E	0	1	1	9	0	5	2

UNIQUE ID

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai

SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

B. Tech. DEGREE

CONTINUOUS ASSESSMENT 4

OCT – 2020: II YEAR – QUARTER 6

CSE 270: Design and Analysis of Algorithm

(Common to B.Tech CSE - AI & ML, CyS & IoT)

Time: 1 hour – 15 minutes **Maximum Mark: 15** **Date: 31/10/2020**

Name: Karan, SathishKumar
Unique Id: E0119039, E0119052

1. Problem Statement:

Design and implement algorithm for online mail merge application. Give detailed report on complexity analysis of the distributed process involved in mail merge application.

2. Programming language: Go(language)

3. Platform:

- Windows,
- Visual Studio Code,
- Go Compiler
- Gmail

A. Deadline for above task: 24-10-2020

4. Description of task:

Mail merge is an essential part of any enterprise application for sending bulk messages. But for a majority of programmers, they use Python, a scripting language to create a performance problem, so, we have created one of the most performant programs using a language developed and used by Google "The Go Lang ". We have also incorporated concurrent programming to perform the mail sending part as it relies on the internet an asynchronous process where the request is sent over the wire to the modem where

Github Repository - https://github.com/karan-vk/Mail-Merge-Go

routing happens and then a DNS server routes the request to a mail server in this case Gmail.

So for a system that allows SMT (Simultaneous multithreading), the program will run concurrently, or the process is serialized to run on a single thread. Nowadays, most computers are multithreaded so, the whole concurrency paradigm will be a non-issue

5. Algorithm development:

Part 1 Creating an email

1. Setting the io related variables
    - csvFile (O(1) declaring a string variable)
    - templateName (O(1) declaring a string variable)
    - fileNameColumn (O(1) declaring a string variable)
    - fileExt (O(1) declaring a string variable)
2. Loading the template for the body of the message (O(1) just a pointer)
3. Loading the csv file for the data of the body of the message (O(1) just a pointer)
4. Setting csv reader delimiter(O(1) only declaration
5. For O(n)
    a. Reading the csv file (O(1) -T O(n)-space n-number of the field in a row)
    b. Creating a new file (O(1) -T O(1) -Space)
    c. Applying the template to the file created (O(1)-T O(n)-S n no. of fields to be filled in the template)

Part 2 Sending the mail

1. Declaring the necessary variables
    a. Files O(1)
    b. Emails O(1)
    c. Channel string O(1)
    d. Root (O(1))
    e. For loop to iterate and store files and emails
    f. Sending mail (O(1) sending the request not requesting whether it was successful)

Github Repository - https://github.com/karan-vk/Mail-Merge-Go

5.Implementation of Algorithm:

main.go

```
package main

import (
    "encoding/csv"
    "flag"
    "io"
    "os"
    "text/template"
)

type TWrap struct{ Fields *[]string }
func main() {
    csvFile := flag.String("c", "list.csv", "The csv file
to parse")                        //O(1)-T O(1)-S
    templateName := flag.String("t", "report.tpl", "The te
mplate to use")                   //O(1)-T O(1)-S
    fileNameColumn := flag.Int("i", 0, "Column of csv to u
se as output file basename") //O(1)-T O(1)-S
    fileExt := flag.String("s", "eml", "Output file suffix
")                                //O(1)-T O(1)-S
    flag.Parse()
                                  //O(1)-T O(n)-S
    template, err := template.ParseFiles(*templateName)
//O(1)-T O(n)-S n - size of the template

if err != nil {
        panic(err)
    } //O(1)-T O(1)-S
    file, err := os.Open(*csvFile) //O(1)-T O(n)-S
    if err != nil {
        panic(err)
    } //O(1)-T O(1)-S

    defer file.Close() //O(1)-T O(1)-
S note space is released by the garbage collector
    reader := csv.NewReader(file) //O(1)-T O(n)-S
    reader.Comma = ','

    for {
        row, err := reader.Read() //O(1)-T O(n)-
S number of column
        if err == io.EOF {
            break //O(1)-T O(1)-S
        } else if err != nil {
            panic(err)
        } //O(1)-T O(1)-S
        f, err := os.Create("./output/" + row[*fileNameCol
umn] + "." + *fileExt) //O(1)-T O(1)-S


        defer f.Close()
//O(1)-
T O(1)S note space is released by the garbage collector
        template.Execute(f, TWrap{Fields: &row})
//O(1)-T O(m)-S m - number of template slots
    } //O(n)-T O(n+m)-S
    sendEmail()
```

