Istanbul Technical University
Faculty of Computer and Informatics

BLG440E Computer Project 2

Project 3B: Mining Chrome Repository

Group No: 17

İrem Ertürk 150140725

Cem Yusuf Aydoğdu 150120251

GitHub Link: https://github.com/cemysf/blg440e_project3b_group17

YouTube Link: https://www.youtube.com/watch?v=pZATGk02KVk&feature=youtu.be

1. Extracting Useful Data

First of all, we determine the working repositary from the Chromium project and clone the whole repository to our desktops by the following command:

$git clone https://chromium.googlesource.com/chromium/src/build/

Then we realize the that, we need to query some useful data sets which helps us while mining chrome repositories.

- File Name List

with folders: *$ls -ld $(find .) | awk '{print $9}' | sed 's/^..//' | sed -n 'p;$='*

without folders: *$ls -ldp $(find .) | grep -v '/$' | awk '{print $9}' | sed 's/^..//'*

- Developer List

alphabetic developer names:

$git log --pretty=format:"%cn" | sort | uniq

developer names and comit counts in numeric order:

$git log --pretty=format:"%cn" | sort | uniq --count | sort -nr

2. Implementation Details

Identify Top Developers

For identying the top developers that contribute most in file changes, we implement the findTopDeveloper() function in 440_S_FindTopStatistics.sh file. That function takes the total commit count and the required percentage as parameter**.**

```
developer_list="440_F_DeveloperList_Numeric.txt"

declare -i total_commit
total_commit="$(git rev-list --all --count)"


findTopDevelopers(){ #$1=total_commit, $2=%x
        declare -i thresold
        declare -i temp
        temp=0
        thresold=total_commit*$2/100


        while IFS= read -r line  #while read line;
        do
                current=$(echo $line | awk '{print $1}')
                percent=$(echo "$(echo "($current*100)/$total_commit" | bc -l)" | awk '{printf
"%.10f\n",$1}')

                temp=$(($temp+$current))
                dev=$(echo $line | awk '{ for(i=2; i<NF; i++) printf "%s ", $i OFS; if(NF) printf
"%s",$NF; printf ORS}')
                echo -e $percent '\t' $dev

                if [ $temp -ge $thresold ]
                then
                        break
                fi
        done < "$developer_list"  #<<<"$(git log --pretty=format:\"%cn\" | sort | uniq --count |
sort -nr)"
}
```

As seen above, findTopDevelopers() function travels through the 440_F_DeveloperList_Numeric.txt which contains the commit counts of each developers with descendent order.In that manner, at the beginning of the function the required commit value(threshold) is calculated with the :: thresold=total_commit*$2/100.($2 represents the percentage value.) And the function stop looking for each line. And the top developers names and their own contribution percentages are written to the 440_FO_TopDevelopers_80.txt file.

Identify Top Edited Files

For finding the top edited files, we need a new file which contains the filename and edition number of each files. For that reason findCommitCountOfFiles() function loop through the "440_F_FileList.txt" file and use *$(git log --oneline -- $line | wc -l)* command to get number of editions to specific file.

```
file_list="440_F_FileList.txt"

findCommitCountOfFiles(){

        while IFS= read -r line
        do
                #echo $line
                echo "$(git log --oneline -- $line | wc -l)" $line
        done < "$file_list"
}
```

Because findTopFiles() function in 440_S_FindTopStatistics.sh file need that the above code creates a new file 440_F_FilesCommitCounts_sorted.txt file. Then findTopFiles() function travel through that by considering total commit count and the required percentage as parameter.

```
topfile_list="440_F_FilesCommitCounts_sorted.txt"
findTopFiles(){
        declare -i thresold
        declare -i temp
        temp=0
        thresold=total_commit*$2/100


        while IFS= read -r line
        do
                #echo $line
                #echo "$(git log --oneline -- $line | wc -l)" $line
                current=$(echo $line | awk '{print $1}')
                percent=$(echo "$(echo "($current*100)/$total_commit" | bc -l)" | awk '{printf
"%.10f\n",$1}')

                temp=$(($temp+$current))
                dev=$(echo $line | awk '{ for(i=2; i<NF; i++) printf "%s ", $i OFS; if(NF) printf
"%s",$NF; printf ORS}')
                echo -e $percent '\t' $dev

                if [ $temp -ge $thresold ]
                then
                        break
                fi

        done < "$topfile_list"

}
```

