pages
Songs-p0-f0	4	[(35, 'ez', '..'), (123, 'cm', '..'), (163, 'kn', '..')], [(188, 'kg', '..'), (210, 'wl', '..'), (358, 'vu', '..')], [(359, 'cl', '..'), (458, 'dx', '..'), (474, 'nt', '..')], [(486, 'vb', '..'), (492, 'oa', '..'), (494, 'bd', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f1	4	[(21, 'lk', '..'), (102, 'ch', '..'), (126, 'my', '..')], [(169, 'ak', '..'), (203, 'kz', '..'), (214, 'nx', '..')], [(229, 'sq', '..'), (234, 'yc', '..'), (288, 'nk', '..')], [(357, 'in', '..'), (396, 'at', '..'), (450, 'gm', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f2	4	[(29, 'uh', '..'), (124, 'xi', '..'), (131, 'fz', '..')], [(167, 'uw', '..'), (168, 'sq', '..'), (230, 'oa', '..')], [(232, 'pm', '..'), (279, 'fz', '..'), (305, 'hp', '..')], [(327, 'zs', '..'), (477, 'gi', '..'), (489, 'uc', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f3	4	[(14, 'nc', '..'), (80, 'br', '..'), (101, 'ya', '..')], [(197, 'rm', '..'), (246, 'ir', '..'), (255, 'uf', '..')], [(301, 'cp', '..'), (318, 'oa', '..'), (320, 'ut', '..')], [(363, 'ma', '..'), (402, 'mn', '..'), (484, 'xd', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f4	4	[(18, 'lx', '..'), (41, 'na', '..'), (77, 'fq', '..')], [(178, 'qf', '..'), (179, 'dg', '..'), (192, 'pc', '..')], [(232, 'pk', '..'), (243, 'hx', '..'), (328, 'jf', '..')], [(379, 'oj', '..'), (432, 'it', '..'), (446, 'ew', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f5	4	[(80, 'zy', '..'), (110, 'um', '..'), (134, 'mp', '..')], [(177, 'fg', '..'), (194, 'ro', '..'), (207, 'hh', '..')], [(292, 'xe', '..'), (359, 'mj', '..'), (364, 'mq', '..')], [(375, 'ur', '..'), (453, 'lq', '..'), (494, 'jf', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f6	4	[(25, 'me', '..'), (66, 'at', '..'), (137, 'rd', '..')], [(166, 'aa', '..'), (194, 'iq', '..'), (242, 'oc', '..')], [(261, 'dr', '..'), (266, 'bd', '..'), (383, 'rx', '..')], [(419, 'qp', '..'), (428, 've', '..'), (440, 'fi', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f7	4	[(23, 'ba', '..'), (70, 'vt', '..'), (85, 'gk', '..')], [(96, 'yo', '..'), (141, 'yb', '..'), (167, 'zp', '..')], [(299, 'ha', '..'), (310, 'un', '..'), (353, 'fr', '..')], [(416, 'nl', '..'), (491, 'uu', '..'), (499, 'dh', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f8	2	[(17, 'oa', '..'), (63, 'iv', '..'), (170, 'hy', '..')], [(429, 'al', '..'), (None, None), (None, None)]

====>> [Pass #] 1

DbFiles	Num Pages	Pages
Songs-p1-f0	16	[(14, 'nc', '..'), (21, 'lk', '..'), (29, 'uh', '..')], [(35, 'ez', '..'), (80, 'br', '..'), (101, 'ya', '..')], [(102, 'ch', '..'), (123, 'cm', '..'), (124, 'xi', '..')], [(126, 'my', '..'), (131, 'fz', '..'), (163, 'kn', '..')], [(167, 'uw', '..'), (168, 'sq', '..'), (169, 'ak', '..')], [(188, 'kg', '..'), (197, 'rm', '..'), (203, 'kz', '..')], [(210, 'wl', '..'), (214, 'nx', '..'), (229, 'sq', '..')], [(230, 'oa', '..'), (232, 'pm', '..'), (234, 'yc', '..')], [(246, 'ir', '..'), (255, 'uf', '..'), (279, 'fz', '..')], [(288, 'nk', '..'), (301, 'cp', '..'), (305, 'hp', '..')], [(318, 'oa', '..'), (320, 'ut', '..'), (327, 'zs', '..')], [(357, 'in', '..'), (358, 'vu', '..'), (359, 'cl', '..')], [(363, 'ma', '..'), (396, 'at', '..'), (402, 'mn', '..')], [(450, 'gm', '..'), (458, 'dx', '..'), (474, 'nt', '..')], [(477, 'gi', '..'), (484, 'xd', '..'), (486, 'vb', '..')], [(489, 'uc', '..'), (492, 'oa', '..'), (494, 'bd', '..')]

DbFiles	Num Pages	Pages
Songs-p1-f1	16	[(18, 'lx', '..'), (23, 'ba', '..'), (25, 'me', '..')], [(41, 'na', '..'), (66, 'at', '..'), (70, 'vt', '..')], [(77, 'fq', '..'), (80, 'zy', '..'), (85, 'gk', '..')], [(96, 'yo', '..'), (110, 'um', '..'), (134, 'mp', '..')], [(137, 'rd', '..'), (141, 'yb', '..'), (166, 'aa', '..')], [(167, 'zp', '..'), (177, 'fg', '..'), (178, 'qf', '..')], [(179, 'dg', '..'), (192, 'pc', '..'), (194, 'iq', '..')], [(194, 'ro', '..'), (207, 'hh', '..'), (232, 'pk', '..')], [(242, 'oc', '..'), (243, 'hx', '..'), (261, 'dr', '..')], [(266, 'bd', '..'), (292, 'xe', '..'), (299, 'ha', '..')], [(310, 'un', '..'), (328, 'jf', '..'), (353, 'fr', '..')], [(359, 'mj', '..'), (364, 'mq', '..'), (375, 'ur', '..')], [(379, 'oj', '..'), (383, 'rx', '..'), (416, 'nl', '..')], [(419, 'qp', '..'), (428, 've', '..'), (432, 'it', '..')], [(440, 'fi', '..'), (446, 'ew', '..'), (453, 'lq', '..')], [(491, 'uu', '..'), (494, 'jf', '..'), (499, 'dh', '..')]

DbFiles	Num Pages	Pages
Songs-p1-f2	2	[(17, 'oa', '..'), (63, 'iv', '..'), (170, 'hy', '..')], [(429, 'al', '..'), (None, None), (None, None)]

====>> [Pass #] 2

DbFiles	Num Pages	Pages
Songs-p2-f0	34	[(14, 'nc', '..'), (17, 'oa', '..'), (18, 'lx', '..')], [(21, 'lk', '..'), (23, 'ba', '..'), (25, 'me', '..')], [(29, 'uh', '..'), (35, 'ez', '..'), (41, 'na', '..')], [(63, 'iv', '..'), (66, 'at', '..'), (70, 'vt', '..')], [(77, 'fq', '..'), (80, 'br', '..'), (80, 'zy', '..')], [(85, 'gk', '..'), (96, 'yo', '..'), (101, 'ya', '..')], [(102, 'ch', '..'), (110, 'um', '..'), (123, 'cm', '..')], [(124, 'xi', '..'), (126, 'my', '..'), (131, 'fz', '..')], [(134, 'mp', '..'), (137, 'rd', '..'), (141, 'yb', '..')], [(163, 'kn', '..'), (166, 'aa', '..'), (167, 'uw', '..')], [(167, 'zp', '..'), (168, 'sq', '..'), (169, 'ak', '..')], [(170, 'hy', '..'), (177, 'fg', '..'), (178, 'qf', '..')], [(179, 'dg', '..'), (188, 'kg', '..'), (192, 'pc', '..')], [(194, 'iq', '..'), (194, 'ro', '..'), (197, 'rm', '..')], [(203, 'kz', '..'), (207, 'hh', '..'), (210, 'wl', '..')], [(214, 'nx', '..'), (229, 'sq', '..'), (230, 'oa', '..')], [(232, 'pk', '..'), (232, 'pm', '..'), (234, 'yc', '..')], [(242, 'oc', '..'), (243, 'hx', '..'), (246, 'ir', '..')], [(255, 'uf', '..'), (261, 'dr', '..'), (266, 'bd', '..')], [(279, 'fz', '..'), (288, 'nk', '..'), (292, 'xe', '..')], [(299, 'ha', '..'), (301, 'cp', '..'), (305, 'hp', '..')], [(310, 'un', '..'), (318, 'oa', '..'), (320, 'ut', '..')], [(327, 'zs', '..'), (328, 'jf', '..'), (353, 'fr', '..')], [(357, 'in', '..'), (358, 'vu', '..'), (359, 'cl', '..')], [(359, 'mj', '..'), (363, 'ma', '..'), (364, 'mq', '..')], [(375, 'ur', '..'), (379, 'oj', '..'), (383, 'rx', '..')], [(396, 'at', '..'), (402, 'mn', '..'), (416, 'nl', '..')], [(419, 'qp', '..'), (428, 've', '..'), (429, 'al', '..')], [(432, 'it', '..'), (440, 'fi', '..'), (446, 'ew', '..')], [(450, 'gm', '..'), (453, 'lq', '..'), (458, 'dx', '..')], [(474, 'nt', '..'), (477, 'gi', '..'), (484, 'xd', '..')], [(486, 'vb', '..'), (489, 'uc', '..'), (491, 'uu', '..')], [(492, 'oa', '..'), (494, 'bd', '..'), (494, 'jf', '..')], [(499, 'dh', '..'), (None, None), (None, None)]

In [14]:
```
# Generate n rows with integer keys in descending order and random strings
def gen_reverse_rows(l, n):
  return [(100 - i, gen_random_string(l), '..') for i in range(n)]
""" Generate a RevRecos table with 100 tuples, with k=3 rows per page"""
RevRecos = DbFile(values=gen_reverse_rows(2, 100), k=3, filepfx='RevRecos')
RevRecos.print_file()
RevRecosSorted = algoV.BigSort(RevRecos)
```

