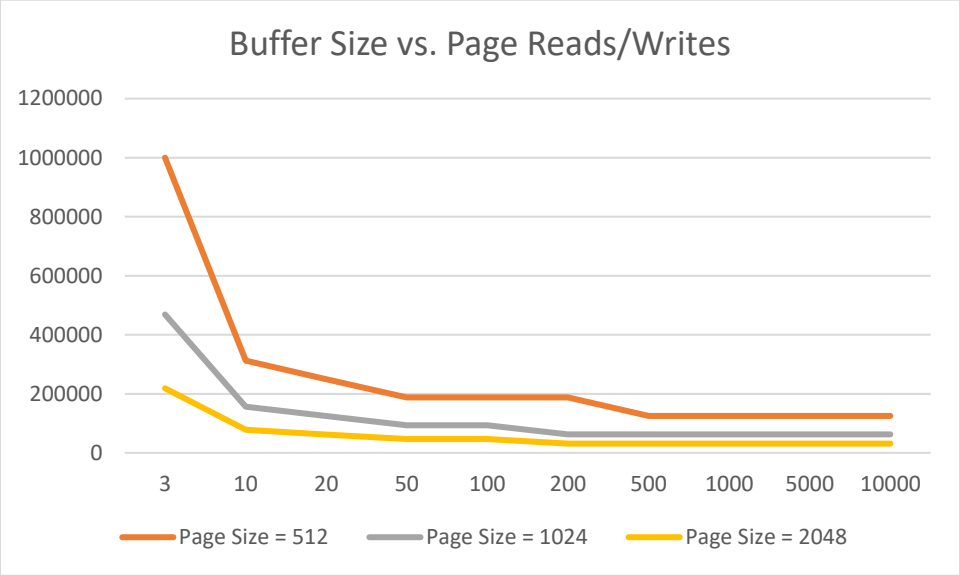
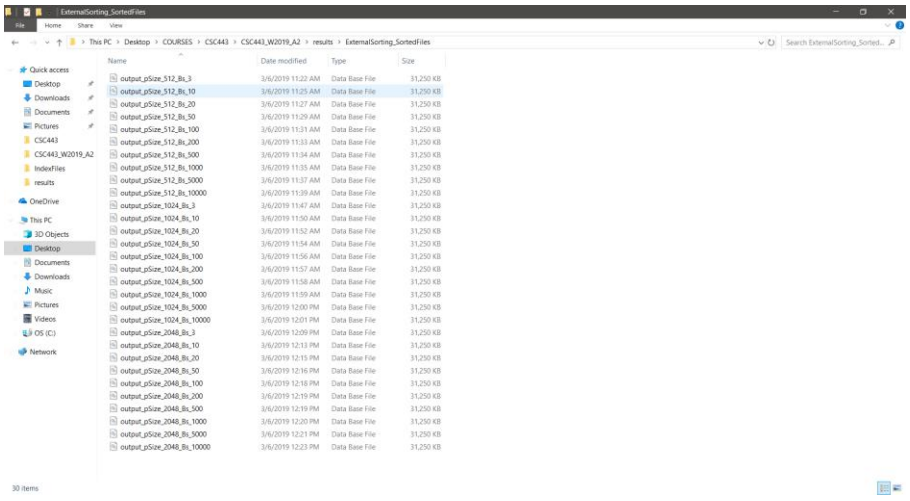
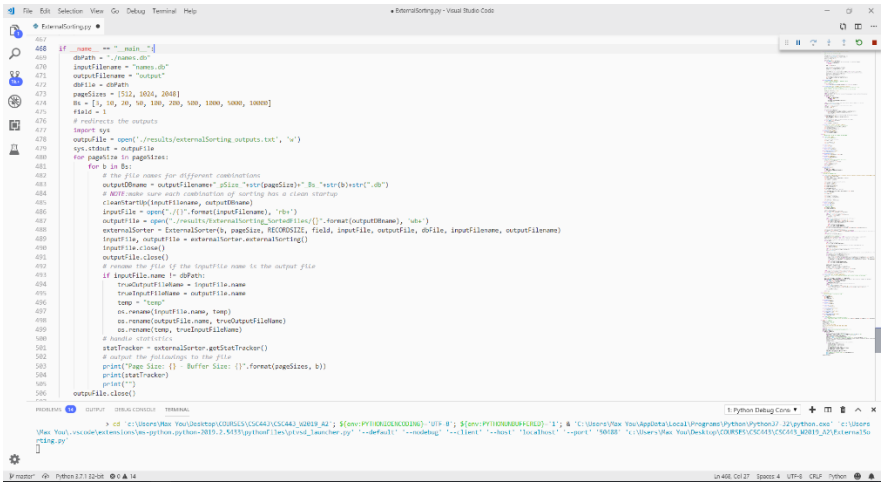
Part one Answers:

