

IŞIK UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

*B. S. Project*

**EXTERNAL SORTING**

by

YASIN NUR

209CS2158

JANUARY 2014



IŞIK UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

*B. S. Project*

**EXTERNAL SORTING**

by

YASİN NUR

209CS2158

JANUARY 2014

**Supervised by:** Yrd.Doç.Dr. **Ali İNAN**

IŞIK UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

**EXTERNAL SORTING**

YASİN NUR

Approved by:

**Project Supervisor:**

Yrd.Doç.Dr. Ali İNAN \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Department Head:**

Prof. Dr. Ercan SOLAK \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Approval Date:** 16/01/2014

**EXTERNAL SORTING**

**Abstract**

External sorting makes it possible to efficiently sort large quantities of data with limited memory. This algorithm is a memory sorting algorithm. It means that to get the work from RAM and give it to hard disk. Data store in hard disk and every run step the algorithm read data from hard disk, sort them, and write back to hard disk. This project analyzes the algorithm efficiency and argues experiments result.

**Acknowledgements**

First, I thank my project supervisor Asst.Prof. Ali İNAN. I am very lucky to study with him because working with him was intellectually rewarding and great experience for me. When I need information about project, Mr. İNAN was available and gave answers satisfactorily and every time I feel his support and constant encouragement. His friendly approach to me gave very big encouragement and I made this project with enthusiasm.

Second I thank to Department of Computer Engineering in Isik University. All my teachers contribute and increase my knowledge. My computer skills and personal attitudes are their success.

Lastly I thank to my friends and my family for their patience and their support.

TABLE OF CONTENTS

Abstract

**Acknowledgements**

**Table of Contents**

List of Figures

List of Tables

[Definitions, Acronyms and Abbreviations](#_Toc377509578)

[1. Introduction 1](#_Toc377509579)

[2. Development 1](#_Toc377509580)

[2.1 Goals and Tradeoffs 1](#_Toc377509581)

[2.2 Desing Issues 2](#_Toc377509582)

[3. Theory of External Sorting 2](#_Toc377509583)

[3.1. Merge Sort 2](#_Toc377509584)

[3.2. External Merge Sort 3](#_Toc377509585)

[4. Experiment of External Sorting Algorithm 5](#_Toc377509586)

[4.1. Time According to Number of Page 9](#_Toc377509587)

[4.2. Time According to Buffer Size 10](#_Toc377509588)

[4.3. Number of IO vs Number of Page and Buffer Size 11](#_Toc377509589)

[5. Discussion Experiment Results 12](#_Toc377509590)

[6. Conclusion 12](#_Toc377509591)

[REFERENCES](#_Toc377509592)

# LIST OF FIGURES

Fıgure 3.1.1 Example of Two Way Merge Sort of a Seven Page File ...……………………………...3

Fıgure 3.2.1 External Merge Sort Pass 0……..…………….……………………………………4

Fıgure 3.2.2 External Merge Sort Pass 1, 2… ………………….…...……………………….……...4

Fıgure 3.2.3 Example of External Merge Sort with B = 3 N=9……….…………...………………….5

Fıgure 4.1 Number of IO vs NumberOfPage with Buffersize =32 ….…………………………….….7

Fıgure 4.2 Number of IO vs Buffersize with NumberOfPage=500 ..……………………………..…..7

Fıgure 4.3 Graph the data at “list 4.3” ...................……….……….………………………………......8

Fıgure 4.1.1 Graph for fixed buffer size = 4 and changing number of pages ……………………….. 9

Fıgure 4.1.2 Graph for fixed buffer size = 32 and changing number of pages ..…………………….. 9

Fıgure 4.2.1 Graph for fixed Number of Page = 10 and changing buffer size .…………………….. 10

Fıgure 4.2.2 Graph for fixed Number of Page = 500 and changing buffer size ...………………….. 10

Fıgure 4.3.1 IO operations and Time Graph for Fixed buffer size=32 ……….....………………….. 11

Fıgure 4.3.2 IO operations and Time Graph for Fixed number of page=500 …...………………….. 11

**LIST OF TABLES**

List 4.1.Number of Passes of External Merge Sort………….………………………………6

List 4.2.Number of IO operation ……………….……..…..…………………………………….6

List 4.3.Time passed of External Merge Sort (time millisecond) ………………….……...8

**DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

**IDE :** Integrated Development Environment

**IO :** Input Output

**MATLAB :** Matrix Laboratory

**RAM :** Random Access Memory

**RPM :** Revolutions per Minute

**SQL :** Structured Query Language

# 1. Introduction

# In database systems, sorting of tuples have an important role. Sorting is to arrange items according to some properties of data and gets ordered data. Sorting problem has attracted area for research because of the need to speed of fast result when operation on millions of records.

# Sorting can use in database because it is required in some situations. ORDER BY clause for SQL query is a good example because the output of the query needs to be sorted. Also sorting can be used on the relation before applying join, union or intersection operations for getting better performance and better efficiency. Although sorting has a complexity, searching algorithm can use sorting such as before sort after search. Additionally, sorting can be used for eliminating duplicate values in the records.

# Sorting is made by main memory when records entirely fits in the main memory and the sorting called internal sorting. However when the data too large to fit in the RAM, the term which is external sorting, to describe this situation. External sorting makes it possible to efficiently sort large quantities of data with limited memory. External sorting uses secondary devices such as hard disk so it called memory sorting. Internal sorting is faster than memory sorting because memory sorting requires I/O operations.

