Deanship of Graduate Studies
Al-Quds University

Performance Evaluation of Algorithms with Different Structure Representations on Power Consumption for Smartphone Devices

IhabZ. M. Al-Kalouti

M.Sc. Thesis

Jerusalem-Palestine

2013/2014

Performance Evaluation of Algorithms with Different Structure Representations on Power Consumption for Smartphone Devices

Prepared By:
IhabZ. M. Al-Kalouti

B.Sc.: Computer Science-Al-Quds University-Palestine

Supervisor: Doctor Raid Al-Zaghal

A thesis Submitted in Partial fulfillment of requirements for the degree of Master of Computer Science / Department of Computer Science / Faculty of Graduate Studies – Al-Quds University.

2013/2014

Al-Quds University
Deanship of Graduate Studies
Computer Science Department

Thesis Approval

Performance Evaluation of Algorithms with Different Structure Representations on Power Consumption for Smartphone Devices

Prepared By: IhabZ. M. Al-Kalouti
Registration No: 20913401

Supervisor: Dr. Raid Al-Zaghal

Master thesis submitted and accepted. Date: / /2014
The names and signatures of the examining committee members are as follows:

1- Head of Committee: Dr.	Signature: ……..…..
2- Internal Examiner: Dr.	Signature: ……..……..
3- External Examiner: Dr.	Signature: ……..……..

Jerusalem – Palestine
2013/2014

Dedication

This work is dedicated…

To my parents, for giving birth to me at the first place and supporting me spiritually throughout my life.

To my beloved wife for the endless support and encouragement and to my son Mohammad

IhabZ. M. AlKalouti

Declaration

I certify that this thesis submitted for the degree of Master, is the result of my own research, except where otherwise acknowledged, and that this study (or any part of the same) has not been submitted for a higher degree to any other university or institution.

Signed……..

IhabZ. M. AlKalouti

Date: / /2014

Acknowledgements

I would like to express my deepest gratitude to Dr. Raid Zaghal for providing his technical expertise and support throughout the entire thesis process. I appreciate his patience, caring, his motivation and immense knowledge. Furthermore, I would like to thank the rest of the thesis committee and computer science department for all their support during my research. Finally I appreciate the excellent guidance especially from Dr. Badie Sartwai, Dr.Rashid Jayousi, Dr. Wael Hasouneh, Dr. Jihad Najjar and Dr. Nidal Kafri.

Abstract

The need for energy awareness on mobile devices is becoming more crucial in the design and development of mobile applications. Since power conservation is a major aspect for mobile users, the thesis will try to explore several methods and techniques in order to evaluate the performance of algorithms with different data structure representations on the power consumption of Smartphone devices. In this thesis, we will control and administer the mobile application in such a way to get quantitative results and perform relevant analysis. We will present experiments that will measure the amount of power and CPU overhead; we will explore how changing the data structures for various algorithms with different time complexities could impact power consumption. Also this thesis will try to investigate other related topics that have impact on the battery device such as local/global variables, reusing objects, arrays versus sets and other factors. The thesis will record and analyze the experiments in terms of CPU cycles and battery consumption and their relation to the above mentioned case studies. The results shows that using different data structure representations on various algorithms (having different time complexities) have impact on power consumption and CPU overhead of Smartphone devices, the figures and the analysis shows the following:

A- That algorithms having time complexity of n cube with its neutral case comes at the top most level of consuming battery life and the top level of CPU overhead.

B- N square comes at the second level of consuming battery life and using CPU with the following ranking (high to low) of algorithms data structure representations: Data types adjustment as double -> linked list based algorithms -> array based algorithms.

C- N LOG N comes at the third level of consuming battery life and using CPU with the following ranking (high to low) of algorithms data structure representations: Data types adjustment as double -> linked list based algorithms -> array based algorithms.

ملخص:

إن الحاجة إلى إتخاذ الطاقة كعامل أساسي في تصميم وتطوير تطبيقات الأجهزة المحمولة أصبح أكثر أهميةَ. حيث أن الحفاظ على الطاقة يشكل جانبا رئيساَ ومهماَ لمستخدمي الهواتف الذكية، من خلال هذه الرسالة سوف نقوم ببحث ودراسة عدة أساليب وتقنيات من أجل خفض إستهلاك الطاقة في الهواتف الذكية. وذلك من خلال تقييم أداء الخوارزميات ذات الإعدادات المختلفة.علما بأن النهج المتبع في هذه الرسالة هو الحصول على نتائج كمية من خلال التحكم بالتطبيقات ومن ثم إجراء التحليلات ذات الصلة.وكذلك من خلال هذه الرسالة سوف نقوم بالبحث عن عوامل أخرى من شأنها أن توثر على استهلاك الطاقة في الهواتف الذكية. ومن هذه العوامل كيفية تأثير نمط البرمجة(كتعريف المتغيرات) في إستهلاك الطاقة.

ومن خلال النتائج تبين أن استخدام الخوارزميات بإعدادات متباينة له تأثير في خفض إستهلاك الطاقة في الهواتف الذكية.

Contents

List of Tables

List of Figures

Chapter One - Introduction

1.1 Introduction

Mobile devices have become widespread due to the continuous advances in mobile technology and mobile applications. It is now becoming more compelling to extend enterprise products to mobile devices in order to reach more users. Smartphone's and mobile device applications are advancing rapidly. The applications being developed can be categorized in several groups such as educational, GPS navigation systems, news, medical files, social, travel, weather, gaming, entertainment, and e-commerce. The applications are stored in a repository called an application store, for each specific mobile operating system a store is available. For example, all Apple devices which utilize the iOS operating system has a store, the Android operating system has a store for their devices. So does Microsoft and BlackBerry mobile operating systems which have online App Stores of their own applications.

A question rises here; what makes these applications vary? These applications are different from desktop or server applications since they are different than the traditional desktop applications; mobile devices have limited capabilities, computing power, CPU power, RAM, battery life, and data storage, which make them different from traditional desktop applications. Mobile devices are critically constrained in their resources while power is the major limited design factor for mobile devices, batteries have slowly improved their capacity at a rate of only few percent per year according to (Yao and Durant 2009, 25). This means that mobile applications need more efficient power to overcome this gap.

1.2 Motivation

Smartphones have surpassed desktop machines in sales in 2011 to become the most prevalent computing platforms according to (Pathak et al. 2012). Despite the significant market of Smartphones devices and exponential increase of their apps market, Smartphones devices will remain severely limited by their battery life time.

Software companies and software developers need to be “power aware” when dealing with mobile devices since the target here is battery-powered systems, not complying to this requirement will lead to unnecessarily power-hungry applications that rely on the operating system for power management, in addition users struggle to determine which applications are energy-efficient. Typically, users blame the operating system or the hardware platform instead of unfortunate and unintentional software design decisions (Zhang et al. 2010). However, when developers first start their applications on desktop or server sides they ignore the power issue altogether, one could say that power was free and of course some of the developers have taken that factor into account. There have been several developers that continue to believe the battery is a hardware component and it can only be examined at the hardware level (Yao and Durant 2009, 25).

This thesis will focus on addressing the issue at the software level. According to (Bunse et al. 2009) there has been several studies in the past that discussed the hardware portion and few studies have dealt with the subject of the software impact on power consumption in mobile devices.

This thesis will focus on algorithms complexities with different data structure settings and their relation to mobile power consumption and CPU overhead. In this thesis we have explored and implemented research to evaluate the performance of various algorithms with various data structures against battery consumption and the CPU of the mobile device. It

will introduce a number of experiments to measure battery consumption for mobile devices that relate to various fields of programming as we will present in the following chapters.

1.3 Research Problem

The fact that Smartphone devices have resource limitations, and the fact that Smartphone devices have limited battery life time, is a crucial aspect and challenging task for developers, so developers need to reduce resource intensive applications in order to overcome this gap.

In this section we present a number of questions that will shed some light on the research problem area. These questions will be answered after we present our results:

- Can software developers help in addressing the power consumption issue without dealing with the hardware level?
- How can the software developer optimize the application in terms of power consumption while maintaining the application's performance?
- Do various implementations of algorithms with different representations of data structure have impact on battery levels?
- What are the factors that could drain battery life during coding phase?

What Makes Smartphone Device Different?

(Wasserman2010) explored several factors that highlight the differences between traditional desktop applications and mobile applications, the author mentioned that mobile applications present some additional requirements that are less commonly found with traditional applications including senor handling, user interface, potential interaction with

other applications, families of hardware and software platforms, native and hybrid applications that includes applications that invoke services over the telephone network or the internet via a web browser and affect data and displays on the device, also other additional factor that we focus on our thesis is power consumption, here the author mentions that many aspects of an application affects its use of the device's power and thus the battery life time of the device. Dedicated devices can be optimized for maximum battery life time, but mobile applications may inadvertently make extensive use of battery-draining resources. Finally the author mentions that the mobile environment, with its dependence on different kinds of networks differs from traditional environments and thus raises some new researches questions and problem areas that needs further study and power consumptions is one of those concerns.

1.4 Contribution

The thesis takes the first steps towards understanding and evaluating various settings at the software level of the Smartphone application, in order to reduce energy cost.

In more details, the thesis will focus on the following four concreate contributions:

1. The thesis will present different algorithms, with different time complexities, these algorithms are the initial step towards starting the performance evaluation.
2. These algorithms are evaluated with different settings for each, such as different data structures representations (linked list, arrays, data types, normal mode), as well each algorithm is represented at different time complexity.
3. The thesis will present experiments in order to measure the energy cost, and will measure the CPU overhead.

4. The thesis as well present various experiments that shows how other factors, such as accelerometer, data objects, programming style, etc. could impact battery life time.

The aim of the thesis, is to present to software developer's guidelines that they could adopt at the design phase for the applications, and to enable them make smart programming choices that would reduce battery consumption and increase its life time.

1.5 Thesis Outline

The thesis includes five chapters; chapter two includes the background and related work and gives general definitions on the terms that will be used during the thesis, and afterwards it defines related works and the taxonomy which presents a comprehensive comparison between various components of Smartphone devices in terms of power consumption. Chapter three describes the thesis methodology and approach that has been used, afterwards the chapter present the scenarios that will be adopted in the thesis, as well will show the plan of work of these scenarios. Chapter four presents the experiments and the results, in this chapter the thesis will present the experiments design and all the related plans and work implemented within the experiments, afterwards the chapter will show the results and the analysis on them using graphs and SPSS. Finally, the chapter describes other factors and experiments that affect the power consumption on Smartphone devices and presents the results. Chapter five shows the conclusion and future work.

Chapter Two- Background and Related Work

2.1 Theoretical Study of Algorithms

This chapter will present an overview of algorithms complexity and definitions of sorting algorithms (these will be used in our experiments), we have looked into several definitions in various books, and found simple and comprehensive definition according to (Nakov and Kolev 2013, 770-772), and the authors mentioned that one cannot talk about efficiency of algorithms and data structures without explaining the term "algorithm complexity".

Algorithm complexity is a measure, which evaluates the order of the count of operations, performed by a given or algorithm as a function of the size of the input data. To put this simpler, complexity is a rough approximation of the number of steps necessary to execute an algorithm. When we evaluate complexity we speak of order of operation count, not of their exact count. For example if we have an order of N^2 operations to process N elements, then $N^2/2$ and $3*N^2$ are of one and the same quadratic order.

Algorithm complexity is commonly represented with the big-O notation, also known as asymptotic notation, where f is the function of the size of the input data. The asymptotic computational complexity O(f) measures the order of the consumed resources (CPU time, memory, etc.) by certain algorithm expressed as function of the input data size.

Complexity can be constant, logarithmic, linear, n*log(n), quadratic, cubic, exponential, etc. This is respectively the order of constant, logarithmic, linear and so on, number of steps, are executed to solve a given problem. For simplicity, sometime instead of "algorithms complexity" or just "complexity" we use the term "running time".

Also the authors in Table (2.1) have shown a comparison between every type of complexity (running time).

