Minesweeper as a Tool of Probability

Gautham B A
Branch-Dept. of Computer Science
College-PES Institute of Technology, BSK 3rd stage,
Bangalore, India

***Abstract*—Minesweeper can be used as a tool for experimenting on probability. The computer guesses a location. The user is asked to pinpoint that location by clicking on the display area. The user has to determine the location guessed by the computer within a stipulated number of clicks. Thus, the probability of determining the location guessed by the computer can be calculated. Several data structures like stack, queues, priority queues, circular doubly linked lists are used. To give the "game" perspective of the proposed method, it has a multiplayer support and also a provision for chatting between the 2 parties. This work tries to show how Minesweeper could be used as a tool for measuring probability, with some changes done to the rules of the trivial minesweeper game.**

Keywords—probability; ServerSocket; Socket Programming; GUI ; Stacks; Queues; Priority Queues; Circular Doubly Linked List; Server; Client; Object Serialization, encryption, Network security;

I. INTRODUCTION

Probability is not the science of foretelling the future, rather, it says how likely the event can occur. To be more precise, probability takes into account all the possible events that might occur and tells how likely each particular event can occur. The likelihood of the event that can occur is expressed as percentage.

The proposed method employs Probability extensively. It tries to show how Minesweeper can be used as a tool for measuring probability. It tells how likely the user could have selected a location that was chosen by the computer as the mine as shown in fig(6).

The proposed method is also advertised as a guessing game. The computer places a mine(bomb) at a random location and the player is expected to find the location where the mine is placed with the aid of a signal detector(Signal bar as shown in fig(8)). As and when the user's click nears the location of the mine selected by the computer, the signal on the signal bar grows. And finally, the mine is disarmed when the user succeeds in locating the randomly chosen location by the computer. There are certain constraints for the user-time and the no. of clicks.

Coming to the technical aspects of the proposed method, it makes an extensive use of Data Structures namely, Stacks, Queues, Priority Queues, Circular Doubly Linked List. It also makes a remarkable use of the network classes offered by Java-ServerSocket and Socket.

A stack or LIFO (last in, first out) is an abstract data type that serves as a collection of elements, with two principal operations:

1. "push" adds an element to the collection.
2. "pop" removes the last element that was added.

The term LIFO stems from the fact that, using these operations, the last element "popped off" a stack in series of pushes and pops is the first element that was pushed in the sequence. This is equivalent to the requirement that, considered as a linear data structure, or more abstractly a sequential collection, the push and pop operations occur only at one end of the structure, referred to as the top of the stack.[1]

A queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. Often a peek or front operation is also entered, returning the value of the front element without dequeuing it. A queue is an example of a linear data structure, or more abstractly a sequential collection.[2]

A doubly-linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Each node contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes. The beginning and ending nodes' previous and next links, respectively, point to some kind of terminator, typically a sentinel node or null, to facilitate traversal of the list. If there is only one sentinel node, then the list is circularly linked via the sentinel node. It can be conceptualized as two singly linked lists formed from the same data items, but in opposite sequential orders.[3]

A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.[4]

Queues are used to establish connection among the clients in the multiplayer mode of the game. Queues are also used to store the conversation that happens between the clients in a chat box and the conversation is restored back to the chat boxes when the 2 of them connect again as shown in the *Chat window* in fig(7).

Stacks are used to keep track of the locations that are clicked by the user as shown in fig(9).

Circular Doubly Linked List is used to encrypt the user's password as shown in fig(5).

Priority Queue is used to maintain a list of the high scores of all the users in a descending order.

All the objects(like stack object, queue object etc.) are written to files by serializing them with the help Object Serialization.

Working of the game is detailed as follows-There is a server called *GameComServer* which runs on any of the computers connected to a network. The clients are run on other computers connected to the same network and the connection to the server is established by providing the IP address of the computer running the *GameComServer* as shown in fig(4). The user has to create his account(or login if he already has) before playing the game as shown in fig(4).

The server keeps track of the highest score of all of the users separately. All the high scores are put together in a Priority Queue and thus a descending order of the scores is obtained.