DbFiles	Num Pages	Pages
RevRecos	34	[(100, 'lm', '..'), (99, 'jv', '..'), (98, 'vc', '..')], [(97, 'ko', '..'), (96, 'xh', '..'), (95, 'di', '..')], [(94, 'wn', '..'), (93, 'qx', '..'), (92, 'uk', '..')], [(91, 'ar', '..'), (90, 'tu', '..'), (89, 'zl', '..')], [(88, 'gy', '..'), (87, 'hr', '..'), (86, 'wm', '..')], [(85, 'uz', '..'), (84, 'pz', '..'), (83, 'hg', '..')], [(82, 'th', '..'), (81, 'lv', '..'), (80, 'nq', '..')], [(79, 'in', '..'), (78, 'ew', '..'), (77, 'vs', '..')], [(76, 'fw', '..'), (75, 'fx', '..'), (74, 'mq', '..')], [(73, 'fl', '..'), (72, 'cp', '..'), (71, 'kx', '..')], [(70, 'sp', '..'), (69, 'cj', '..'), (68, 'hb', '..')], [(67, 'rr', '..'), (66, 'kl', '..'), (65, 'au', '..')], [(64, 'ft', '..'), (63, 'zm', '..'), (62, 'mf', '..')], [(61, 'zu', '..'), (60, 'gz', '..'), (59, 'yc', '..')], [(58, 'zz', '..'), (57, 'ss', '..'), (56, 'ei', '..')], [(55, 'fq', '..'), (54, 'zu', '..'), (53, 'gi', '..')], [(52, 'qx', '..'), (51, 'pr', '..'), (50, 'ak', '..')], [(49, 'dv', '..'), (48, 'ln', '..'), (47, 'wy', '..')], [(46, 'mp', '..'), (45, 'zb', '..'), (44, 'wk', '..')], [(43, 'fy', '..'), (42, 'rt', '..'), (41, 'uo', '..')], [(40, 'aj', '..'), (39, 'wg', '..'), (38, 'rj', '..')], [(37, 'kr', '..'), (36, 'ck', '..'), (35, 'ev', '..')], [(34, 'va', '..'), (33, 'ru', '..'), (32, 'jb', '..')], [(31, 'yn', '..'), (30, 'ik', '..'), (29, 'yg', '..')], [(28, 'ba', '..'), (27, 'ct', '..'), (26, 'ny', '..')], [(25, 'lv', '..'), (24, 'pd', '..'), (23, 'sx', '..')], [(22, 'vp', '..'), (21, 'bh', '..'), (20, 'km', '..')], [(19, 'te', '..'), (18, 'in', '..'), (17, 'io', '..')], [(16, 'sw', '..'), (15, 'kk', '..'), (14, 'bw', '..')], [(13, 'ht', '..'), (12, 'va', '..'), (11, 'bp', '..')], [(10, 'xy', '..'), (9, 'lt', '..'), (8, 'tt', '..')], [(7, 'ov', '..'), (6, 'ib', '..'), (5, 'ep', '..')], [(4, 'il', '..'), (3, 'bi', '..'), (2, 'rq', '..')], [(1, 'ox', '..'), (None, None), (None, None)]

Big Sort RevRecos

====>> [Pass #] 0

DbFiles	Num Pages	Pages
RevRecos-p0-f0	4	[(89, 'zl', '..'), (90, 'tu', '..'), (91, 'ar', '..')], [(92, 'uk', '..'), (93, 'qx', '..'), (94, 'wn', '..')], [(95, 'di', '..'), (96, 'xh', '..'), (97, 'ko', '..')], [(98, 'vc', '..'), (99, 'jv', '..'), (100, 'lm', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f1	4	[(77, 'vs', '..'), (78, 'ew', '..'), (79, 'in', '..')], [(80, 'nq', '..'), (81, 'lv', '..'), (82, 'th', '..')], [(83, 'hg', '..'), (84, 'pz', '..'), (85, 'uz', '..')], [(86, 'wm', '..'), (87, 'hr', '..'), (88, 'gy', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f2	4	[(65, 'au', '..'), (66, 'kl', '..'), (67, 'rr', '..')], [(68, 'hb', '..'), (69, 'cj', '..'), (70, 'sp', '..')], [(71, 'kx', '..'), (72, 'cp', '..'), (73, 'fl', '..')], [(74, 'mq', '..'), (75, 'fx', '..'), (76, 'fw', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f3	4	[(53, 'gi', '..'), (54, 'zu', '..'), (55, 'fq', '..')], [(56, 'ei', '..'), (57, 'ss', '..'), (58, 'zz', '..')], [(59, 'yc', '..'), (60, 'gz', '..'), (61, 'zu', '..')], [(62, 'mf', '..'), (63, 'zm', '..'), (64, 'ft', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f4	4	[(41, 'uo', '..'), (42, 'rt', '..'), (43, 'fy', '..')], [(44, 'wk', '..'), (45, 'zb', '..'), (46, 'mp', '..')], [(47, 'wy', '..'), (48, 'ln', '..'), (49, 'dv', '..')], [(50, 'ak', '..'), (51, 'pr', '..'), (52, 'qx', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f5	4	[(29, 'yg', '..'), (30, 'ik', '..'), (31, 'yn', '..')], [(32, 'jb', '..'), (33, 'ru', '..'), (34, 'va', '..')], [(35, 'ev', '..'), (36, 'ck', '..'), (37, 'kr', '..')], [(38, 'rj', '..'), (39, 'wg', '..'), (40, 'aj', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f6	4	[(17, 'io', '..'), (18, 'in', '..'), (19, 'te', '..')], [(20, 'km', '..'), (21, 'bh', '..'), (22, 'vp', '..')], [(23, 'sx', '..'), (24, 'pd', '..'), (25, 'lv', '..')], [(26, 'ny', '..'), (27, 'ct', '..'), (28, 'ba', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f7	4	[(5, 'ep', '..'), (6, 'ib', '..'), (7, 'ov', '..')], [(8, 'tt', '..'), (9, 'lt', '..'), (10, 'xy', '..')], [(11, 'bp', '..'), (12, 'va', '..'), (13, 'ht', '..')], [(14, 'bw', '..'), (15, 'kk', '..'), (16, 'sw', '..')]

DbFiles	Num Pages	Pages
RevRecos-p0-f8	2	[(1, 'ox', '..'), (2, 'rq', '..'), (3, 'bi', '..')], [(4, 'il', '..'), (None, None), (None, None)]

====>> [Pass #] 1

DbFiles	Num Pages	Pages
RevRecos-p1-f0	16	[(53, 'gi', '..'), (54, 'zu', '..'), (55, 'fq', '..')], [(56, 'ei', '..'), (57, 'ss', '..'), (58, 'zz', '..')], [(59, 'yc', '..'), (60, 'gz', '..'), (61, 'zu', '..')], [(62, 'mf', '..'), (63, 'zm', '..'), (64, 'ft', '..')], [(65, 'au', '..'), (66, 'kl', '..'), (67, 'rr', '..')], [(68, 'hb', '..'), (69, 'cj', '..'), (70, 'sp', '..')], [(71, 'kx', '..'), (72, 'cp', '..'), (73, 'fl', '..')], [(74, 'mq', '..'), (75, 'fx', '..'), (76, 'fw', '..')], [(77, 'vs', '..'), (78, 'ew', '..'), (79, 'in', '..')], [(80, 'nq', '..'), (81, 'lv', '..'), (82, 'th', '..')], [(83, 'hg', '..'), (84, 'pz', '..'), (85, 'uz', '..')], [(86, 'wm', '..'), (87, 'hr', '..'), (88, 'gy', '..')], [(89, 'zl', '..'), (90, 'tu', '..'), (91, 'ar', '..')], [(92, 'uk', '..'), (93, 'qx', '..'), (94, 'wn', '..')], [(95, 'di', '..'), (96, 'xh', '..'), (97, 'ko', '..')], [(98, 'vc', '..'), (99, 'jv', '..'), (100, 'lm', '..')]

DbFiles	Num Pages	Pages
RevRecos-p1-f1	16	[(5, 'ep', '..'), (6, 'ib', '..'), (7, 'ov', '..')], [(8, 'tt', '..'), (9, 'lt', '..'), (10, 'xy', '..')], [(11, 'bp', '..'), (12, 'va', '..'), (13, 'ht', '..')], [(14, 'bw', '..'), (15, 'kk', '..'), (16, 'sw', '..')], [(17, 'io', '..'), (18, 'in', '..'), (19, 'te', '..')], [(20, 'km', '..'), (21, 'bh', '..'), (22, 'vp', '..')], [(23, 'sx', '..'), (24, 'pd', '..'), (25, 'lv', '..')], [(26, 'ny', '..'), (27, 'ct', '..'), (28, 'ba', '..')], [(29, 'yg', '..'), (30, 'ik', '..'), (31, 'yn', '..')], [(32, 'jb', '..'), (33, 'ru', '..'), (34, 'va', '..')], [(35, 'ev', '..'), (36, 'ck', '..'), (37, 'kr', '..')], [(38, 'rj', '..'), (39, 'wg', '..'), (40, 'aj', '..')], [(41, 'uo', '..'), (42, 'rt', '..'), (43, 'fy', '..')], [(44, 'wk', '..'), (45, 'zb', '..'), (46, 'mp', '..')], [(47, 'wy', '..'), (48, 'ln', '..'), (49, 'dv', '..')], [(50, 'ak', '..'), (51, 'pr', '..'), (52, 'qx', '..')]