Sendmail.go

```go
package main

import (
    "fmt"
    "io/ioutil"
    "net/smtp"
    "os"
    "path/filepath"
    "strings"
    "time"
)

func sendEmail() {
    var files []string//O(1)-T O(1)-S
    var emails []string //O(1)-T O(1)-S
    c := make(chan string)//O(1)-T O(1)-S

    root := "./output" //O(1)-T O(1)-S
    err := filepath.Walk(root, func(path string, info os.Fi
leInfo, err error) error { //O(n)-T O(n)-S

        emails = append(emails, strings.Split(path[7:], ".e
ml")...)//O(1)-T O(1)-S
        files = append(files, path)//O(1)-T O(1)-S
        return nil
    })

    if err != nil {//O(1)-T O(1)-S
        panic(err)
    }
```

Github Repository - https://github.com/karan-vk/Mail-Merge-Go

```go
    for index, file := range files {
        fmt.Println(file)//O(1)-T O(1)-S
        if index != 0 {
            data, err := ioutil.ReadFile(file)
//O(1)-T O(n)-S

            if err != nil {
                fmt.Println("File reading error", err)
//O(1)-T O(1)-S
                return
            }

            go send(emails[index], string(data),c)//O(1)-
T O(n)-S


        }
    }//O(n)-T O(n)-S
    for l := range c {
        e := l
        go func(l string,e string) {
            // time.Sleep(5*time.Second)
            send(l,e, c)
        }(l,e)
    }//O(n)-T O(n)-S

    //fmt.Println()
}
```

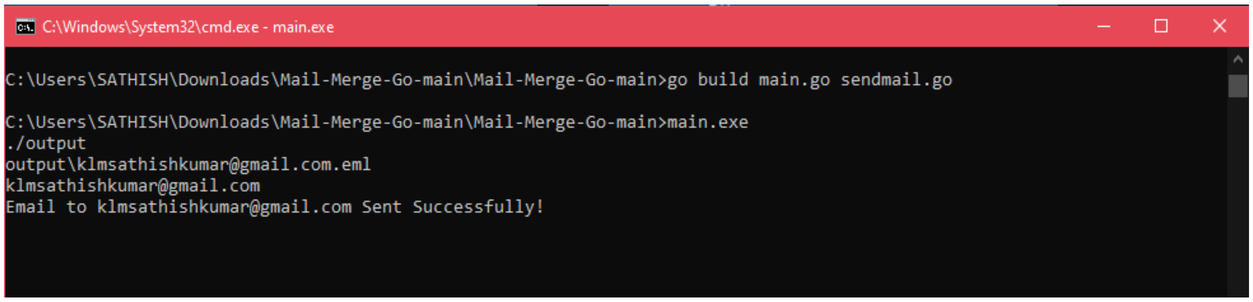
Github Repository - https://github.com/karan-vk/Mail-Merge-Go

```go
func send(email string, body string, c chan string ) {
    timer1:=time.Now().UnixNano()
    from := "testuser.sret@gmail.com"//O(1)-T O(1)-S
    password := "testuser"//O(1)-T O(1)-S

    to := []string{
        email,
    }//O(1)-T O(1)-S
    fmt.Println(to[0])//O(1)-T O(1)-S
    smtpHost := "smtp.gmail.com"//O(1)-T O(1)-S
    smtpPort := "587"//O(1)-T O(1)-S
    message := []byte(body)//O(1)-T O(n)-S
    auth := smtp.PlainAuth("", from, password, smtpHost)
//O(1)-T O(1)-S

    err := smtp.SendMail(smtpHost+":"+smtpPort, auth, from,
 to, message)//O(1)-T O(n)-S
    if err != nil {
        fmt.Println(err)//O(1)-T O(1)-S
        c <- "err"
        return
    }
    s := "Email to " + email + " Sent Successfully!"
//O(1)-T O(1)-S

    fmt.Println(s)//O(1)-T O(1)-S
    timer2:=time.Now().UnixNano()
    fmt.Println(timer2-timer1)
    c <- s
    return
}
```