Total page reads/writes										
Page Size\B	3	10	20	50	100	200	500	1000	5000	10000
512	1000000	312500	250000	187500	187500	187500	125000	125000	125000	125000
1024	468750	156250	125000	93750	93750	62500	62500	62500	62500	62500
2048	218750	78125	62500	46875	46875	31250	31250	31250	31250	31250
Total Passes (unit in pages)										
Page size\B	3	10	20	50	100	200	500	1000	5000	10000
512	16	5	4	3	3	3	2	2	2	2
1024	15	5	4	3	3	2	2	2	2	2
2048	14	5	4	3	3	2	2	2	2	2



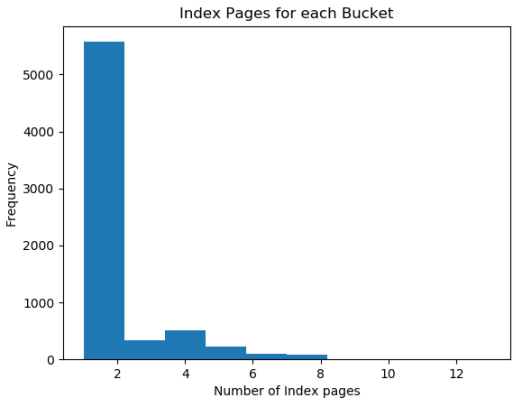
Observations: From the line chart, we see that the number of pages read (y-axis) diminish as the number of buffer pages (x-axis) increases. These empirical results match the equation provided in class, which is $\log_{B-1}\frac{N}{B}$, as B increases, the number gets smaller.

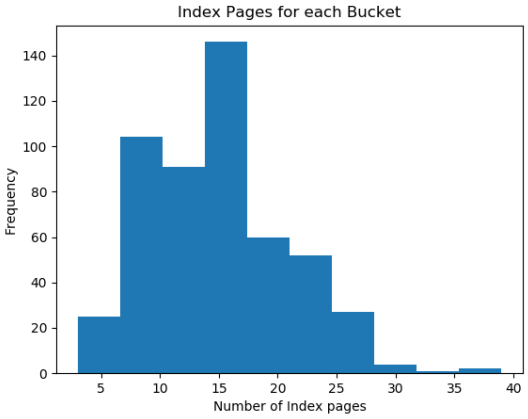




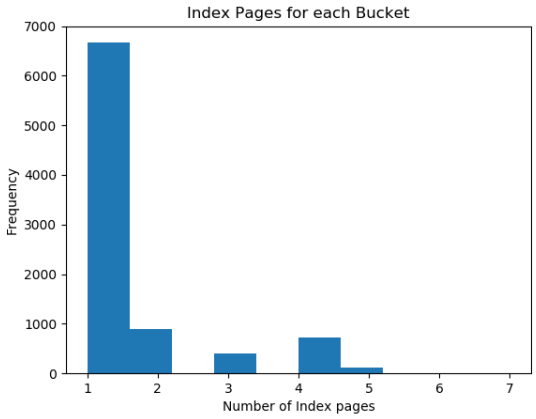
Part two answers:

Index Pages for Linear Hashing Index File	Index Pages for Static Hashing Index File





Index Pages for Extendible Hash Index File

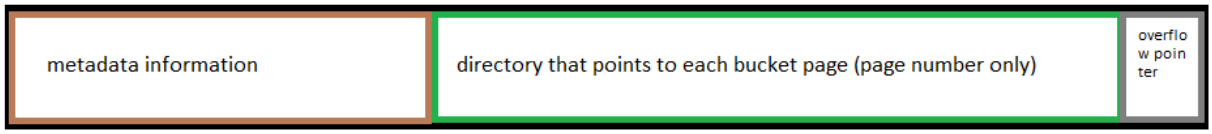


Outputs from the program:

	number of buckets	Regular index pages	Overflow pages
Static hash index	512	512	7084
Extendible hash index	8814	9843	4393
Linear hash index	6845	6845	4967

Static Hash Index File **header page** Format contains the following information in order:

- Page size – 2 bytes (determine each page size of the index file)
- Index file type – 1 byte (to identify the index type of this index file)
- Fixed number of buckets – 1 byte (number of fixed buckets in this index file)
- Overflow page pointer size – 1 byte (number of bytes to read for the overflow page number)
- Record entry size – 1 byte (the size of each record entry in index and overflow pages)
- Bytes reserved to store number of record entries are stored in index page (exclude directory page) – 1 byte
- Directory entry size – 2 bytes (the size of each directory entry)
- A directory of buckets that points to the index page for the rest of the header page
- Overflow pointer for the next directory page – 2 bytes (points to the next directory page)



Static Hash index File **directory page** Format contains the following information in order:

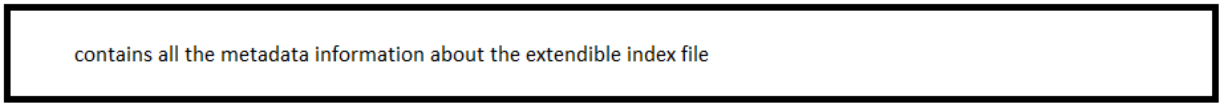
- Number of directory entries in this page – 2 bytes (number of fixed size directory entries in this page)
- A list of directory entries (a sequential list of fixed size directory entries)
- Overflow directory pointer to the next directory page – 2 bytes (the page number for the next directory page)

Static Hash index File **index and overflow page** format contains the following information in order:

- Number of records in this page – 2 bytes (number of fixed size record entries in this page)
- All the record entries (a sequential list of fixed size record entries)
- At the end, there is an overflow page pointer – 2 bytes (the page number for the overflow page of this bucket)

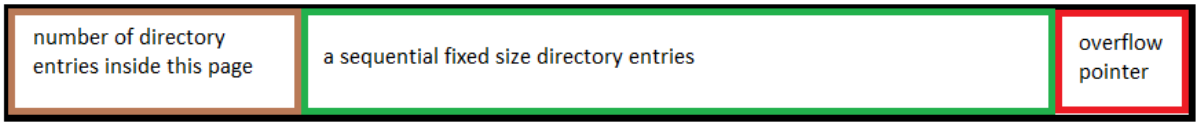
Extendible Hash Index File **header page** Format contains the following information in order:

- Page size – 2 bytes (determine each page size of the index file)
- Index file type – 1 byte (to identify the index type of this index file)
- First directory page start location – 1 byte (the page number of the first directory page)
- Global depth – 1 byte (the global depth is needed for the hash function)
- Overflow page pointer size – 1 byte (number of bytes to read to interpret the overflow page number)
- Directory entry size – 1 byte (size of each directory entry in directory pages)
- Record entry size – 1 byte (size of each record entry in index and overflow pages)
- Bytes reserved to store the local depth for each bucket page – 1 byte (number of bytes to read to interpret the local depth of the index page)
- Bytes reserved to store number of record entries are stored in index page (exclude directory page) – 1 byte (number of bytes to read to know how many records in this page)



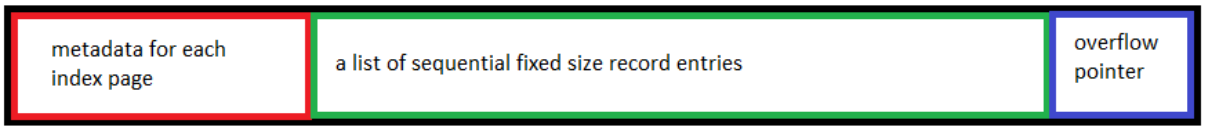
Extendible Hash Index File **directory page** Format contains the following information in order:

- Number of directory entries in this page – 2 bytes (number of fixed size directory entries in this page)
- A list of directory entries (a sequential list of fixed size directory entries)
- Overflow directory pointer to the next directory page – 2 bytes (the page number for the next directory page)



Extendible Hash Index File **index page of each bucket** contains the following information in order:

- Local depth of this bucket – 2 bytes (the local depth of this index page, this is required to determine whether to double the directories)
- Number of record entries in this page – 2 bytes (the number of record entries currently in this page)
- A sequential fixed size list of record entries (a list of fixed size record entries)
- Overflow page pointer – 2 bytes (the overflow page number of the next overflow page of this bucket)

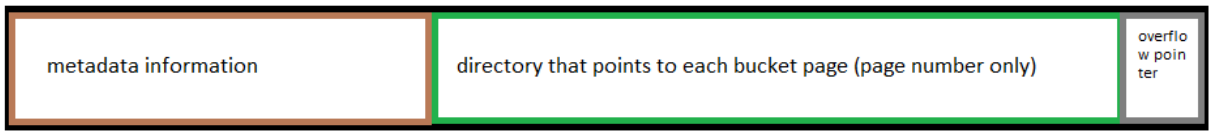


Extendible Hash Index File **overflow page of each bucket** contains the following information in order:

- Number of records in this page – 2 bytes (number of fixed size record entries in this page)
- All the record entries (a sequential list of fixed size record entries)
- At the end, there is an overflow page pointer – 2 bytes (the page number for the overflow page of this bucket)

Linear Hash Index File **header page** Format:

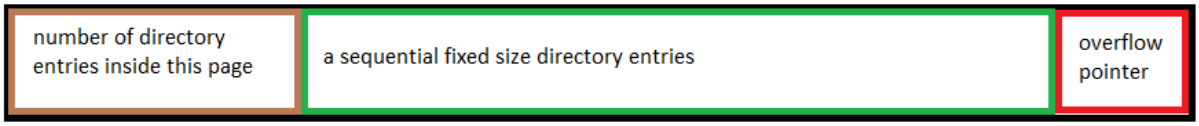
- Page size – 2 bytes (determine each page size of the index file)
- Index file type – 1 byte (to identify the index type of this index file)
- Initial number of buckets – 1 byte (the initial number of buckets to begin with, need for the hash function)
- Current level – 1 byte (the current level, it is need for the hash function)
- Next bucket pointer – 1 byte (the bucket that the *next* pointer points to, needed to determine how many bits to interpret)
- Overflow page pointer size – 1 byte (number of bytes to read to interpret the overflow page number)
- Record entry size – 1 byte (the size of each record entry)
- Bytes reserved to store number of record entries are stored in index page (exclude directory page) – 1 byte (number of bytes to read to interpret the number of records in the current page)
- Directory entry size (size of each directory entry)
- A directory of buckets that points to the index page for the rest of the header page (a list of fixed-size directory entries)
- Overflow pointer for the next directory page – 2 bytes (the page number that points to the next directory page)



Linear Hash Index File **directory page** Format contains the following information in order:

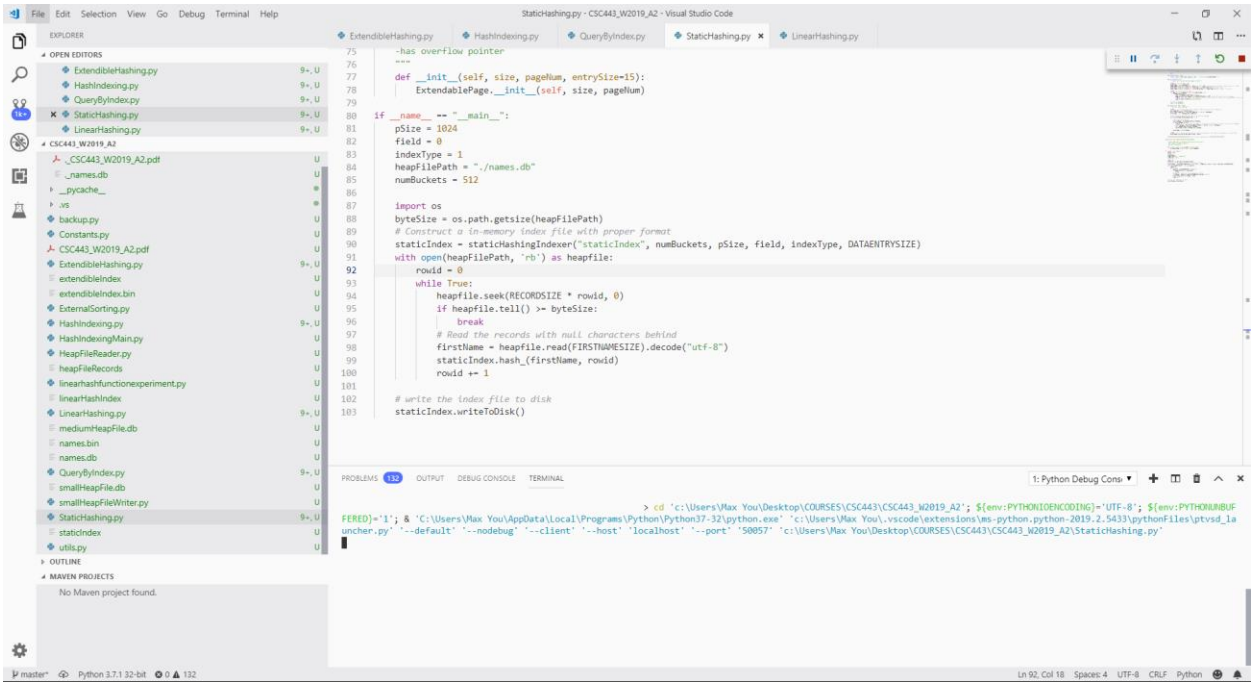
- Number of directory entries in this page – 2 bytes (number of fixed size directory entries in this page)