DbFiles	Num Pages	Pages
RevRecos-p1-f2	2	[(1, 'ox', '..'), (2, 'rq', '..'), (3, 'bi', '..')], [(4, 'il', '..'), (None, None), (None, None)]

====>> [Pass #] 2

DbFiles	Num Pages	Pages
RevRecos-p2-f0	34	[(1, 'ox', '..'), (2, 'rq', '..'), (3, 'bi', '..')], [(4, 'il', '..'), (5, 'ep', '..'), (6, 'ib', '..')], [(7, 'ov', '..'), (8, 'tt', '..'), (9, 'lt', '..')], [(10, 'xy', '..'), (11, 'bp', '..'), (12, 'va', '..')], [(13, 'ht', '..'), (14, 'bw', '..'), (15, 'kk', '..')], [(16, 'sw', '..'), (17, 'io', '..'), (18, 'in', '..')], [(19, 'te', '..'), (20, 'km', '..'), (21, 'bh', '..')], [(22, 'vp', '..'), (23, 'sx', '..'), (24, 'pd', '..')], [(25, 'lv', '..'), (26, 'ny', '..'), (27, 'ct', '..')], [(28, 'ba', '..'), (29, 'yg', '..'), (30, 'ik', '..')], [(31, 'yn', '..'), (32, 'jb', '..'), (33, 'ru', '..')], [(34, 'va', '..'), (35, 'ev', '..'), (36, 'ck', '..')], [(37, 'kr', '..'), (38, 'rj', '..'), (39, 'wg', '..')], [(40, 'aj', '..'), (41, 'uo', '..'), (42, 'rt', '..')], [(43, 'fy', '..'), (44, 'wk', '..'), (45, 'zb', '..')], [(46, 'mp', '..'), (47, 'wy', '..'), (48, 'ln', '..')], [(49, 'dv', '..'), (50, 'ak', '..'), (51, 'pr', '..')], [(52, 'qx', '..'), (53, 'gi', '..'), (54, 'zu', '..')], [(55, 'fq', '..'), (56, 'ei', '..'), (57, 'ss', '..')], [(58, 'zz', '..'), (59, 'yc', '..'), (60, 'gz', '..')], [(61, 'zu', '..'), (62, 'mf', '..'), (63, 'zm', '..')], [(64, 'ft', '..'), (65, 'au', '..'), (66, 'kl', '..')], [(67, 'rr', '..'), (68, 'hb', '..'), (69, 'cj', '..')], [(70, 'sp', '..'), (71, 'kx', '..'), (72, 'cp', '..')], [(73, 'fl', '..'), (74, 'mq', '..'), (75, 'fx', '..')], [(76, 'fw', '..'), (77, 'vs', '..'), (78, 'ew', '..')], [(79, 'in', '..'), (80, 'nq', '..'), (81, 'lv', '..')], [(82, 'th', '..'), (83, 'hg', '..'), (84, 'pz', '..')], [(85, 'uz', '..'), (86, 'wm', '..'), (87, 'hr', '..')], [(88, 'gy', '..'), (89, 'zl', '..'), (90, 'tu', '..')], [(91, 'ar', '..'), (92, 'uk', '..'), (93, 'qx', '..')], [(94, 'wn', '..'), (95, 'di', '..'), (96, 'xh', '..')], [(97, 'ko', '..'), (98, 'vc', '..'), (99, 'jv', '..')], [(100, 'lm', '..'), (None, None), (None, None)]

In [15]:
```
""" Generate a Listens table with 100 tuples, with k=7 rows per page"""

random.seed(4242)
Listens = DbFile(values=gen_random_rows(2, 100), k=7, filepfx='Listens')
Listens.print_file()

ListensSorted = algoV.BigSort(Listens) # Sort Listens table
```

DbFiles	Num Pages	Pages
Listens	15	[(442, 'ne', '..'), (15, 'tm', '..'), (195, 'ki', '..'), (95, 'qp', '..'), (13, 'pf', '..'), (80, 'sy', '..'), (345, 'pd', '..')], [(14, 'eq', '..'), (73, 'fk', '..'), (63, 'fw', '..'), (122, 'ws', '..'), (343, 'uh', '..'), (466, 'ob', '..'), (468, 'ik', '..')], [(3, 'kf', '..'), (265, 'kv', '..'), (183, 'tm', '..'), (207, 'kx', '..'), (158, 'zf', '..'), (177, 'tr', '..'), (249, 'io', '..')], [(41, 'zx', '..'), (433, 'pi', '..'), (86, 'tf', '..'), (467, 'jp', '..'), (143, 'cd', '..'), (373, 'mn', '..'), (341, 'op', '..')], [(118, 'ph', '..'), (54, 'to', '..'), (213, 'rn', '..'), (114, 'hl', '..'), (366, 'gq', '..'), (196, 'cj', '..'), (457, 'zb', '..')], [(274, 'gd', '..'), (217, 'ax', '..'), (13, 'xr', '..'), (220, 'tv', '..'), (296, 'gz', '..'), (486, 'iw', '..'), (419, 'np', '..')], [(497, 'gn', '..'), (181, 'hf', '..'), (411, 'nz', '..'), (257, 'aq', '..'), (479, 'mu', '..'), (58, 'pz', '..'), (490, 'th', '..')], [(239, 'lv', '..'), (152, 'dr', '..'), (280, 'we', '..'), (442, 'aa', '..'), (463, 'fb', '..'), (264, 'oe', '..'), (107, 'hk', '..')], [(190, 'ze', '..'), (162, 'dg', '..'), (236, 'ot', '..'), (9, 'ji', '..'), (127, 'wl', '..'), (50, 'vf', '..'), (346, 'vs', '..')], [(69, 'ev', '..'), (37, 'ec', '..'), (294, 'ld', '..'), (14, 'xo', '..'), (442, 'fi', '..'), (210, 'gw', '..'), (285, 'fg', '..')], [(427, 'rs', '..'), (227, 'me', '..'), (438, 'pd', '..'), (448, 'dh', '..'), (463, 'iq', '..'), (205, 'wz', '..'), (31, 'yj', '..')], [(460, 'mt', '..'), (483, 'nf', '..'), (443, 'yy', '..'), (392, 'iy', '..'), (67, 'az', '..'), (468, 'up', '..'), (22, 'ko', '..')], [(286, 'mf', '..'), (448, 'ee', '..'), (166, 'nv', '..'), (113, 'vh', '..'), (183, 'pd', '..'), (148, 'qv', '..'), (429, 'zh', '..')], [(396, 'gx', '..'), (142, 'vj', '..'), (29, 'xf', '..'), (229, 'co', '..'), (219, 'xe', '..'), (367, 'kc', '..'), (478, 'ce', '..')], [(335, 'dh', '..'), (116, 'et', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

Big Sort Listens

====>> [Pass #] 0

DbFiles	Num Pages	Pages
Listens-p0-f0	4	[(3, 'kf', '..'), (13, 'pf', '..'), (14, 'eq', '..'), (15, 'tm', '..'), (41, 'zx', '..'), (63, 'fw', '..'), (73, 'fk', '..')], [(80, 'sy', '..'), (86, 'tf', '..'), (95, 'qp', '..'), (122, 'ws', '..'), (143, 'cd', '..'), (158, 'zf', '..'), (177, 'tr', '..')], [(183, 'tm', '..'), (195, 'ki', '..'), (207, 'kx', '..'), (249, 'io', '..'), (265, 'kv', '..'), (341, 'op', '..'), (343, 'uh', '..')], [(345, 'pd', '..'), (373, 'mn', '..'), (433, 'pi', '..'), (442, 'ne', '..'), (466, 'ob', '..'), (467, 'jp', '..'), (468, 'ik', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f1	4	[(13, 'xr', '..'), (54, 'to', '..'), (58, 'pz', '..'), (107, 'hk', '..'), (114, 'hl', '..'), (118, 'ph', '..'), (152, 'dr', '..')], [(181, 'hf', '..'), (196, 'cj', '..'), (213, 'rn', '..'), (217, 'ax', '..'), (220, 'tv', '..'), (239, 'lv', '..'), (257, 'aq', '..')], [(264, 'oe', '..'), (274, 'gd', '..'), (280, 'we', '..'), (296, 'gz', '..'), (366, 'gq', '..'), (411, 'nz', '..'), (419, 'np', '..')], [(442, 'aa', '..'), (457, 'zb', '..'), (463, 'fb', '..'), (479, 'mu', '..'), (486, 'iw', '..'), (490, 'th', '..'), (497, 'gn', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f2	4	[(9, 'ji', '..'), (14, 'xo', '..'), (22, 'ko', '..'), (31, 'yj', '..'), (37, 'ec', '..'), (50, 'vf', '..'), (67, 'az', '..')], [(69, 'ev', '..'), (127, 'wl', '..'), (162, 'dg', '..'), (190, 'ze', '..'), (205, 'wz', '..'), (210, 'gw', '..'), (227, 'me', '..')], [(236, 'ot', '..'), (285, 'fg', '..'), (294, 'ld', '..'), (346, 'vs', '..'), (392, 'iy', '..'), (427, 'rs', '..'), (438, 'pd', '..')], [(442, 'fi', '..'), (443, 'yy', '..'), (448, 'dh', '..'), (460, 'mt', '..'), (463, 'iq', '..'), (468, 'up', '..'), (483, 'nf', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f3	3	[(29, 'xf', '..'), (113, 'vh', '..'), (116, 'et', '..'), (142, 'vj', '..'), (148, 'qv', '..'), (166, 'nv', '..'), (183, 'pd', '..')], [(219, 'xe', '..'), (229, 'co', '..'), (286, 'mf', '..'), (335, 'dh', '..'), (367, 'kc', '..'), (396, 'gx', '..'), (429, 'zh', '..')], [(448, 'ee', '..'), (478, 'ce', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