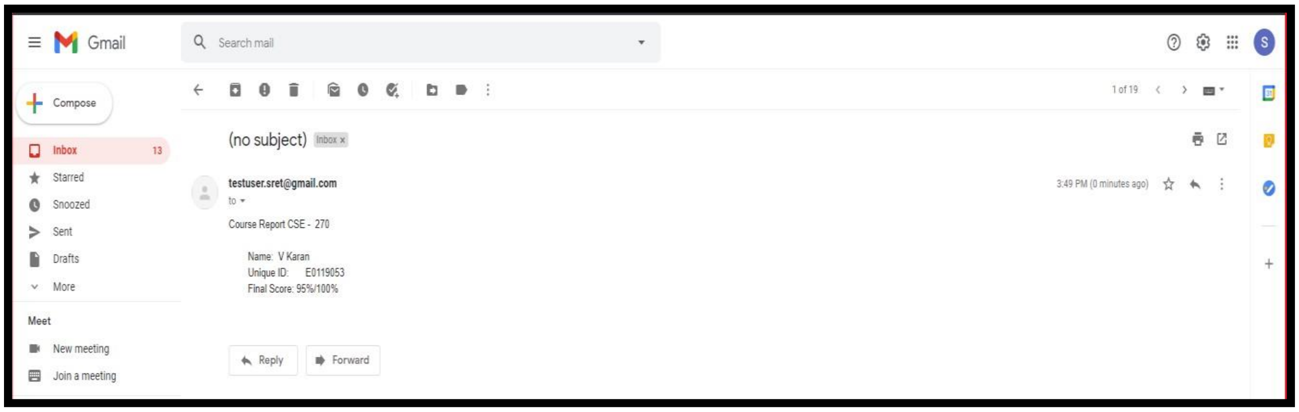
Github Repository - https://github.com/karan-vk/Mail-Merge-Go

OUTPUT:

Executing Golang file



```
C:\Users\SATHISH\Downloads\Mail-Merge-Go-main\Mail-Merge-Go-main>go build main.go sendmail.go

C:\Users\SATHISH\Downloads\Mail-Merge-Go-main\Mail-Merge-Go-main>main.exe
./output
output\klmsathishkumar@gmail.com.eml
klmsathishkumar@gmail.com
Email to klmsathishkumar@gmail.com Sent Successfully!
```



(no subject)

testuser.sret@gmail.com
to
3:49 PM (0 minutes ago)

Course Report CSE - 270

Name: V Karan
Unique ID: E0119053
Final Score: 95%/100%

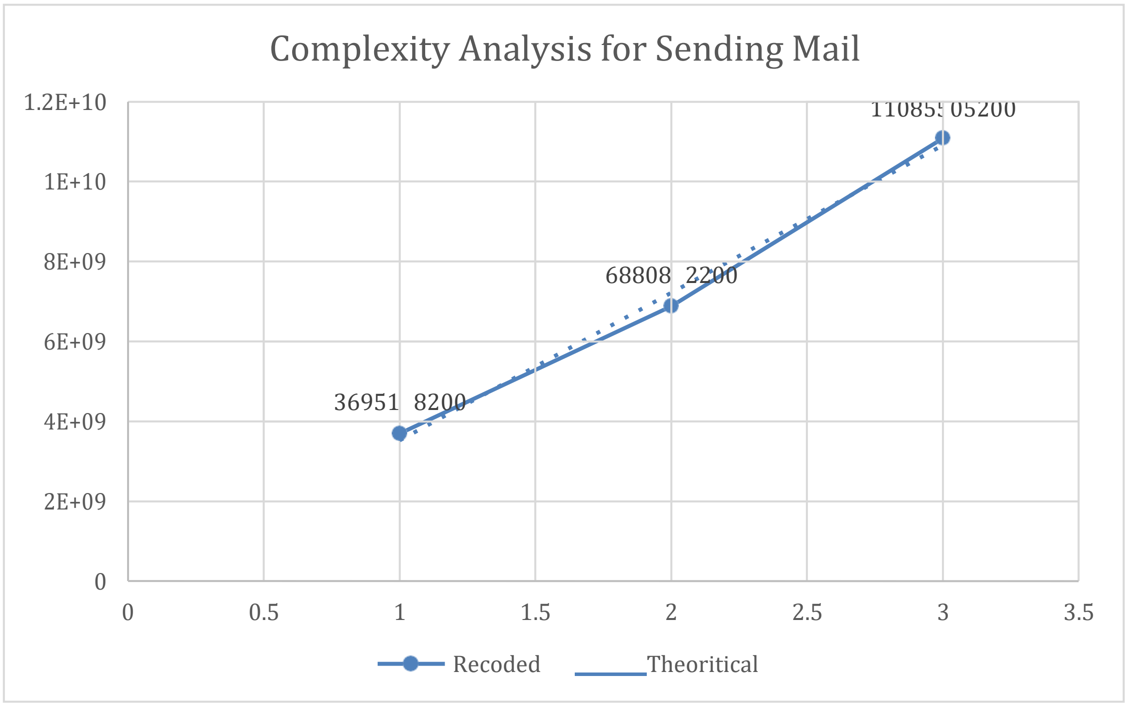
Mail sent Successfully!!

B. Deadline for above task: 28-10-2020

Github Repository - https://github.com/karan-vk/Mail-Merge-Go

6. Results on Complexity analysis:

Time complexity for network operations



- Mail Merge process – O(n)
- Send Mail process – O(n)

C. Deadline for above task and Final submission : 30-10-2020 (Continuous Assessment -4)

Github Repository - https://github.com/karan-vk/Mail-Merge-Go