As seen above, findTopFiles() function travels through the 440_F_FilesCommitCounts_sorted.txt which contains the file names in alphabetic order with the edition numbers.In that manner, at the beginning of the function the required commit value(threshold) is calculated with the :: thresold=total_commit*$2/100.($2 represents the percentage value.) And the function stop looking for each line whenever temp value reacehes the predefined threshold value. And the top developers names and their own contribution percentages are written to the 440_FO_TopFiles_80.txt file.

Create Adjacency Matrix

```
#! /bin/bash
committer_list="440_F_DeveloperList_Alphabetic.txt"
file_list="440_F_FileList.txt"

declare -A names

# print commiter names in the first line
declare -i index
index=0
while IFS= read -r name
do
        names["\"$name\""]=$index
        ((index++))
done < "$committer_list"
```

Map each developers name to an integer value and use it whenever the developer contibute the change in file.

```
# Construct adj matrix
declare -i column_number
declare -i row_number

initializeMatrix()
{
        local index=0
        local total=$(($row_number * $column_number))

        while [ "$index" -lt "$total" ]
        do
                AdjMatrix[$index]=0
                let "index += 1"
        done

}
printMatrix()
{
        local index=0
        for ((r=0; r<row_number;r++))
        do
                for ((c=0; c<column_number; c++))
                do
                        let "index = $r*$column_number + $c"
                        echo -n "${AdjMatrix[$index]} "
                done
                echo ""
        done
}

printRow()      #$1:row
{
        local index=0

                for ((c=0; c<column_number; c++))
                do
                        let "index = $1*$column_number + $c"

                        echo -n "${AdjMatrix[$index]} "
                done
                echo ""
}
```

To create the adjacency matrix for representing contributions to each files, we first create and initialize the Adjacency matrix with all zeros by using the initializeMatrix() function. Rows represent the files and the columns represents the developers.

```
modifyValue()   ## $1=row $2=col $3=value
{
        local index=0

        let "index = $1*$column_number + $2"
        AdjMatrix["$index"]=$3
}

# for each file in the list, check its collaborators
declare -i row_count
declare -i temp
declare -i constant_v

row_count=0
temp=0
constant_v=1
declare -a AdjMatrix

row_number=1

column_number=${#names[@]}

while IFS= read -r item
do

        initializeMatrix

        while read n;
        do

           temp=${names[$n]}

            modifyValue 0 ${names[$n]} 1

        done <<<"$(git log  --pretty=format:\"%cn\" -- $item | sort | uniq )"

        printRow

done < "$file_list"
```