# The main idea of external sorting is ‘divide and conquer’. The algorithm include to divide data that each fit in RAM and sorts each chunks then merges the sorted chunks together. The external sorting main concern is minimizing the number of times to move data between the external storage (Hard disk) and the main memory (RAM).

# 2. Development

# 2.1 Goals and Tradeoffs

# External sorting is type of memory sorting. It needs IO operation so algorithm’s major goal is minimizing the IO operation. Algorithm tries to minimize time cost. Users want to get result quickly and correctly. The algorithm’s second goal is stability. In the theory, we have a formula for to calculate run order according to buffer size and page size. Algorithm needs to evaluate run order with the same result with theory. The algorithm’s third goal is consistency and error preserving.

# Developing the algorithm part, some tradeoffs happen. Firstly, beginning of the algorithm, each page must be sorted. When sorting the pages, I have to choose an internal sorting algorithm. I chose quick sort but If the pages are mostly sorted then quick sort is the bad choice because it has O(n2) complexity for worst case but in best case and average case performance is O(n logn) is very good. Secondly, I use scanner for read write operation because of using scanner is easy to control. I can use binary IO stream for increase the performance and decrease the time for read write operations. Lastly, I added some additional check conditions for stability. These extra conditions increase the program’s executing time but it give stability.

# 2.2 Design Issues

# In coding part, I chose Java for the programming language and I chose the NetBeans for Java IDE. I created 4 classes for algorithm part, 1 class for main class and 1 class for testing class. Evaluating data and plotting data part, I used MATLAB.

# Testing the data with respect to time, I use computer that has this features;

# Acer Aspire 5740, Intel Core i3 (i3-330M) / 2.13 GHz, Dual-Core, 3MB L3 Cache, 3GB DDR3 RAM. Also the most important part is hard disk. My hard disk features;

# Model: HITACHI HTS545032B9A300

# Capacity : 320 GB

# Rotational Speed: 5400 RPM

# Rotation Time: 11.11 ms

# Average Rotational Latency: 5.56 ms

# 3. Theory of External Sorting

# The main idea of external sort is if the entire file does not fit into main memory, sort the data by breaking it the smaller subfiles that fit the main memory. Sort each subfiles and then merge them. Each step the main memory read subfiles from external memory such as hard drive, and sorts it then writes back to external memory.

# 3.1. Merge Sort

# Merge sort is a comparison based sorting algorithm. It has average case performance O(n logn). It has also best and worst case performance is same with O(n logn). Merge sort is invented by John von Neumann in 1945 and the main idea is divide and conquer.

# Assume that the number of pages is 2k for some integer value of k and merge sort is two way merge sort then,

# Pass 0 produces 2k sorted file with single page

# Pass 1 produces 2k-1 sorted file with two page

# Pass 2 produces 2k-2 sorted file with four page and it is go on.

# The number of pass based on the number of page with respect to 2pass-1 > number of pages and number of pases = where N is number of input pages. We have 2 IO operation for each page so the overall cost is .

# For example in figure 3.1.1, we have 7 input pages. Number of pases = 4 so we know that the sorting algorithm has Pass 0, Pass 1, Pass 2 and Pass 3.

# C:\Users\yasin\Desktop\merge.jpg

# Figure 3.1.1 Example of Two Way Merge Sort of a Seven Page File [1]

# 3.2. External Merge Sort

# The main idea of external merge sort is use the available buffer pages in memory suppose that available buffer pages number is B. Assume we need to sort a large file with N pages. The purpose of the external sorting is trying to minimize the number of passes because it also decrease the number of IO operations.

# Every data for external merge sorting has Pass 0. Initially, each page has to sort internally.

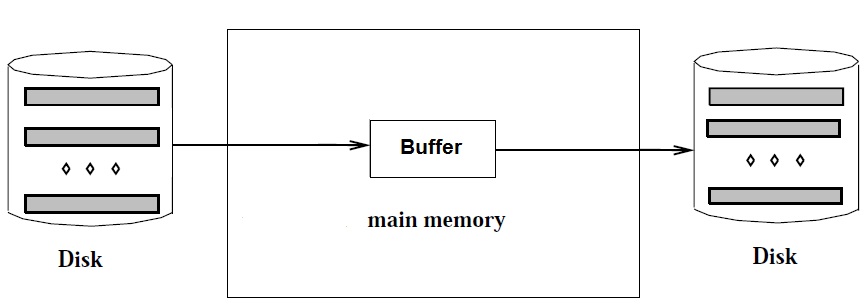


Figure 3.2.1 External Merge Sort Pass 0 [2]

In Pass 0, we need to use only 1 buffer. For every page, algorithm reads from disk, sorts it and writes back to disk. After Pass 0, pages ready to merge.