====>> [Pass #] 1

DbFiles	Num Pages	Pages
Listens-p1-f0	15	[(3, 'kf', '..'), (9, 'ji', '..'), (13, 'pf', '..'), (13, 'xr', '..'), (14, 'eq', '..'), (14, 'xo', '..'), (15, 'tm', '..')], [(22, 'ko', '..'), (29, 'xf', '..'), (31, 'yj', '..'), (37, 'ec', '..'), (41, 'zx', '..'), (50, 'vf', '..'), (54, 'to', '..')], [(58, 'pz', '..'), (63, 'fw', '..'), (67, 'az', '..'), (69, 'ev', '..'), (73, 'fk', '..'), (80, 'sy', '..'), (86, 'tf', '..')], [(95, 'qp', '..'), (107, 'hk', '..'), (113, 'vh', '..'), (114, 'hl', '..'), (116, 'et', '..'), (118, 'ph', '..'), (122, 'ws', '..')], [(127, 'wl', '..'), (142, 'vj', '..'), (143, 'cd', '..'), (148, 'qv', '..'), (152, 'dr', '..'), (158, 'zf', '..'), (162, 'dg', '..')], [(166, 'nv', '..'), (177, 'tr', '..'), (181, 'hf', '..'), (183, 'pd', '..'), (183, 'tm', '..'), (190, 'ze', '..'), (195, 'ki', '..')], [(196, 'cj', '..'), (205, 'wz', '..'), (207, 'kx', '..'), (210, 'gw', '..'), (213, 'rn', '..'), (217, 'ax', '..'), (219, 'xe', '..')], [(220, 'tv', '..'), (227, 'me', '..'), (229, 'co', '..'), (236, 'ot', '..'), (239, 'lv', '..'), (249, 'io', '..'), (257, 'aq', '..')], [(264, 'oe', '..'), (265, 'kv', '..'), (274, 'gd', '..'), (280, 'we', '..'), (285, 'fg', '..'), (286, 'mf', '..'), (294, 'ld', '..')], [(296, 'gz', '..'), (335, 'dh', '..'), (341, 'op', '..'), (343, 'uh', '..'), (345, 'pd', '..'), (346, 'vs', '..'), (366, 'gq', '..')], [(367, 'kc', '..'), (373, 'mn', '..'), (392, 'iy', '..'), (396, 'gx', '..'), (411, 'nz', '..'), (419, 'np', '..'), (427, 'rs', '..')], [(429, 'zh', '..'), (433, 'pi', '..'), (438, 'pd', '..'), (442, 'aa', '..'), (442, 'fi', '..'), (442, 'ne', '..'), (443, 'yy', '..')], [(448, 'dh', '..'), (448, 'ee', '..'), (457, 'zb', '..'), (460, 'mt', '..'), (463, 'fb', '..'), (463, 'iq', '..'), (466, 'ob', '..')], [(467, 'jp', '..'), (468, 'ik', '..'), (468, 'up', '..'), (478, 'ce', '..'), (479, 'mu', '..'), (483, 'nf', '..'), (486, 'iw', '..')], [(490, 'th', '..'), (497, 'gn', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

In [16]:
```
algoV.BNLJ(Songs, Listens, 4)
```

BNLJ: Songs JOIN Listens

====>> 0

DbFiles	Num Pages	Pages
join/Songs-bnlj-Listens	4	[(486, 'vb', '..', 'iw', '..'), (210, 'wl', '..', 'gw', '..'), (229, 'sq', '..', 'co', '..'), (396, 'at', '..', 'gx', '..')], [(29, 'uh', '..', 'xf', '..'), (80, 'br', '..', 'sy', '..'), (14, 'nc', '..', 'eq', '..'), (14, 'nc', '..', 'xo', '..')], [(41, 'na', '..', 'zx', '..'), (80, 'zy', '..', 'sy', '..'), (177, 'fg', '..', 'tr', '..'), (207, 'hh', '..', 'kx', '..')], [(419, 'qp', '..', 'np', '..'), (166, 'aa', '..', 'nv', '..'), (63, 'iv', '..', 'fw', '..'), (429, 'al', '..', 'zh', '..')]

Out[16]: <__main__.DbFile at 0x1100ab450>

In [17]:
```
algoV.HPJ(Songs, Listens, 4)
```

HashPartiton: Songs

====>> 0

DbFiles	Num Pages	Pages
tmp/Songs.0	7	[(188, 'kg', '..'), (492, 'oa', '..'), (288, 'nk', '..')], [(396, 'at', '..'), (232, 'pm', '..'), (124, 'xi', '..')], [(168, 'sq', '..'), (80, 'br', '..'), (320, 'ut', '..')], [(484, 'xd', '..'), (328, 'jf', '..'), (192, 'pc', '..')], [(432, 'it', '..'), (232, 'pk', '..'), (364, 'mq', '..')], [(80, 'zy', '..'), (292, 'xe', '..'), (440, 'fi', '..')], [(428, 've', '..'), (416, 'nl', '..'), (96, 'yo', '..')]

DbFiles	Num Pages	Pages
tmp/Songs.1	8	[(21, 'lk', '..'), (229, 'sq', '..'), (169, 'ak', '..')], [(357, 'in', '..'), (477, 'gi', '..'), (29, 'uh', '..')], [(305, 'hp', '..'), (489, 'uc', '..'), (101, 'ya', '..')], [(197, 'rm', '..'), (301, 'cp', '..'), (77, 'fq', '..')], [(41, 'na', '..'), (177, 'fg', '..'), (453, 'lq', '..')], [(137, 'rd', '..'), (25, 'me', '..'), (261, 'dr', '..')], [(353, 'fr', '..'), (85, 'gk', '..'), (141, 'yb', '..')], [(429, 'al', '..'), (17, 'oa', '..'), (None, None)]

DbFiles	Num Pages	Pages
tmp/Songs.2	11	[(458, 'dx', '..'), (210, 'wl', '..'), (494, 'bd', '..')], [(486, 'vb', '..'), (474, 'nt', '..'), (358, 'vu', '..')], [(450, 'gm', '..'), (102, 'ch', '..'), (214, 'nx', '..')], [(126, 'my', '..'), (234, 'yc', '..'), (230, 'oa', '..')], [(318, 'oa', '..'), (402, 'mn', '..'), (246, 'ir', '..')], [(14, 'nc', '..'), (18, 'lx', '..'), (446, 'ew', '..')], [(178, 'qf', '..'), (194, 'ro', '..'), (110, 'um', '..')], [(494, 'jf', '..'), (134, 'mp', '..'), (166, 'aa', '..')], [(266, 'bd', '..'), (242, 'oc', '..'), (194, 'iq', '..')], [(66, 'at', '..'), (310, 'un', '..'), (70, 'vt', '..')], [(170, 'hy', '..'), (None, None), (None, None)]

DbFiles	Num Pages	Pages
tmp/Songs.3	9	[(359, 'cl', '..'), (163, 'kn', '..'), (123, 'cm', '..')], [(35, 'ez', '..'), (203, 'kz', '..'), (167, 'uw', '..')], [(131, 'fz', '..'), (327, 'zs', '..'), (279, 'fz', '..')], [(363, 'ma', '..'), (255, 'uf', '..'), (243, 'hx', '..')], [(179, 'dg', '..'), (379, 'oj', '..'), (375, 'ur', '..')], [(359, 'mj', '..'), (207, 'hh', '..'), (419, 'qp', '..')], [(383, 'rx', '..'), (491, 'uu', '..'), (299, 'ha', '..')], [(23, 'ba', '..'), (167, 'zp', '..'), (499, 'dh', '..')], [(63, 'iv', '..'), (None, None), (None, None)]

HashPartiton: Listens

====>> 0

DbFiles	Num Pages	Pages
tmp/Listens.0	3	[(80, 'sy', '..'), (468, 'ik', '..'), (196, 'cj', '..'), (220, 'tv', '..'), (296, 'gz', '..'), (152, 'dr', '..'), (280, 'we', '..')], [(264, 'oe', '..'), (236, 'ot', '..'), (448, 'dh', '..'), (460, 'mt', '..'), (392, 'iy', '..'), (468, 'up', '..'), (448, 'ee', '..')], [(148, 'qv', '..'), (396, 'gx', '..'), (116, 'et', '..'), (None, None), (None, None), (None, None), (None, None)]

DbFiles	Num Pages	Pages
tmp/Listens.1	4	[(13, 'pf', '..'), (345, 'pd', '..'), (73, 'fk', '..'), (265, 'kv', '..'), (177, 'tr', '..'), (249, 'io', '..'), (41, 'zx', '..')], [(433, 'pi', '..'), (373, 'mn', '..'), (341, 'op', '..'), (213, 'rn', '..'), (457, 'zb', '..'), (217, 'ax', '..'), (13, 'xr', '..')], [(497, 'gn', '..'), (181, 'hf', '..'), (257, 'aq', '..'), (9, 'ji', '..'), (69, 'ev', '..'), (37, 'ec', '..'), (285, 'fg', '..')], [(205, 'wz', '..'), (113, 'vh', '..'), (429, 'zh', '..'), (29, 'xf', '..'), (229, 'co', '..'), (None, None), (None, None)]

DbFiles	Num Pages	Pages
tmp/Listens.2	5	[(442, 'ne', '..'), (14, 'eq', '..'), (122, 'ws', '..'), (466, 'ob', '..'), (158, 'zf', '..'), (86, 'tf', '..'), (118, 'ph', '..')], [(54, 'to', '..'), (114, 'hl', '..'), (366, 'gq', '..'), (274, 'gd', '..'), (486, 'iw', '..'), (58, 'pz', '..'), (490, 'th', '..')], [(442, 'aa', '..'), (190, 'ze', '..'), (162, 'dg', '..'), (50, 'vf', '..'), (346, 'vs', '..'), (294, 'ld', '..'), (14, 'xo', '..')], [(442, 'fi', '..'), (210, 'gw', '..'), (438, 'pd', '..'), (22, 'ko', '..'), (286, 'mf', '..'), (166, 'nv', '..'), (142, 'vj', '..')], [(478, 'ce', '..'), (None, None), (None, None), (None, None), (None, None), (None, None), (None, None)]

DbFiles	Num Pages	Pages
tmp/Listens.3	4	[(15, 'tm', '..'), (195, 'ki', '..'), (95, 'qp', '..'), (63, 'fw', '..'), (343, 'uh', '..'), (3, 'kf', '..'), (183, 'tm', '..')], [(207, 'kx', '..'), (467, 'jp', '..'), (143, 'cd', '..'), (419, 'np', '..'), (411, 'nz', '..'), (479, 'mu', '..'), (239, 'lv', '..')], [(463, 'fb', '..'), (107, 'hk', '..'), (127, 'wl', '..'), (427, 'rs', '..'), (227, 'me', '..'), (463, 'iq', '..'), (31, 'yj', '..')], [(483, 'nf', '..'), (443, 'yy', '..'), (67, 'az', '..'), (183, 'pd', '..'), (219, 'xe', '..'), (367, 'kc', '..'), (335, 'dh', '..')]