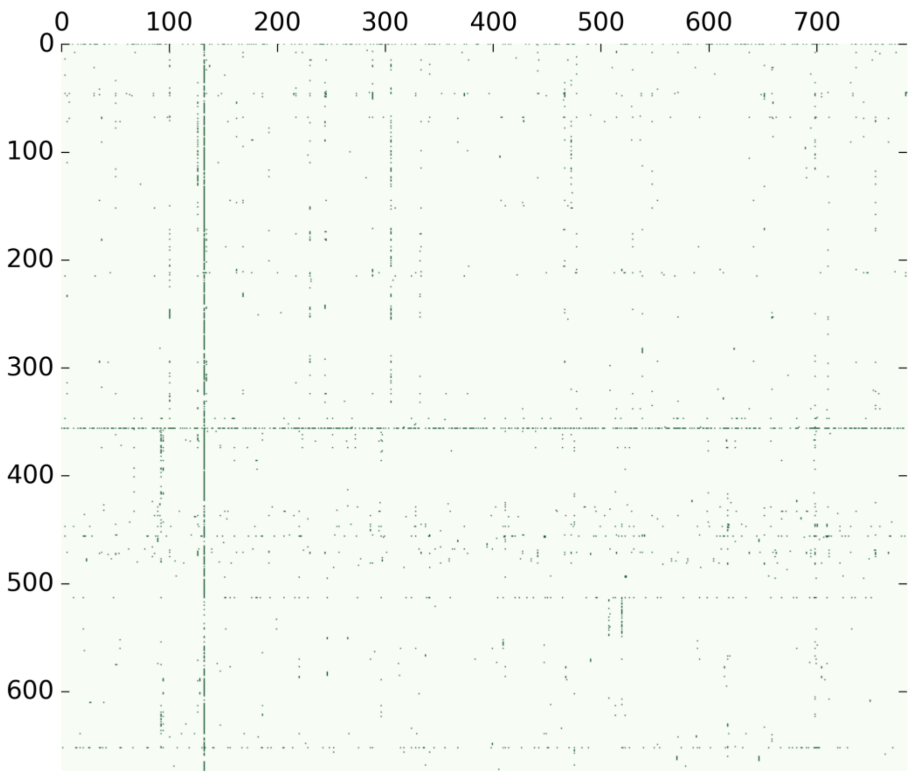
Modify value fuction takes the row, column and new_value as parameters. Because we store adjacency matrix in one one dimensional array rather than 2d array, we calculate the index by using row and column values. And change the value with one.

The main flow of algorithm is done here. The blue highlighted code loops through the "440_F_FileList.txt" file and for each file finds the file contributers. As you can see in the inner while loop we call the modifyValuefunction to change value in adjacency matrix. After all contributers of files modified in inner loop we print the rows.