The paper is organized as follows 1. Introduction 2. Literature survey 3. Ease of use 4. Design and implementation 5. A note on object serialization 6. Data Structures 7. Results 8. Conclusion. 9. References.

II. LITERATURE SURVEY

1. <u>*Minesweeper game:*</u>

Minesweeper is a single-player puzzle video game. The objective of the game is to clear a rectangular board containing hidden "mines" without detonating any of them, with help from clues about the number of neighboring mines in each field. The game originates from the 1960s, and has been written for many computing platforms in use today. It has many variations and offshoots.[6]

The proposed method shares the name of the most popular game called Minesweeper. Yet, it is quite different, even from the fundamental aspects except for the aim of the game which is to find the mine. The minesweeper game neither provides any information about the probability that persisted at the time of locating the mine nor does it provide any information that could be made use of to ascertain the probability. The proposed method on the other hand is a tool for determining the probability. It lets the user know about the probability in the form of the score that the user gets at the time of locating the mine as shown in fig(6) (The probability in the figure is expressed as percentage).

2. <u>*Minesweeper: A Statistical and Computational Analysis*</u>

How hard is it to win the game Minesweeper, and what algorithms can be developed to play it efficiently? This question is interesting because it allows us to examine a problem in Linear Algebra, a branch of mathematics, in a fun an interesting way. This is because the question of playing the game Minesweeper can be shown to be equivalent to solving a certain math problem. This means that the results that this paper has found are applicable not only for players of the game but to mathematicians who study certain problems in Linear Algebra.[7]

The paper[7] tries to bring out the computational aspect that is involved in the Minesweeper game by dealing with Linear Algebra. It gives strategies to win the game via a Mathematical approach.

The proposed method is different from [7] as it is a tool which doesn't try to solve any problem but answers the question of the probability that was residing at the time the mine was located as shown in fig(6).

3. <u>*2*n Minesweeper Consistency Problem is in P*</u>

Minesweeper is a popular single player game included in Window OS. Since Richard Kaye proved that "Minesweeper is NP complete" in 2000, it has been studied by many researchers. Meredith Kadlac had shown that 1D Minesweeper consistency problem is regular and can be recognized by a deterministic finite automaton. This consistency problem is extended to 2xn Minesweeper which is 2D but with its one dimension restricted to 2. This problem is tractable and design a finite automaton which can solve 2xn Minesweeper consistency problem in linear time.

The paper[8] tries to show that 2xn Minesweeper consistency problem is also in P.

The proposed method is different from that proposed in paper[8] as it does not deal with determinism and does not try to construct an automaton. The proposed method tries to reveal the probability of disarming the mine at the last click as shown in fig(6).

4. *Other tools for probability:*

   A preponderance of tools for experimenting and determining probability can be found on internet like

   http://www.shodor.org/interactivate/activities/ExpProbability/

   http://stattrek.com/online-calculator/binomial.aspx

   Yet, the proposed method is unique and is quite different from all of them as it employs a very unique way of determining the probability which is described further in the paper.

   Its very manifestation is different from all other tools. It serves the purpose of a game (fig(8)) for those who seek an ephemeral entertainment and as a tool for those who want to experiment with fun.

III. EASE OF USE

A. *Usage*

1. The proposed method can be a very essential tool for those who want to experiment on Probability.
2. For those who seek entertainment, the proposed method still holds good because it manifests itself to the user as a guessing game.
3. It even includes a multiplayer mode. So, any number of people can connect and play against each other.
4. The entertainment aspect is enhanced as there is a provision for people who are playing in the multiplayer mode to chat with each other with the help of an elegant and robust chat box as shown in fig(7).

B. *System Requirements*

1. Any computer running Java.
2. If 2 or more computers are used, they must be connected to the same network.

IV. DESIGN AND IMPLEMENTATION

The proposed method is implemented entirely in Java and makes an extensive use of Data Structures and Computer Networking tools offered by the Java classes-ServerSocket and Socket.

