**Analysis of External Memory Algorithms and**

**Streaming Algorithms in Computational Geometry**

A

Bachelor Thesis Project

Report Submitted in Partial Fulfillment of the

Requirements for the Degree of

Bachelor of Technology

*by*

**Shrinivas Acharya**

**(10010164)**

Under the guidance of

**Dr. S.V. Rao**

****

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, ASSAM

**CERTIFICATE**

*This is to certify that the work contained in this thesis entitled "****Analysis of External Memory Algorithms and Streaming Algorithms in computational Geometry****" is a bona fide work of* ***Shrinivas Acharya (Roll No. 10010164)****, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree*.

Supervisor: **Dr. S. V. Rao**

Associate Professor,

Nov, 2013 Department of Computer Science & Engineering,

Guwahati. Indian Institute of Technology Guwahati, Assam.

( i )

**Acknowledgements**

I acknowledge with gratitude the constant encouragement and inspiration provided by my guide Dr. S. V. Rao and thank him for his able guidance, keen interest and valuable suggestions in the preparation of this thesis. Under his guidance I was not only able to appreciate a new domain of research but also realized the essential elements of research like defining a problem, constructing an approach, taking relevant assumptions, among others.

My thanks are also due to my friends and a number of people, whom I discussed with the various nuances of the problem at hand that helped me give a better structure to my ideas.

Last, but surely not the least, I am deeply grateful to my family, for the endearing encouragement and emotional support.

**Shrinivas Acharya**

( ii )

**Table of Content**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Abstract** | | | | | (iv) |
|  | | | | | |
| **Chapter 1** | **INTRODUCTION** | | | | 1 |
|  | | | | | |
| **Chapter 2** | **Review of Prior Work** | | | | 2 |
| 2.1 | External Merge Sort | | | 2 |
| 2.1.1 | | Algorithm | 4 |
| 2.1.2 | | I/O Complexity Analysis | 3 |
| 2.2 | Lower Bound for sorting operation on N numbers | | | 4 |
| 2.2.1 | | Internal Memory | 4 |
| 2.2.1 | | External Memory | 4 |
|  | | | | | |
| **Chapter 3** | **An Approximate Algorithm for finding minimum enclosing ball** | | | | 6 |
| 3.1 | | Algorithm | | 6 |
| 3.2 | | Analysis of Approximation Factor | | 8 |
| 3.3 | | Time Complexity Analysis | | 10 |
|  | | | | | |
| **Chapter 4** | **Implementation And Results** | | | | 11 |
| 3.1 | | Graph for d = 5 | | 12 |
| 3.2 | | Graph for d = 100 | | 13 |
| 3.3 | | Table of Results | | 16 |
|  | | | | | |
| **Chapter 5** | **FUTURE WORK** | | | | 17 |
|  | | | | |  |
| **Bibliography** | | | | | 18 |

(iii)

**Abstract**

When data set is too large and massive to fit inside the main memory, the input/output communication between the fast internal memory and slower external memory becomes a major performance bottleneck. In this paper we analyze some external memory algorithms. These are often referred as **EM** algorithms or **I/O** algorithms or **out-of-core** algorithms. EM algorithms are analyzed or measured in the performance of their time complexity (the CPU time), number of I/O operations and amount of space required.

In this paper we present the analysis of some fundamental external memory algorithms. We discuss some bounds in terms of I/O for external memory algorithms. In the end we discuss the approximate streaming algorithm for computing the minimum enclosing ball for a given set of points in any dimension(particularly higher dimensions), its implementation in C and it’s time complexity analysis (CPU time), one of the measures of performance for EM algorithms and streaming algorithms, with real time data.

( iv )

**Chapter1**

**Introduction**

The study of EM algorithms or I/O-efficient algorithms has been receiving increased attention as advances in processor speeds is increasing the difference between the throughput rates of main memory and external memories. Main memory access is very fast in comparison to external memory and an increase in the speed of the processor will not compensate the expensiveness of I/O operations. In the field of computational geometry where we deal with huge data set (mainly in form of graphs), I/O efficiency becomes more crucial [1].

1.