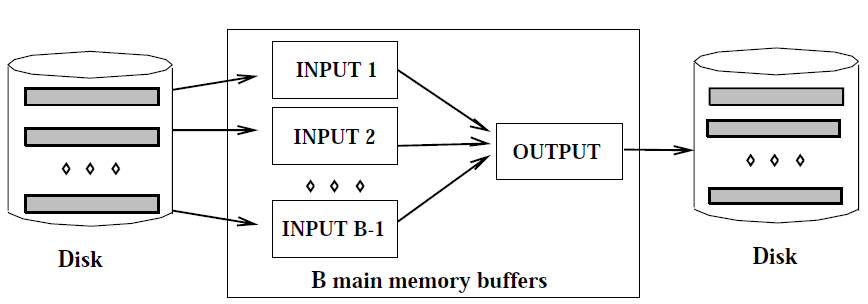
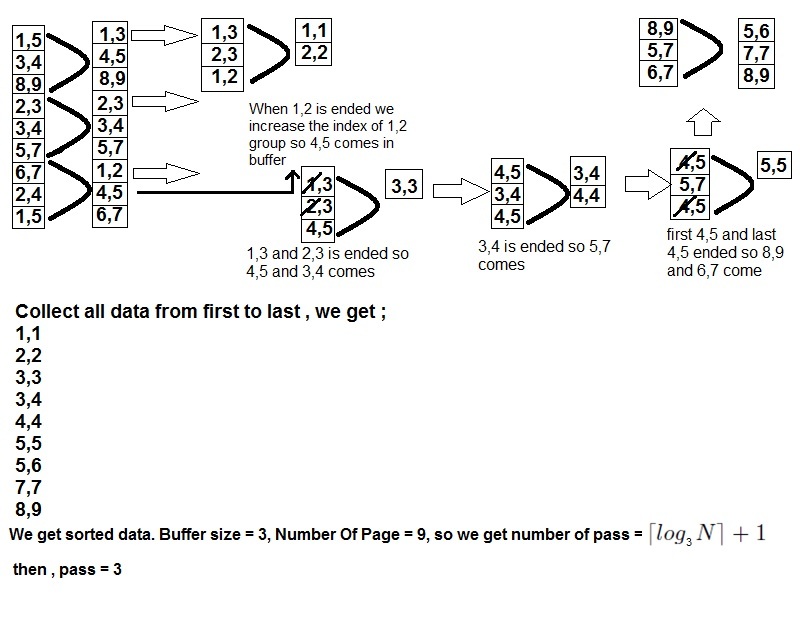
****

Figure 3.2.2 External Merge Sort Pass 1, 2… [3]

Other Passes (Pass1, Pass2… ), pages read into input buffers and find minimum element step by step and write it to output buffer. When output buffer is full, flush the buffer and write it to disk. We use indexes for input pages because each finding minimum we have to increase this page index for finding next element in page. Repeat this until reaching end of the pages.

Number of Passes = 

Total IO cost = 2N \* (Number of Passes)

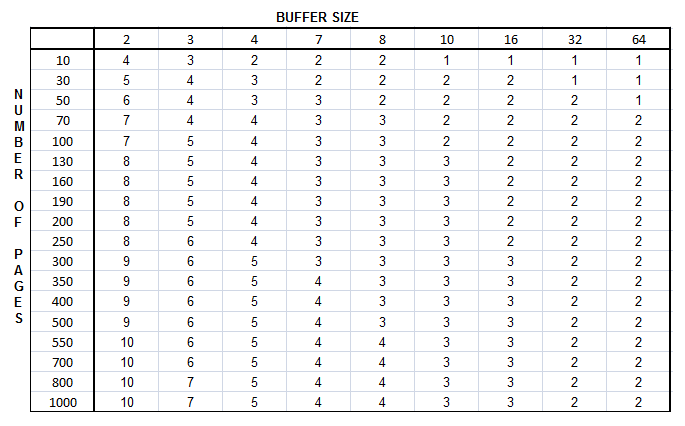
Figure 3.2.3 Example of External Merge Sort with B = 3 N=9

**4. Experiment of External Sorting Algorithm**

In experiment part, we try to find algorithm efficiency with making some trial. We will refer that;

* Input page number as Number of page
* Each sorted subfile as a Run
* Number of available input buffer as buffer size (p.s. we need one more buffer for output buffer. In experiment step I ignore the output buffer)
* Element number inside the pages as number of element

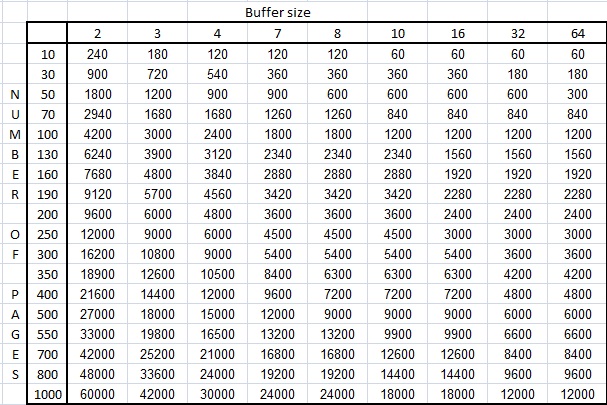
External sorting says us to check if “BufferSizeNumberOfPasses >NumberOfPages”, all file becomes sorted. As we know the main idea of external sorting is reduce the number of read write operations. Potential gain of algorithm is use all available input buffers to decrease number of pass. For this reason, the algorithm chooses possible minimum number of passes.



List 4.1.Number of Passes of External Merge Sort

We can simply analysis this list. For example choose input buffer size = 2 and number of pages = 30 then we can calculate 2n>30 then we calculate n = 5.

We know that Total IO cost = 2N \* (Number of Passes). In algorithm keep a counter for all IO operation and fill the table.

