# Chapter 1

# INTRODUCTION

File is a data structure on secondary storage, which acts as a non-volatile container for data. File is a name given to any kind of document stored in any type of storage device, which can be reached by the computer. A file is identified by a name followed by a filename extension.

File Structure is a pattern for arranging data in a file. It is a combination of representations for data in files and of operations for accessing the data.

## 1.1 Brief history of File Structures

**Early Work:**

Early work assumed that files were on tape. Access was sequential. The cost of access grew in direct proportion to the size of file.

**The emergence of Disks and Indexes:**

As files grew very large, sequential access was not a good solution. Disks allowed for direct access. Indexes were made it possible to keep a list of keys and pointers in a small file that could be searched very quickly. With the key and pointer, the user had direct access to the large, primary file.

**The emergence of Tree Structures:**

Indexes also have a sequential flavour. When they grow too much, they also become difficult to manage. The idea of using tree structures to manage the index emerged in the early 60’s. However, tress can grow very unevenly as records are added and deleted resulting in long searches requires many disk accesses to find a record.

**Balanced Trees:**

In 1963, researchers came up with the idea of AVL trees for data in memory. However, AVL tress did not apply to files because they work well when tree nodes are composed of single record rather than dozens of hundreds of them. In the 1970’s came the idea of B-Trees which require an O($\log_k$ N) access time where N is the number of entries in the file and k is the number of entries indexed in a single block of the B-Tree structure. B-Tree can guarantee that we can find an entry among millions of others with only 3 or 4 trips to the disk.

**Hash Tables:**

Retrieving entries in 3 or 4 accesses is good but it does not reach the goal of accessing data with a single request. Hashing was a good way to reach this goal with files that do not change size greatly over time. Recently, Extendable Dynamic Hashing guarantees one or at most two disk accesses no matter how big a file becomes.

## 1.2 Design goals of File Structure

Disks are slow. They are also technological marvels, packing hundreds of megabytes on disks that can fit into a notebook computer. Only a few years ago, disks with that kind of capacity looked like small washing machines. However, relative to other parts of computer, disks are slow. The time taken to get information back from even relatively slow electronic random access memory is about 120 nanoseconds or 120 billionth of a second. Getting the same information from a typical disk might take 30milliseconds, or 30 thousandths of a second. The general goals of research.

File Structure is a combination of representations for data in files and operations for accessing the data. It allows applications to read, write and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order. Computer Memory (RAM) is very fast, but limited. Secondary storage such as disks can save thousands of megabytes in a small physical location. However, access to secondary storage is very slow.
For example, getting information from RAM takes about 100 nanoseconds, while getting information from disk takes 30 milliseconds.

# Chapter 2

# Problem Statement

## 2.1 Description

Implement a project, which predicts the number of collisions based on Poisson formula.

## 2.2 What is data collision ?

A data collision may take place when hashing data or when calculating a checksum. A hash function reduces data to a smaller value and is often used in compression and cryptography. While the hash operation may save disk space, it is possible that two different inputs may produce the same output. Multiple hash functions can be used to avoid duplicate values when a collision occurs.

Similarly, checksum are not guaranteed to be unique since they are smaller than the original data. While the probability is often very low, two different data sets can theoretically produce the same checksum value. A well-designed algorithm should minimize this risk.

## 2.3 What is Hashing ?

Hashing is generating a value or values from a string of text using a mathematical function. Hashing is one way to enable security during the process of message transmission when the message is intended for a particular recipient only. A formula generates the hash, which helps to protect the security of the transmission against tampering. Hashing is also a method of sorting key values in a database table in an efficient manner.

# Chapter 3

# Design

To predict the number of collisions occurring in our data set on any applied hash function, we are going to use Poisson's formula.

## 3.1 Poisson Distribution

A Poisson distribution is a statistical distribution showing the likely number of times that an event will occur within a specified period of time. It is used for independent events, which occur at a constant rate within a given interval of time. The Poisson distribution is a discrete function, meaning that the event can only be measured as occurring or not as occurring, meaning the variable can only be measured in whole numbers. Fractional occurrences of the event are not a part of the model.