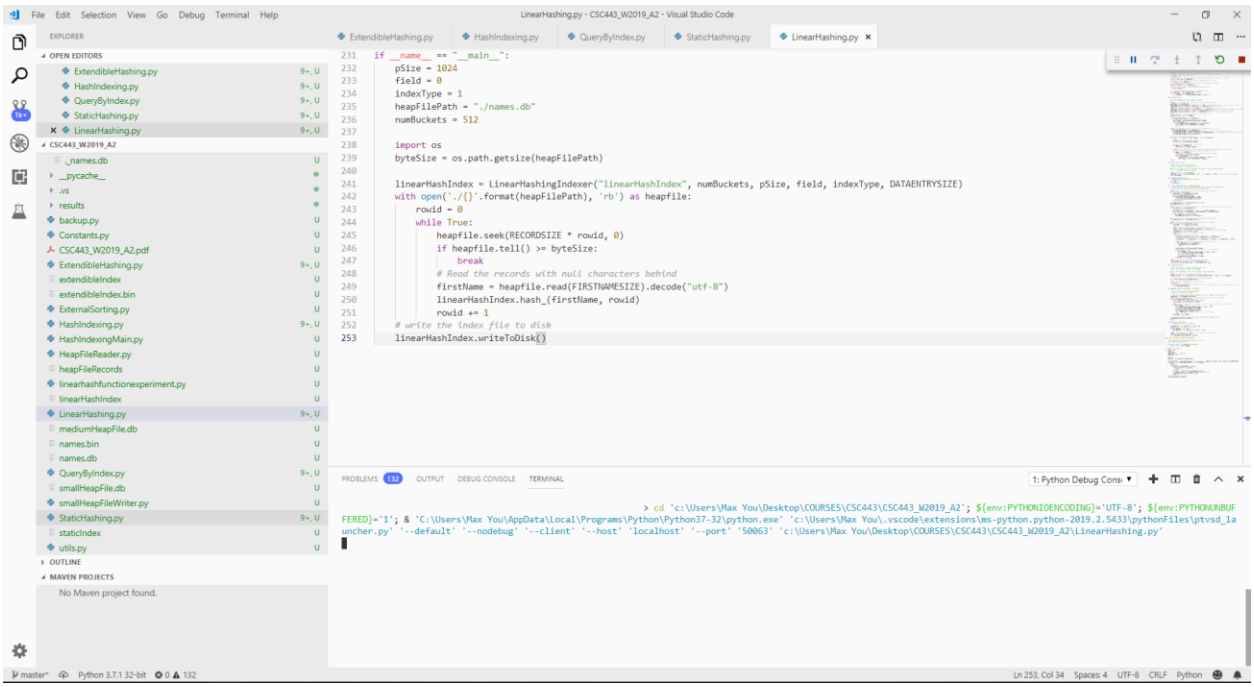
- A list of directory entries (a sequential list of fixed size directory entries)
- Overflow directory pointer to the next directory page – 2 bytes (the page number for the next directory page)