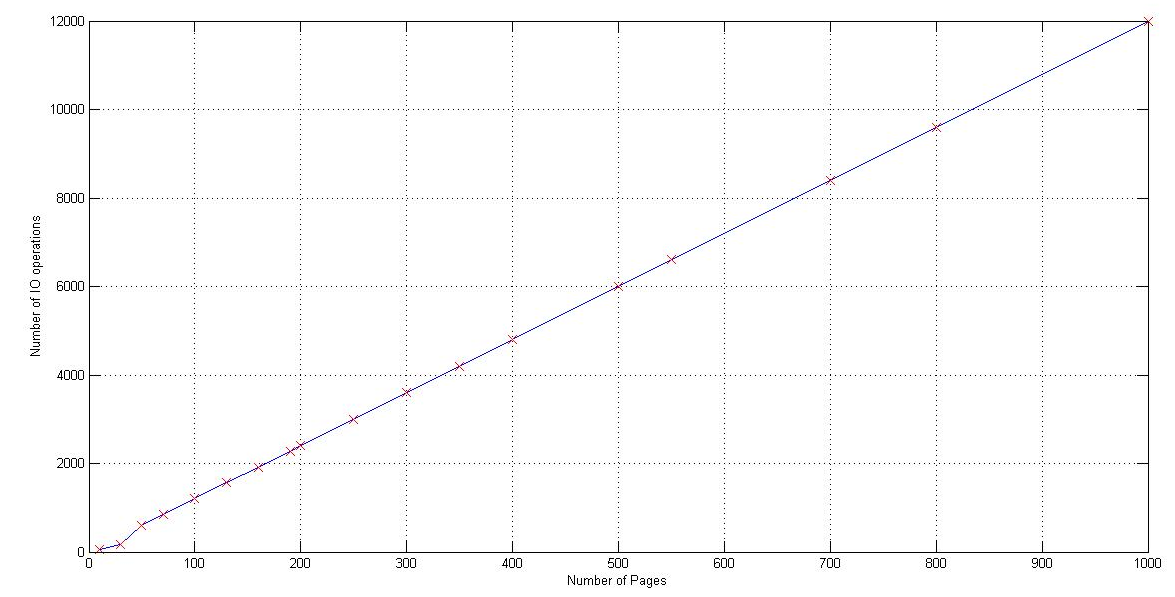


List 4.2.Number of IO operation

For list 4.2, we observe;

Total IO cost = 2\*NumberOfPages\*NumberOfPasses\*NumberOfElementsInsideThePage.

2 coming from 1 for reading operation and 1 for writing operation.IO operation was made by each element.

 Figure 4.1.Number of IO vs NumberOfPage with Buffersize =32

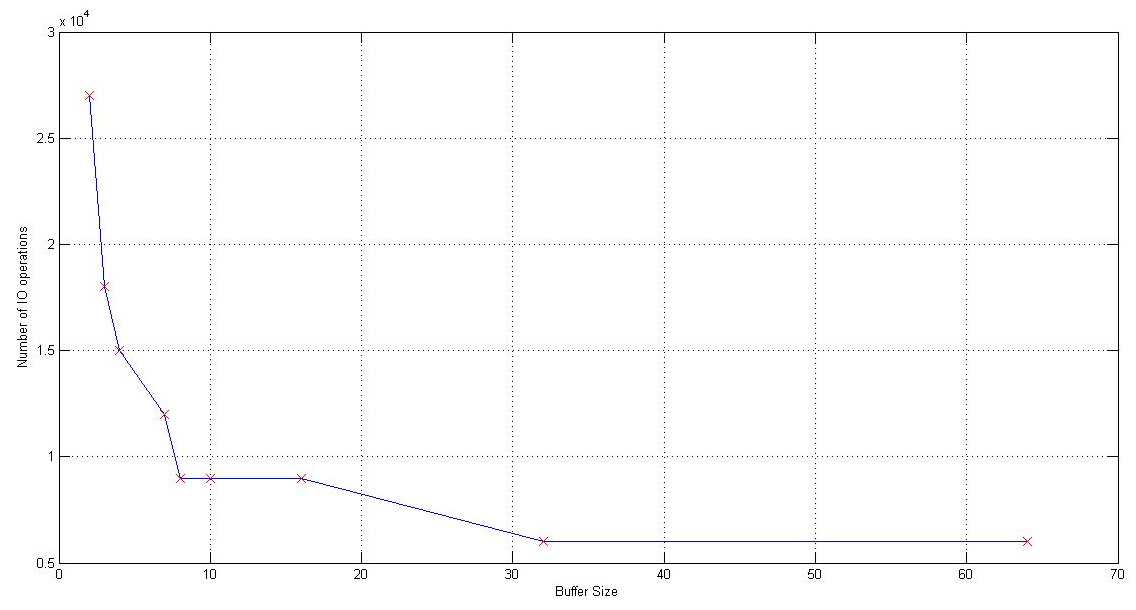
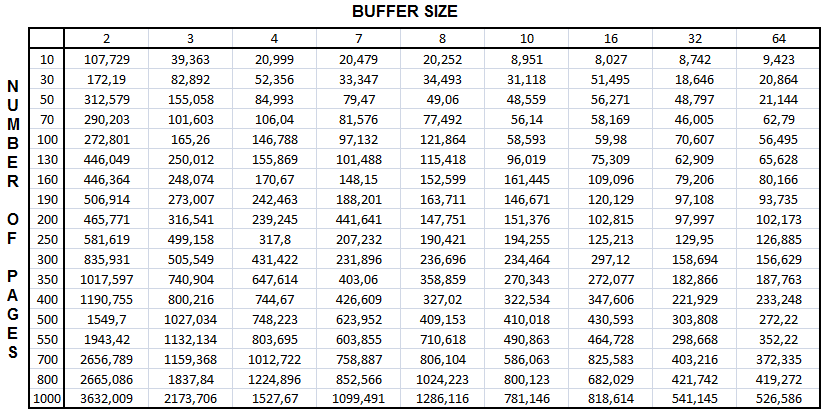


Figure 4.2.Number of IO vs Buffersize with NumberOfPage=500

These figures have some distortion because I choose number of pages randomly. If number of page was selected specific values then graphs have better shape.