## 3.2 Poisson Distribution Formula

An event can occur 0, 1, 2, … times in an interval. The average number of events in an interval is designated $\mu$. $\mu$ the event rate, also called the rate parameter. The probability of observing k events in an interval is given by the equation

$$P(k\ events\ in\ interval) = e^{-\mu}\frac{\mu^k}{k!}$$

Where,

- $\mu$ is average number of events per interval
- $e$ is the number 2.71.. (euler's number) the base of the natural logarithm
- $k$ takes values 0,1,2,….
- $k$! = k*(k-1)*(k-2)*….. 2*1 is the factorial of k

## 3.3 Example

Consider the following problem statement. Given N available addresses and r records to be stored, what is the probability of an address getting 0,1,2,,….. records assigned to it ?

Collision occurs when an address is assigned to more than one record, therefore we use Poisson distribution to calculate the number of overflowing records. Lambda=r/N where N is the available addresses and r are the number of records.

Let us consider r=1000 and N=1000

P(0)=0.368

P(1) =0.368

P(2) =0.184

P(3) =0.161

P(4) =0.015

P(5) =0.003

P(6) =0.001

Number of addresses which have 0 records =1000 x 0.368=368

Number of addresses which have 0 records =1000 x 0.368=368

Number of addresses which have 0 records =1000 x 0.184=184

Number of addresses which have 0 records =1000 x 0.061=61

Number of addresses which have 0 records =1000 x 0.015=15

Number of addresses which have 0 records =1000 x 0.003=3

Number of addresses which have 0 records =1000 x 0.001=1

Overflow = (184*1)+(61*2)+(15*3)+(3*4)+(1*5)=368

Therefore out of 1000 records 368 records are over flowing.

# Chapter 4

# Implementation

PROBLEM STATEMENT 1:

Given N available address and r record to be stored what is the probability of an address getting 0,1,2,3…. records assigned to it ? If an address is assigned more than one record, then a collision occurs. .Poisson distribution is used to calculate the number of overflowing record where $\mu$ = R/N. Where R= no. of records stored and N is available address considering 1000 records to be stored in 1000 location.

```
from scipy.stats import poisson
import matplotlib.pyplot as plt
import numpy as np
r=1000
n=1000
mu=r/n
prob=[]
k=list(range(0,7))

for i in k:
    prob.append(round(poisson.pmf(i, mu),4))

y_pos = np.arange(len(prob))
plt.bar(y_pos, prob, align='center',color=['black', 'red',
'green', 'blue', 'cyan'],edgecolor='blue')
#plt.xticks(y_pos, objects)
plt.ylabel('probability')
plt.title('probability of collision')
plt.xlabel('K')

for i in range(len(prob)):
    print('the probability of an address getting',i,'records
assigned-->',prob[i],"\n")
print("--------------------*******----------------------******-
-------------------------")
for i in range(len(prob)):
    print('the number of address which have',i,'records--
>',int(round(prob[i]*1000,0)),"\n")
```

```
sum=0
for j in list(range(2,7)):
    sum=sum+int(round(prob[j]*1000,0))*(j-1)
print("*******************************************")
print('*  Total number if over flowing is ',sum," *")
print("*******************************************")
```

PROBLEM STATEMENT 2:

Analyzing the change in packing density with change in $\mu$.

```
from scipy.stats import poisson
import matplotlib.pyplot as plt
import numpy as np

bar_width=.30
n_group=8
mu1=0.33
mu2=0.59
mu3=1.29
p1=[]
p2=[]
p3=[]
for i in range(0,8):
    p1.append(round(poisson.pmf(i, mu1),6))
    p2.append(round(poisson.pmf(i, mu2),6))
    p3.append(round(poisson.pmf(i, mu3),6))
print("---------------------------------------------------------------
------------")
print("value of K\tp1\t\tp2\t\tp3")
print("---------------------------------------------------------------
------------")

for i in range(0,6):
    print(i,"\t\t",p1[i],"\t",p2[i],"\t",p3[i])


fig,ax=plt.subplots()
index=np.arange(n_group)
rects1=plt.bar(index,p1,bar_width,color='b',label="p1")
rects1=plt.bar(index+bar_width,p2,bar_width,color='g',label="p
2")
```