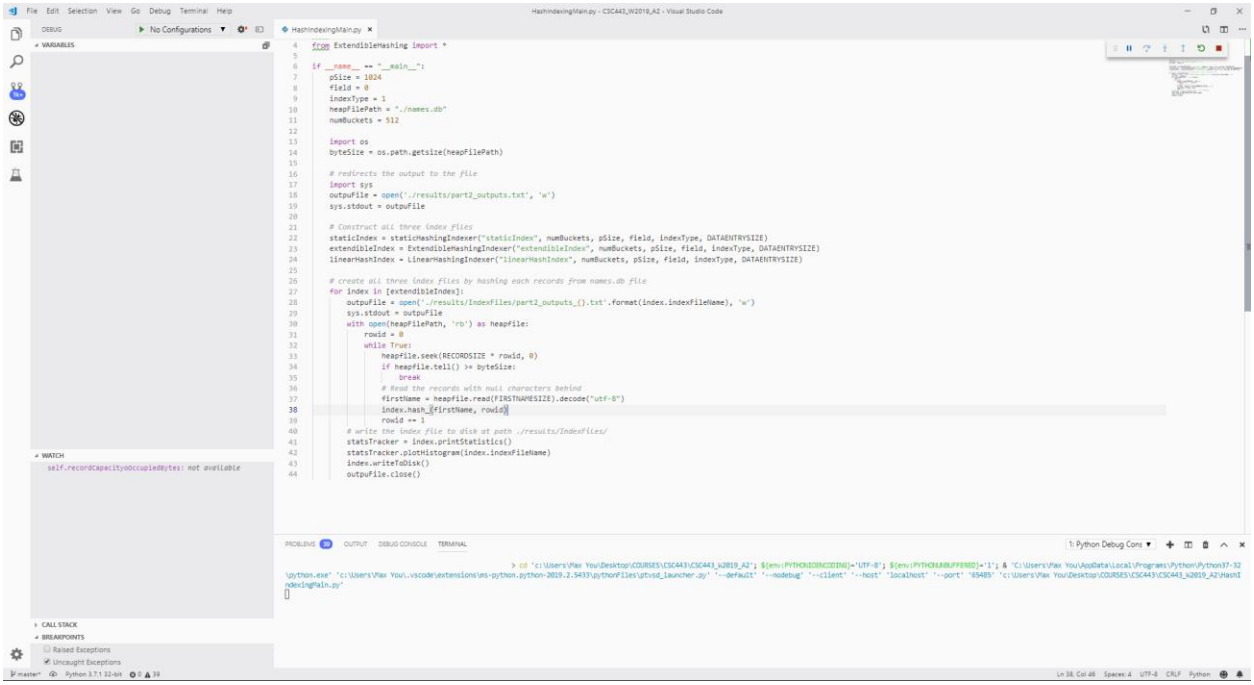


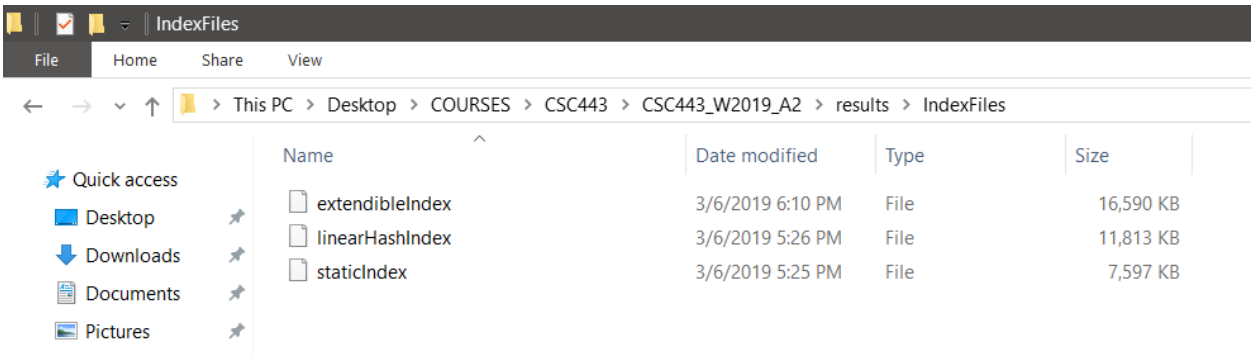
Linear Hash Index File **index and overflow page** Format:

- Number of records in this page – 2 bytes (number of fixed size record entries in this page)
- All the record entries (a sequential list of fixed size record entries)
- At the end, there is an overflow page pointer – 2 bytes (the page number for the overflow page of this bucket)




File Edit Selection View Go Debug Terminal Help
LinearHashing.py - CSC443_W2019_A2 - Visual Studio Code
EXPLORER
OPEN EDITORS
ExtendibleHashing.py
HashIndexing.py
QueryByIndex.py
StaticHashing.py
LinearHashing.py
CSC443_W2019_A2
_names.db
__pycache__
.vs
results
backup.py
Constants.py
CSC443_W2019_A2.pdf
ExtendibleHashing.py
extendibleIndex
extendibleIndex.bin
ExternalSorting.py
HashIndexing.py
HashIndexingMain.py
HeapFileReader.py
heapFileRecords
linearhashfunctionexperiment.py
linearHashIndex
LinearHashing.py
mediumHeapFile.db
names.bin
names.db
QueryByIndex.py
smallHeapFile.db
smallHeapFileWriter.py
StaticHashing.py
staticIndex
utils.py
OUTLINE
MAVEN PROJECTS
No Maven project found.
231 if __name__ == "__main__":
232 pSize = 1024
233 field = 0
234 indexType = 1
235 heapFilePath = "./names.db"
236 numBuckets = 512
237
238 import os
239 byteSize = os.path.getsize(heapFilePath)
240
241 linearHashIndex = LinearHashingIndexer("linearHashIndex", numBuckets, pSize, field, indexType, DATAENTRYSIZE)
242 with open('./{}'.format(heapFilePath), 'rb') as heapfile:
243 rowid = 0
244 while True:
245 heapfile.seek(RECORDSIZE * rowid, 0)
246 if heapfile.tell() >= byteSize:
247 break
248 # Read the records with null characters behind
249 firstName = heapfile.read(FIRSTNAMESIZE).decode("utf-8")
250 linearHashIndex.hash_(firstName, rowid)
251 rowid += 1
252 # write the index file to disk
253 linearHashIndex.writeToDisk()
PROBLEMS 132
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Cons
> cd 'c:\Users\Max You\Desktop\COURSES\CSC443\CSC443_W2019_A2'; ${env:PYTHONIOENCODING}='UTF-8'; ${env:PYTHONUNBUFFERED}='1'; & 'C:\Users\Max You\AppData\Local\Programs\Python\Python37-32\python.exe' 'c:\Users\Max You\.vscode\extensions\ms-python.python-2019.2.5433\pythonFiles\ptvsd_launcher.py' '--default' '--nodebug' '--client' '--host' 'localhost' '--port' '50063' 'c:\Users\Max You\Desktop\COURSES\CSC443\CSC443_W2019_A2\LinearHashing.py'
master*
Python 3.7.1 32-bit
0 0 132
Ln 253, Col 34
Spaces: 4
UTF-8
CRLF
Python