During the experiment step, I keep the data for time elapsed according to buffer size and number of pages. For stabilization and decrease the computer’s measurement fault, I make this calculation 10 times and I get an average value. Because of the computer speed, the time difference is too small so I keep the data for nanosecond observation and keep the data millisecond for graphs.



List 4.3.Time passed of External Merge Sort (time millisecond)

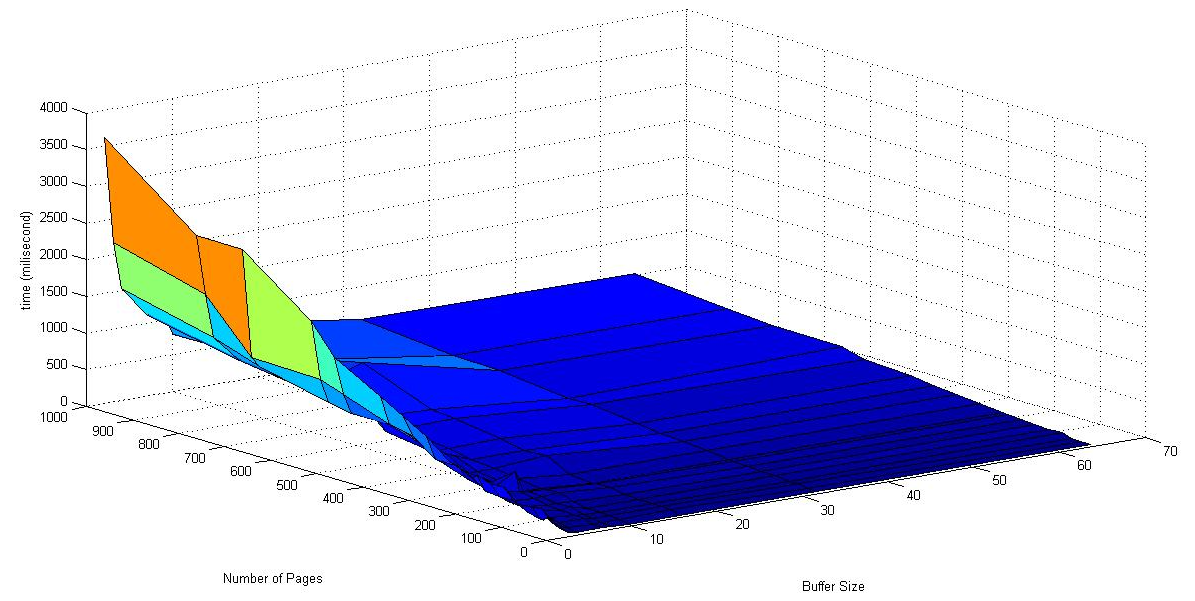


Figure 4.3.Graph the data at “list 4.3”

This figure show us when the number of pages increase and buffer size decrease, then time increase. Graph has max point for number of page = 1000 and buffer size = 2 for my data. (max page number, min buffer size).

**4.1. Time According to Number of Page**

In this step, input buffer size is fixed as 4 but number of page is different for checking time according to number of page.

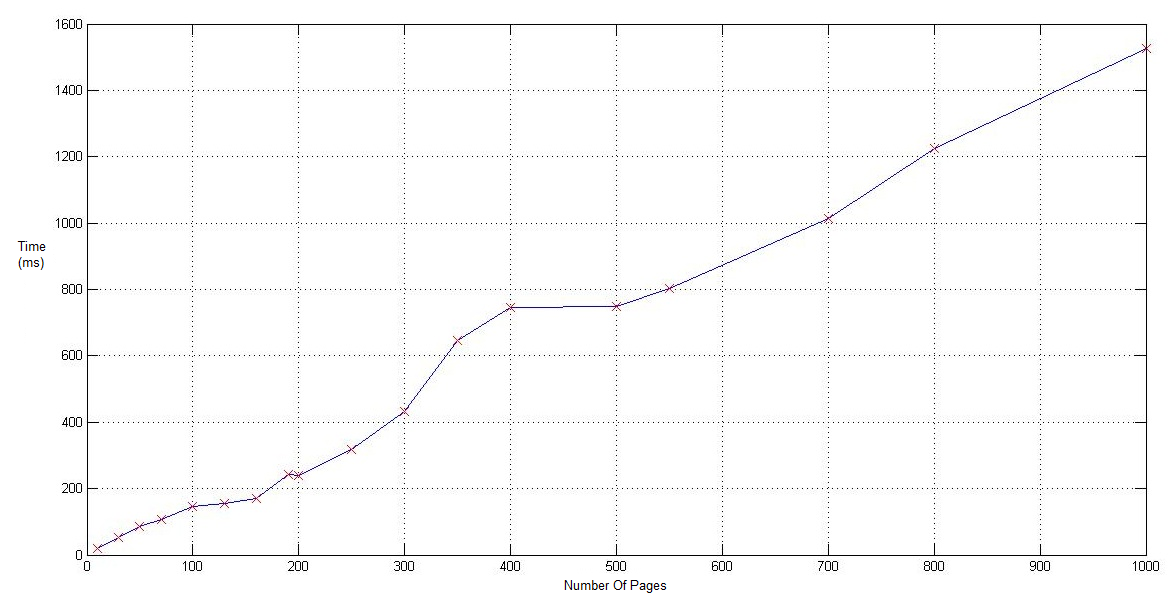


Figure 4.1.1.Graph for fixed buffer size =4 and changing number of pages

In this step I changed the buffer size as 32 and simply I observed the time is less than buffer size 4.