**Chapter2**

**Review of Prior Works**

In the following sections the computational model we use, consists of a single processor with a small local memory connected to a large external memory. The parameters of the model are as follows: [2]

**Notations**

1. Size of the dataset (in bytes) stored in external memory.
2. Block Size (in bytes), in secondary memory (Reading and writing can only take place in units of this block size).
3. Size of Data (in blocks).
4. Size of RAM (in bytes).
5. Size of RAM (in blocks).

**Assumptions**

The first external memory algorithm we discuss is External Memory Merge Sort. This algorithm is the most fundamental in the process of understanding of external memory algorithms.

**2.1 External Memory Merge Sort**

Let's assume the dataset is in a file. Before we look into the algorithm we define the term Run. **Run** is defined as a **sorted sub-file**.

**2.1.1 Algorithm**

1: **For** times **do**

2: Take blocks from external memory into RAM and sort them using any in place sorting algorithm.( Use quick sort, insertion sort or radix sort, this is not going to affect much as all of the data is in main memory, so we use a really quick sorting algorithm). Clearly there are blocks in a run now.

3: **While** thereare more than one run, iteratively **do**

4: Merge runs into a single run

2.

**2.1.2 I/O Complexity Analysis**

In pass 1 we have total of runs with total of no of I/O’s. Each run has no of blocks.As we iteratively repeat the process, the runs start to have blocks each in second pass and then blocks each in third pass and so on. In each pass we read every block once and write back once. So total of I/O’s are performed in each pass.

Run1

Bring *m* blocks, do in place sorting algorithm

Block *B*12

Block *B*11

*m* I/O’s

*m* I/O’s

Block *B*1m

*m* I/O’s

*m* I/O’s

Run2

Block *B*2m

Block *B*2m

**RAM**

RAM

Block *B*2m

*m* I/O’s

*m* I/O’s

Runn/m

Block *B*n/m, 2

Block *B*n/m, 1

Block *B*n/m, m

**Fig.1:** Pass 1 of External memory merge sort

3.

And after every pass, no of blocks present in one run, increases with a factor of till only one run is left. So total no of passes are performed.

Hence we have total I/O’s.

**2.2 Lower Bound for sorting operation on N numbers** [3]

**2.2.1 Internal Memory**

Assume that the only operation allowed is the comparison of two numbers. There are permutations of numbers. Take any two numbers . In permutations comes before and in other permutations comes before . So after one comparison we can eliminate half of the possibilities. Let us assume that it takes number of comparisons or steps to reach to the sorted order. Therefore,

.

Now using ***Stirling's approximation*** we have,

.

Thus is the lower bound for sorting *N* elements in internal memory.

**2.2.2 External Memory**

**Assumptions:**

1. Transfer of elements is the only allowed operation. No new or duplicate elements can be created.
2. The entire block should be transferred.

Following the same procedure of elimination of possible permutations, after input operations and arbitrary number of output operations, number of possibilities is . However, on looking closely the denominator of this term appears to be slightly different. (In the denominator) will be there only when a new block is fetched from the External memory. Since there are new blocks that can be fetched, the number of possibilities would be

4.

Using approximations [3]

Thus the lower bound for sorting in external memory is .

5.

**Chapter3**

**An Approximate Algorithm to Find**

**Minimum Enclosing Ball**

Implementation of a simple approximation streaming algorithm for computing the minimum enclosing ball of n points in high dimensions. This algorithm runs in liner time for a fixed dimension using minimum space in just one pass over the data points.

In this report, the focus is on the cases when d may be large.

**3.1 The Algorithm** [4]

1:Ball

2:*.c* *pₒ*

3:*.r*

4: **For** each point *p* in the input stream *P* **do**

5: **If** *p* is outside *B* **do**

6:*r* i-1 *r*

7:*c* i-1 *c*

8:i *p c*i-1 *r*i-1

9:*c**p c*i-1

10:*r**r* i-1  i

This streaming algorithm is very simple to understand. It returns minimum radius ball *B* enclosing points *P.*

Let’s discuss how this algorithm works.

Initially the radius is zero and center is first input point.