File Edit Selection View Go Debug Terminal Help
HashIndexingMain.py - CSC443_W2019_A2 - Visual Studio Code
DEBUG
No Configurations
HashIndexingMain.py
VARIABLES
WATCH
self.recordCapacityOccupiedBytes: not available
CALL STACK
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
4 from ExtendibleHashing import *
5
6 if __name__ == "__main__":
7 pSize = 1024
8 field = 0
9 indexType = 1
10 heapFilePath = "./names.db"
11 numBuckets = 512
12
13 import os
14 byteSize = os.path.getsize(heapFilePath)
15
16 # redirects the output to the file
17 import sys
18 outpuFile = open('./results/part2_outputs.txt', 'w')
19 sys.stdout = outpuFile
20
21 # Construct all three index files
22 staticIndex = staticHashingIndexer("staticIndex", numBuckets, pSize, field, indexType, DATAENTRYSIZE)
23 extendibleIndex = ExtendibleHashingIndexer("extendibleIndex", numBuckets, pSize, field, indexType, DATAENTRYSIZE)
24 linearHashIndex = LinearHashingIndexer("linearHashIndex", numBuckets, pSize, field, indexType, DATAENTRYSIZE)
25
26 # create all three index files by hashing each records from names.db file
27 for index in [extendibleIndex]:
28 outpuFile = open('./results/IndexFiles/part2_outputs_{}.txt'.format(index.indexFileName), 'w')
29 sys.stdout = outpuFile
30 with open(heapFilePath, 'rb') as heapfile:
31 rowid = 0
32 while True:
33 heapfile.seek(RECORDSIZE * rowid, 0)
34 if heapfile.tell() >= byteSize:
35 break
36 # Read the records with null characters behind
37 firstName = heapfile.read(FIRSTNAMESIZE).decode("utf-8")
38 index.hash_(firstName, rowid)
39 rowid += 1
40 # write the index file to disk at path ./results/IndexFiles/
41 statsTracker = index.printStatistics()
42 statsTracker.plotHistogram(index.indexFileName)
43 index.writeToDisk()
44 outpuFile.close()
PROBLEMS 30
OUTPUT
DEBUG CONSOLE
TERMINAL
1: Python Debug Cons
> cd 'c:\Users\Max You\Desktop\COURSES\CSC443\CSC443_W2019_A2'; ${env:PYTHONIOENCODING}='UTF-8'; ${env:PYTHONUNBUFFERED}='1'; & 'C:\Users\Max You\AppData\Local\Programs\Python\Python37-32\python.exe' 'c:\Users\Max You\.vscode\extensions\ms-python.python-2019.2.5433\pythonFiles\ptvsd_launcher.py' '--default' '--nodebug' '--client' '--host' 'localhost' '--port' '65485' 'c:\Users\Max You\Desktop\COURSES\CSC443\CSC443_W2019_A2\HashIndexingMain.py'
master*
Python 3.7.1 32-bit
0 0 30
Ln 38, Col 46
Spaces: 4
UTF-8
CRLF
Python

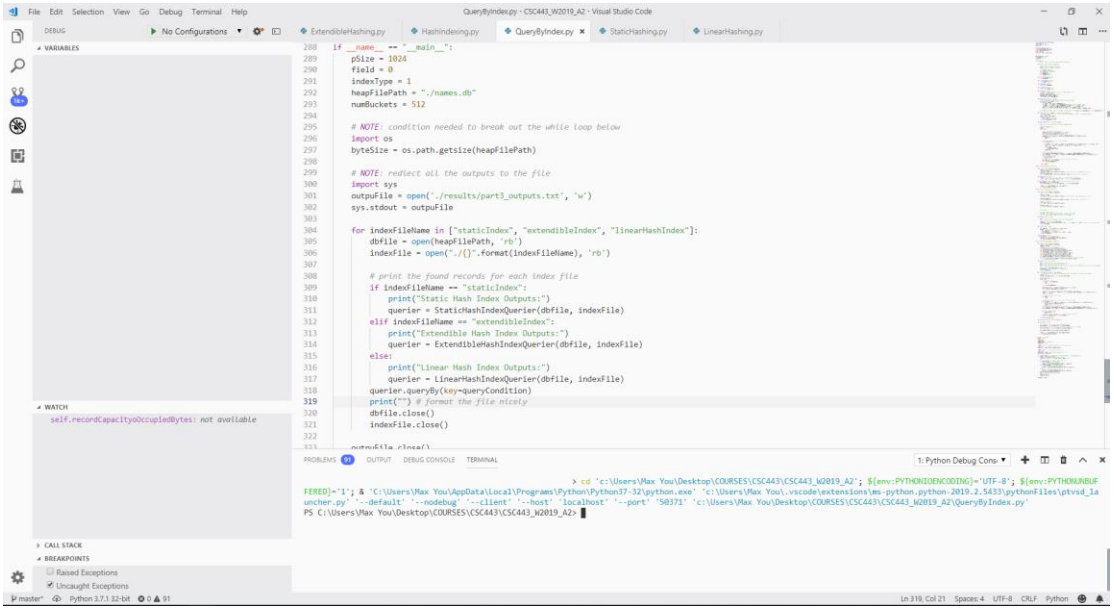


Part three answers:

Index type \ metadata	Index Page Reads	Data Page Reads	Index Page Writes	Data Page Writes
Static hashing index	18	39	0	0
Extendible hashing index	3	39	0	0
Linear hashing index	2	39	0	0

Observations:

Firstly, we noticed that static hashing index has the most index page reads since mostly like it has the greatest number of overflow pages at that bucket due to its property of fixed number of buckets. Secondly, we see that extendible hash index has 3 index page reads, compare to other hash index files, it needs to read an extra page that contains the desired directory entry. Lastly, we noticed that linear hash index has very little index page reads, it makes sense when combine the histogram results from above, we see most of the buckets have only 1-2 index pages. The number of data page reads should be the same across all three type of index file since record entry contains the rowid as the value; each time we see the rowid with the matched first name, we need to fetch a page from the database file to get the actual records. Finally, query operations should not have write cost, we only need to read in the data page to find the records.



The queried results are the same across all three hash index files, as follows:

First name: Nona - Last name: Buentello - Email: nona.buentello@yahoo.com

First name: Nona - Last name: Kluesner - Email: nona.kluesner@hotmail.com

First name: Nona - Last name: Burgoon - Email: nona.burgoon@aol.com

First name: Nona - Last name: Bloomberg - Email: nona.bloomberg@gmail.com

First name: Nona - Last name: Gilmartin - Email: nona.gilmartin@gmail.com

First name: Nona - Last name: Evan - Email: nona.evan@yahoo.co.uk

First name: Nona - Last name: Rowlands - Email: nona.rowlands@aol.com

First name: Nona - Last name: Cabe - Email: nona.cabe@aol.com

First name: Nona - Last name: Quackenbush - Email: nona.quackenbush@charter.net

First name: Nona - Last name: Lucht - Email: nona.lucht@yahoo.ca

First name: Nona - Last name: Mapp - Email: nona.mapp@aol.com

First name: Nona - Last name: Enlow - Email: nona.enlow@microsoft.com

First name: Nona - Last name: Hulbert - Email: nona.hulbert@msn.com

First name: Nona - Last name: Mcclurg - Email: nona.mcclurg@rediffmail.com

First name: Nona - Last name: Neil - Email: nona.neil@msn.com

First name: Nona - Last name: Sipp - Email: nona.sipp@gmail.com

First name: Nona - Last name: Catchings - Email: nona.catchings@bellsouth.net

First name: Nona - Last name: Skalski - Email: nona.skalski@gmail.com

First name: Nona - Last name: Mccloskey - Email: nona.mccloskey@aol.com

First name: Nona - Last name: Lilly - Email: nona.lilly@yahoo.com

First name: Nona - Last name: Hirth - Email: nona.hirth@verizon.net

First name: Nona - Last name: Arias - Email: nona.arias@aol.com

First name: Nona - Last name: Lansberry - Email: nona.lansberry@yahoo.com

First name: Nona - Last name: Cote - Email: nona.cote@yahoo.co.in

First name: Nona - Last name: Forest - Email: nona.forest@exxonmobil.com

First name: Nona - Last name: Vizcaino - Email: nona.vizcaino@gmail.com

First name: Nona - Last name: Achenbach - Email: nona.achenbach@rediffmail.com

First name: Nona - Last name: Belliveau - Email: nona.belliveau@gmail.com

First name: Nona - Last name: Sheehan - Email: nona.sheehan@hotmail.com

First name: Nona - Last name: Edgar - Email: nona.edgar@yahoo.com

First name: Nona - Last name: Cyr - Email: nona.cyr@gmail.com

First name: Nona - Last name: Peek - Email: nona.peek@shell.com

First name: Nona - Last name: Budd - Email: nona.budd@gmail.com

First name: Nona - Last name: Preston - Email: nona.preston@aol.com

First name: Nona - Last name: Vizcaino - Email: nona.vizcaino@gmail.com

First name: Nona - Last name: Snook - Email: nona.snook@hotmail.com

First name: Nona - Last name: Lamas - Email: nona.lamas@apple.com

First name: Nona - Last name: Ventimiglia - Email: nona.ventimiglia@gmail.com

First name: Nona - Last name: Woodworth - Email: nona.woodworth@gmail.com