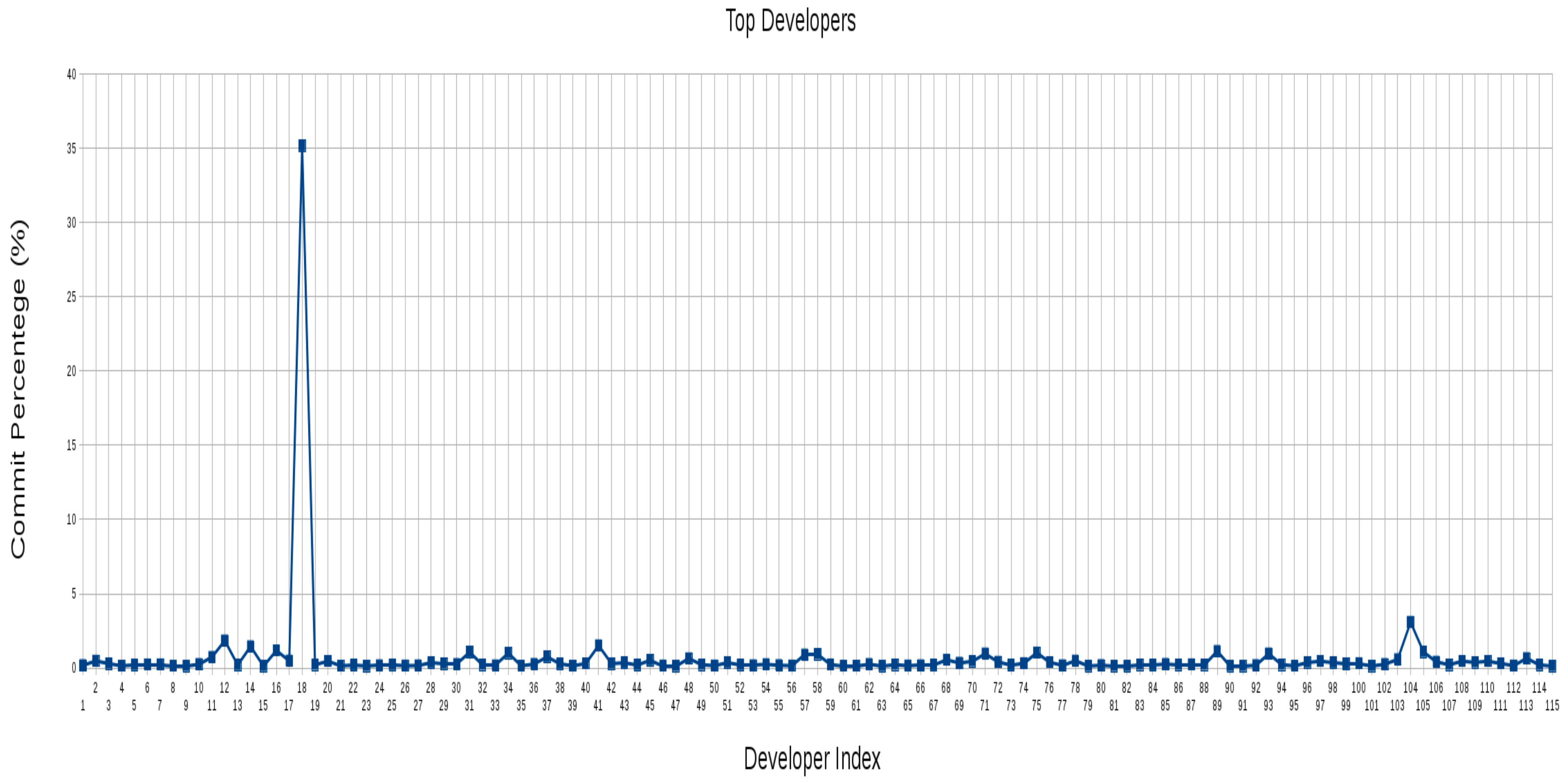
3. Matrix Visualization

At the last step of our project, we encounter with problem about visualization tools. First of all, there exists no download link in given SocialAction website. Then, we try to use Gephi social network visualization tool. However, its comma separated format is not suitable for our adjacency matrix format. For that reason we write a python code which represents our adjacency matrix.