C:\Users\Shrinivas\Desktop\BTP Report\radiusIsZero.jpg

Now with every new point in the input stream we do the following:

After reading the next point from the input stream, we check whether this point lies inside

6.

the circle of minimum enclosing ball or not. If it is inside the circle then no update is required in minimum enclosing ball and hence we read the next point in input stream. If it is outside the circle we update the minimum enclosing ball as follows:

Let be the point causing ith update to the current minimum enclosing ball , having radius and center at .

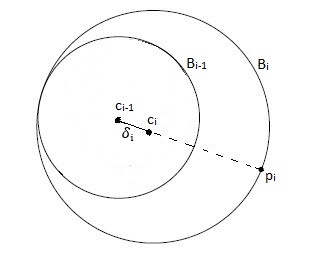
Now we move the center from to in the direction of the line joining to by the value of .

This we can write as equation below:

And new radius as:

Here

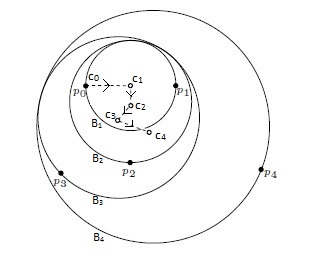
The following diagram shows how to move from to and update minimum enclosing ball accordingly



**Fig. 2:** inclusion of an external point into the minimum enclosing ball

At every new point from input stream we check for update in minimum enclosing ball. Every time after update, its center moves in the direction of the input point responsible for the update, with the increment of in radius. The movement of the center would look like:

7.



**Fig. 3:** center of minimum enclosing ball moving in the direction of next point

**3.2 Analysis of Approximation Factor**

The algorithm discussed above computes a 3/2 **approximation** to the minimum enclosing ball of points in time using extra space. The proof of this statement is as follows:

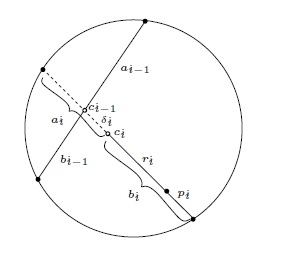
**Proof:**

For , let \* be the minimum enclosing ball with radius and be the minimum enclosing ball calculated by the algorithm above with radius and center described earlier. are described earlier.

The ’s lie inside \*, and so do the’s (as one can see by induction, due to the convexity of \*).For each > 0, consider the chord of \* which passes through and (which passes through as well). The point splits this chord into two segments, one containing and the other containing. Let be the length of the former segment and be the length of the latter segment. The key to the whole proof lies in finding the right invariant, which turns out to be the following:

Claim: .

8.



**Fig. 4** [4]

The proof is done by induction on .

**Base Case**

Since is midpoint of

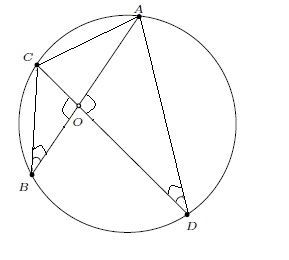
**Now, let**

We can say that:

We can prove above relation as follows.

For any circle having two intersecting chords, product of the segments of one chord equals the product of the segments of the other chord.

That is

Proof:

and

(All angles made by one chord on

circumference are same)

(Similar triangles)

**Fig. 5:**

9.

(as )

(as and is monotonically increasing for )

Assume \* and clearly and we have proven

(as and is monotonically decreasing for )

(as )

**3.3 Time Complexity Analysis**

If we look at algorithm we can see that at step 4, 5, 7, 8 and 9 it takes ) time. As the algorithm runs for the entire input stream, therefore total time of execution is ). We can say it takes linear time for a fixed dimension. And overall time complexity is .

10.

**Chapter4**

**Implementation and Results**

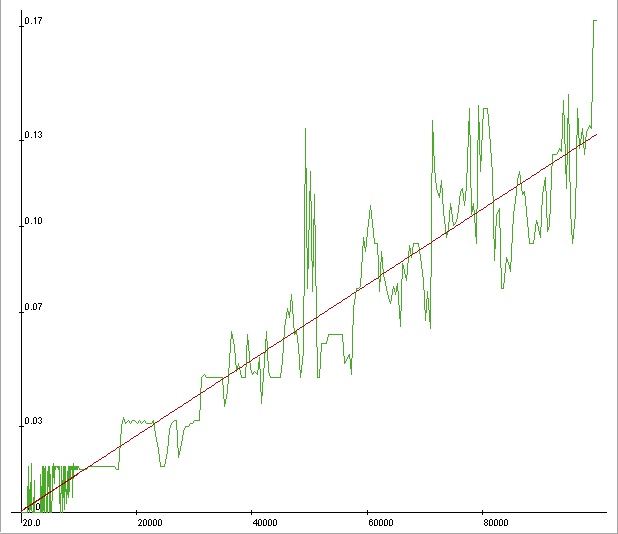
Algorithm is implemented and tested in C language.