BNLJ: tmp/Songs.0 JOIN tmp/Listens.0

====>> 0

DbFiles	Num Pages	Pages
join/Songs.0-bnlj-Listens.0	1	[(80, 'br', '..', 'sy', '..'), (396, 'at', '..', 'gx', '..'), (80, 'zy', '..', 'sy', '..'), (None, None)]

BNLJ: tmp/Songs.1 JOIN tmp/Listens.1

====>> 0

DbFiles	Num Pages	Pages
join/Songs.1-bnlj-Listens.1	2	[(229, 'sq', '..', 'co', '..'), (29, 'uh', '..', 'xf', '..'), (41, 'na', '..', 'zx', '..'), (177, 'fg', '..', 'tr', '..')], [(429, 'al', '..', 'zh', '..'), (None, None), (None, None), (None, None)]

BNLJ: tmp/Songs.2 JOIN tmp/Listens.2

====>> 0

DbFiles	Num Pages	Pages
join/Songs.2-bnlj-Listens.2	2	[(486, 'vb', '..', 'iw', '..'), (210, 'wl', '..', 'gw', '..'), (14, 'nc', '..', 'eq', '..'), (14, 'nc', '..', 'xo', '..')], [(166, 'aa', '..', 'nv', '..'), (None, None), (None, None), (None, None)]

BNLJ: tmp/Songs.3 JOIN tmp/Listens.3

====>> 0

DbFiles	Num Pages	Pages
join/Songs.3-bnlj-Listens.3	1	[(207, 'hh', '..', 'kx', '..'), (419, 'qp', '..', 'np', '..'), (63, 'iv', '..', 'fw', '..'), (None, None)]

In [18]:
```
algoV.SMJ(Songs, Listens, 4)
```

Big Sort Songs

====>> [Pass #] 0

DbFiles	Num Pages	Pages
Songs-p0-f0	4	[(35, 'ez', '..'), (123, 'cm', '..'), (163, 'kn', '..')], [(188, 'kg', '..'), (210, 'wl', '..'), (358, 'vu', '..')], [(359, 'cl', '..'), (458, 'dx', '..'), (474, 'nt', '..')], [(486, 'vb', '..'), (492, 'oa', '..'), (494, 'bd', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f1	4	[(21, 'lk', '..'), (102, 'ch', '..'), (126, 'my', '..')], [(169, 'ak', '..'), (203, 'kz', '..'), (214, 'nx', '..')], [(229, 'sq', '..'), (234, 'yc', '..'), (288, 'nk', '..')], [(357, 'in', '..'), (396, 'at', '..'), (450, 'gm', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f2	4	[(29, 'uh', '..'), (124, 'xi', '..'), (131, 'fz', '..')], [(167, 'uw', '..'), (168, 'sq', '..'), (230, 'oa', '..')], [(232, 'pm', '..'), (279, 'fz', '..'), (305, 'hp', '..')], [(327, 'zs', '..'), (477, 'gi', '..'), (489, 'uc', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f3	4	[(14, 'nc', '..'), (80, 'br', '..'), (101, 'ya', '..')], [(197, 'rm', '..'), (246, 'ir', '..'), (255, 'uf', '..')], [(301, 'cp', '..'), (318, 'oa', '..'), (320, 'ut', '..')], [(363, 'ma', '..'), (402, 'mn', '..'), (484, 'xd', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f4	4	[(18, 'lx', '..'), (41, 'na', '..'), (77, 'fq', '..')], [(178, 'qf', '..'), (179, 'dg', '..'), (192, 'pc', '..')], [(232, 'pk', '..'), (243, 'hx', '..'), (328, 'jf', '..')], [(379, 'oj', '..'), (432, 'it', '..'), (446, 'ew', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f5	4	[(80, 'zy', '..'), (110, 'um', '..'), (134, 'mp', '..')], [(177, 'fg', '..'), (194, 'ro', '..'), (207, 'hh', '..')], [(292, 'xe', '..'), (359, 'mj', '..'), (364, 'mq', '..')], [(375, 'ur', '..'), (453, 'lq', '..'), (494, 'jf', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f6	4	[(25, 'me', '..'), (66, 'at', '..'), (137, 'rd', '..')], [(166, 'aa', '..'), (194, 'iq', '..'), (242, 'oc', '..')], [(261, 'dr', '..'), (266, 'bd', '..'), (383, 'rx', '..')], [(419, 'qp', '..'), (428, 've', '..'), (440, 'fi', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f7	4	[(23, 'ba', '..'), (70, 'vt', '..'), (85, 'gk', '..')], [(96, 'yo', '..'), (141, 'yb', '..'), (167, 'zp', '..')], [(299, 'ha', '..'), (310, 'un', '..'), (353, 'fr', '..')], [(416, 'nl', '..'), (491, 'uu', '..'), (499, 'dh', '..')]

DbFiles	Num Pages	Pages
Songs-p0-f8	2	[(17, 'oa', '..'), (63, 'iv', '..'), (170, 'hy', '..')], [(429, 'al', '..'), (None, None), (None, None)]

====>> [Pass #] 1

DbFiles	Num Pages	Pages
Songs-p1-f0	16	[(14, 'nc', '..'), (21, 'lk', '..'), (29, 'uh', '..')], [(35, 'ez', '..'), (80, 'br', '..'), (101, 'ya', '..')], [(102, 'ch', '..'), (123, 'cm', '..'), (124, 'xi', '..')], [(126, 'my', '..'), (131, 'fz', '..'), (163, 'kn', '..')], [(167, 'uw', '..'), (168, 'sq', '..'), (169, 'ak', '..')], [(188, 'kg', '..'), (197, 'rm', '..'), (203, 'kz', '..')], [(210, 'wl', '..'), (214, 'nx', '..'), (229, 'sq', '..')], [(230, 'oa', '..'), (232, 'pm', '..'), (234, 'yc', '..')], [(246, 'ir', '..'), (255, 'uf', '..'), (279, 'fz', '..')], [(288, 'nk', '..'), (301, 'cp', '..'), (305, 'hp', '..')], [(318, 'oa', '..'), (320, 'ut', '..'), (327, 'zs', '..')], [(357, 'in', '..'), (358, 'vu', '..'), (359, 'cl', '..')], [(363, 'ma', '..'), (396, 'at', '..'), (402, 'mn', '..')], [(450, 'gm', '..'), (458, 'dx', '..'), (474, 'nt', '..')], [(477, 'gi', '..'), (484, 'xd', '..'), (486, 'vb', '..')], [(489, 'uc', '..'), (492, 'oa', '..'), (494, 'bd', '..')]

DbFiles	Num Pages	Pages
Songs-p1-f1	16	[(18, 'lx', '..'), (23, 'ba', '..'), (25, 'me', '..')], [(41, 'na', '..'), (66, 'at', '..'), (70, 'vt', '..')], [(77, 'fq', '..'), (80, 'zy', '..'), (85, 'gk', '..')], [(96, 'yo', '..'), (110, 'um', '..'), (134, 'mp', '..')], [(137, 'rd', '..'), (141, 'yb', '..'), (166, 'aa', '..')], [(167, 'zp', '..'), (177, 'fg', '..'), (178, 'qf', '..')], [(179, 'dg', '..'), (192, 'pc', '..'), (194, 'iq', '..')], [(194, 'ro', '..'), (207, 'hh', '..'), (232, 'pk', '..')], [(242, 'oc', '..'), (243, 'hx', '..'), (261, 'dr', '..')], [(266, 'bd', '..'), (292, 'xe', '..'), (299, 'ha', '..')], [(310, 'un', '..'), (328, 'jf', '..'), (353, 'fr', '..')], [(359, 'mj', '..'), (364, 'mq', '..'), (375, 'ur', '..')], [(379, 'oj', '..'), (383, 'rx', '..'), (416, 'nl', '..')], [(419, 'qp', '..'), (428, 've', '..'), (432, 'it', '..')], [(440, 'fi', '..'), (446, 'ew', '..'), (453, 'lq', '..')], [(491, 'uu', '..'), (494, 'jf', '..'), (499, 'dh', '..')]