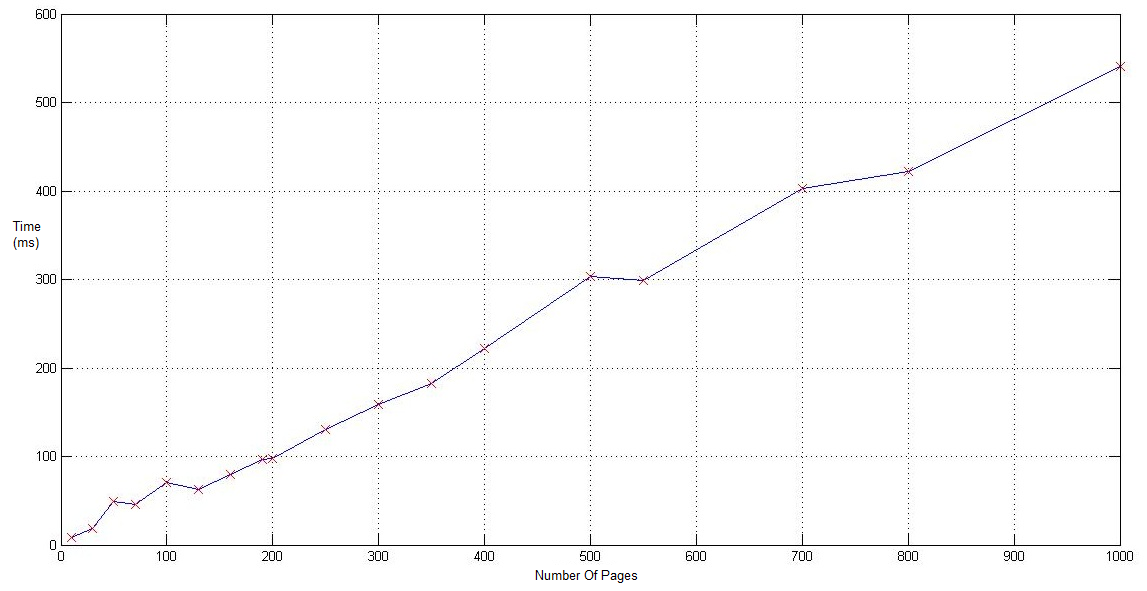


Figure 4.1.2.Graph for fixed buffer size =32 and changing number of pages

**4.2. Time According to Buffer Size**

In this step, number of page is fixed as 10 but buffer size is different for checking time according to buffer size.

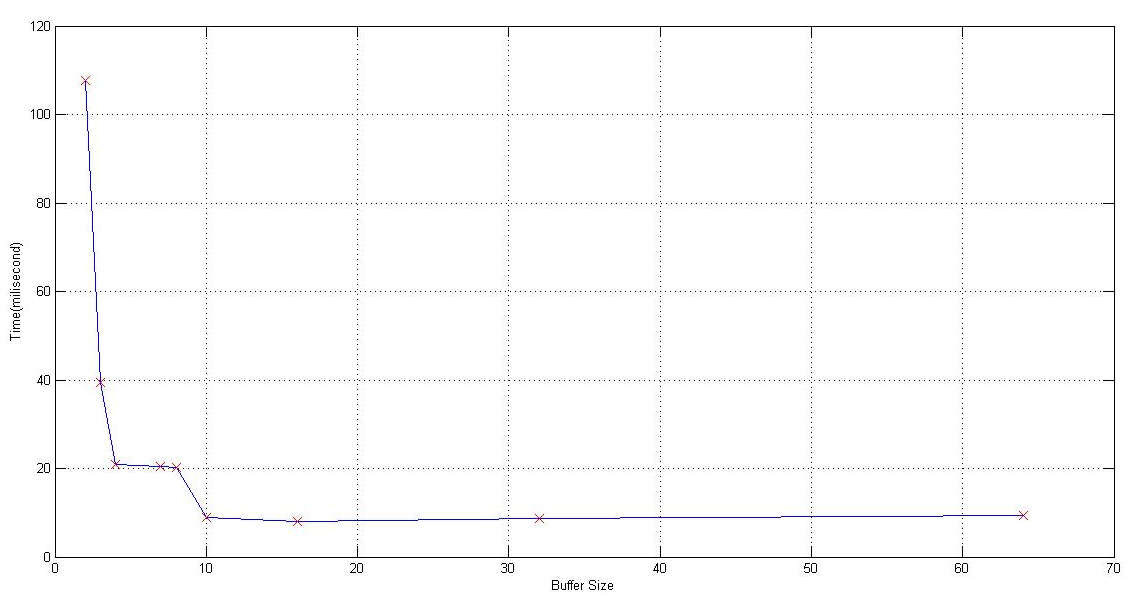


Figure 4.2.1.Graph for fixed Number of page= 10 and changing buffer size

In this step I changed number of page as 500 and simply I observed the time is more than number of page 10.