Table 2.1: Various algorithms complexities

Complexity	Running Time	Description
Constant	O(1)	It takes a **constant number of steps** for performing a given operation (for example 1, 5, 10 or other number) and this count does not depend on the size of the input data.
Logarithmic	O(log(N))	It takes the order of **log(N) steps**, where the base of the logarithm is most often 2, for performing a given operation on N elements. For example, if N = 1,000,000, an algorithm with a complexity O(log(N)) would do about 20 steps (with a constant precision). Since the base of the logarithm is not of a vital importance for the order of the operation count, it is usually omitted.
Linear	O(N)	It takes nearly the **same amount of steps as the number of elements** for performing an operation on N elements. For example, if we have 1,000 elements, it takes about 1,000 steps. Linear complexity means that the number of elements and the number of steps are linearly dependent, for example the number of steps for N elements can be N/2 or 3*N.
Logarithmic	O(n*log(n))	It takes **N*log(N) steps** for performing a given operation on N elements. For example, if you have 1,000 elements, it will take about 10,000 steps.
quadratic	$O(n^2)$	It takes the order of **N^2 number** of steps, where the N is the size of the input data, for performing a given operation. For example if N = 100, it takes about 10,000 steps. Actually we have a quadratic complexity when the number of steps is in quadratic relation with the size of the input data. For example for N elements the steps can be of the order of $3*N^2/2$.
Cubic	$O(n^3)$	It takes the order of **N^3 steps**, where N is the size of the input data, for performing an operation on N elements. For example, if we have 100 elements, it takes about 1,000,000 steps.
exponential	$O(2^n)$, O(N!), $O(n^k)$, …	It takes a number of steps, which is with an **exponential** dependability with the size of the input data, to perform an operation on N elements. For example, if N = 10, the exponential function 2^N has a value of 1024, if N = 20, it has a value of 1 048 576, and if N = 100, it has a value of a number with about 30 digits. The exponential function N! grows even faster: for N = 5 it has a value of 120, for N = 10 it has a value of 3,628,800 and for N = 20 – 2,432,90,008,176,640,000.

A- Bubble Sort Algorithm:

Bubble sort is one of the simplest sorting algorithms, in this algorithm two sequential values are compared, and if one of the values is greater than the other, swapping is done and would be sorted in ascending order, what is known about bubble sort is that it takes a lot of time (processing time).

B- Insertion Sort Algorithm:

Insertion sort operates in a different way than the bubble sort, insertion sort compares two elements, and then swap them if needed, and then the algorithm continues to operate by selecting the following item, and deal with as needed, if the element is less than the other two elements, the algorithm insert the item in the correct place and so on.

C- Selection Sort Algorithm:

Selection sort operates in the following method, the first value is compared with the other elements in the list, i.e. (n-1), and the less value element is swapped to the first of the list, and so on.

D- Quick Sort Algorithm:

Quick sort algorithm is a partition algorithm (divide and conquer), the algorithm starts by setting a pivot point, and the small elements values are placed aside, while the large elements values are places on the other side, then again a new pivot point is set, and the same process is repeated again, till the elements at both sides are sorted, finally the two sets are combined together to have a completed sorted list.

E- Merge Sort Algorithm

The merge sort works as follows: the algorithm starts it work by dividing the main list into two sub lists, each list will have (approximately) the same number of elements, afterwards the two sub lists is again divided into four sub lists, and then to eight and so on. The next step of the algorithm is that the sub lists are combined back, two at a time to create a new sorted list, again sub lists are combined again to finally produce the final sorted list.

(Kamthane2013, 427-433) and (Sherrod 2007) *provides further clarifications, definitions and examples of each algorithm.*

2.2 Related Work

Software design has impact and effect on the power consumption of Smartphone devices, this statement has been proven by couple of papers which our thesis in one way or another linked to.

Summary of researchers that focus on the software part

In a paper by (Pathak et al. 2012), the researchers study the issue of power consumption in Smartphone devices by characterizing and detecting (no sleep) energy bugs in Smartphone apps. This study makes the first advances towards understanding and automatically detecting software energy bugs in Smartphones. The idea behind their research is the following: All components in the Smartphone device stay off or in an idle state, unless the application in the Smartphone device instructs the device to keep it on. Mainly, all Smartphone devices such as Android-based and iPhones stay at sleeping mode which almost consume zero power at this state, but this policy severely affects Smartphone applications, according to the authors the application may implement a crucial service or task with other interfaces and this could result in no activity when performing a special task. As stated by the authors, such procedure encumbers the developers to juggle power control APIs, that is to keep the elements of the Smartphone devices active while running the application and off otherwise, the issue is that during this policy a non-sleep bugs could arise, and that will cause extensive use of the battery life. This policy has given the app developers a deep paradigm shift in Smartphone programming, the authors in their paper call it power encumbered programming, and this new shift put additional difficulties for the programmers to deal with power consumption in Smartphone devices, dealing with this causes what is called no-sleep bugs which are defined as energy bugs resulting from misunderstanding at the coding phase for an application, and causing unexpected high

energy consumption. As reported by authors, the difficulties and the complexities for power encumbered programming is due to the event-based nature of Smartphones applications. And it is different compared to desktop/server applications, and thus the developers need to be alerted on these issues.

In another research by (Abdul Elminaam et al. 2009), they explore the issue of power consumption and did a performance evaluation of symmetric encryption algorithms for wireless devices. They found that power consumption is a major concern on wireless devices and thus there is a need to maximize battery life while choosing the appropriate algorithms setup. The authors provided an evaluation of six of the most common encryption algorithms on power consumption for wireless devices, the following are the list of the algorithms: AES, DES, 3DES, RC2, Blowfish and RC6. A comparison has been made of these six algorithms at different settings of each algorithm. Settings defined such as different sizes of data blocks and different data types. The goal of their experiments was to collect the following metrics: Encryption time, throughput, power consumption (micro joule/byte) and power consumption (percent in battery consumed). Results of the evaluation show that algorithms at different setup have impact of power consumption for wireless devices. The authors used two methods to measure the power consumption: First by readings from the actual device and second by using the cycles, the operating voltage of the CPU and the average current drawn for each cycle.

Table (2.2) will present a number of researches and their main objective and contributions, and will present the link to the thesis.

Table 2.2: Researches and related work

Reference	Objective and Contribution	Link to our thesis
[13], Tien and Lee 1996	Reduce power consumption by the appropriate choice of the algorithms and other higher level decisions about the design of the software.	Here we followed up the expert recommendations, and we showed in our experiments that choosing the appropriate algorithms could reduce energy cost
[2], Abdul Elminaam et al. 2009	Reduce power consumption by evaluation the performance of security algorithms.	Here we followed up the same power consumption measurements and we made different algorithms setups.
[6], Larsson 2010	Improve energy efficiency by using high performance algorithms and data structures.	Here we followed up the expert suggestions, and we showed in our experiments that data structure has impact on battery life.
[10], Pathak et al. 2012	Reduce power consumption using the software applications (energy bugs).	Here we linked our work to their in applications, in that designing the appropriate application in terms of power saves battery life.

Narrative and summary for the conclusion of each research

In their paper (Tien and Lee 1996), mentioned that the choice of the algorithm and other higher level decisions about the design of the software component can affect system power consumption in a big way, the authors informed that the design of the system software, the actual application source code, and the process of translation into machine instructions – all of these determine the power cost of software component. And thus in light of the above, there is a clear need for considering the power consumption in systems from the point of view of software.

As for [2], (Abdul Elminaam et al. 2009), they present and evaluate six algorithms at different settings, and they evaluate the algorithms against power consumption, their results shows changing in power consumption levels while using different the algorithms setups.

However in [6], (Larsson 2010), advised to improve energy efficiency by using high performance algorithms and data structures that complete tasks faster, allowing the processor to idle. As well advised that if the requirements allow, an alternate approach is to investigate the suitability of less complex (and more energy efficient) algorithms, the author stated also that for instance, heavily recursive algorithms can be energy inefficient, as they often add overhead by using or exercising more stack than non-recursive algorithms.

The authors in the papers shed the light on the importance of the software portion to minimize the energy cost. (Pathak et al. 2012) [10], advised and provided essential guidelines for the developers during the coding phase of the applications to avoid high energy costs.

Link with the thesis

In our thesis we present performance evaluation of algorithms with different complexities and various representations on power consumption for Smartphone devices, we use different settings of algorithms with different time complexities, we first present the traditional algorithms, and then we used different settings such as data types, linked list, arrays. We measure the CPU overhead and the power consumption. At the end we

introduce guidelines for the developers in order to minimize battery cost during the coding phase.

In our thesis we followed the recommendations of (Tien and Lee 1996) and (Larsson 2010), their perspective was to reduce power consumption by choosing the appropriate algorithms and to improve energy efficiency by using high performance algorithms and data structures within the design of the software. And we present experiments and results that shows different algorithms with same time complexities and different setups which affect power consumption

In our thesis as well, we extended the research of (Abdul Elminaam et al. 2009), in such a way that we added different algorithms with different time complexities, and as well different settings of each algorithm, Adding on top of that we followed the same measurement methods that are used by them.

Again we focus on the applications portion (i.e. coding phase), as we see that this portion together with other elements (explained in the taxonomy section), would reduce the cost of energy, this perspective is extremely obvious in the research of (Pathak et al. 2012), in our thesis we followed their direction towards concentrating on the software portion.

2.3 Taxonomy

As reported by several studies (Pathak et al. 2012) & (Pathak et al. 2011) the components in the Smartphones devices are divided into two main categories traditional components such as CPU, Wi-Fi, 3G radio, memory, screen and storage, and the other category are exotic components such as GPS, camera, various sensors including accelerometer, magnetometer, proximity and gyroscope and all of which has impact of the battery life. So as power consumption is a crucial factor in Smartphone devices, the thesis will try to present a taxonomy of power for Smartphone devices.

In this thesis, we have looked into several research studies in order to get a clear idea about the factors that cost power most, and we summarized a taxonomy from different papers and according to different statistics; [9] (Pathak et al. 2011), [7] (Maloney et al. 2012)and [1] (Abdelmotalib et al. 2012). We have categorized the Smartphone components into six categories (See Table (2.3) below), afterwards we present a hierarchy of the power consumption for each component, from the least consuming battery life to the most consuming based on the above mentioned researches (see Fig. (2.1) below). Finally we present a small summary and narrative for each component, all of which are summarized in Table (2.3).

Table 2.3: Categories of Smartphone components in terms of power consumption

Software Factors	Hardware Factors	External Factors (Networking Technologies)	Smartphone Sensors	Smartphone Components	Unknown Energy Bugs
Software updates	Faulty battery	Wi-Fi NIC	Accelerometer	GSM	specific apps
OS configuration	Exterior damage	3G	Bluetooth	CPU	not able to identify
Applications		GSM	Microphone	RAM	
Loop bugs		Wireless handovers	GPS	GPU	
			Video camera	LCD	
				Backlight	

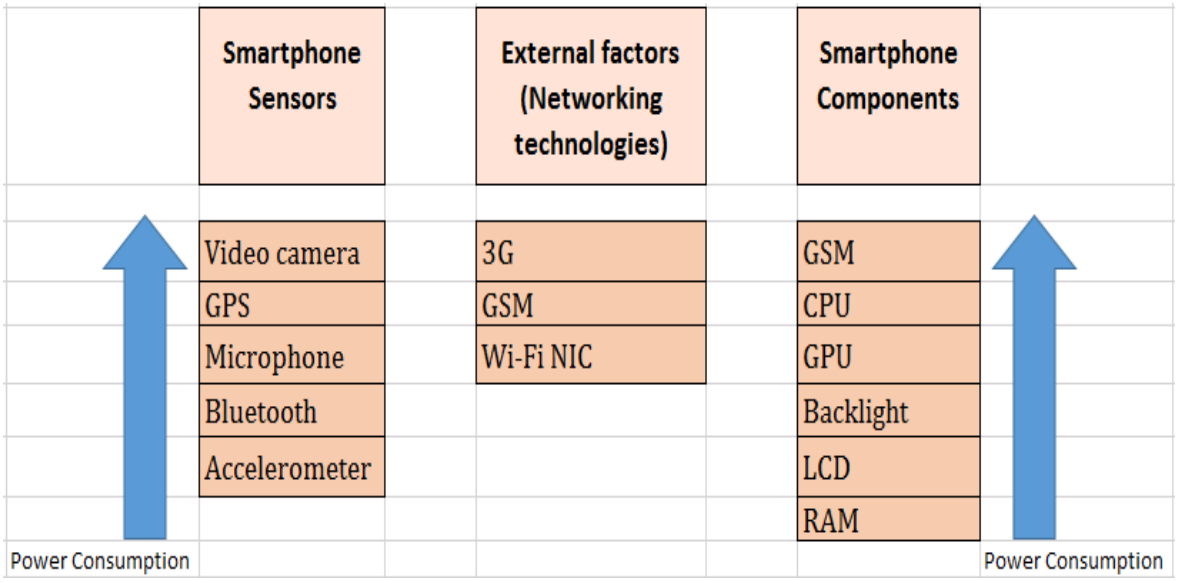

Figure2.1: Hierarchy of the power consumption [1], [7] & [9].

Software Factors:

- OS updates for the Smartphone devices cause power consumption, the main reason behind that is that there could be an energy bug in the updates software, or there could be an energy bug in the applications that are installed within the OS update. For example, the "Suspend" process in Android was observed to run in the background and thus draining energy.
- OS configuration changes as well may cause drain in battery life.
- Now looking at the applications energy bugs, according to the authors, this is the most prominent. For example, no-sleep bugs in applications results in draining unexpected power, as the tasks are still active and no need for them to work anymore, a no-sleep bug is an error done by the application developer at the coding phase, where an application acquires a wake lock for a component which wakes the component up, but does not release the lock even after the task is completed, the no-sleep bugs has been found in many Smartphone applications such as Facebook, location listener, Google calendar, email apps, dialer apps, weather apps, GPS based apps (including Google maps, which turn on GPS and do not turn it off even after their exit), gallery apps (triggered by "no sleep" of the motion sensor).
- Loop bugs is another cause of energy consumption, loop bugs occurs when a Smartphone application enters a looping state, performing unnecessary tasks, and thus draining battery. Such as a simple routine calling itself by mistake.