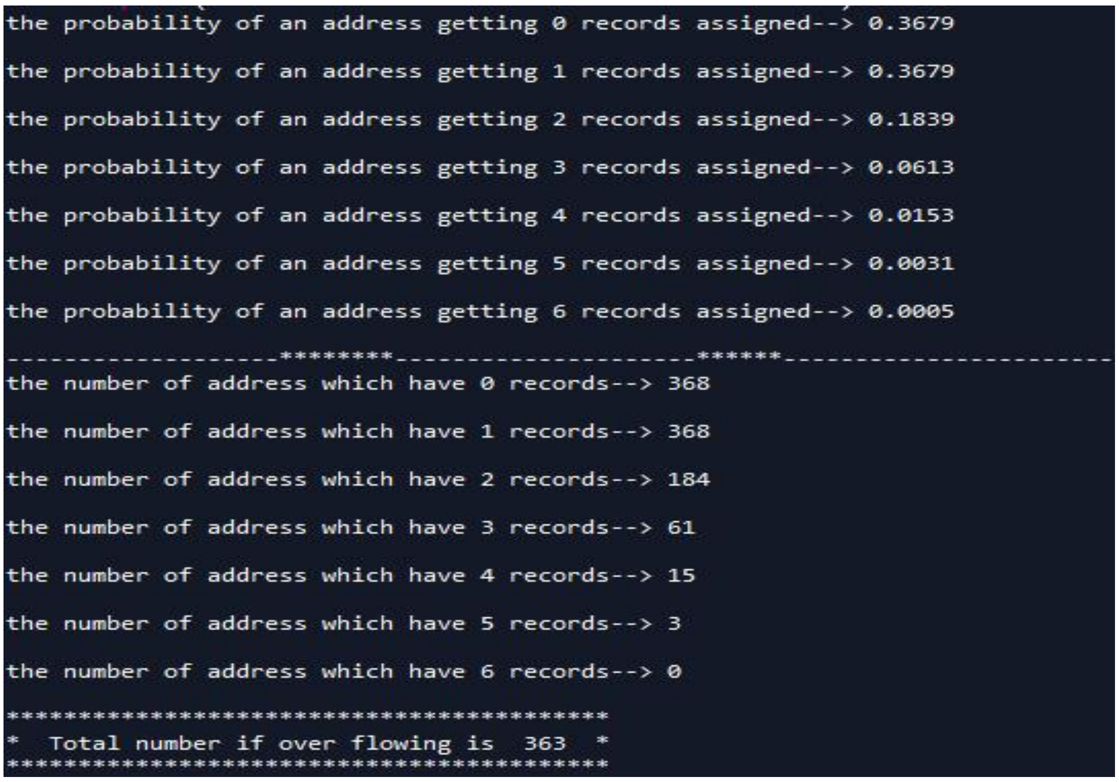
```
rects1=plt.bar(index+2*bar_width,p3,bar_width,color='y',label=
"p3")
plt.xticks(index + bar_width, list(range(0,8)))
plt.ylabel('probability')
plt.title('probability bargraph')
plt.xlabel('K')
```

# Chapter 5

# Results and Analysis

Finding the probability of an address getting K record assigned for problem statement 1.

In addresses that contain only one record, there is no collision. In 184 addresses ,which are assigned 2 records ,only one can be accommodated therefore ,184 records overflow .in 61 addresses only 3 records are assigned, whereas 61 records can be accommodated, but (61*2) records will overflow, so the final no.of overflowing records are (184*1) +(61*2)+(15*3)+(3*4)+(1*5) = 368. Therefore, if 1000 records are stored in 1000 addresses then 368 records are overflow.



```
the probability of an address getting 0 records assigned--> 0.3679

the probability of an address getting 1 records assigned--> 0.3679

the probability of an address getting 2 records assigned--> 0.1839

the probability of an address getting 3 records assigned--> 0.0613

the probability of an address getting 4 records assigned--> 0.0153

the probability of an address getting 5 records assigned--> 0.0031

the probability of an address getting 6 records assigned--> 0.0005

-------------------********---------------------******----------------------
the number of address which have 0 records--> 368

the number of address which have 1 records--> 368

the number of address which have 2 records--> 184

the number of address which have 3 records--> 61

the number of address which have 4 records--> 15

the number of address which have 5 records--> 3

the number of address which have 6 records--> 0

******************************************
*  Total number if over flowing is  363  *
******************************************
```

Fig.1 Calculating probability and total number of over flowing.