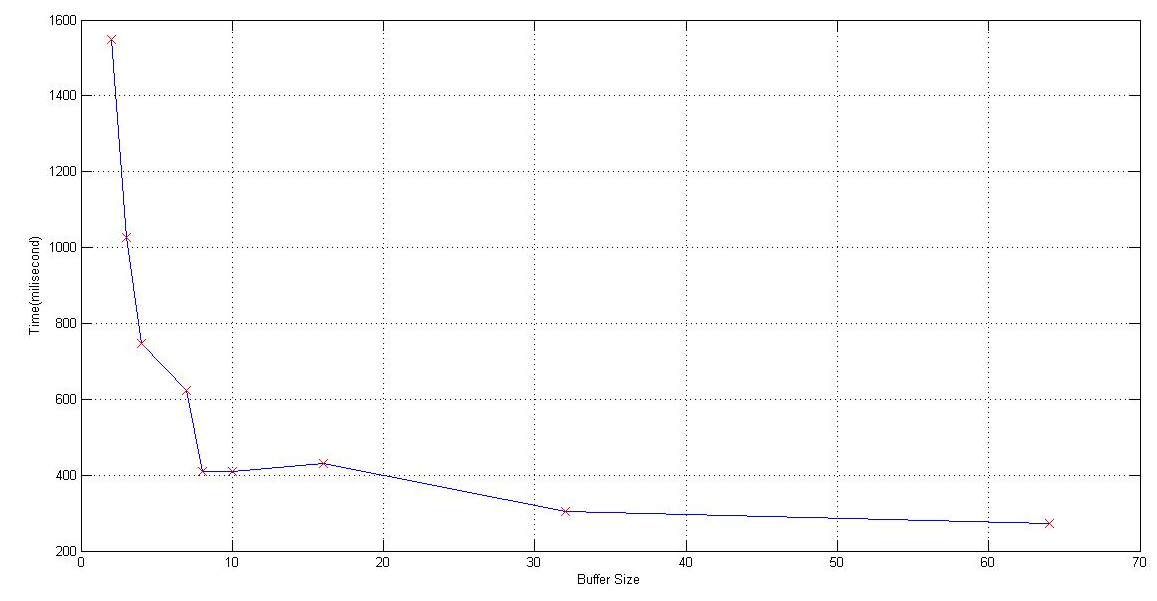


Figure 4.2.2.Graph for fixed Number of page= 500 and changing buffer size

## 4.3. Number of IO vs Number of Page and Buffer Size

In this part,firstly we try to observe how IO operations number relates with time according to fixed buffer size and changing number of pages.

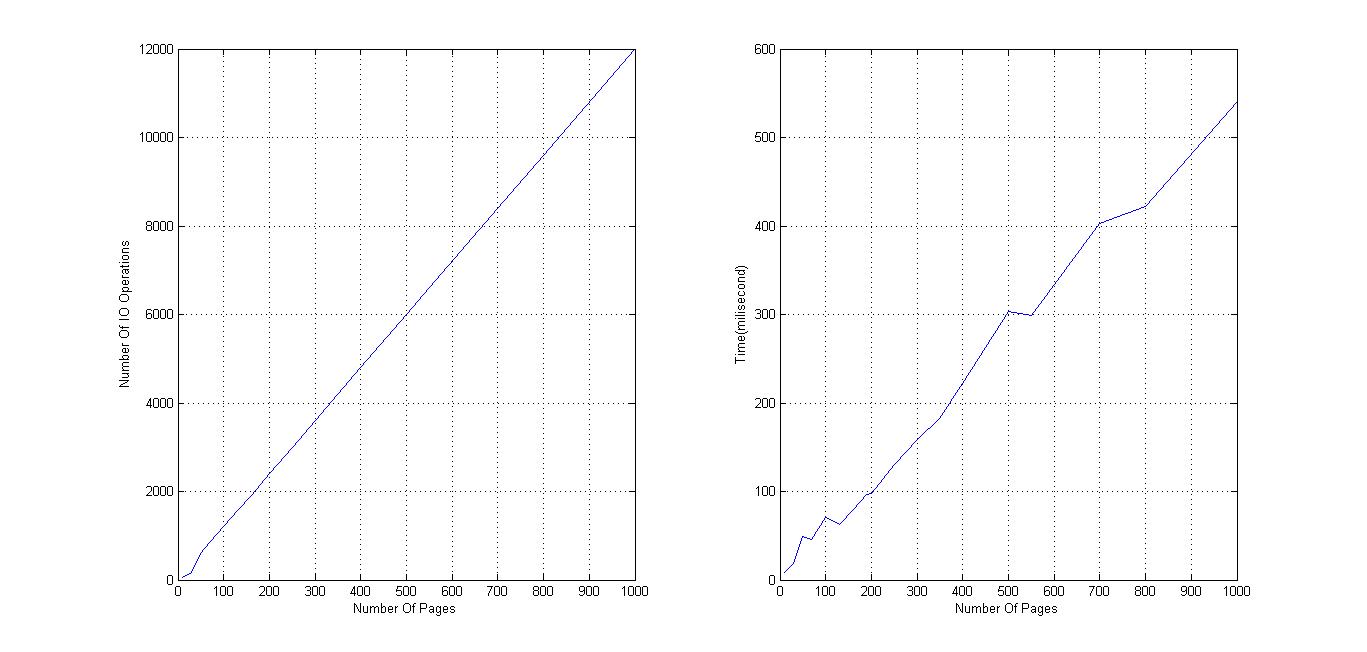


Figure 4.3.1.IO operations and Time Graph for Fixed buffer size=32

Secondly, we try to observe how IO operations number relates with time according to fixed number of pages and changing buffer size.