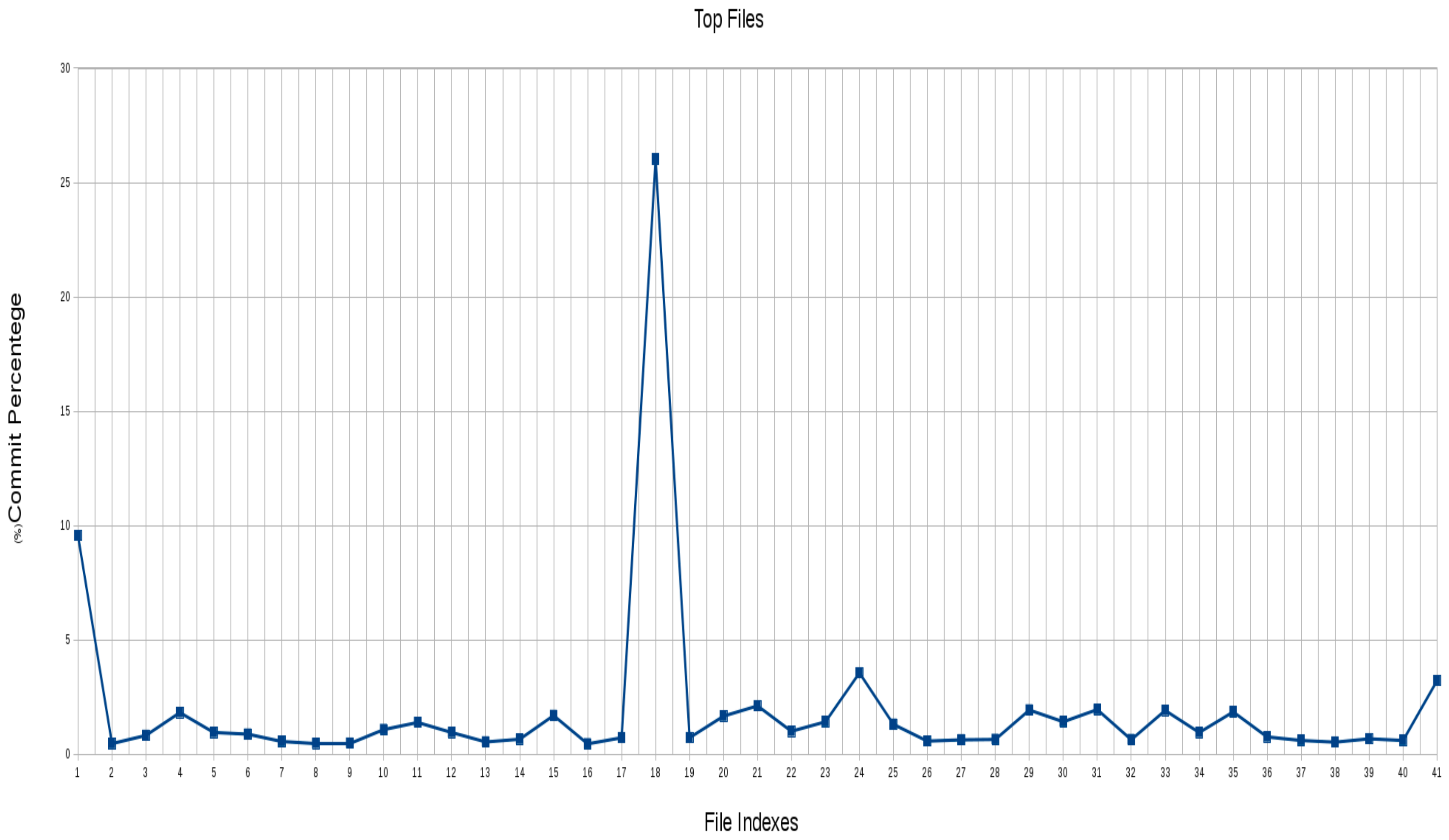
```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import numpy as np

input_file="440_FO_Matrix.txt"

matrix=[]

# Read file into an array
with open(input_file) as f:
    for line in f:
        matrix.append(line.split())

# Copy matrix into numpy array
numpy_array=np.array(matrix, dtype=np.float)

# Plot numpy array
fig = plt.figure()
ax= fig.add_subplot(1,1,1)
ax.imshow(numpy_array, cmap=plt.cm.Greens, origin="lower")
ax.xaxis.tick_top()
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.gca().invert_yaxis()

# Show and write
plt.show()
fig.savefig("adjMatrix_output.png", dpi=350)
```