DbFiles	Num Pages	Pages
Songs-p1-f2	2	[(17, 'oa', '..'), (63, 'iv', '..'), (170, 'hy', '..')], [(429, 'al', '..'), (None, None), (None, None)]

====>> [Pass #] 2

DbFiles	Num Pages	Pages
Songs-p2-f0	34	[(14, 'nc', '..'), (17, 'oa', '..'), (18, 'lx', '..')], [(21, 'lk', '..'), (23, 'ba', '..'), (25, 'me', '..')], [(29, 'uh', '..'), (35, 'ez', '..'), (41, 'na', '..')], [(63, 'iv', '..'), (66, 'at', '..'), (70, 'vt', '..')], [(77, 'fq', '..'), (80, 'br', '..'), (80, 'zy', '..')], [(85, 'gk', '..'), (96, 'yo', '..'), (101, 'ya', '..')], [(102, 'ch', '..'), (110, 'um', '..'), (123, 'cm', '..')], [(124, 'xi', '..'), (126, 'my', '..'), (131, 'fz', '..')], [(134, 'mp', '..'), (137, 'rd', '..'), (141, 'yb', '..')], [(163, 'kn', '..'), (166, 'aa', '..'), (167, 'uw', '..')], [(167, 'zp', '..'), (168, 'sq', '..'), (169, 'ak', '..')], [(170, 'hy', '..'), (177, 'fg', '..'), (178, 'qf', '..')], [(179, 'dg', '..'), (188, 'kg', '..'), (192, 'pc', '..')], [(194, 'iq', '..'), (194, 'ro', '..'), (197, 'rm', '..')], [(203, 'kz', '..'), (207, 'hh', '..'), (210, 'wl', '..')], [(214, 'nx', '..'), (229, 'sq', '..'), (230, 'oa', '..')], [(232, 'pk', '..'), (232, 'pm', '..'), (234, 'yc', '..')], [(242, 'oc', '..'), (243, 'hx', '..'), (246, 'ir', '..')], [(255, 'uf', '..'), (261, 'dr', '..'), (266, 'bd', '..')], [(279, 'fz', '..'), (288, 'nk', '..'), (292, 'xe', '..')], [(299, 'ha', '..'), (301, 'cp', '..'), (305, 'hp', '..')], [(310, 'un', '..'), (318, 'oa', '..'), (320, 'ut', '..')], [(327, 'zs', '..'), (328, 'jf', '..'), (353, 'fr', '..')], [(357, 'in', '..'), (358, 'vu', '..'), (359, 'cl', '..')], [(359, 'mj', '..'), (363, 'ma', '..'), (364, 'mq', '..')], [(375, 'ur', '..'), (379, 'oj', '..'), (383, 'rx', '..')], [(396, 'at', '..'), (402, 'mn', '..'), (416, 'nl', '..')], [(419, 'qp', '..'), (428, 've', '..'), (429, 'al', '..')], [(432, 'it', '..'), (440, 'fi', '..'), (446, 'ew', '..')], [(450, 'gm', '..'), (453, 'lq', '..'), (458, 'dx', '..')], [(474, 'nt', '..'), (477, 'gi', '..'), (484, 'xd', '..')], [(486, 'vb', '..'), (489, 'uc', '..'), (491, 'uu', '..')], [(492, 'oa', '..'), (494, 'bd', '..'), (494, 'jf', '..')], [(499, 'dh', '..'), (None, None), (None, None)]

Big Sort Listens

====>> [Pass #] 0

DbFiles	Num Pages	Pages
Listens-p0-f0	4	[(3, 'kf', '..'), (13, 'pf', '..'), (14, 'eq', '..'), (15, 'tm', '..'), (41, 'zx', '..'), (63, 'fw', '..'), (73, 'fk', '..')], [(80, 'sy', '..'), (86, 'tf', '..'), (95, 'qp', '..'), (122, 'ws', '..'), (143, 'cd', '..'), (158, 'zf', '..'), (177, 'tr', '..')], [(183, 'tm', '..'), (195, 'ki', '..'), (207, 'kx', '..'), (249, 'io', '..'), (265, 'kv', '..'), (341, 'op', '..'), (343, 'uh', '..')], [(345, 'pd', '..'), (373, 'mn', '..'), (433, 'pi', '..'), (442, 'ne', '..'), (466, 'ob', '..'), (467, 'jp', '..'), (468, 'ik', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f1	4	[(13, 'xr', '..'), (54, 'to', '..'), (58, 'pz', '..'), (107, 'hk', '..'), (114, 'hl', '..'), (118, 'ph', '..'), (152, 'dr', '..')], [(181, 'hf', '..'), (196, 'cj', '..'), (213, 'rn', '..'), (217, 'ax', '..'), (220, 'tv', '..'), (239, 'lv', '..'), (257, 'aq', '..')], [(264, 'oe', '..'), (274, 'gd', '..'), (280, 'we', '..'), (296, 'gz', '..'), (366, 'gq', '..'), (411, 'nz', '..'), (419, 'np', '..')], [(442, 'aa', '..'), (457, 'zb', '..'), (463, 'fb', '..'), (479, 'mu', '..'), (486, 'iw', '..'), (490, 'th', '..'), (497, 'gn', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f2	4	[(9, 'ji', '..'), (14, 'xo', '..'), (22, 'ko', '..'), (31, 'yj', '..'), (37, 'ec', '..'), (50, 'vf', '..'), (67, 'az', '..')], [(69, 'ev', '..'), (127, 'wl', '..'), (162, 'dg', '..'), (190, 'ze', '..'), (205, 'wz', '..'), (210, 'gw', '..'), (227, 'me', '..')], [(236, 'ot', '..'), (285, 'fg', '..'), (294, 'ld', '..'), (346, 'vs', '..'), (392, 'iy', '..'), (427, 'rs', '..'), (438, 'pd', '..')], [(442, 'fi', '..'), (443, 'yy', '..'), (448, 'dh', '..'), (460, 'mt', '..'), (463, 'iq', '..'), (468, 'up', '..'), (483, 'nf', '..')]

DbFiles	Num Pages	Pages
Listens-p0-f3	3	[(29, 'xf', '..'), (113, 'vh', '..'), (116, 'et', '..'), (142, 'vj', '..'), (148, 'qv', '..'), (166, 'nv', '..'), (183, 'pd', '..')], [(219, 'xe', '..'), (229, 'co', '..'), (286, 'mf', '..'), (335, 'dh', '..'), (367, 'kc', '..'), (396, 'gx', '..'), (429, 'zh', '..')], [(448, 'ee', '..'), (478, 'ce', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

====>> [Pass #] 1

DbFiles	Num Pages	Pages
Listens-p1-f0	15	[(3, 'kf', '..'), (9, 'ji', '..'), (13, 'pf', '..'), (13, 'xr', '..'), (14, 'eq', '..'), (14, 'xo', '..'), (15, 'tm', '..')], [(22, 'ko', '..'), (29, 'xf', '..'), (31, 'yj', '..'), (37, 'ec', '..'), (41, 'zx', '..'), (50, 'vf', '..'), (54, 'to', '..')], [(58, 'pz', '..'), (63, 'fw', '..'), (67, 'az', '..'), (69, 'ev', '..'), (73, 'fk', '..'), (80, 'sy', '..'), (86, 'tf', '..')], [(95, 'qp', '..'), (107, 'hk', '..'), (113, 'vh', '..'), (114, 'hl', '..'), (116, 'et', '..'), (118, 'ph', '..'), (122, 'ws', '..')], [(127, 'wl', '..'), (142, 'vj', '..'), (143, 'cd', '..'), (148, 'qv', '..'), (152, 'dr', '..'), (158, 'zf', '..'), (162, 'dg', '..')], [(166, 'nv', '..'), (177, 'tr', '..'), (181, 'hf', '..'), (183, 'pd', '..'), (183, 'tm', '..'), (190, 'ze', '..'), (195, 'ki', '..')], [(196, 'cj', '..'), (205, 'wz', '..'), (207, 'kx', '..'), (210, 'gw', '..'), (213, 'rn', '..'), (217, 'ax', '..'), (219, 'xe', '..')], [(220, 'tv', '..'), (227, 'me', '..'), (229, 'co', '..'), (236, 'ot', '..'), (239, 'lv', '..'), (249, 'io', '..'), (257, 'aq', '..')], [(264, 'oe', '..'), (265, 'kv', '..'), (274, 'gd', '..'), (280, 'we', '..'), (285, 'fg', '..'), (286, 'mf', '..'), (294, 'ld', '..')], [(296, 'gz', '..'), (335, 'dh', '..'), (341, 'op', '..'), (343, 'uh', '..'), (345, 'pd', '..'), (346, 'vs', '..'), (366, 'gq', '..')], [(367, 'kc', '..'), (373, 'mn', '..'), (392, 'iy', '..'), (396, 'gx', '..'), (411, 'nz', '..'), (419, 'np', '..'), (427, 'rs', '..')], [(429, 'zh', '..'), (433, 'pi', '..'), (438, 'pd', '..'), (442, 'aa', '..'), (442, 'fi', '..'), (442, 'ne', '..'), (443, 'yy', '..')], [(448, 'dh', '..'), (448, 'ee', '..'), (457, 'zb', '..'), (460, 'mt', '..'), (463, 'fb', '..'), (463, 'iq', '..'), (466, 'ob', '..')], [(467, 'jp', '..'), (468, 'ik', '..'), (468, 'up', '..'), (478, 'ce', '..'), (479, 'mu', '..'), (483, 'nf', '..'), (486, 'iw', '..')], [(490, 'th', '..'), (497, 'gn', '..'), (None, None), (None, None), (None, None), (None, None), (None, None)]

SMJ: Songs JOIN Listens

====>> 0

DbFiles	Num Pages	Pages
join/Songs-p2-f0-smj-Listens-p1-f0	4	[(14, 'nc', '..', 'eq', '..'), (14, 'nc', '..', 'xo', '..'), (29, 'uh', '..', 'xf', '..'), (41, 'na', '..', 'zx', '..')], [(63, 'iv', '..', 'fw', '..'), (80, 'br', '..', 'sy', '..'), (80, 'zy', '..', 'sy', '..'), (166, 'aa', '..', 'nv', '..')], [(177, 'fg', '..', 'tr', '..'), (207, 'hh', '..', 'kx', '..'), (210, 'wl', '..', 'gw', '..'), (229, 'sq', '..', 'co', '..')], [(396, 'at', '..', 'gx', '..'), (419, 'qp', '..', 'np', '..'), (429, 'al', '..', 'zh', '..'), (486, 'vb', '..', 'iw', '..')]