The project can be divided into 2 parts-Server and the client. The server can run on any computer which is connected to a network. When the client program is run from a different computer connected to the same network, the user at the client side is requested to create his/her account by providing a username and password as shown in fig(4). The user can login on successful creation of the account.

The user can now explore the various features of the proposed method. The single player mode requires the user to locate the mine which is selected randomly by the computer and whose location isn't revealed to the user. The user has to locate the mine by randomly clicking at several places on the screen. A progress bar which functions as a signal detector calculates the distance between the mine(a location chosen by the computer) and the randomly clicked location by the user and expresses it as percentage in the progress bar as shown in the *Signal bar* in fig(8).

The user is also subjected to several constraints. The user has to locate the mine within 60 seconds which is kept track by the timer as shown in fig(8) and also, the number of clicks that he can make is limited.

In multiplayer mode, the one who would like to host the game sends his request to the server along with this IP address. His request is added onto a queue. Now, another person who wants to join a game sends his request to the server. The server checks if the queue is not empty. If so, the server replies to the recent user by send him the data(IP address) that was previously put onto the queue. Now, a connection between 2 people is established and they can start playing the game against each other.

The score that is rewarded is the product of probability that is expressed as percentage and a factor of the number of clicks that remain. It is a factor of probability of locating the mine at the n^{th} click. The probability is first computed and expressed as percentage as shown in fig(7).

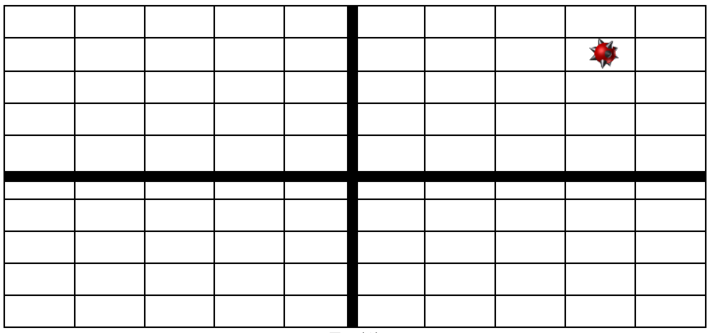
Each and every time the user runs the project, the probability (or rather, the score) is sent to the server. The server uses a priority queue so as to arrange the incoming scores in a descending order.

A. *Equations*

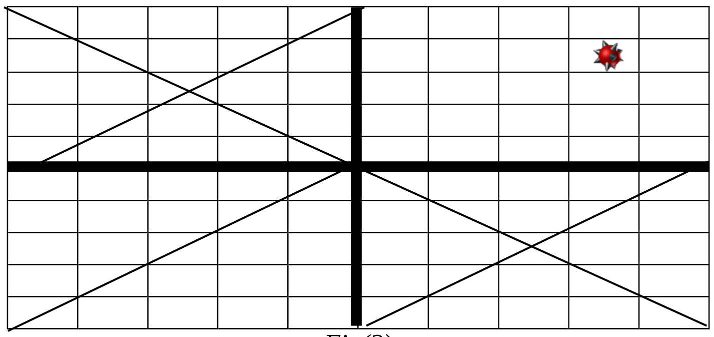
BOXES/=FACTOR;

SCORE=1/(BOXES*(MAX_CLICKS-CLICKS));

B. *Calcutaing the probability-*

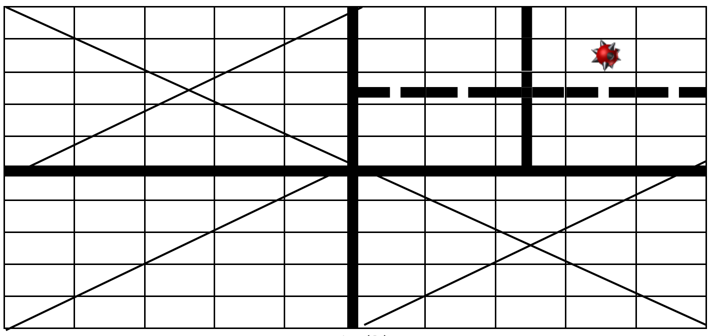


Fig(1)
10x10 boxes in which the mine is in (9,2) with top-left box as origin.