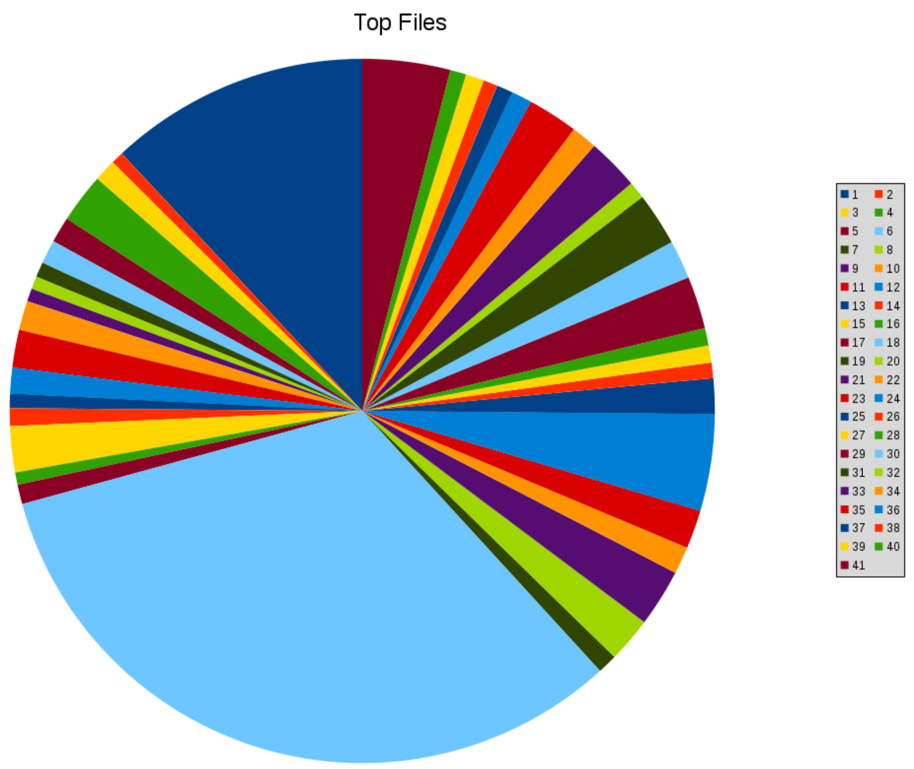
3. Results


Top Developers
Commit Percentege (%)
Developer Index
0
5
10
15
20
25
30
35
40

%	Developer
0.1835872958	**aberent@chromium.org**
0.477326969	agl@chromium.org
0.2937396732	ajwong@chromium.org
0.1468698366	andrewhayden@chromium.org
0.2019460253	apatrick@chromium.org
0.2294841197	aurimas@chromium.org
0.2386634845	ben@chromium.org
0.1468698366	blundell@chromium.org
0.1376904718	boliu@chromium.org
0.2570222141	bradnelson@chromium.org
0.734349183	bradnelson@google.com
1.8450523224	brettw@chromium.org
0.2019460253	Brett Wilson
1.4503396365	bulach@chromium.org
0.128511107	cevans@chromium.org
1.1749586929	cjhopman@chromium.org
0.4865063338	cmp@chromium.org
35.1569671379	Commit bot
0.2203047549	cpu@chromium.org
0.4589682394	craigdh@chromium.org
0.1468698366	csharp@chromium.org
0.1927666605	davemoore@chromium.org
0.1376904718	derat@chromium.org
0.174407931	dfalcantara@chromium.org
0.2019460253	digit@chromium.org
0.1560492014	dkegel@google.com
0.1652285662	dmichael@chromium.org
0.3579952267	dpranke@chromium.org
0.2937396732	erg@chromium.org
0.2570222141	eugenis@chromium.org
1.0648063154	evan@chromium.org
0.2203047549	evanm@google.com
0.1652285662	fischman@chromium.org
1.0005507619	frankf@chromium.org
0.1468698366	frankf@google.com
0.2662015789	gkanwar@chromium.org
0.7527079126	glider@chromium.org
0.2845603084	hans@chromium.org
0.1468698366	hclam@chromium.org
0.302919038	iannucci@chromium.org
1.51459519	ilevy@chromium.org
0.2937396732	jam@chromium.org
0.3488158619	jamesr@chromium.org
0.2019460253	james.wei@intel.com
0.5140444281	jbudorick@chromium.org
0.1560492014	jknotten@chromium.org
0.1376904718	jln@chromium.org
0.6425555352	jochen@chromium.org
0.2203047549	joi@chromium.org
0.1560492014	joth@chromium.org
0.3579952267	jrg@chromium.org
0.2019460253	jrg@google.com
0.174407931	jschuh@chromium.org
0.2478428493	jshin@chromium.org
0.1835872958	kjellander@chromium.org
0.1560492014	kkania@chromium.org
0.8903983844	mark@chromium.org
0.9271158436	maruel@chromium.org
0.2294841197	michaelbai@chromium.org
0.1468698366	michaelbai@google.com
0.1468698366	mithro@mithis.com
0.2570222141	mkosiba@chromium.org
0.128511107	mmentovai@google.com
0.2203047549	mmoss@chromium.org
0.1560492014	mmoss@google.com
0.174407931	mnaganov@chromium.org
0.1927666605	mostynb@opera.com
0.5507618873	navabi@google.com
0.3396364972	newt@chromium.org
0.4497888746	Nico Weber
0.9638333027	nileshagrawal@chromium.org
0.4038920507	nsylvain@chromium.org
0.1927666605	oshima@chromium.org
0.3120984028	peter@chromium.org
1.0280888563	phajdan.jr@chromium.org
0.3855333211	piman@chromium.org
0.174407931	pkasting@chromium.org
0.4865063338	pliard@chromium.org
0.1376904718	primiano@chromium.org
0.1835872958	qsr@chromium.org
0.1376904718	rmcilroy@chromium.org
0.128511107	robertshield@chromium.org
0.2203047549	rsesek@chromium.org
0.1927666605	rsleevi@chromium.org
0.2662015789	sadrul@chromium.org
0.2111253901	saintlou@chromium.org
0.2386634845	sbc@chromium.org
0.2203047549	scherkus@chromium.org
1.1382412337	scottmg@chromium.org
0.1376904718	scottmg@google.com
0.1376904718	sebmarchand@chromium.org
0.1835872958	sergeyu@chromium.org
0.9638333027	sgk@google.com
0.2203047549	shashishekhar@chromium.org
0.1468698366	shouqun.liu@intel.com
0.3671745915	sivachandra@chromium.org
0.4589682394	sky@chromium.org
0.3579952267	skyostil@chromium.org
0.2753809436	spang@chromium.org
0.3120984028	stuartmorgan@chromium.org
0.1376904718	tapted@chromium.org
0.2570222141	tc@google.com
0.5782999816	tfarina@chromium.org
3.0934459335	thakis@chromium.org
1.083165045	thestig@chromium.org
0.4130714155	thomasvl@chromium.org
0.2203047549	timurrrr@chromium.org
0.4589682394	tony@chromium.org
0.3763539563	tonyg@chromium.org
0.4589682394	torne@chromium.org
0.302919038	wangxianzhu@chromium.org
0.1560492014	willchan@chromium.org
0.6517348999	yfriedman@chromium.org
0.2111253901	yongsheng.zhu@intel.com
0.128511107	zty@chromium.org