The C code was run on the GHz Intel i-5 processor with total of 4GB internal memory.

For time complexity analysis, code was tested against dimensions with data points in each dimension, which is total of Million different data points.

The results, in the form of different graphs, of these 2M data points are as follows:

11.



**Fig6:** data pointsv/stime taken for

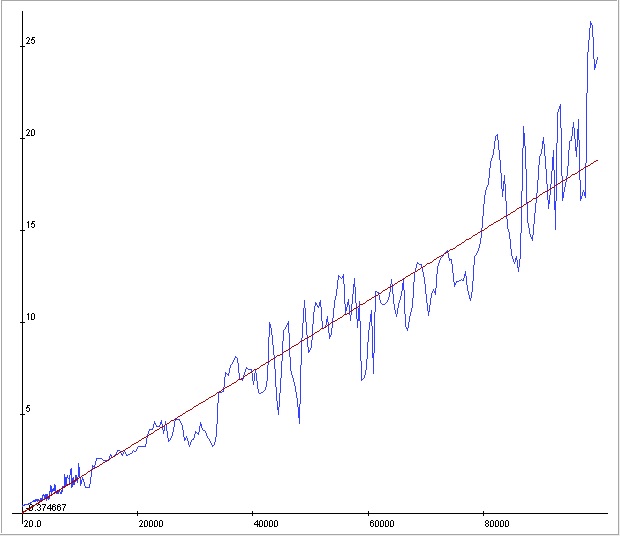
X-axis: No. of data points

Y-axis: Time in sec.

Time taken to find minimum enclosing ball for random points is seconds. Linear regression for the curve is also shown. Initially for small no of points code stops almost instantaneously taking seconds

Slope is seconds per data point.

12.



**Fig7:** data pointsv/stime taken for

X-axis: No. of data points

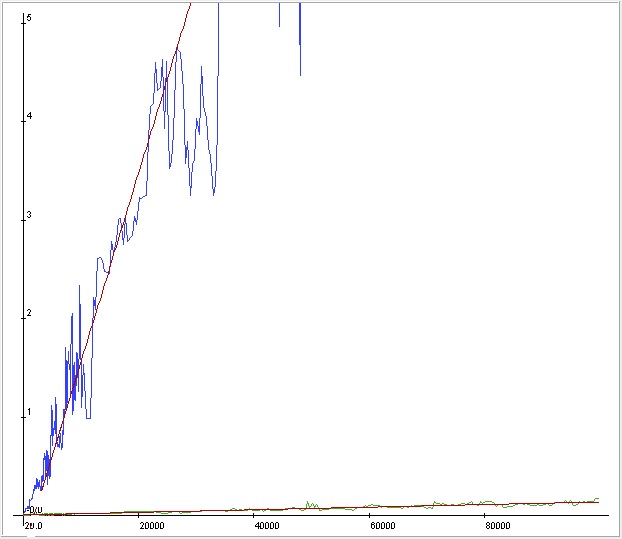
Y-axis: Time in sec.

The graph is for minimum enclosing ball for points lying in 100 dimensions from points to data points.

Slope is seconds per data point.

Clearly both graphs have linear dependency, as expected, between time-taken to find minimum enclosing ball v/s no of points for minimum enclosing ball. But both graph put together following graphs clear the idea that time complexity is not independent of dimension of points.

13.



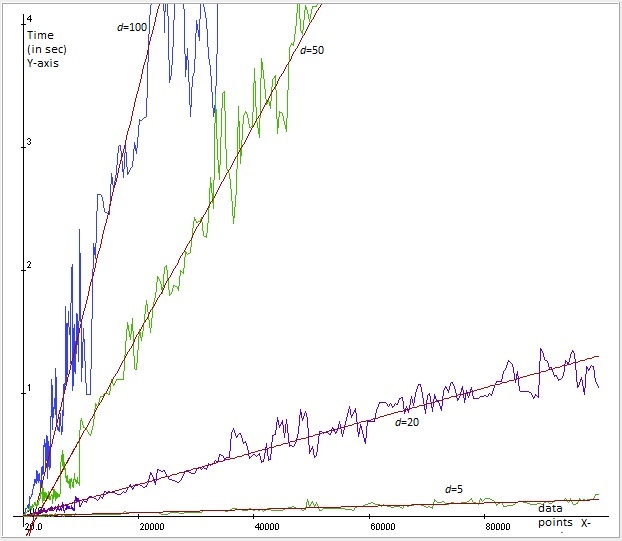
**Fig8:** data pointsv/stime taken for and

X-axis: No. of data points

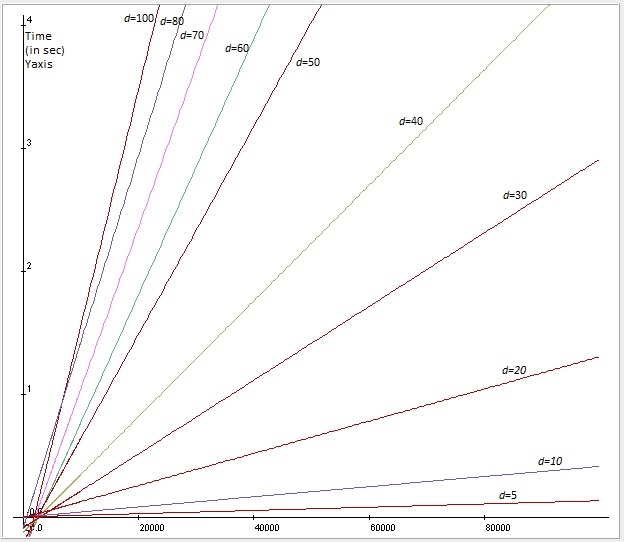
Y-axis: Time in sec.

The curve for both 5 dimension data set and 100 dimensions data set are coming liner but difference in slope is very huge. If we take their ratio it comes out to be:

14.



**Fig. 9:** curve for , , and

****

**Fig. 10**: curve for

|  |  |
| --- | --- |
|  | Slope( in sec per data point) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Table:** slope for Fig. 9

We can see the effect of dimension on the time taken (CPU time) for different input data points. So the real time output (CPU time, one of the measure of performance for EM algorithms) of the above algorithm is also lineally dependent on size of input data and also on the dimension of these points.

16.

**Chapter5**

**Future Work**

The important measure of performance for external memory algorithms is I/O. It is also the bottleneck for these algorithms.

The main focus of my work would be to

* Learn I/O minimization algorithms and streaming algorithms in the field of computational geometry.
* Implement these algorithms and do the I/O analysis.

17.

# **Bibliography**

|  |  |
| --- | --- |
| [1] | J. S. Vitter, "External Memory Algorithms and Data Structures," *DIMACS Series in Discrete Mathematics,* p. 38, 1998. |
| [2] | J.-J. T. D. E. V. J. S. V. Michael T. Goodrich, "External-Memory Computational Geometry". |
| [3] | Aggarwal and Vitter, "The Input/Output Complexity of Sorting and," *Communications of the ACM, ,* vol. 31(9), pp. 1116-1127, 1998. |
| [4] | T. M. C. Timothy M. Chan, "A Simple Streaming Algorithm for Minimum Enclosing Balls," *CCCG 2006, Kingston, Ontario,,* p. 4, 2006. |
| [5] | N. Zeh, "I/O Efficient Graph Algorithms," *Lecture notes from EEF Summer School on Massive Data Set,* p. 70, 2002. |
| [6] | V. P. Timothy M. Chan, "Streaming and dynamic algorithms for minimum enclosing balls in high dimensions," *Proc. 12th Algorithms and Data Structures Symposium (WADS), Lecture Notes in Computer Science ,* vol. 6844, pp. 195-206, 2011. |

18.