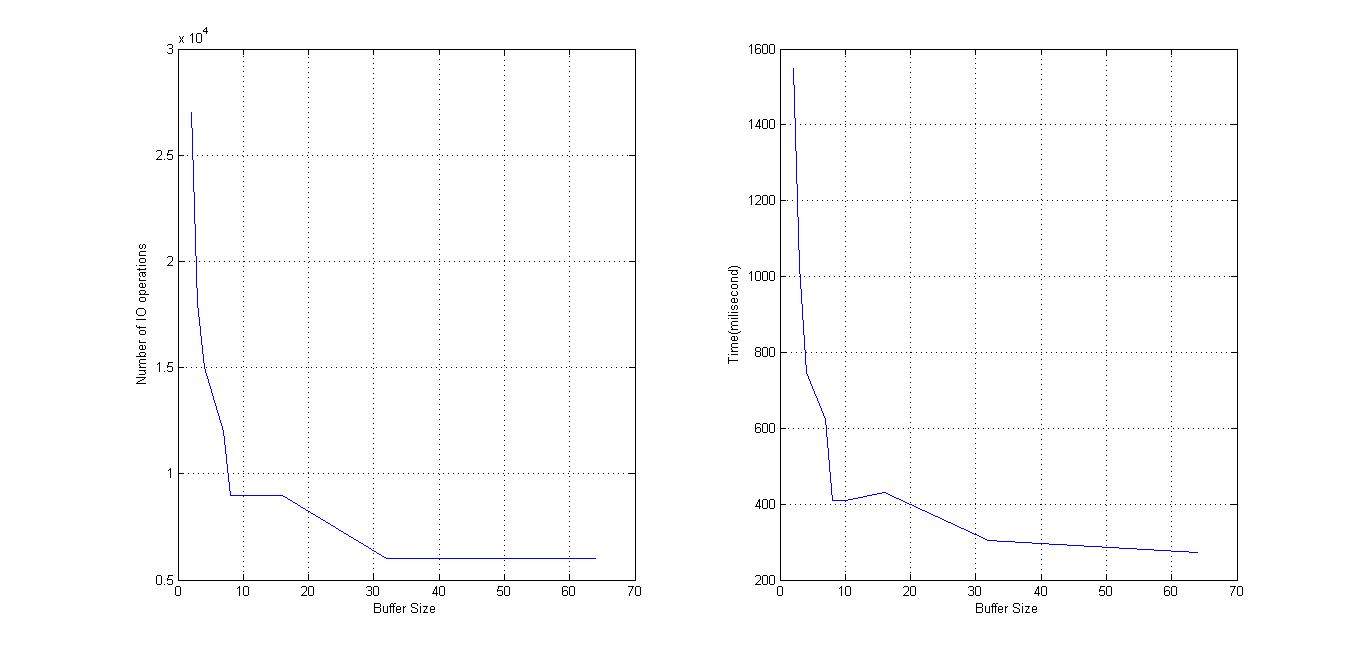


Figure 4.3.2.IO operations and Time Graph for Fixed number of page=500

Graphs have different scale. We observe that only the shape is likely similar. It means that when IO operations happen, time increase immediately and time is dependent IO operations.

**5. Discussion Experiment Results**

All experiment shows us, time has direct proportion with number of pages and time has inverse proportion with buffer size. It has not an optimal buffer size because it is relevant to RAM.

Number of passes is important issue for evaluate time. When number of passes change for decreasing order, the time decreases sharply. For example, Number of pages = 10, Buffer size=2 has number of passes = 4 and time = 107.7 ms, after this value let check buffer size =3. Same number of pages and buffer size =3 has number of passes = 3 and time = 39.4 ms. This example show us number of passes and time has direct proportion. We can also realize that their graph is very similar. Figure 4.1 and figure 4.2.1 which are time versus buffer size with fixed number of page, are similar and figure 4.2 and figure 4.1.1 which are time versus number of page with fixed buffer size, are similar.

Evaluating the data which are time according to number of page, we realize that number of pages is relevant to time with like linearly. It has also some sharply increasing part because of increasing number of passes.

The part of “Time According to Buffer Size”, we can analyze that buffer size and time has exponential decay relationship. It is because “BufferSizeNumberOfPasses >NumberOfPages” and increasing buffer size led to decreasing number of passes and as a result of this time decrease.

Additionally, we observed that if buffer size greater than number of pages, than time became similar. It is because the number of passes equal 1 and this algorithm run only once and sort the data. To result of this property, increasing buffer size isn’t decrease time every time.

Time graphs have some distortion because of computer. I tried to minimize these distortions with making experiment 10 times and getting average value.

**6. Conclusion**

Sorting is very crucial issue in computer science. Growing the data size led increase the complexity of getting specific information from data. In these days, some users store too much data in their databases. When they want to sort their data, internal sorting algorithms don’t work because the size of main memory (RAM) is small than data size. External sorting algorithm help us this situation. External sorting uses some part of RAM for buffer but actually it uses mostly hard disk to read and write operation.

External sorting algortihm’s major goal is minimizing the IO operations. Time is directly related the IO operations and in this project, I examined time according to buffer size which is related RAM, number of input pages. Simply I saw increasing buffer size and decreasing number of page decrease the number of IO operations therefore time decreases. I tested it and record the data and check it in theoretical values.

**REFERENCES**

[1] Ramakrishnan, R. (1997). Database management systems / Raghu Ramakrishnan. New York : WCB/McGraw-Hill, 1997 pp.304.

[2] Cornell University lecture notes

http://www.cs.cornell.edu/courses/cs432/2003fa/slides/ExternalSorting.pdf

[3] Ramakrishnan, R. (1997). Database management systems / Raghu Ramakrishnan. New York : WCB/McGraw-Hill, 1997 pp.306.