Top Files
(%)Commit Percentege
File Indexes
0
5
10
15
20
25
30
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

%	File
9.5557187443	all.gyp
0.4589682394	android/adb_install_apk.py
0.8169634661	android/buildbot/bb_device_status_check.py
1.817514228	android/buildbot/bb_device_steps.py
0.9454745732	android/buildbot/bb_run_bot.py
0.8720396549	android/envsetup.sh
0.5507618873	android/gyp/javac.py
0.4681476042	android/gyp/util/build_utils.py
0.477326969	android/PRESUBMIT.py
1.0739856802	android/provision_devices.py
1.386084083	android/pylib/device/device_utils.py
0.9454745732	android/pylib/gtest/filter/content_browsertests_disabled
0.5324031577	android/pylib/gtest/gtest_config.py
0.6517348999	android/pylib/perf/test_runner.py
1.689003121	android/test_runner.py
0.4497888746	android/tombstones.py
0.7159904535	build_config.h
26.0234991739	common.gypi
0.7159904535	config/android/config.gni
1.6614650266	config/android/internal_rules.gni
2.1112539012	config/android/rules.gni
1.0005507619	config/BUILDCONFIG.gn
1.4136221773	config/BUILD.gn
3.5524141729	config/compiler/BUILD.gn
1.3034697999	config/features.gni
0.5691206169	config/linux/BUILD.gn
0.6241968056	config/win/BUILD.gn
0.6425555352	get_landmines.py
1.9276666055	gn_migration.gypi
1.4136221773	gyp_chromium
1.946025335	install-build-deps.sh
0.6333761704	isolate.gypi
1.9093078759	java_apk.gypi
0.9454745732	java.gypi
1.8450523224	linux/system.gyp
0.7527079126	sanitizers/tsan_suppressions.cc
0.5966587112	toolchain/gcc_toolchain.gni
0.5232237929	toolchain/mac/BUILD.gn
0.6700936295	toolchain/win/BUILD.gn
0.5966587112	vs_toolchain.py
3.2127776758	whitespace_file.txt


Top Files