Fig(2)

After a click if the mine isn't located, 3/4th of the boxes are discarded. If the user had clicked in the quadrant where the mine is present, The progress bar(indicator of proximity) will show a value greater than 50% and hence the user can get to know that he is more probable to locate the mine since he has just now located the quadrant in which the mine is present.



Fig(3)

Upon the next click, the first quadrant is further divided into 4 sub-quadrants and the algorithm repeats until either the user locates the mine or he runs out of chances.

1. *The screen is divided to* n *number of boxes as shown in fig(1).*
2. A suitable factor *t* is chosen.
3. At each click, *n/t* boxes are discarded. Here, rather than just clicking around randomly, the user is required to make an educated guess, judging on the value displayed by the progress bar. If the progress bar displays a value greater than 50% then the user is assured that he is in the right quadrant. If the value displayed on the progress bar is lesser than 50%, the user should realize that he has not picked the quadrant in which the mine lies and has to make the further clicks cautiously by not considering the current quadrant for his next moves.
4. e.g. Let us assume that there are 4 quadrants. This implies that the FACTOR=4, i.e. *t=4*. If the user clicks on the first quadrant and the progress bar displays its reading to be more than 50%, the user is well assured that the mine is located somewhere in the first quadrant. The dynamics of the game anticipates that the user should choose a location well-within the first quadrant as his next click. So, the other 3 quadrants containing n*(3/4) boxes are discarded.

V. A NOTE ON OBJECT SERIALIZATION

Fate of an object:

When an object comes to life, memory is allocated for it and each object will have a copy of all the attributes and the methods. It has a definite structure. But when the object dies (when the program ends), the memory set aside for it will be de-allocated and the object no longer exists.

But the fate of an object can be altered with the help of an advanced concept called Object Serialization.

With the help of class ObjectOutputStream, the "live object" can be written into a file as it is. It maintains its structure along with the data stored in its attributes along with the methods that are associated with it. So, the object continues to exist even after the program ends.

With the help of the class ObjectInputStream, the dead object can be revived back to life. Which means, the object that is written into a file can be revived by allocating the same memory space that was allocated to it earlier.

So, by using Object Serialization to write the objects of Data Structures into memory, efficient storage can be achieved because the object maintains its structure along with the data when it is written.

This concept of serializing an object is used for storing the live objects of a stack containing the clicked locations, object of queue of conversation between the 2 parties, object of a list of the usernames of all the users, object of doubly linked list of their passwords and object of priority queue having the highest score along with the username in the descending order.

VI. DATA STRUCTURES

The data structures used in this game are stacks, queues, circular doubly linked list and priority queue.

1. Stacks:

When the user clicks a location, it is pushed onto the stack. All the locations are popped out and is displayed on the screen at the end of the game.

2. Queues:

a. In the GameComServer, 2 queues are maintained. One for the bomber and the other for the disarmer. When the player requests a connection, his IP address (the port also in case of bomber) is enqueued onto to the queue. The queue is dequeued when the 2 opposite parties wish to connect.

b. The conversation as shown in fig(7) between the bomber and the disarmer is facilitated with the help of a queue. Each message sent by any one of them is enqueued onto a queue that stores the conversation between the two. The same conversation again shows up if both of them reconnect later when they play the game.

***3.** Circular doubly linked list:*

It is used to encrypt the password of the user. The algorithm to encrypt is as follows.

<u>*Encryption Algorithm:*</u>

a. At the time of registration of the account, when the user keys in his password, each character is populated into a circular doubly linked list.
b. A random number lesser than the length of the password is generated.
c. Each character of the password is shifted to the left as many times as that of the random number generated.
d. The random number that is generated is inserted at the head of the circular doubly linked list as shown in fig(5).

The algorithm to decrypt is as follows.

<u>*Decryption Algorithm:*</u>

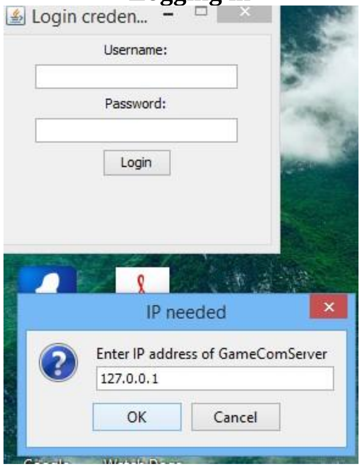
a. When the user tries to log in, his password is retrieved by the server.
b. The number (key) is removed.
c. The list is shifted to the right as many times as the key.

***4.** Priority queue:*

The priority queue is used to maintain a sorted list of the high scores. Each time the user plays the game, his score is sent to the server. The server then fetches the score corresponding to the username and updates his score only if the new score is greater than the previous score. Therefore, the priority queue aids in maintaining a list of high scores in a descending order.

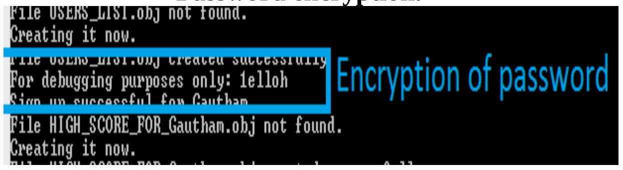
VII. Results

Logging in



Fig(4)

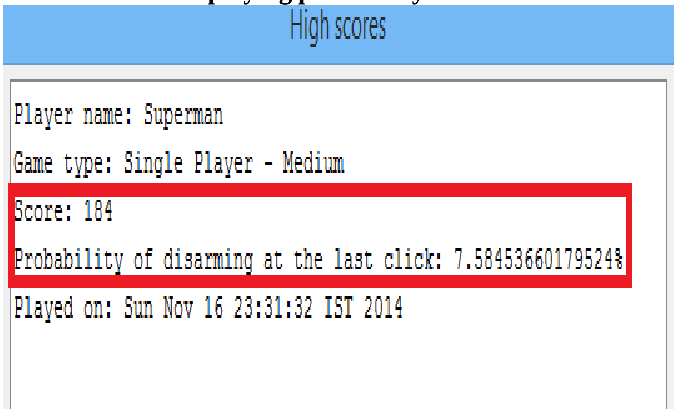
Password encryption.



```
File USERS_LIST.obj not found.
Creating it now.
File USERS_LIST.obj created successfully
For debugging purposes only: 1elloh
Sign up successful for Gautham
File HIGH_SCORE_FOR_Gautham.obj not found.
Creating it now.
```

Fig(5)

Displaying probability

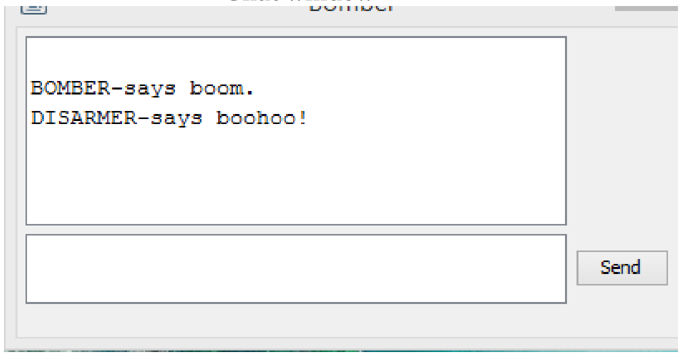


```
Player name: Superman
Game type: Single Player - Medium
Score: 184
Probability of disarming at the last click: 7.58453660179524%
Played on: Sun Nov 16 23:31:32 IST 2014
```

Fig(6)

(The probability as shown in fig(6) is expressed in percentage)

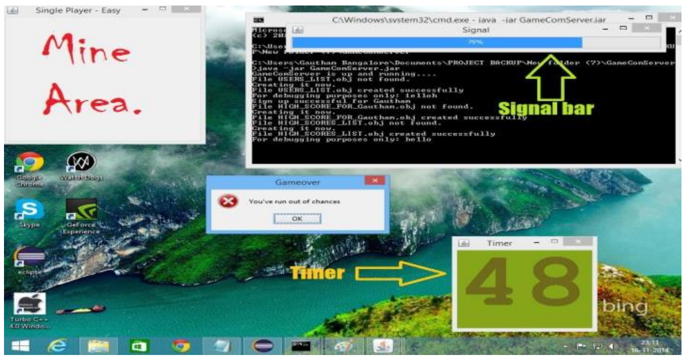
Chat window



```
BOMBER-says boom.
DISARMER-says boohoo!
```

Fig(7)

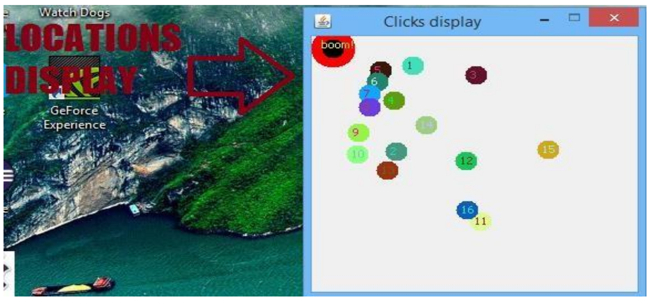
Gameplay.


Single Player - Easy
Mine Area.
C:\Windows\system32\cmd.exe - java -jar GameComServer.jar
Signal
Signal bar
Gameover
You've run out of chances
OK
Timer
Timer
48

```
C:\Users\Gautham Bangalore\Documents\PROJECT BACKUP\New folder (7)\GameComServer
>java -jar GameComServer.jar
GameComServer is up and running....
File USERS_LIST.obj not found.
Creating it now.
File USERS_LIST.obj created successfully
For debugging purposes only: 1elloh
Sign up successful for Gautham
File HIGH_SCORE_FOR_Gautham.obj not found.
Creating it now.
File HIGH_SCORE_FOR_Gautham.obj created successfully
File HIGH_SCORES_LIST.obj not found.
Creating it now.
File HIGH_SCORES_LIST.obj created successfully
For debugging purposes only: hello
```

Fig(8)

Displaying locations of the clicks.



Fig(9)

VIII. CONCLUSION

The proposed method serves as a powerful tool for experimenting probability and it provides a very accurate measure. The proposed method can be used as a framework to enhance *Network Security*. Referring to fig(1), the password or the encrypted text in general, can be placed in the location where the mine is placed. From the security point of view, the hacker should not be able to locate the mine(the password in this case). Only the legitimate user who knows the password will know the location of the password. Thereby, we can measure the probability of the hacker decrypting the password with the help of the proposed method. The analogy between *Network Security* and the proposed method is that, only the legitimate user can locate the mine but the hacker may/may not locate the mine and the number of attempts that he makes could be treated as invalid clicks in terms of the proposed method. People can have fun by experimenting and indulging themselves in the mystery of probability.

The proposed method can further be improved by introducing more advanced concepts in probability. Other data structures could be used to improve the performance. The game aspect of the proposed method can by embellished by enhancing the security feature. The game’s aesthetic appeal can be magnified by using more robust classes that Java offers for game design.

IX. REFERENCES

[1] http://en.wikipedia.org/wiki/Stack_(abstract_data_ type)

[2] http://en.wikipedia.org/wiki/Queue_(abstract_data _type)

[3] http://en.wikipedia.org/wiki/Doubly_linked_list

[4] http://en.wikipedia.org/wiki/Priority_queue

[5] Michael T. Goodrich, Roberto Tamassia-Data structures and algorithms with Java, 3rd edition.

[6] http://en.wikipedia.org/wiki/Minesweeper_(video_ game)

[7] Minesweeper: A Statistical and Computational Analysis by Andrew Fowler Andrew Young.

[8] 2*n Minesweeper Consistency Problem is in P by Shu-Chiung Hu and Shun-Shii Lin.