Bar graph representation of fig 1

Where k is the variable in x-axis representing no. of records and y-axis, represent probability.

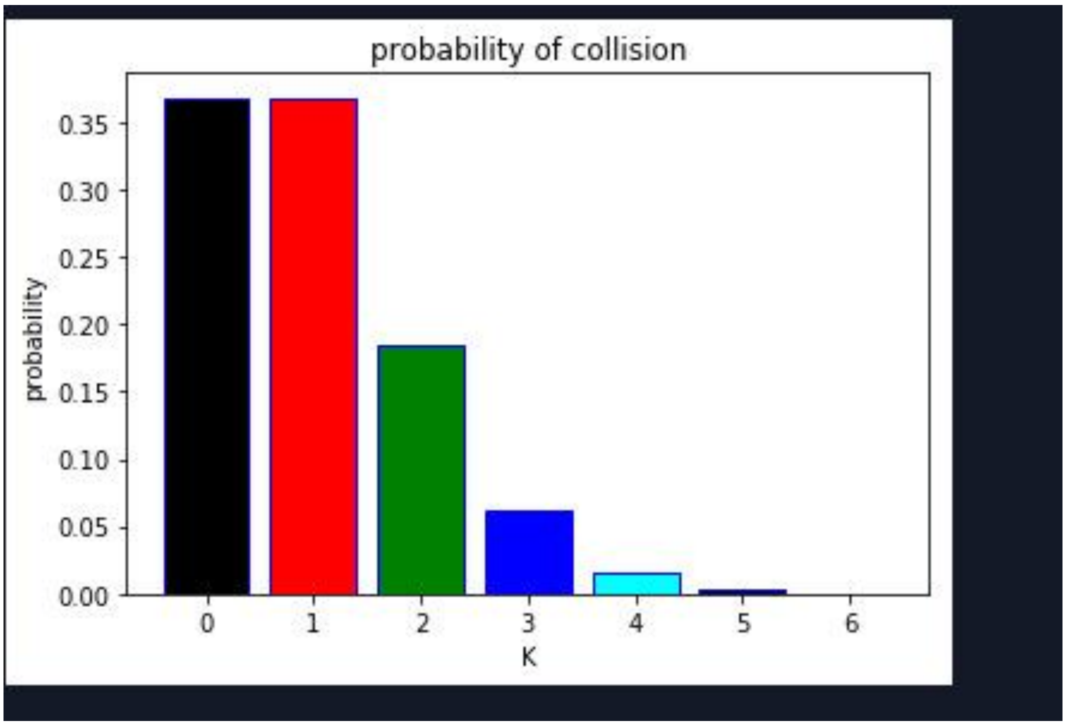


Fig.2 Bar graph

This one show the comparisons among different $\mu$ value, in problem statement 2.