Out[18]: <__main__.DbFile at 0x110094f50>

JOINs, GROUPBY IO Cost

In [19]:
```
################################################
## IO Cost Equations
################################################
class DBOptimizer:
  # setting up IO, e.g, Read/Write costs in IOs
  def __init__(self, B, C_r, C_w):
    self.B = B
    (self.C_w, self.C_r) = (C_w, C_r)

  ##########
  # Sort(R) and HP(R)
  def SortCost(self, R):
    # assume repacking optimization, and B-way merge. Assume B ~= B+1
    P_R = R.P()
    if P_R == 0:
      return 0
    return (self.C_r+self.C_w)*P_R*(ceil(log(P_R/(2*self.B), self.B)) + 1)

  def HPCost(self, R):
    # C_r*R.P() to read pages, and C_w*R.P() to write out partition
    return ((self.C_r+self.C_w)*R.P())

  def Sort(self, R):
    R.Sort()
    return self.SortCost(R)
  def HP(self, R):
    R.HP()
    return self.HPCost(R)

  #########
  #### JOIN Algorithms
  #### We'll handle OUT outside the function.
  # JOIN Algo1: P(R), P(S) and B: comute BNLJ IO Cost.
  def BNLJCost(self, R, S):
    ioCost = self.C_r*(R.P() + S.P()*ceil(R.P()/self.B))
    if R == S: # self-join: assume no special optimization.
      pass     # TODO(student): Think if you can improve BNLJ for self-joins?
    return ioCost

  # JOIN Algo2: P(R), P(S), B. Are R and S sorted?
  # Should we assume small backups (linear? Depends on duplicates, and small B)?
  def SMJCost(self, R, S, smallBackup):
    ioCost = 0.0
    if (not R.isSorted): # then eval sort R
      ioCost += self.SortCost(R)
    if (not S.isSorted): # then eval sort S
      ioCost += self.SortCost(S)
    if (smallBackup): # enuf B, and non-duplicates
      ioCost += self.C_r*(R.P() + S.P())
    else:  # assume worst case, and doing BNLJ
```

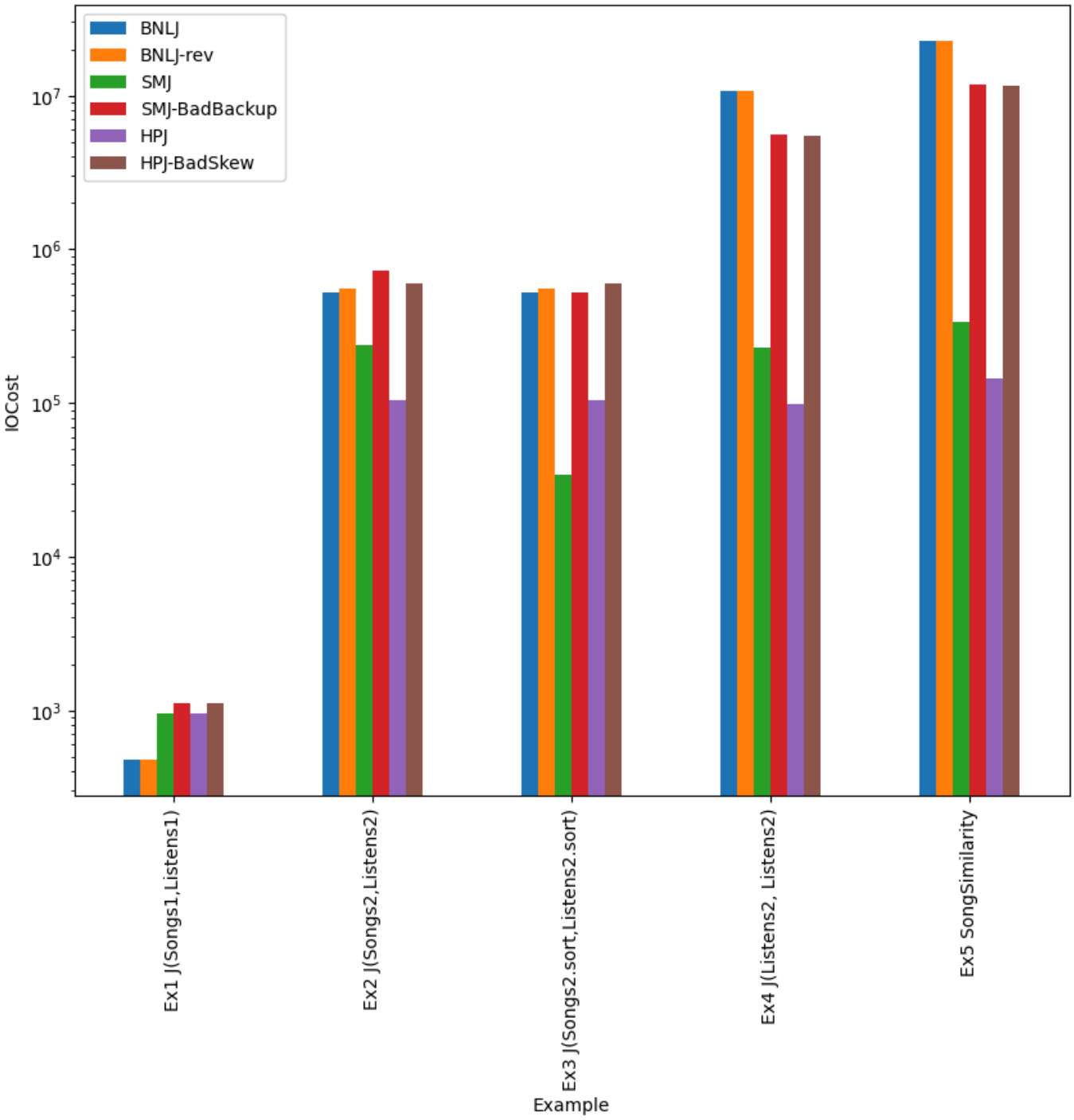
```
    ioCost += min(self.BNLJCost(R, S), self.BNLJCost(S, R))
  if (R == S): # self-join? Halve the cost. No need to Sort(R) twice, etc.
    ioCost /= 2
  return ioCost

# JOIN Algo 3: P(R), P(S), B. Are R and S hash partioned?
# Should we assume few collisions (linear? Depends on hash function, and B)
def HPJCost(self, R, S, fewCollisions):
  ioCost = 0.0
  if (not R.isHPed): # then eval hash-partition R
    ioCost += self.HPCost(R)
  if (not S.isHPed): # then eval hash-partition S
    ioCost += self.HPCost(S)
  if fewCollisions:
    ioCost += self.C_r*(R.P() + S.P())
  else: # worst-case,  and doing BNLJ
    ioCost += min(self.BNLJCost(R, S), self.BNLJCost(S, R))
  if (R == S): # self-join? Halve the cost. Don't need to HP twice, etc.
    ioCost /= 2
  return ioCost

def JoinOUTEstimator(self, R, S, k_WHERE, special = ''):
  # In general, this is a hard problem to model accurately. DBs keep
  # a variety of stats (from prior queries) to predict the size of results.
  # Let's sketch some rough ideas to get some flavor
  #     1. What's the rowsize of OUT?
  #        Simple model = add R.rowsize + S.rowsize
  #        More accurate model: factor in specific columns projected.
  #     2. What's the # rows in OUT?
  #       (a) Simple model = For general tables,
  #           Worst case: R.T()*S.T() rows.
  #           With WHERE filtering: use probability of row satisfying WHERE
  #       (b) Special cases: When JOIN key is unique (e.g. JOIN on studentID),
  #           OUT will have appx min(R.T(), S.T()) rows.
  #       (c) For other special cases (e.g., outerjoin, with WHERE, etc.)
  #           model using probability
  #     We'll use a simple estimator.
  if (R == S):
    sizeRowOUT = R.RowSize()
  else:
    sizeRowOUT = R.RowSize() + S.RowSize()
  numRowsOUT = k_WHERE*R.T()*S.T()
  if (special == 'uniq'): # fewer tuples
    numRowsOUT = min(R.T(), S.T())*k_WHERE
  return (numRowsOUT, sizeRowOUT)

def GroupBYEstimator(self, R, k_HAVING):
  # Same ideas as JOIN estimator
  sizeRowOUT = R.RowSize()
  numRowsOUT = R.T()*k_HAVING
  return (numRowsOUT, sizeRowOUT)

## GROUPBY on Column X.
# GROUPBY Algo1: Sorting R (on columns X) will group rows in sorted order.
# We can then scan the groups to compute aggregates.
def GroupBySort(self, R):
  return self.SortCost(R)

# GROUPBY Algo2: Hash partititioning R (on columns X) will place relevant groups
# in the same bucket, making it easy to compute aggregates.
def GroupByHash(self, R):
  return self.HPCost(R)

## Basic "v1" query optimizer
# DB checks cost of all plans. Picks lowest cost, and executes that plan.
# 1. In this "v1", we assume (a) no backup, and (b) no data skew for simplicity.
```

```
  # In a more sophisticated version, the optimer will keep stats to estimate
  # the shape of the data (e.g., duplicates, skew, etc.).
  # 2. Also, we ignore OUT because all the JOINs will produce the same result,
  # and will cost the same to output the OUT.

  def EvaluateJoinPlans(self, R, S):
    plans = {}
    plans["BNLJ"] = self.BNLJCost(R, S)
    plans["BNLJ-rev"] = self.BNLJCost(S, R)
    plans["SMJ"] = self.SMJCost(R, S, True)
    plans["SMJ-BadBackup"] = self.SMJCost(R, S, False)
    plans["HPJ"] = self.HPJCost(R, S, True)
    plans["HPJ-BadSkew"] = self.HPJCost(R, S, False)

    return plans

  def EvaluateGroupByPlans(self, R):
    plans = {}
    plans["GroupBy-Sort"] = self.GroupBySort(R)
    plans["GroupBy-Hash"] = self.GroupByHash(R)
    return plans

  def QueryOptimizerV1(self, R, S):
    plans = self.EvaluateJoinPlansV1(R, S)
    print(plans)
```