Hardware Factors:

- Faulty battery is one cause of power consumption.

- Exterior damage could unexpectedly drain battery life, as this could trigger other parts of the phone, such as software, to consume power. For example a faulty home button could result in truing on and off the backlight and the CPU several times causing loss of power.
- Exterior damage could also be from a scratched SIM, thus increasing its internal resistance, which in turn led to more power consumption.
- Exterior damage as well includes damage of the external SDCard, according to the authors, a damaged SDCard could trigger buggy apps into a looping state, which cause a huge drain of power.

External Factors (Networking technologies):

- Wi-Fi NIC consume a lot of energy, what has a major impact as well is weak wireless signals, as this cause the network interface controller NIC to increase its activity and thus resulting in major power drain (over 4G, 3G, EDGE, Wi-Fi and GPRS).
- Wireless handovers as well has an impact on battery life. For example, repeated network handovers (3G to EDGE or 3G to 4G and vice versa).
- Networking technologies is one of the main cost of energy are the networking technologies, such as 3G, GSM and Wi-Fi, it has been reported that power consumption is proportional to the workload occurred as a result of data transfer, rather than the size of the data. Different communication technologies have different data transfer throughputs and thus have different cost of energy.

Smartphone Sensors:

- The increase of sensors in Smartphone devices, increase the risk of power consumption, accelerometer could be a good example, as changing the rate of the accelerometer impacts the battery life. GPS as well is a major concern, GPS increase the cost of power.

Smartphone Components:

- Smartphone devices components as well plays a role in battery life, some components cost less energy that other, for instance as the Fig. (2.1) shows that RAM consume less energy than the CPU, LCD consume less energy than backlight. Of course this is due to several reasons, and studies is focusing on how to reduce the power for each component by itself.

Unknown Energy Bugs:

- Sometimes it is not clear where does the energy goes, specific apps such as browser, juice defender, skype, music players, task killers, themes and some more applications cause battery life to decrease due to issues in the applications.
- Also it has been reported that sometimes it is difficult to identify the main cause of high power consumption.

Chapter Three - Methodology and Approach

3.1 Approach

The approach is to present to the software developers experiments and analysis on how can different data structure and algorithms representations and other factors affects the power consumption. The thesis approach as well is to show how can different settings of the same algorithms affect battery life time, different settings include using arrays, linked list, different data types.

Also the approach is to explore how can other factors have impact on the battery life, and how can the setup of these different factors affect battery life time, this factors include accelerometer, reusing data objects, programming style such as local and global variables.

As known algorithms are measured in terms of time complexity (*Big-O* complexity) that measure the time and performance, and space complexity which measures memory usage. The thesis here will explore and focus on algorithms with different complexities and with different data structure representations. This thesis will adopt the following *Big-O* complexities:

O(logn),O(n), O(nlogn), O(n^2), O(n^3), O(n^4)

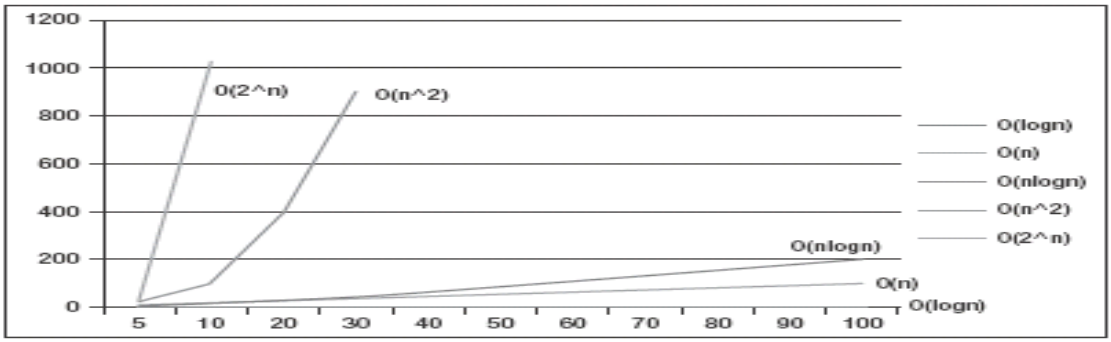

Figure 3.1: Results of each *Big-O* notation when the input (variable n) increases.

(Vo 2011, 92).

3.2 Methodology

In this section we outline the methodology of our research:

1. The experiments will be based on and compiled using Smartphone devices, so in order to create applications (as a test bed for this thesis) on these devices we need to study its programming language and install all related kits (SDK) for it.
2. Specify and design the experiments to be introduced and categorize them. The experiments will explore the relation between algorithms having various time complexities and having different data structure representations against power consumption in Smartphone devices. And will be measured in two ways first the actual readings from Smartphone device using a log file and the second one is using the formula, further details are on the experiments design chapter.
3. Specify other factors that could drain power, such as programming style, reusing objects, accelerometers and others, apply experiments and analyses the collected data.
4. Implement the experiments on a real mobile device.
5. Collect the data for each experiment.
6. Analyze the collected data and present the results in relevant diagrams.
7. Come out with recommendations based on the collected data.

3.3 Scenarios

Below Table (3.1) shows the structure on how the comparison of the collected data will be implemented, so as it would give a clear idea on what to be compared and to get their performance evaluation in terms of CPU overhead and battery consumption.

The four scenarios (with six time complexity of each scenario) that are to be evaluated are the neutral case, real case with array based algorithms, real case with linked list based algorithms, and finally the real case including the modifications on the data type for various time complexities of *O(logn), O(n), O(nlogn), O(n^2), O(n^3) and O (n^4).*

So for the first phase the thesis will present twenty four experiment on a real Smartphone device. (Four scenarios with six time complexity of each).

Table 3.1: Comparison between different time complexities of algorithms.

Time Complexity Algorithms	Scenarios							
	Neutral Case Algorithms		Real Case Algorithms using Arrays		Real Case Algorithms + Data Type Modifications		Real Case Algorithms using Linked List	
	CPU Work load	Power Consumption	CPU Work load	Power Consumption	CPU Work load	Power Consumption	CPU Work load	Power Consumption
O(logn)								
O(n)								
O(nlogn)								
O(n^2)								
O(n^3)								
O(n^4)								

As for the second phase, the following are the scenarios that are evaluated in terms of power consumption:

1- Programming style (global and local variables):

   The programing style is to show how can programming style affects power consumption, for the experiment we present two scenarios, the first is using local variables settings in a piece of code, while the second is using global variable in the same piece of code.

2- Device orientation:

   The thesis will show how can device orientation affects battery life, the two scenarios are the known device orientations portrait and landscape.

3- Device accelerometer (change rate):

   The accelerometer is device used to measure acceleration forces ("g-force"), and it senses the velocity and the motion, the thesis will show how accelerometer can affect battery life, the thesis will present three scenarios on different rates for the accelerometer.

4- Various data representations.(such as array and set):

   The thesis here will show scenarios on how different already defined functions can have impact on battery life. Such as union or intersection that are used in sets.

5- Reusing data objects:

   In this section the thesis will show two scenarios, the first one include a piece of code that reuses data objects, while the other initiates new objects.

Chapter Four - Experiments and Results

4.1 Experimental Design

4.1.1 Device Used

For the thesis experiments we used a mobile device of 1.4 GHz Scorpion Android OS with the specifications in Table (4.1) below, any device with the same specifications can generate the same results as well, we choose this device as it was almost with common specifications at the time of implementation, other devices with other specification could be used, taking into account that data that will be generated will be different, but the results will be the same in terms of concept, for instance other device will be faster, of course the data will be different but the main core and the concept of the results will be the same, i.e., if the results shows that arrays consume power less then linked list, this fact will not be changed on other device, but the data will vary.

4.1.2 Experiments Environment

All experiments are implemented using Java code under Android SDK environment, and using Eclipse as an integrated development environment (IDE). Also we started to test Apple SDK environment, using objective C language, in order to test the power consumption, but we faced many difficulties, such as there is no API to measure the readings of power consumption.

4.1.3 Experiments Setup

The experiment used six algorithms that have time complexities of *O(logn),O(n), O(nlogn), O(n^2), O(n^3),* and *O(n^4).* To start setting up the experiment the following has been done:

1. We created algorithms with the simplest code of loops. These algorithms are of type *O(logn),O(n), O(nlogn), O(n^2), O(n^3),* and *O(n^4).* The first case is considered as a neutral base case which represents the best case scenario, as in this case this code is created with the simplest *for loops* statements to reach the intended time complexity, that is algorithms are created for each of the above mentioned time complexity, one algorithms for each of these types: *O(logn),O(n), O(nlogn), O(n^2), O(n^3)* and *O(n^4),* on each of these algorithms we created a Java instance of class, as each Java object has its own data, and we dynamically enlarge this object during the execution of every and each iteration for each algorithm so as we can monitor and evaluate two things here, first the power consumption of the mobile device and seconds the CPU overhead of each algorithm, we executed the code on the phone device and created a log file as well on the phone device in order to monitor the algorithms performance and CPU overhead. It is Worth mentioning that the first case have no major calculations and this is intended as this is a neutral case but this instance has been added to monitor the above mentioned metrics of battery life and CPU overhead. The above is considered as the first measurement and the initial experiment step.
2. We added algorithms with the same time complexities but in this case with more calculations and more CPU overhead, doing this we can monitor and evaluate the performance of the algorithms, that have more instructions and calculation, over the CPU and the battery life, in this step we selected real common sorting algorithms

that are commonly used by programmers namely: Bubble sort, insertion sort, selection sort, quick sort and finally merge sort. To monitor and evaluate the battery life and the CPU overhead. This is considered as the real case. Section 2.1 shows a theoretical study for these types of algorithms.

3. Third level has been added to the experiments, in this step we used the same sorting algorithms but this time the algorithms was linked list based instead of array based.
4. Finally, fourth level has been added to the experiments, in this step we used the same sorting algorithms but at this level we changed the data types. That is we replaced integer data types with float data type, so as to evaluate if this could have any impact on battery life and CPU overhead.

4.1.4 Experiments Metrics and Measurement

Several performance metrics are collected in this thesis:

1. CPU process time.
2. Power consumption (micro joule/byte).
3. Power consumption (percent in battery consumed, reading from the log file created in the phone device).

The CPU process time: As the CPU carries out all the instructions of a program, so the CPU overhead is the CPU time for the algorithm to be executed and we recorded this time in milliseconds as this is the time to execute the algorithm instructions.

For consumption of the energy cost, this technique is used by some of the researches in this domain (Abdul Elminaamet al. 2009).By using the cycles, the operating voltage of the CPU, and the average current drawn for each cycle, we can calculate the energy consumption of the intended algorithms. For example, on average, each cycle consumes

approximately 270 *mA* on an Intel 486DX2 processer (Wei et al. 2001) or 180 *mA* on Intel Strong ARM (Sinha and Chandrakasan 2009, 220-225). Then, the amount of energy consumed by program *P* to achieve its goal is given by $E=Vcc*I*T$ joules, since for a given hardware *Vcc* are fixed and *I* also depends on the hardware. Further explanation will be presented later in the data collection and analysis chapter.

The other method to measure the power consumption is from the readings from the device. This method monitors the battery level and reads the battery level after each iteration and collects the data in the log file. That is this method collects the changes in the battery level percentage readings from the phone device itself, almost similar approach has been used by (Abdul Elminaamet al. 2009).

4.1.5 Experiments Tasks

The thesis tries to explore and to implement the following tasks:

1. A comparison is conducted between the results of the selected different time complexities of algorithms that are executed over various scenarios: Neutral case, real case array based, real case linked list based and finally real case with data type modifications.

2. A thorough and comprehensive analysis is conducted on the generated results to try to evaluate the performance of the different time complexities algorithms and their scenarios.

Table4.1: Specifications of the mobile device that were used during the thesis.” The specs has been taken from phone manual and from the following web site http://www.gsmarena.com/samsung_galaxy_w_i8150-4114.php”

GENERAL	**2G Network**	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 900 / 2100
	SIM	Mini-SIM
	Announced	2011, August
	Status	Available. Released 2011, October
BODY	**Dimensions**	115.5 x 59.8 x 11.5 mm (4.55 x 2.35 x 0.45 in)
	Weight	114.7 g (4.02 oz)
DISPLAY	**Type**	TFT capacitive touchscreen, 16M colors
	Size	480 x 800 pixels, 3.7 inches (~252 ppi pixel density)
	Multitouch	Yes
SOUND	**Alert types**	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
		- DNSe sound enhancement
MEMORY	**Card slot**	microSD, up to 32 GB
	Internal	1.7 GB storage, 512 MB RAM, 2 GB ROM
DATA	**GPRS**	Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps
	EDGE	Class 12
	Speed	HSDPA, 14.4 Mbps; HSUPA, 5.76 Mbps
	WLAN	Wi-Fi 802.11 b/g/n, Wi-Fi hotspot
	Bluetooth	Yes, v3.0 with A2DP
	USB	Yes, microUSB v2.0
CAMERA	**Primary**	5 MP, 2592x1944 pixels, autofocus, LED flash, check quality
	Features	Geo-tagging, touch focus, face and smile detection
	Video	Yes, 720p@30fps, check quality
	Secondary	Yes, VGA
FEATURES	**OS**	Android OS, v2.3.5 (Gingerbread)
	Chipset	Qualcomm MSM8255T Snapdragon
	CPU	1.4 GHz Scorpion
	GPU	Adreno 205
	Sensors	Accelerometer, proximity, compass
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM, RSS
	Browser	HTML, Adobe Flash
	Radio	Stereo FM radio with RDS
	GPS	Yes, with A-GPS support
	Java	Yes, via Java MIDP emulator
	Colors	Black, White
		- SNS integration - MP4/DivX/XviD/WMV/H.264/H.263 player - MP3/WAV/eAAC+/Flac player - Organizer - Image/video editor - Document viewer/editor - Google Search, Maps, Gmail, YouTube, Calendar, Google Talk, Picasa - Voice memo/dial/commands

		- Predictive text input (Swype)
BATTERY		Li-Ion 1500 mAh battery
	Stand-by	Up to 570 h (2G) / Up to 420 h (3G)
	Talk time	Up to 17 h 50 min (2G) / Up to 8 h 20 min (3G)
TESTS	**Display**	Contrast ratio: 853:1 (nominal)
	Loudspeaker	Voice 69dB / Noise 66dB / Ring 67dB
	Audio quality	Noise -87.8dB / Crosstalk -87.3dB
	Camera	Photo / Video

The thesis will present the major issues and obstacles that have faced us during implementing the experiments and how we managed to find solutions.

4.2 Difficulties and Constraints during the Experiment

Problem One: operating system and platform:

To be able to develop applications on mobile devices, I have learned *Objective C* language for iPhone mobile devices, and then I downloaded the *SDK* and installed the *Xcode* on a MacBook device and started developing applications for the experiments. I faced many problems while configuring and installing the applications on this device, but at the end it went smoothly. Afterwards, we started to get the rate of power consumption of the application being developed, unfortunately a problem occurred: There is no API that can get the exact reading of the power form the mobile device; the reading can only be extracted on steps of 5 percent decrements (so it goes like this: 95% then 90% then 85% and so on). Then we tried to install a patch that can set the reading at the LCD of the mobile device every one percent, and got the readings manually after every transaction of the algorithm. The problem with this manual method is that it takes a lot of time and as well not accurate data will be collected, readings from the following authors (Vo 2011) and (Kochan 2003).

After several readings and consultation with experts in this domain, we decided to change the development environment altogether and switched to Android. We installed Eclipse and started developing applications for the experiments using Android operating system. The power consumption reading was more accurate on Android; it was different from the iOS environment and gave me the power consumption readings on a 1% steps. Worth mentioning that while reading on this issue, we found a research (Pathak et al. 2012) by Microsoft and Purdue University that recommends using the Android OS environment rather than the iOS as they faced issues with the environment of the iOS. There are also certain issues with the iOS device power system and battery that makes it difficult to get the exact reading of power and battery level at any given moment.

Problem Two: Algorithms complexity

To start the experiments we need to design algorithms with time complexities of *O(logn), O(n), O(nlogn), O(n^2), O(n^3), and O(n^4)* as a test cases. The thesis took common sorting algorithms such as *bubble sort* and *insertion sort* that have a time complexity of *O(n^2),* the *quick sort* algorithm that has a time complexity of *O(nlogn),* and a matrix multiplication algorithm that has a time complexity of *(n^3)*. But two problems appeared regarding the inaccuracy between the various types of algorithms. The first problem was that the bubble sort algorithm has a time complexity of *O(n^2),* and insertion sort algorithm has as well a time complexity of *O(n^2).* One algorithm will consume more power than the other. The second problem was that the same algorithm has best, worst and average cases. The other algorithms have the same properties, so the thesis should avoid using the worst case in one algorithm and the best case in another algorithm; the results will be inaccurate.

So in order to overcome this problem that may cause inaccurate data, we used a standard format to represent all algorithms (simple and regular coding format) to generate uniform

complexity. And then we took the second real case algorithm with common sorting algorithms and took the average from the collected data.

Problem Three: Device memory

When we started to implement the algorithms at the first time (i.e., the bubble sort, insertion sort, quick sort, matrix multiplication …) the input for these algorithms was a function of random number (a set of array of random number).The array has been increased every time in order to measure the battery drain from the mobile device. While increasing the array another problem appeared with the device memory. The insufficient memory in the mobile device and (then) the application crashed. This occurs in certain array limit of each function.

The problem was solved by amending the code and do partitioning in the code in order to avoid the application crash. That is to set a part of the code fixed and amend the other. The data collected has been put in a file in order to analyze it. Then the data has been exported to SPSS. After analyzing the data in SPSS, but unfortunately this test did not succeed at all since the data was incorrect.

4.3 Data Collection

Only three out of six time complexity algorithms succeed to work, i.e., in total nine experiments (out of 24) succeed to get the data, while the other time complexities did not generate the accurate results, and to get the intended data, and the following will justify that:

1. For algorithms of time complexities of O*(logn) & O(n)*, the runtime was very fast thus no records are generated for evaluation, all of the data read zero in these cases. Even when the experiments are implemented on the real device, the runtime was too fast, in such a way that the readings in the log file was almost zero, or even very small decimals, that cannot be accurate in analyzing them, this is justified from the fact that O*(logn) & O(n)*are very fast time complexities, and that is why algorithms of those kinds are always used, or even in algorithms complexities reduction is needed.
2. As for $O(n^4)$ the simplest code was too slow that it takes hours to generate one reading and then the runtime crashes, in some cases the battery was fully consumed while not having a single reading, and that is why time complexity is important factor for each algorithm, taking into account that sometimes a bug in the application could have the same impact, for instance loop bugs occurs when Smartphone application enters a looping state, performing unnecessary tasks, and thus draining battery, such as a simple routine calling itself by mistake.

   As for $O(n^3)$ data collection, we only managed to generate data for the neutral case scenario only, while for other scenarios the application crashes all the times, again the same justification as above.

Worth mentioning, that we tried hardly to collect some data for the above time complexities, as we planned to apply a sort of forecast on the collected data to meet the

goal, but the device runtime crashes all the times because of the limited capabilities of Smartphone devices in general. Below Table (4.2) illustrates the succeeded experiments.

Table 4.2: Successful experiments

Time Complexity Algorithms	Scenarios							
	Neutral Case Algorithms		Real Case Algorithms using Arrays		Real Case Algorithms + Data Type Modifications		Real Case Algorithms using Linked List	
	CPU Work load	Power Consumption	CPU Work load	Power Consumption	CPU Work load	Power Consumption	CPU Work load	Power Consumption
$O(logn)$								
$O(n)$								
$O(nlogn)$	Y	Y	Y	Y	Y	Y	Y	Y
$O(n^2)$	Y	Y	Y	Y	Y	Y	Y	Y
$O(n^3)$	Y	Y						
$O(n^4)$								

At the initial phase we collected the data for the neutral case algorithms which represents the best case scenario, the code has been created for the six time complexities, then the code has been executed on the Smartphone device to get the readings, a special log file has been created on the device in order to collect the readings of the CPU time and the device power consumption levels for the intended algorithms. The collected CPU process time is recorded in milliseconds, as this is the time that the CPU carries out the all the instructions of a program, as well the mobile device battery level has been collected and registered in the same log file.

Following up on collecting data, same approach has been followed and implemented for the second scenario for the real case array based algorithms, and then the third scenario for real case linked list based algorithms, and finally the fourth case that include the data types modifications.

All the data collected from the text log file in the mobile device has been converted into excel sheets in order to analyze the results. The data includes the time complexity of the

algorithms which are *O(nlogn), O(n^2) &O(n^3)* based on the scenarios previously discussed. Second is the time it took the algorithm to finish executing the code and is registered in milliseconds. And final column is the amount of power consumed when the algorithm finished executing the code for each transaction. Afterwards the data has been analyzed initially by using the SPSS Statistics software package and then by using excel.

With the above discussion we covered the implemented methods for data collection for two mercies, 1- the CPU process time & 2- the Power consumption (percent in battery consumed, reading from the log file created in the phone device).

Now we will explain how we collect the data for the third measurement, which is power consumption (micro joule/byte).

This measurement method has been used by some researchers and studies digging around the same domain, such as (Abdul Elminaam et al. 2009) and (Wei et al. 2001), according to their studies they reported that by using the cycles, the operating voltage of the CPU, and the average current drawn for each cycle, the energy consumption of the intended algorithms can be calculated.

And the researches gives examples on that, for example on average, each cycle consumes approximately 270 *mA* on an Intel 486DX2 processer or 180 *mA* on Intel Strong ARM. Then, the amount of energy consumed by program *P* to achieve its goal is given by: *E=Vcc*I*T* joules, since for a given hardware *Vcc* are fixed and *I* also depends on the hardware.

Looking at (Wei et al. 2001) explanation in more details, they explained the method as follows: Let a program *P* run in *T* seconds to achieve its goal, *Vcc* be the supply voltage of the system, and *I* be the average current in ampere drawn from the power source for *T* seconds. We can rewrite *T* as *T=N * t*, where *N* is the number of clock cycles and *t* is the

clock period. Then, the amount of energy consumed by *P* to achieve its goal is given by: *E = Vcc * I * N * t* joules. Since for a given hardware, both *Vcc* and *t* are fixed, *E ὰ I * N*. However, at the application level, it is more meaningful to talk about *T* than *N*, and therefore we express energy E ὰ I * T. at the end we have a formula given by *E=Vcc*I*T* joules, as *Vcc* is fixed and *I* is fixed depending on the device.

In our thesis, in top of collecting the data using the battery level from the device, we used as well the same method of (Abdul Elminaam et al. 2009) and (Wei et al. 2001), we used the same formula to calculate the amount of energy, *E=Vcc*I*T*, as *Vcc* is fixed and *I* is fixed we collected these information as average from the device data sheet, each cycle consumes approximately180 *mA* on ARM and *Vcc* is 5 fixed for the device. Then, the amount of energy *E*=5*180* the duration for the program to execute which is *T* seconds. As we mentioned previously the time to execute a program for a specific transaction has been collected as well.

Based on the collected data and the results, the thesis present the following graphs:

1. Fig (4.1): Performance evaluation of algorithms with different data structure settings on power consumption, this graph shows the algorithms that consume less power (in percent), to the algorithms that consume more power.
2. Fig (4.2): Performance evaluation of algorithms with different data structure settings on CPU, this graph shows the algorithms that consume less CPU overhead (in percent), to the algorithms that consume more CPU overhead. (We only presented the neutral case as to have a clear picture on the differences).
3. Fig (4.3): Performance evaluation of algorithms with different data structure settings on power consumption (micro joule/byte), this graph shows the algorithms that consume less power (in percent), to the algorithms that consume more power.

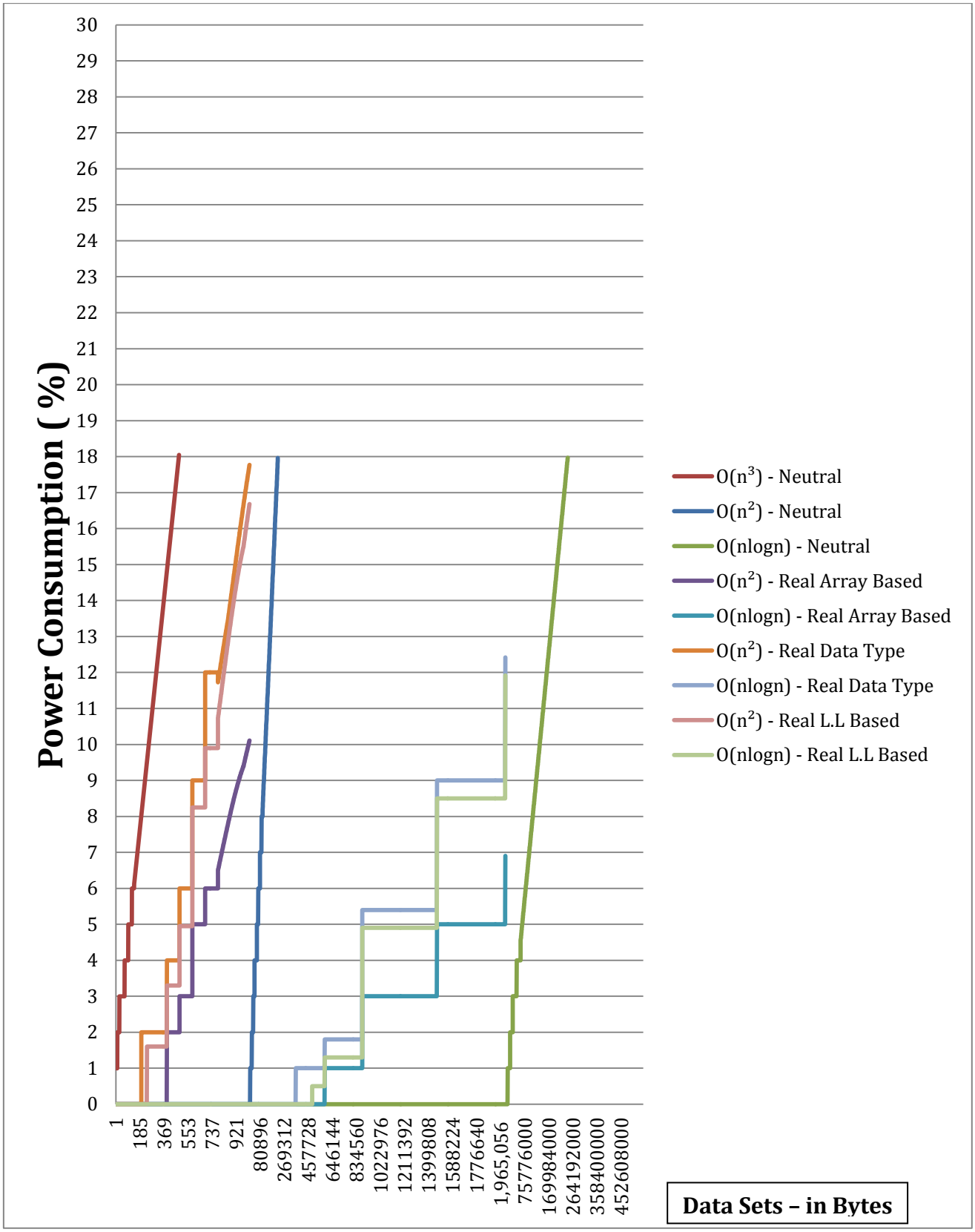

Figure4.1: Performance evaluation of algorithms with different data structure representations on power consumption

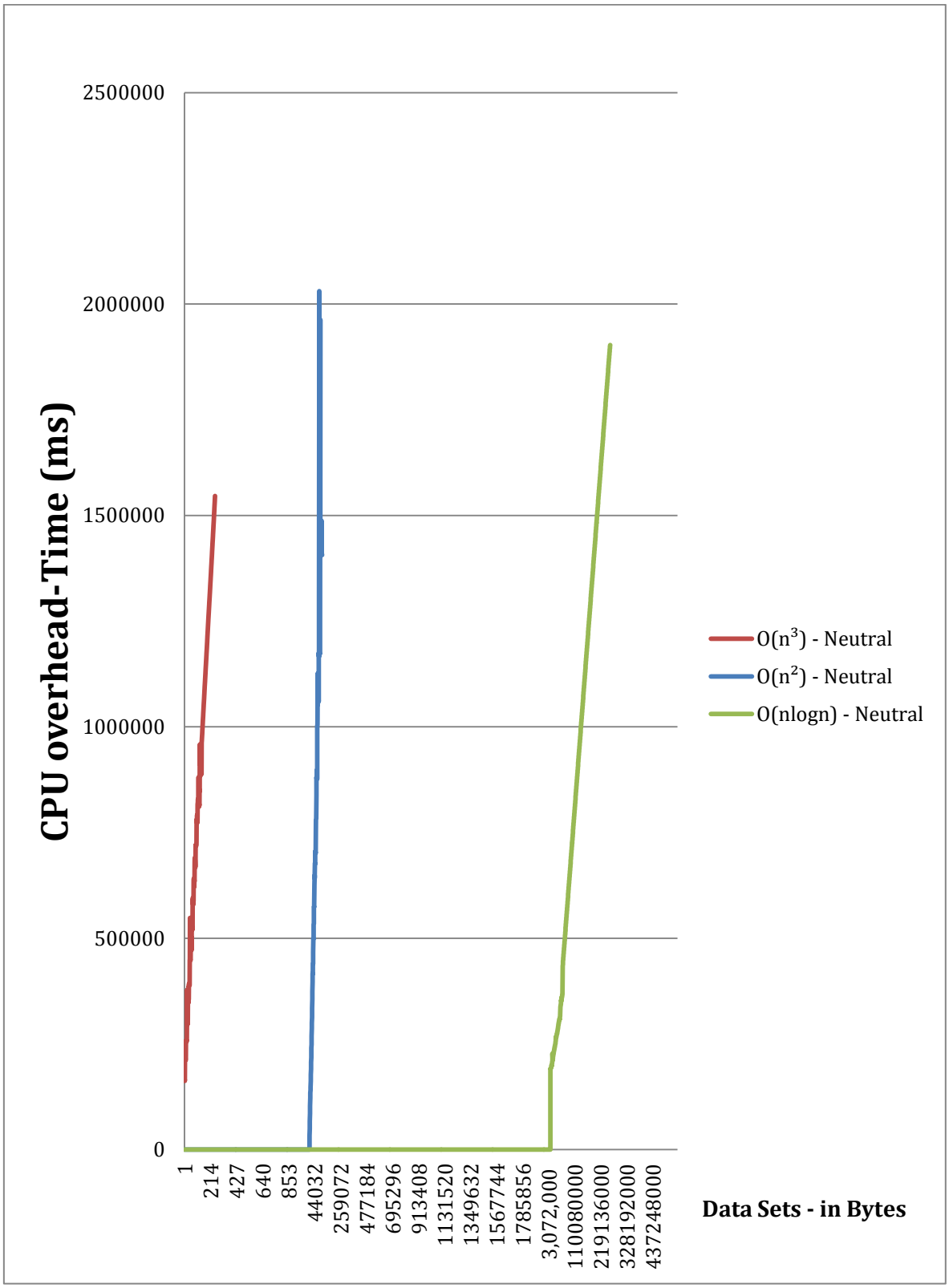

Figure 4.2: Performance evaluation of algorithms with different structure representations on CPU

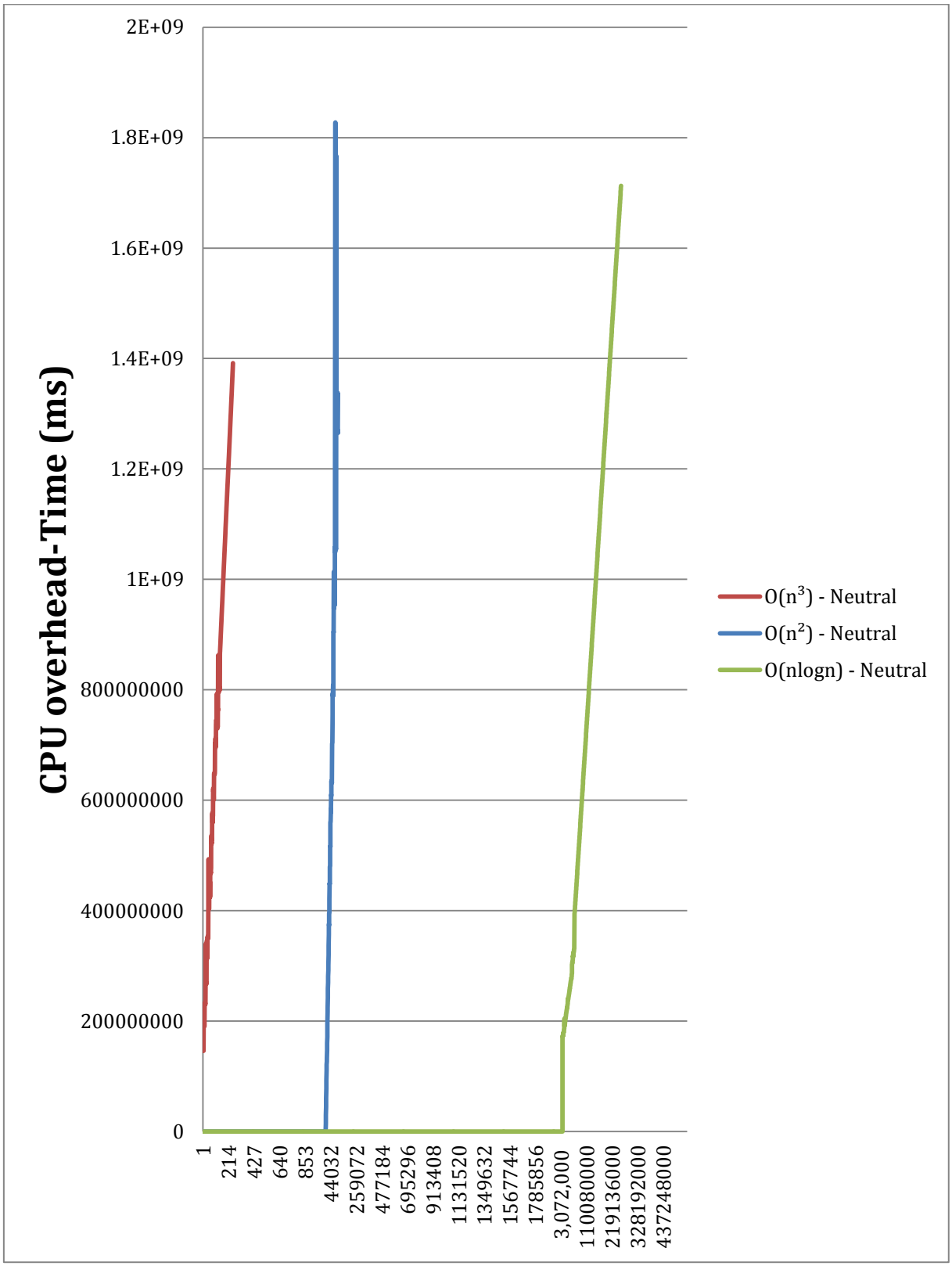

Figure4.3: Performance evaluation of algorithms with different structure representations on micro joule/byte

4.4 Using the SPSS Statistics Software Package.

We run all the collected data for the various implemented scenarios on SPPS Statistics software package in order to find out the significance factor between the data (i.e., the relation factor), according to SPSS a factor of less than 0.5 percent means that there exists a significance between the data (i.e., between X & Y), which means in our case a relation between the data sets of various scenarios and between the power consumed from the Smartphone device, below Table (4.3) shows SPPS data which shows a zero significance value for each element in our case study.

Table 4.3: SPSS data

Coefficients(a)

Model		Unstandardized Coefficients		T	Sig.
		B	Std. Error		
1	(Constant)	-35	0.532	-66.113	0.000
	O(n^3) Scn1	29	0.049	585.751	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	491	33.607	14.623	0.000
	O(n^2) Scn1	12207	7.741	1,576.896	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	524973	18,000	29.165	0.000
	O(nlogn) Scn1	13511671	4,544	2,973.478	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	219	3.532	61.991	0.000
	O(n^2) Scn2	84	0.721	115.810	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	151198	4,376.631	34.547	0.000
	O(nlogn) Scn2	321768	1,871.640	171.918	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	167	2.971	56.217	0.000
	O(n^2) Scn3	51	0.334	151.633	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	126113	3,992.211	31.590	0.000
	O(nlogn) Scn3	183074	946.413	193.440	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	191	3.151	60.685	0.000
	O(n^2) Scn4	53	0.389	136.879	0.000

a. Dependent Variable: X

Coefficients(a)

Model		Unstandardized Coefficients		t	Sig.
		B	Std. Error		
1	(Constant)	115946	3,738.287	31.016	0.000
	O(nlogn) Scn4	188638	902.900	208.924	0.000

a. Dependent Variable: X

4.5 Additional Points for Discussion.

Looking further and further on the collected data and the graphs we can evaluate the performance of the various scenarios in couple of points:

1. If we took the first and the second graph (Fig4.1) & (Fig4.2) that illustrates the relation between different algorithms having different time complexities and having various data structure representations against the Smartphone battery levels and against the CPU overhead we find out the following:
   A. That algorithms having time complexity of n cube with its neutral case as best case scenario comes at the top most level of consuming battery life and the top level of CPU overhead. This is obvious from the collected data and the graphs representations.
   B. N square comes at the second level of consuming battery life and using CPU with the following ranking (High to Low) of algorithms data structure representations: Data types adjustment as double -> linked list based algorithms -> array based algorithms.
   C. N LOG N comes at the third level of consuming battery life and using CPU with the following ranking (high to low) of algorithms data structure representations: Data types adjustment as double -> linked list based algorithms -> array based algorithms.
2. Based on evaluating the performance of various algorithms having different data structure, it is obvious that choosing the write data structure at the coding phase have impact of the battery life of the Smartphone device and as well the CPU overhead.

3. Based on the results of the data from the graphs and from the SPPS we come out with linear relation.

In order to summarize the outputs of the experiments and the analysis, we present as well a hierarchy of the collected results, Fig (4.4) show the hierarchy of the experiments from the least to the most consumption of power and CPU overhead.

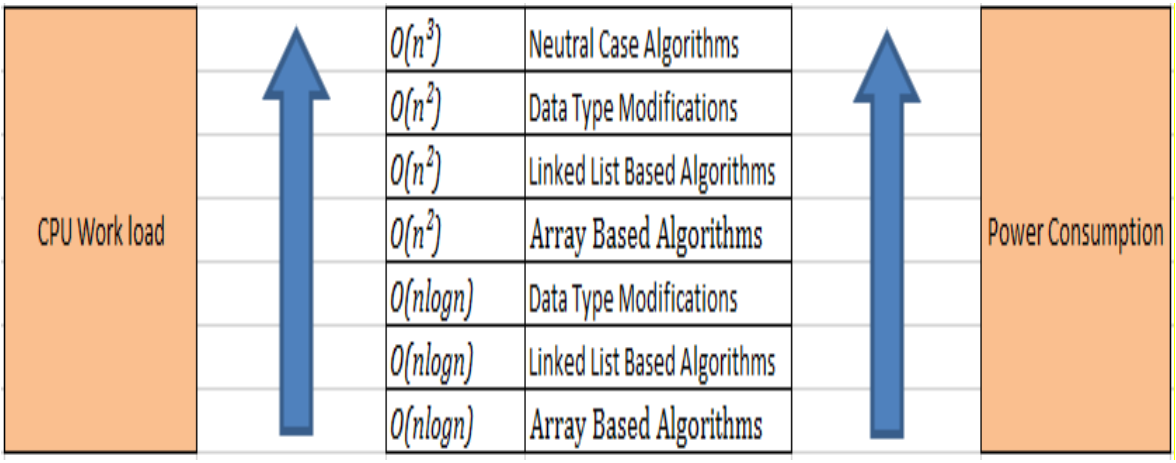

Figure 4.4: Hierarchy of power consumption and CPU overhead

4.6 Recommendations for Research

The following recommendations are offered for this research

- Given the changing nature of technology towards more implementations on mobile Smartphone devices, and given the fact that power management is a crucial factor in mobile devices implementation it is significant that software developers should start designing applications that are power efficient.
- Based on the results of the research and based on evaluating the performance of algorithms with different data structure representations on power consumption and CPU overhead of Smartphone devices it is crucial to take into account, at the coding

phase, the impact on battery level and as well the impact on CPU, while choosing between different data structure representations for various time complexities.

4.7 Other Factors

In this section the thesis will introduce some other—related methods and experiments that have impact on mobile device battery life. The thesis explore methods and techniques for reducing power consumption within the software life cycle.

The following points are discussed in relation to power consumption:

I. Programming style (global and local variables).
II. Device orientation.
III. Device accelerometer (change rate).
IV. Various data representations (such as array and set).
V. Reusing data objects.

However, (Larsson 2010) has focused on other guidelines on how to optimize applications for energy efficiency. Such as loops, performance libraries/extensions, algorithms, complier optimization, drivers, programming language and others, the author described briefly each factor and how can it be optimized. For example the author regarding the loops advised to minimize the use of tight small loops. To reduce the comparison and testing overhead associated with these loops, the performance/power relationship can be improved by loop unrolling. To achieve this goal, the instructions that are called in multiple iterations of the loop are combined into a single iteration. This approach will speed up the program if the overhead instructions of the loop impair performance significantly. Side effects may include increased register usage and expanded code size. As well other factors have been discussed in this paper.

4.7.1 Programming Style (global and local variables).

For our experiment we will use global and local variables in order to figure out if these could have any effect on battery life during the coding phase. So for our experiment we did the following: First, local variables have been defined in the code and in the iteration loop section. Second, global variables have been defined in the code. The input data has been set in megabytes. The code has been executed to measure the amount of power consumption that has been used for each of the variables together with the time it took each algorithm to execute. The time has been measured in nanoseconds, and the power consumption in percent rate. The results have been read for each of the inputs so as to be able to analyze the data. In this experiment the data sets in megabytes, starting from 1 megabyte and reaching 31 megabytes. (31 megabytes is the maximum the device was able to take readings, after 31 megabytes the application crashes and error arise, the error that arises is insufficient memory).

The results of the experiment are presented in the below tables Table (4.4) & Table (4.5) and the analysis is illustrated in Fig (4.5) & Fig (4.6).

Table 4.4: Local variables average readings.

Programming Style	Method Used	data set Megabytes	Bytes	milliseconds	seconds	minutes	Battery usage
Local variables declarations	Iteration loops	1	1,048,576	115631	115.631	1.927183	1%
		2	2,097,152	107727	107.727	1.79545	1%
		3	3,145,728	112663	112.663	1.877717	1%
		4	4,194,304	133973	133.973	2.232883	1%
		5	5,242,880	130193	130.193	2.169883	1%
		6	6,291,456	129130	129.13	2.152167	1%
		7	7,340,032	114720	114.72	1.912	1%
		8	8,388,608	125494	125.494	2.091567	1%
		9	9,437,184	138265	138.265	2.304417	1%
		10	10,485,760	134863	134.863	2.247717	1%
		11	11,534,336	129050	129.05	2.150833	1%
		12	12,582,912	136231	136.231	2.270517	1%
		13	13,631,488	143232	143.232	2.3872	1%
		14	14,680,064	163417	163.417	2.723617	1%
		15	15,728,640	189124	189.124	3.152067	2%
		16	16,777,216	209391	209.391	3.48985	2%
		17	17,825,792	225040	225.04	3.750667	2%
		18	18,874,368	229648	229.648	3.827467	2%
		19	19,922,944	235881	235.881	3.93135	2%
		20	20,971,520	233801	233.801	3.896683	2%
		21	22,020,096	242574	242.574	4.0429	2%
		22	23,068,672	249751	249.751	4.162517	2%
		23	24,117,248	232801	232.801	3.880017	2%
		24	25,165,824	252168	252.168	4.2028	2%
		25	26,214,400	246360	246.36	4.106	2%
		26	27,262,976	258000	258	4.3	2%
		27	28,311,552	230122	230.122	3.835367	2%
		28	29,360,128	227033	227.033	3.783883	2%
		29	30,408,704	241535	241.535	4.025583	2%
		30	31,457,280	260350	260.35	4.339167	2%
		31	32,505,856	232134	232.134	3.8689	2%

The table shows the data of the local variables implemented in the algorithm and the data column shows the input size in megabytes, the time in milliseconds is the time it took the algorithm to execute the code then it is converted into seconds and minutes to be more readable for the user and at the end battery usage % column is the amount of power consumption it took the algorithm to execute the code.

Table 4.5: Global variables average readings.

Programming Style	Method Used	data set Megabytes	Bytes	milliseconds	seconds	minutes	Battery usage
Global variables declarations	Iteration loops	1	1,048,576	118116	118.116	1.9686	1%
		2	2,097,152	122850	122.85	2.0475	1%
		3	3,145,728	128819	128.819	2.146983	1%
		4	4,194,304	143321	143.321	2.388683	1%
		5	5,242,880	122575	122.575	2.042917	1%
		6	6,291,456	121128	121.128	2.0188	1%
		7	7,340,032	134302	134.302	2.238367	1%
		8	8,388,608	139388	139.388	2.323133	1%
		9	9,437,184	120881	120.881	2.014683	1%
		10	10,485,760	141282	141.282	2.3547	1%
		11	11,534,336	140920	140.92	2.348667	1%
		12	12,582,912	130464	130.464	2.1744	1%
		13	13,631,488	161107	161.107	2.685117	1%
		14	14,680,064	149280	149.28	2.488	1%
		15	15,728,640	148662	148.662	2.4777	1%
		16	16,777,216	138695	138.695	2.311583	1%
		17	17,825,792	150370	150.37	2.506167	1%
		18	18,874,368	142648	142.648	2.377467	1%
		19	19,922,944	165849	165.849	2.76415	1%
		20	20,971,520	159607	159.607	2.660117	1%
		21	22,020,096	139467	139.467	2.32445	1%
		22	23,068,672	174450	174.45	2.9075	1%
		23	24,117,248	150535	150.535	2.508917	1%
		24	25,165,824	166846	166.846	2.780767	1%
		25	26,214,400	176483	176.483	2.941383	1%
		26	27,262,976	173391	173.391	2.88985	1%
		27	28,311,552	188049	188.049	3.13415	1%
		28	29,360,128	159640	159.64	2.660667	1%
		29	30,408,704	175120	175.12	2.918667	1%
		30	31,457,280	175285	175.285	2.921417	1%
		31	32,505,856	183665	183.665	3.061083	1%

The table shows the data of the global variables implemented in the algorithm and the data column shows the input size in megabytes, the time in milliseconds is the time it took the algorithm to execute the code then it is converted into seconds and minutes to be more readable for the user and at the end battery usage % column is the amount of power consumption it took the algorithm to execute the code.

Results and Analysis

Fig (4.5) & Fig (4.6) show the results of the experiment. It represents the difference in terms of power consumption when choosing between local variables and global variables.

Fig (4.5) shows that using global variables consume less power on mobile devices. It consumes only 1% power starting from one megabyte till the end of the experiment (thirty one megabyte), while local variable consumes 2% starting 15 megabytes and reaching the end. Of course this rate of power consumption is not huge, but if added to other factors it makes a difference. (Please note that local and global variables consume the same amount of energy from 1 till 15 megabyte, this is because as data increases, power consumption increases).

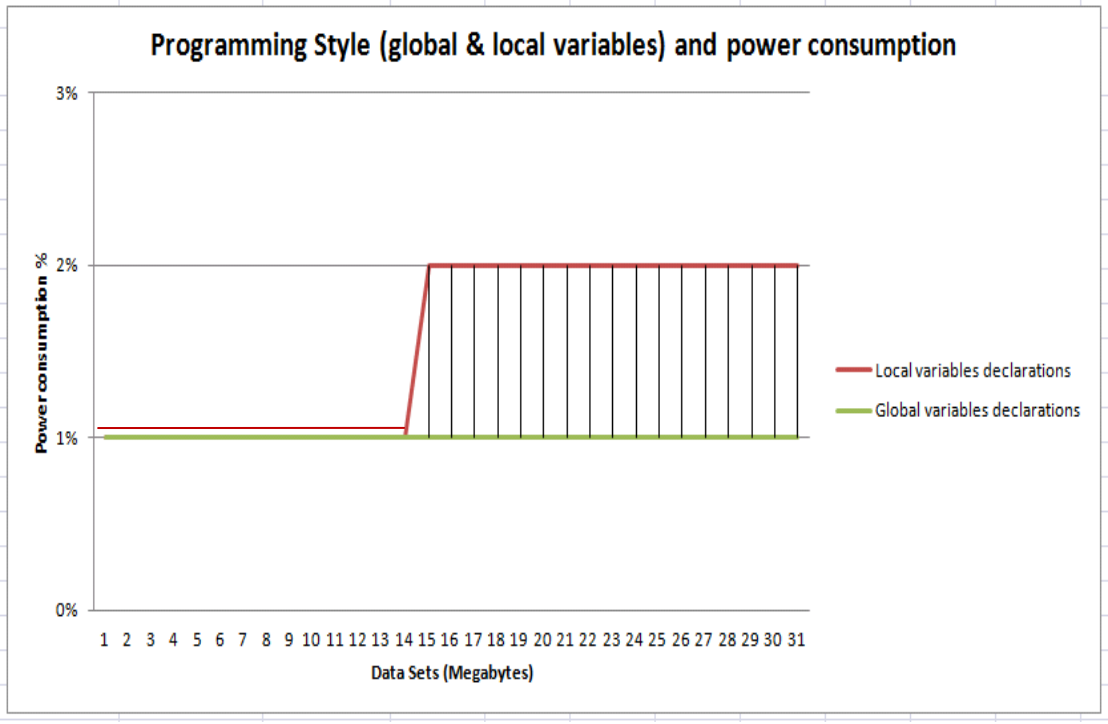

Figure 4.5: Programming Style and time power consumption.

Fig (4.6) presents the relation between the two variables and time consumption. It is noted that global variable consumes less time than local variable from 15 megabytes onwards, in contrast to the local variable which consumes less time starting from 1 megabytes and reaching 14 megabytes. The first half (1-14 megabytes) in the below graph shows that the data is still instable, while as the experiment and data continues, the data seems more stable and the graph takes it shape, and even the graph shape can expected on how it will look like(forecast).

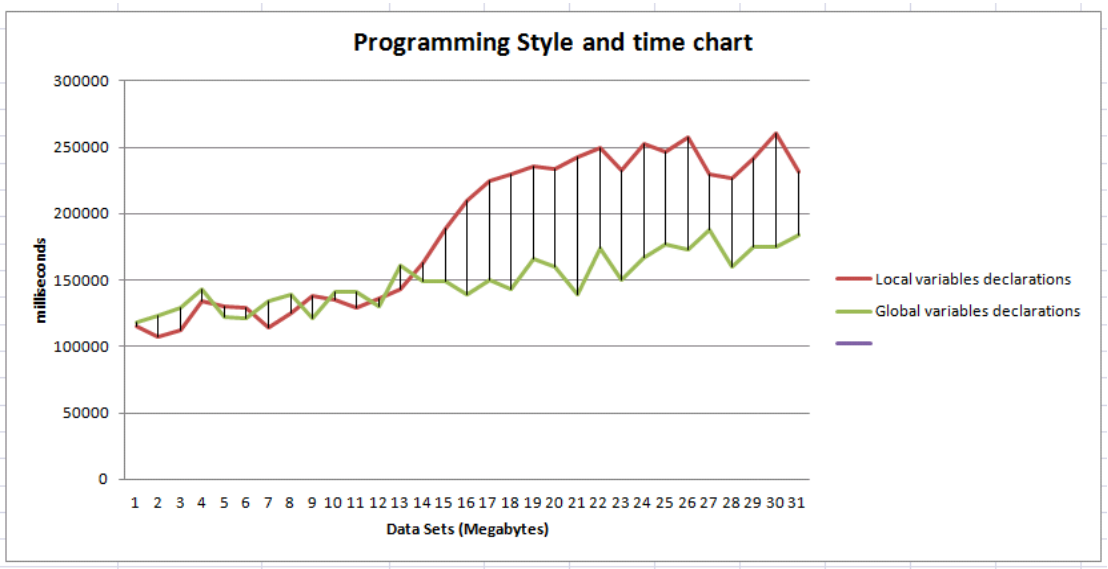

Figure 4.6: Programming Style and time chart.

4.7.2 Device Orientation.

For our experiment we will try to explore if the device orientations have any impact on battery life time, device orientation means changing the mode of the device from portrait to landscape depending on how the user hold the device. So in this experiment we did two tests, in the first one we changed the device orientation between portrait and landscape for a fixed time and on the second test we removed the option the changes the device orientation and tested the code for the same fixed time in the previous test and here are the results that we collected illustrated in Graph Fig (4.7), the results show the setting the device with this option consume more power while keeping this option off will reduce the power consumption , but looking deeply at the results it shows that the difference is minor.

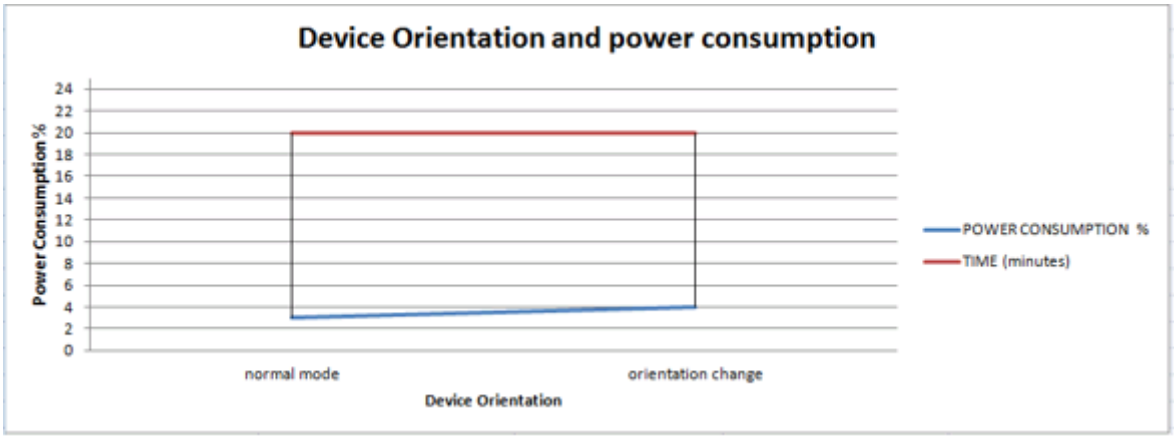

Figure 4.7: Device orientation and power consumption.

4.7.3 Device Accelerometer (change rate).

The experiment will explore if changing the rate of the device accelerometer affects mobile device battery life time, we applied three change rates to test their impact, however change rate can be defined as a value that can be set during the coding phase from which you can control the rate when the accelerometer can work, but of course this could have effect on the quality of the mobile device software application, that is increasing the change rate delay the accelerometer for identified period, and thus could have impact on the quality of the application, for instance the change rate can be set as 0.1 seconds (or to be more accurate 0.1 part of the second), or 0.5 seconds, i.e., activation of the accelerometer every 0.1 part of the second, take for example when a user plays a game that uses the accelerometer, the reaction time must be high in order for the game to be of good quality, but according to the results of the experiment, the accelerometer has impact on the battery life, and this could be replaced for instance by buttons.

The collected results are illustrated in graph Fig (4.8) and the results shows a significance difference of power consumption between various change rates.

The experiment shows that 0.1 change rate can consume 12% of the battery life time in 20 minutes, while using a change rate of 0.5 consume 6% of the battery life time at the same time. A difference that is considered high. It is noted from the results, that the accelerometer use a lot of overhead when configured.

We conclude that decreasing the change rate will result in draining more battery life time as the device will have more overhead, and on the contrary when increasing the change rate this will result in consuming less power, as this will result in less overhead but this would definitely affect the quality of the application.

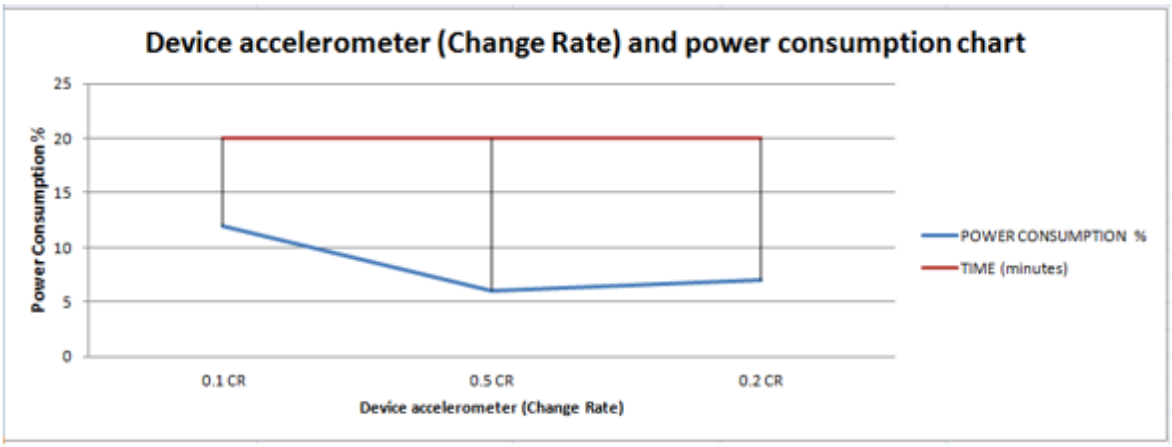

Figure 4.8: Device accelerometer and power consumption.

4.7.4 Various Data Representations (such as array and set).

The experiment we will explore if programming style could have any impact on mobile device battery life, so for our experiment we will explore the difference between arrays and sets in terms of power consumption, so we created a code, and in the same piece of code one time we used arrays and on the other test we used sets, and we executed the code over the mobile device on both tests, and we collected the results from the log file that we created. The collected results are illustrated in graph Fig (4.9), and the results show significance difference of power consumption between using arrays and using sets. If we further investigated the graph we see that using array would consume more power than using sets, as for the same data sets, arrays consume power while in sets the reading shows zero power consumption, also as shown in the graph increasing data sets will increase power consumption. So in short it is concluded that using arrays consumes power in mobile device while using set consume no power as well consume little time.

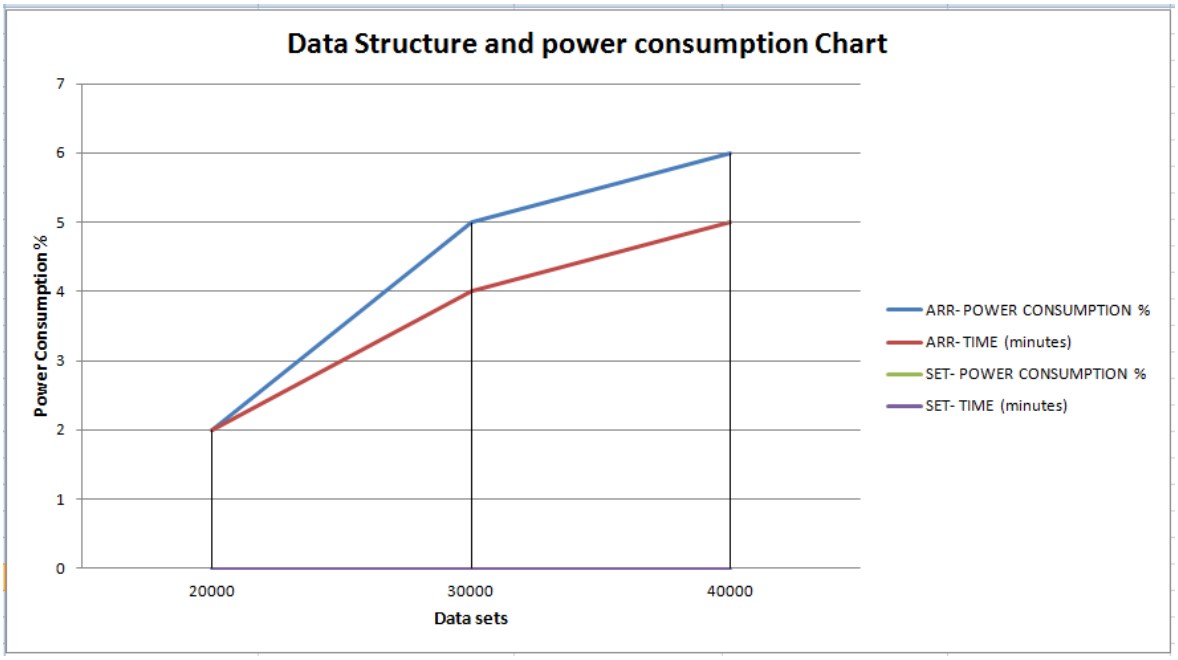

Figure 4.9: Various data representations and power consumption.

4.7.5 Reusing Data Objects.

For our experiment we will try to explore if reusing data objects could have any impact on mobile device battery life, as known one of the most performance issues of an application is the creation and the cancelation of an object within the object life cycle. Our experiment will try to explore if reusing object could save some power in the mobile devices. In our experiment we created two applications, the first application is a grid table view that uses the same object while scrolling, that is it will not create new object within the same grid view, while on the second application, the grid table view creates new object while scrolling, the aim is to investigate the differences in terms of power consumption between the two cases. The collected results are illustrated in graph Fig (4.10) and the results show significance difference of power consumption between the two cases. From the collected data of the experiment it is shown that reusing objects consumes less power in mobile device compared to creation of new object at a time.

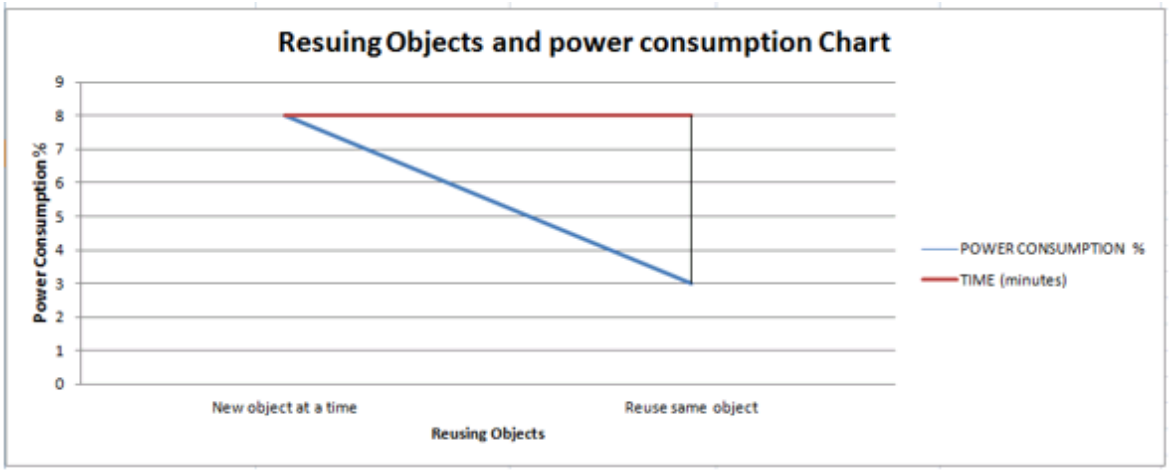

Figure 4.10: Reusing objects and power consumption.

4.8 Validation of the Results

Evaluation and Validation of the results are based on the following points:

1. Our thesis extended and generalized the work of (Abdul Elminaam et al. 2009); in this work, the authors evaluated six of the most commonly used encryption security algorithms, namely: AES, DES, 3DES, RC2, Blowfish and RC6. Their study focused on wireless devices where they compared and evaluated the algorithms against different factors, such as power consumption and CPU overhead. They implemented different setup of various algorithms in order to evaluate the algorithms: such as different sizes of data blocks and different types of algorithms. Their results concluded that using different types of algorithms have significant impact on power consumption of wireless devices.

   In our research, we selected a more generalized set of algorithms; we implemented our experiments on a set of different time complexities of *O(logn), O(n), O(nlogn), O(n^2), O(n^3) and O(n4)*. With each time complexity we implemented different scenarios, (4 scenarios), with different settings of each algorithm; we used the same measurement methodology that was adopted by (Abdul Elminaam et al. 2009); the authors used two methods to measure the power consumption: First by readings from the actual device and second by using the cycles, the operating voltage of the CPU and the average current drawn for each cycle. We followed up the same methodology in order to collect the results and get consistent results. In addition to that, we also explored other set of factors that could have impact on the battery

lifetime on smartphone devices such as scope of variables (global vs. local), reusing data objects and programming style.

2. Our results show that using different settings of algorithms with different time complexities have significant impact on the battery lifetime and CPU overhead. This is consistent with several previous studies (such as Larsson 2010), which have also recommends that using different settings of algorithms' complexities have impact on the power performance. Also researches reported that using different data structure definitions also affects the performance. In our thesis we proved by experiment and using different types of algorithms and different data structures definitions that these have impact on battery lifetime as well, for example we have proved that simple linear array-based structures yields less power consumption and thus less CPU cycles than a linked list; this also consistent with previous studies which found a linear relationship between the data structure and algorithm complexity versus CPU overhead.

   Also according to (Pathak et al. 2012), they suggested and recommended to focus on the software coding phase to solve the issue of power consumption on Smartphone devices; they reported that Smartphone components have presented a shift in the traditional way of programming to what they have called "power encumbered programming", and that this shift places a significant burden on application developers to deal with the power issue. The authors evaluated certain common applications in terms of power consumption, such as Facebook, Google Maps, Email apps and others. They evaluated the algorithms in what is called no-sleep bugs or power bugs. And they defined these bugs as a mistake during the

coding phase of the Smartphone application, which will lead to severe and extensive power drain of the device. They have identified certain programming bugs in common applications that will lead to significant power drain and these should be taken into account and tested during the coding phase.

Our thesis followed the same approach of (Pathak et al. 2012), by working at the software portion and presenting useful guidelines to the Smartphone application developers, in order to take into consideration at the design and coding phase of the Smartphone. According to the authors this bugs could be as simple as a loop bug, where a routine calling itself by mistake, or could be complicated for instance a code that not handles unexpected external events, such as a remote server crash or any other event, and thus draining battery as a result of the repeated tries.

Validation of the results (Comparison):

In order to further validate the results, the thesis will make a comparison between our findings and results generated by (Abdul Elminaam et al. 2009); as stated previously, the authors evaluated six of the most commonly used encryption security algorithms, namely: AES, DES, 3DES, RC2, Blowfish and RC6. And that their study focused on wireless devices.

The authors did not take the time complexity as a point in their study, their work was to evaluate the six encryption security algorithms in terms of power consumption and CPU overhead by changing the setup of the algorithms, such as, by passing text file, audio file, video file and other settings.

In order to have a comparison environment between our findings and theirs, we explored the time complexities of the six encryption security algorithms, we have looked into

several resources such as (Mathur and Kesarwani 2013), and other resources over the internet, and we found that all these encryption security algorithms have a time complexity of *O(n),* and that these algorithms should be fast and should have a high performance, and that is due to their tasks, that is they need to be of high performance as they are used to encrypt data over wireless devices, or for exchanging data and other procedures. And of course the performance of different algorithms is different according to data loads.

For instance, Blowfish is known for its quite slow key schedule (which takes as long as encrypting about 4 KB of data), but this is still *O(n).* So in short all algorithms six algorithms have time complexity of *O(n),* but in order to validate the authors results with our results we did the following:

1. We did an assumption so as to compare and validate the data.
2. The assumption is the following: As the six security algorithms have a time complexity of *O(n),* we will do an assumption to change the time complexity of the security algorithms to *O(nlogn), O(n^2), O(n^3),* so as to compare them with our findings.
3. For instance, let us take Blowfish that have a time complexity of *O(n),* we can change it to *O(n^2),* by adding a *for loop* from *1..N* (times the Blowfish function) which results in a complexity of *O(n^2).* The same applies for *O(n^3),* by adding two nested loops from 1..N (times the times the Blowfish function).
4. Now let us take the results for (Abdul Elminaam et al. 2009), from one of the setups that they have implemented, as mentioned previously they changed the setup to read text file, video files. For validation we will take the data for the text file, (we can consider it as best case scenario) Table (4.6), and video files from which we can consider as worst case scenario) Table (4.7) presents their data together with the three amended time complexities.

Table 4.6: (Abdul Elminaam et al. 2009) results and amended time complexities – Text file

Algorithm	Original Big(O)	Power Consu-mption%	Amended Big(O)	Power Consumption %	Amended Big(O)	Power Consumption %	Amended Big(O)	Power Consumption %
AES	*O(n)*	*0.005*	*O(nlogn)*	*logn*0.005*	*O(n^2)*	*N*0.005*	*O(n^3)*	*N*N*0.005*
DES	*O(n)*	*0.005*	*O(nlogn)*	*logn*0.005*	*O(n^2)*	*N*0.005*	*O(n^3)*	*N*N*0.005*
3DES	*O(n)*	*0.006*	*O(nlogn)*	*logn*0.006*	*O(n^2)*	*N*0.006*	*O(n^3)*	*N*N*0.006*
RC2	*O(n)*	*0.006*	*O(nlogn)*	*logn*0.006*	*O(n^2)*	*N*0.006*	*O(n^3)*	*N*N*0.006*
Blowfish	*O(n)*	*0.001*	*O(nlogn)*	*logn*0.001*	*O(n^2)*	*N*0.001*	*O(n^3)*	*N*N*0.001*
RC6	*O(n)*	*0.003*	*O(nlogn)*	*logn*0.003*	*O(n^2)*	*N*0.003*	*O(n^3)*	*N*N*0.003*

Table 4.7: (Abdul Elminaam et al. 2009) results and amended time complexities – Video file

Algorithm	Original Big(O)	Power Consu-mption%	Amended Big(O)	Power Consumption %	Amended Big(O)	Power Consumption %	Amended Big(O)	Power Consumption %
AES	*O(n)*	*0.007*	*O(nlogn)*	*logn*0.007*	*O(n^2)*	*N*0.007*	*O(n^3)*	*N*N*0.007*
DES	*O(n)*	*0.007*	*O(nlogn)*	*logn*0.007*	*O(n^2)*	*N*0.007*	*O(n^3)*	*N*N*0.007*
3DES	*O(n)*	*0.007*	*O(nlogn)*	*logn*0.007*	*O(n^2)*	*N*0.007*	*O(n^3)*	*N*N*0.007*
RC2	*O(n)*	*0.007*	*O(nlogn)*	*logn*0.007*	*O(n^2)*	*N*0.007*	*O(n^3)*	*N*N*0.007*
Blowfish	*O(n)*	*0.001*	*O(nlogn)*	*logn*0.001*	*O(n^2)*	*N*0.001*	*O(n^3)*	*N*N*0.001*
RC6	*O(n)*	*0.003*	*O(nlogn)*	*logn*0.003*	*O(n^2)*	*N*0.003*	*O(n^3)*	*N*N*0.003*

Based on the above data and results from (Abdul Elminaam et al. 2009), and based on the assumption implemented, we can figure out the linear relation generated, linear relation is obvious were the ranking in terms of power consumption is as follows: *O(n)*, *O(nlogn)*, *O(n^2)* and finally *O(n^3)*, these results are aligned with our results presented in Fig (4.4).

Chapter Five – Conclusion and Future Work

5.1 Conclusion

Battery life time is a limited resource in mobile devices – based on studies (Yao and Durant 2009, 25) that have shown batteries have slowly improved their capacities at a rate of only few percent per year and they are expected not to increase as time moves compared to computing and communication filed. Hence, this thesis presents an effective approach to evaluate the performance of algorithms having different data structure representations and having different time complexities against the Smartphone battery life time. We have developed and conducted several experiments to evaluate the performance. Our results clearly show a difference in power consumption in various scenarios. In this thesis, we have presented and discussed experiments in detail and offered suggestions and concepts for the software designers to substantially minimize the expected energy consumption in the mobile device. In the experiments we used algorithms that have time complexities of *O(logn),O(n), O(nlogn), O(n^2), O(n^3),* and *O(n^4) with neutral case as best cases, with real cases, with different data structure representations* namely array based algorithms and linked-list based algorithms. We amended data types on algorithms and added data objects to algorithms as well. Results show that algorithms having time complexities of order n3 with neutral case as best case scenario come at the top level of consuming battery life time and the top level of CPU overhead. Order n2 algorithms come at the second level of consuming battery life and CPU cycles. Order (nlogn) algorithms come next, and so on. Regarding data structure representations, the algorithms can be ranked from high (power and CPU time consumption) to low: Doubly linked-list, singly linked-list, and array based algorithms.

Finally, the thesis explored other factors as well, which also have impact on battery levels. These include: Programming style (global and local variables), device orientation, device accelerometer (change rate), and various other representations such as arrays and sets and reusing data objects.

5.2 Future Work

Given the fact that huge IT sectors are moving towards designing mobile applications on Smartphone devices, and given the fact that power management is a crucial factor in the mobile software development, and the mobile industry at large; it is significant that software developers should start designing applications that are more power efficient and battery-friendly. So we think that more data structure representations should be explored and have experiments applied to investigate further in this field. Similarly, other power-draining factors (in addition to CPU overhead) should be considered like radio-transmission and screen illumination.

References

1. Abdelmotalib, A. and Wu, Z. (2012): “Power Consumption in Smartphones (Hardware Behaviourism)”. In: International Journal of Computer Science Issues, Volume 9, No.3.

2. Abdul Elminaam, D., Abdul Kader, H., Hadhoud, M. (2009): “Performance Evaluation of Symmetric Encryption Algorithms on Power Consumption for Wireless Devices”. In: International Journal of Computer Theory and Engineering, Volume 1, No.4.

3. Bunse, C., H¨opfner, H., Mansour, E., Roychoudhury, S. (2009): “Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments”. In: Mobile IEEE.

4. Kamthane, A. (2007): *Introduction to Data Structures in C*, First Edition. Pearson Education India.

5. Kochan, S. (2003): *Programming in Objective C*, First Edition. Addison-Wesley Professional Pub.

6. Larsson, P. (2010): “Energy-Efficient Software Guidelines”. In: Intel Software Solutions Group- White paper.

7. Maloney, S. and Boci, I. (2012): “Survey: Techniques for Efficient energy consumption in Mobile Architectures”. In: Wireless and Mobile Networking Conference, IEEE, Toulouse.

8. Nakov, S. andKolev, V. (2013): *Fundamentals of Computer Programming with C#*. Faber, Veliko, Tarnovo.

9. Pathak, A., Hu, Y., Zhang, M. (2011): “Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices”. In: Hotnets ’11, USA.

10. Pathak, A., Hu, Y., Jindal, A., Midkiff, S. (2012): “What is keeping my phone awake? Characterizing and Detecting No-Sleep Energy Bugs in Smartphone Apps”. In: MobiSys’12, UK.

11. Sherrod, A. (2007): *Data Structures and Algorithms for Game Developers*, First Edition. Course Technology PTR.

12. Sinha, A. andChandrakasan, A. (2001): “Joule Track A Web Based Tool for Software energy Profiling”. In: Proceedings of the 38th Design Automation Conference, USA, 220-225.

13. Tien, M. and Lee, C. (1996): “Instruction Level Power Analysis and Optimization of Software”. In: Journal of VLSI Signal Processing.

14. Vo, K. (2011): *Pro iOS Apps Performance Optimization*, First Edition. Apress.

15. Wasserman, A. (2010): “Software Engineering Issues for Mobile application Development”. In: ACM.

16. Wei, S. and Naik, K. (2001): “Software Implementation Strategies for Power-Conscious Systems”, Mobile Networks and Applications, 291-305.

17. Yao, P. and Durant, D. (2009): *Programming .NET Compact 3.5*, Second Edition. Addison-Wesley Professional Pub.

18. Zhang, L., Tiwana, B., Dick, R., Qian, Z., Mao, Z., Wang, Z., Yang, L. (2010): “Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones”. In: CODES+ISSS’10, USA.