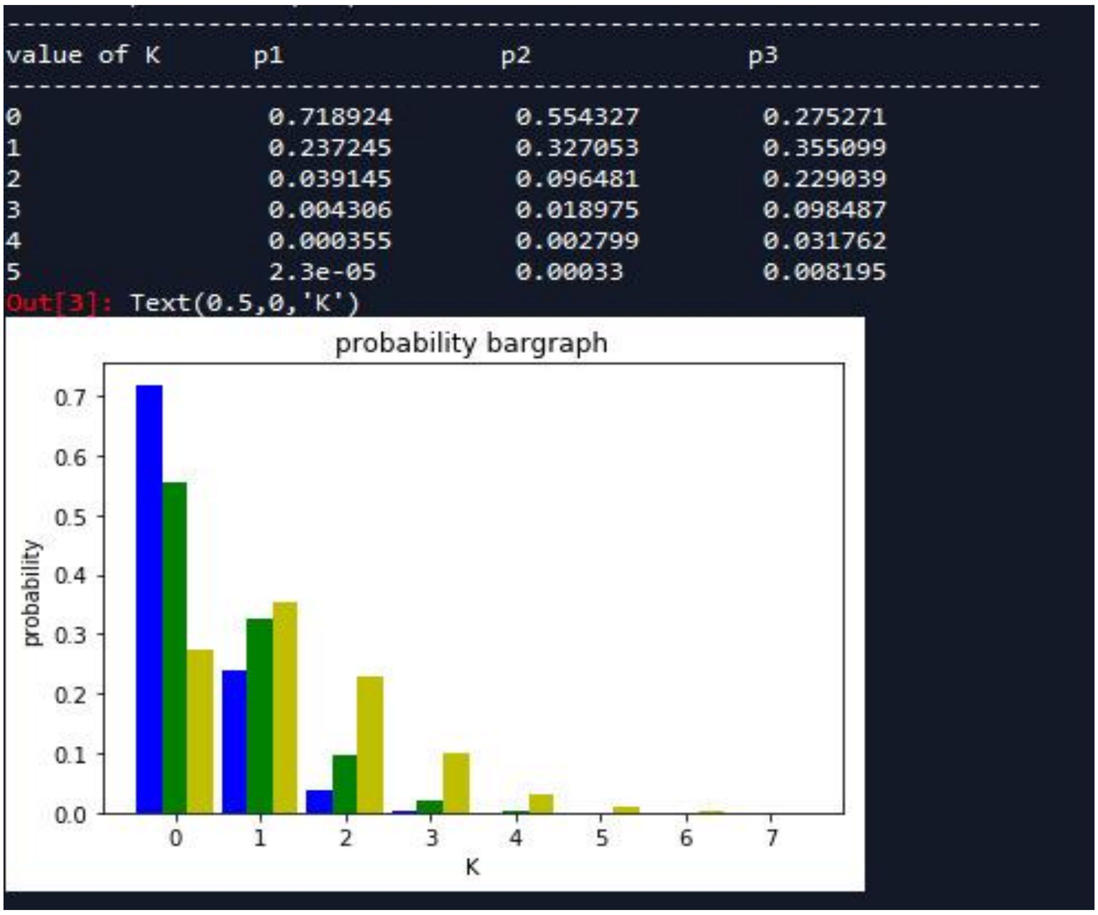
Where k is the variable in x-axis representing no. of records and y-axis, represent probability.

Three different colour used are:

Blue having $\mu1 = 0.33$

Red having $\mu2 = 0.59$

Yellow having $\mu3 = 1.29$



```
-----------------------------------------------------------------
value of K       p1              p2              p3
-----------------------------------------------------------------
0                0.718924        0.554327        0.275271
1                0.237245        0.327053        0.355099
2                0.039145        0.096481        0.229039
3                0.004306        0.018975        0.098487
4                0.000355        0.002799        0.031762
5                2.3e-05         0.00033         0.008195
Out[3]: Text(0.5,0,'K')
```

Fig. 3 Bar graph with change in $\mu$.

# Chapter 6

# Conclusion

Unless your list is static and you have created a perfect hashing function, there will be collisions. Therefore, every hashing algorithm must have a way of dealing with collisions.

It is concluded with the output that when $\mu$ =1 then P (K=0) = P (K=1).
It is also concluded that probability is high when near the mean and concluded that with Change in mean probability also change for any value of K.

From this program we can find out the number of collisions occurring by using Poisson distribution formula and by predicting the number of collisions we can take measures accordingly by developing hash functions that reduce the number of collisions. Collision prediction can be used in real life, as we implemented it in a program where we predict the number of students in a class.

The Poisson distribution may be useful to model events such as

- The number of meteorites greater than 1 meter diameter that strike Earth in a year
- The number of patients arriving in an emergency room between 10 and 11 pm
- The number of photons hitting a detector in a particular time interval

# References

- Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3 rd Edition, Pearson Education, 1998.
- http://faculty.kfupm.edu.sa/ICS/saquib/ICS202/Unit29_Hashing2.pdf
- http://interactivepython.org/runestone/static/pythonds/SortSearch/Hashing.html
- https://www.numpy.org/
- https://www.numpy.org/
- http://matplotlib.org/