In [20]:

```
from IPython.display import display, HTML
import pandas as pd

def addToPd(desc, plans):
  return([desc, plans.get("BNLJ"), plans.get("BNLJ-rev"),
          plans.get("SMJ"), plans.get("SMJ-BadBackup"),
          plans.get("HPJ"), plans.get("HPJ-BadSkew"),
          plans.get("GroupBy-Sort"), plans.get("GroupBy-Hash"),
          plans.get("OUT"), plans.get("OUT-uniq")])

ex = []
db100 = DBOptimizer(100, 1.0, 1.0)
# Example1: Evaluate JOINing two non-big tables
Songs1 = Table(10.0*GB, 1024.0*Bytes)
Listens1 = Table(10.0*GB, 1024.0*Bytes)
ex.append(addToPd("Ex1 J(Songs1,Listens1)",
                  db100.EvaluateJoinPlans(Songs1, Listens1)))

# Example2: Evaluate JOINing two big tables
Songs2 = Table(100.0*GB, 1024.0*Bytes)
Listens2 = Table(2.0*TB, 1024.0*Bytes)
ex.append(addToPd("Ex2 J(Songs2,Listens2)",
                  db100.EvaluateJoinPlans(Songs2, Listens2)))

# Example3: Assume Songs2 and Listens2 are sorted.
Songs2.Sort()
Listens2.Sort()
ex.append(addToPd("Ex3 J(Songs2.sort,Listens2.sort)",
                  db100.EvaluateJoinPlans(Songs2, Listens2)))

# Example4: Self-join
Songs2.Reset()
Listens2.Reset()
ex.append(addToPd("Ex4 J(Listens2, Listens2)",
                  db100.EvaluateJoinPlans(Listens2, Listens2)))

# Example5: from Spotify song_similarity CTE
listens_rowSize = 32.0*Bytes
listens_numRows = pow(10.0, 11) # 100 billion listens
```

```
listens_sizeinMBs = listens_rowSize*listens_numRows/MB
Listens = Table(listens_sizeinMBs, 32.0*Bytes)
ex.append(addToPd("Ex5 SongSimilarity",
                  db100.EvaluateJoinPlans(Listens, Listens)))

pdf = pd.DataFrame(ex, columns= ["Example", "BNLJ", "BNLJ-rev", "SMJ",
                                 "SMJ-BadBackup", "HPJ", "HPJ-BadSkew",
                                 "GroupBy-Sort", "GroupBy-Hash",
                                 "OUT", "OUT-uniq"])

import matplotlib.pyplot as plt
%matplotlib inline
pdf.plot.bar(x='Example', y = ['BNLJ', 'BNLJ-rev', 'SMJ', 'SMJ-BadBackup',
                               'HPJ', 'HPJ-BadSkew',
                               'GroupBy-Sort', 'GroupBy-Hash', 'OUT', 'OUT-uniq'],
                               figsize=(10,8), logy=True)
plt.ylabel("IOCost")
plt.show()
```



OBSERVATIONS

In Example 1, BNLJ is cheaper than SMJ. Why? Songs1 and Listens1 (10 GBs) are pretty small vs B. (100 page = 100*64MBs = 6.4GBs). Intuitively, makes sense. That is, if the data mostly fits in RAM, BNLJ will JOIN in RAM, without any pre-processing (Sort or HP) overhead.

In other cases, we see SMJ and HPJ are doing better than BNLJ. Especially, when the number of pages in join tables is big compared to B. Of course, in the worst case, SMJ (with bad backbackup) or HPJ (with bad skew) could perform poorly. SMJ and HPJ do even better, if they were pre-sorted or pre-partitioned (perhaps for another query or index).

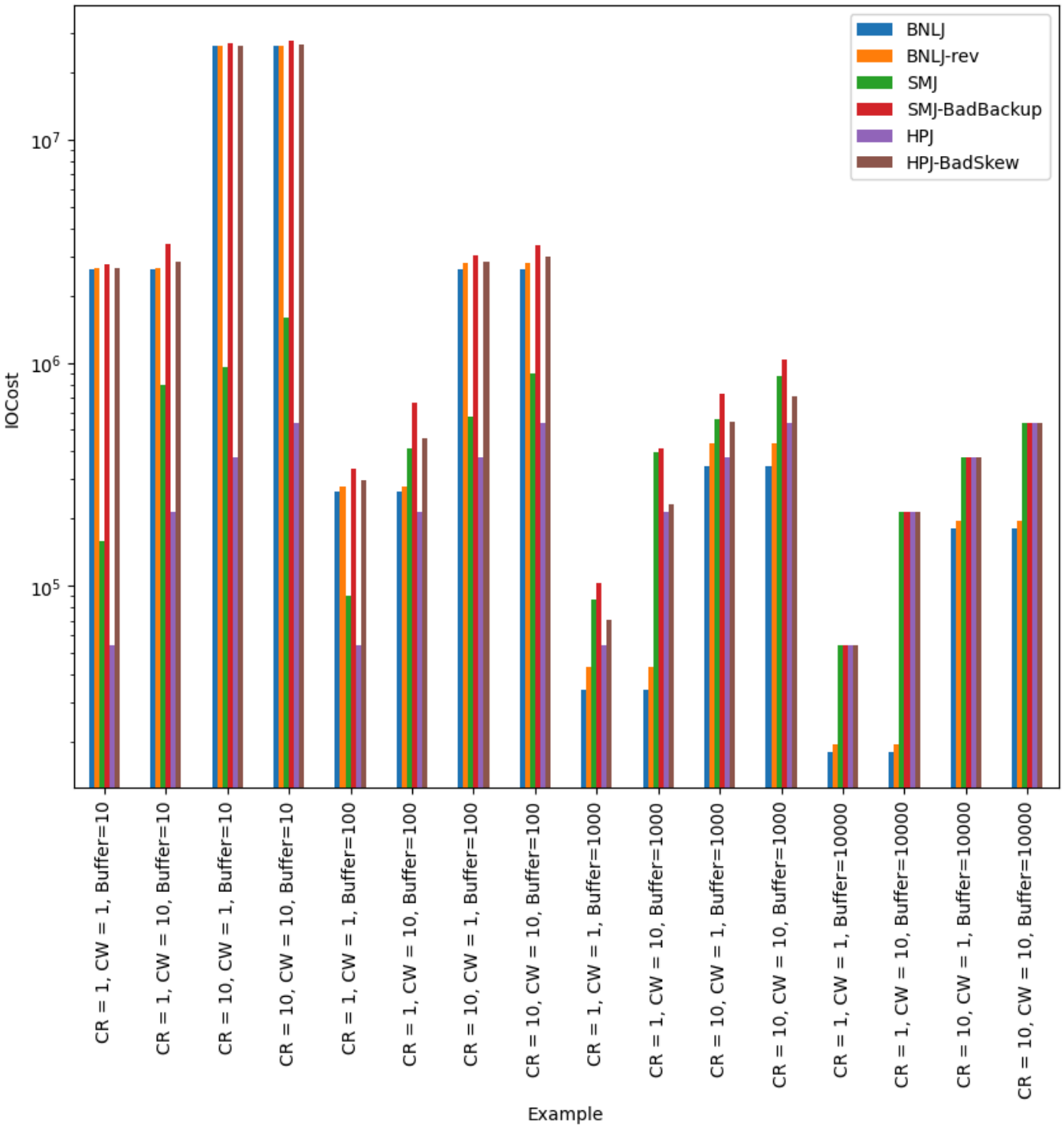
In practice, the query optimizer will evaluate the costs, and pick the algorithm with the least expected cost (e.g. based on prior history or statistics it maintains about the likelihood of duplicates.)

Exercise 2 Problems

Consider Songs3=100GB and Listens3 = 1.0TB. Both have row size = 1024.0 Bytes. Let's explore the impact of different machine configurations on IO costs.

Machines have different RAM buffer sizes (e.g., 32 GBs to 640GBs). Also, when handling multiple parallel queries, query optimizers may use only a portion of the available RAM per query (e.g. 20% for one query, 60% for another, etc). IOdevices often have different C_r and C_w costs. In some devices, C_w is 10x as much as C_r, and in some devices C_w is much faster than C_r.

In [44]:
```
""" Problem 2.1:
Compute IOcosts of BNLJ, SMJ, HPJ for B = 10, 100, 1000, 10000, for different
values of C_r and C_w. Ignore OUT (cost of writing output) in these calculations.
What are the relative costs of SMJ and HPJ versus BNLJ?
"""

ex2 = []
Songs3 = Table(100.0*GB, 1024.0*Bytes)
Listens3 = Table(1.0*TB, 1024.0*Bytes)

DBOptimizerObjects= []

for B in [10, 100, 1000, 10000]:
    for CR in [1,10]:
        for CW in [1, 10]:
            dbo = DBOptimizer(B, CR, CW)
            ex2.append(addToPd("CR = "+str(CR)+", CW = "+str(CW)+", Buffer="+str(B),d

pdf = pd.DataFrame(ex2, columns= ["Example", "BNLJ", "BNLJ-rev", "SMJ",
                                  "SMJ-BadBackup", "HPJ", "HPJ-BadSkew",
                                  "GroupBy-Sort", "GroupBy-Hash",
                                  "OUT", "OUT-uniq"])

%matplotlib inline
pdf.plot.bar(x='Example', y = ['BNLJ', 'BNLJ-rev', 'SMJ', 'SMJ-BadBackup',
                               'HPJ', 'HPJ-BadSkew',
                               'GroupBy-Sort', 'GroupBy-Hash', 'OUT', 'OUT-uniq'],
                               figsize=(10,8), logy=True)
plt.ylabel("IOCost")
plt.show()
```



From the graph we can see that:

1. BNLJ is less when the buffer size is of a larger size.
2. SMJ and and HPJ are more costly as compared to BNLJ when the buffer is of a larger size and vice versa when the buffer size is of a smaller size.

In [ ]: