Word Segmentation

To separate the connected columns of a given binary image.

Sounak Roy

Word Segmentation

To separate the connected columns of a given binary image.

Introduction:

A bitmap or bit-vector is a mapping form of some domain (almost always in the range of integer) to the values in the set of {0,1}. The values can be represented as light/dark, absent/present, unlocked/locked, in-valid/valid, etc. The point is that there are only two possible values and they can be stored n one bit. The array can be viewed as a sub-set of domain, where “1” indicates a number in the set and “0” a number not in the set. This data structure uses “n/w” words of space where “w” is the number of bit(s) in each machine word. Weather the last significant bit or the most significant bit indicates the smallest index number is highly irrelevant, but the format is to be followed.

Objective:

In this project we have to analyze a bitmap or bit-vector, and for that we can use two approaches –

1. Take a binary matrix as input from the user
2. Complete the process with a previously created binary matrix stored in a file.

By using “C” code we select the columns that have at least one number of “1” and the process continues till we find a blank column. Then the previous rows are copied and stored in a separate file. The process is then continued accordingly.

After the process we see that from a single source file (binary matrix) we can create multiple files (segmented arrays). The input is taken as one file and the output is delivered in several segmented file.

Software(s) Used:

Windows 7, Ultimate (Operating System)

CodeBlocks [GNU GCC Compiler]

Notepad

How To Use:

The stand alone executable file is present such that if the user wants to run it without any code editing harassments. There is an option of making your file (text file, .txt format), naming it Source.txt and place it in the required destination (where ever the executable file is saved in your local disk).

Source Code:

```
/* Including Header Files */

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
#include<malloc.h>

/* Defining The Value Of Max */

#define MAXX 1000

/* Function To Insert Manually */

void insert(void)
{
	FILE *p;
	int i,j,n,max_row,max_colm;
	p=fopen("Source.txt","w");
	printf("\n Enter Size Of Row ");
	scanf("%d",&max_row);
	printf("\n Enter Size Of Column ");
	scanf("%d",&max_colm);
	for(i=0;i<max_row;i++)
	{
		printf("\n Enter %d row : ",i+1);
		for(j=0;j<max_colm;j++)
		{
			scanf("%d",&n);
			fprintf(p,"%d",n);
		}
		fprintf(p,"\n");
	}
	fclose(p);
}

/* Function To Obtain The Number Of Rows And Columns From The Source Matrix */

int getmax(void)
{
	FILE *p;
	int c=0;
	char ch;
```

```
	p=fopen("Source.txt","r");
	if(p==NULL)
	{
		printf("\n Sorry, Source File Not Found !");
		getch();
		exit(0);
	}
	do
	{
		ch=getc(p);



		if(ch=='0'||ch=='1')
		{
			c++;
		}
	}while(ch!='\n');
	return c;
	}

/* Main Function */

int main()
{
	FILE *fp,*fp1;
	int *a[MAXX],max_row,max_colm;
	int i=0,j=0,k=-1,prev=0,new=0,flag,l,pos,x,choice;
	char ch,fname[20],no[5]="0",temp[5];
	printf("\n \n Word Segmentation Program \n \n");
	printf("\n Do You Want To Insert Your Own Matrix Or Go With The Pre-Created Source File \n");
	printf("\n Press 1. To Insert Your Own Matrix \n Press
			2. To Go With The Pre-Created Matrix \n\n ");
	scanf("%d",&choice);
	switch(choice)
	{
		case 1:
			insert();
			 break;
		case 2:
			break;
		default:
			break;
	}
	max_colm=getmax();
	fp=fopen("source.txt","r");
	a[i]=(int *)malloc(max_colm*sizeof(int));
	do
	{
		ch=fgetc(fp);
```

```
		{
			printf("\n Source file error !");
			getch();
			return;
		}
		switch(ch)
		{
			case '0':
				*(a[i]+j)=0; j++;
				break;
			case '1':
				*(a[i]+j)=1;
				 j++;
				break;


			case '\n':
				i++;
				a[i]=(int *)malloc(max_colm*sizeof(int));
				j=0;
				break;
			default:
				 break;
		};
	}while(ch!=EOF);
	max_row=i; fclose(fp);
	printf("\n Source File Read Succssfully\n");
	printf("\n\n Starting Segmentation...\n");

/* Starting Segmentation */

	for(j=0;j<max_colm;j++)
	{
		flag=0; for(i=0;i<max_row;i++)
		{
			if(*(a[i]+j)==1)
			{
				prev=new;
				new=1;
				flag=1;
				break;
			}
		}
		if(flag==0)
		{
			prev=new;
			new=0;
		}
		if(prev==0&&new==1)
		{
```

/* Creating Part Files Or Segmented Files */

```
                        k++;
                        strcpy(fname,"Part ");
                        l=strlen(no);
                        step: if(no[l-1]=='9')
                        {
                                no[l-1]='0';
                                if(l==1)
                                {
                                        strcpy(temp,no);
                                        strcpy(no,"1");
                                        strcat(no,temp);
                                }
                                if(no[l-2]=='9')
                                {
                                        l--;
                                        goto step;
                                }
                                else
                                        no[l-2]++;
                        }
                        else
                                no[l-1]++;
                        strcat(fname,no);
                        strcat(fname,".txt");
                        remove(fname);
                        pos=j;
                }
                if(((new==0)&&(prev==1))||((j==max_colm-1)&&(new==1)))
                {
                        fp1=fopen(fname,"w");
```

/* Writing Into The Segmented Files Previously Created */

```
                for(i=0;i<max_row;i++)
                {
                        for(x=pos;x<j+new;x++)
                        {
                                fprintf(fp1,"%d",*(a[i]+x));
                        }
                        fprintf(fp1,"\n");
                }
                        fclose(fp1);
                        printf("\n %s File Created \n",fname);
                }
        }
        printf("\n\n %d Total Part File(s) Created Successfully\n",k+1);
        printf("\n\n To View The Segmented Files \n Go To The Same Location Where The .exe File Of The
        Program Is Saved\n");
```

```
        return 0;
}
```

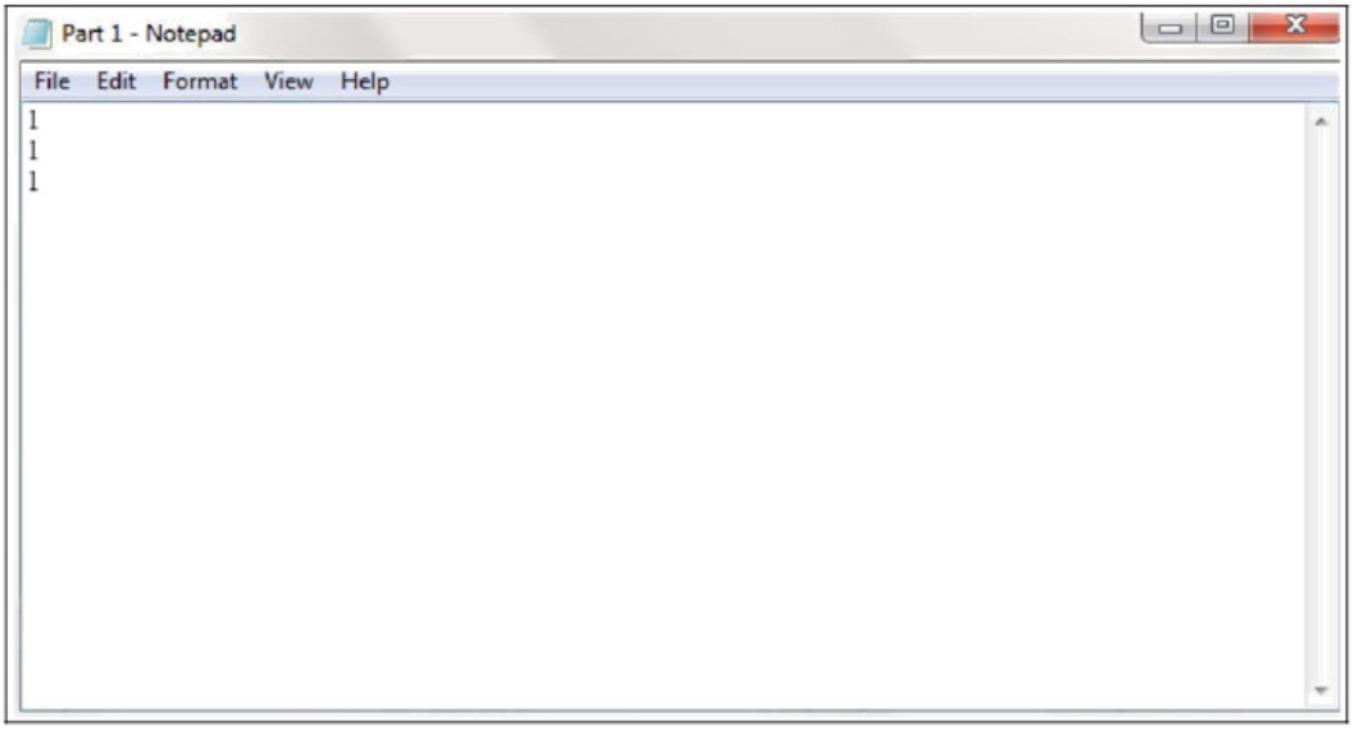
How It Works:

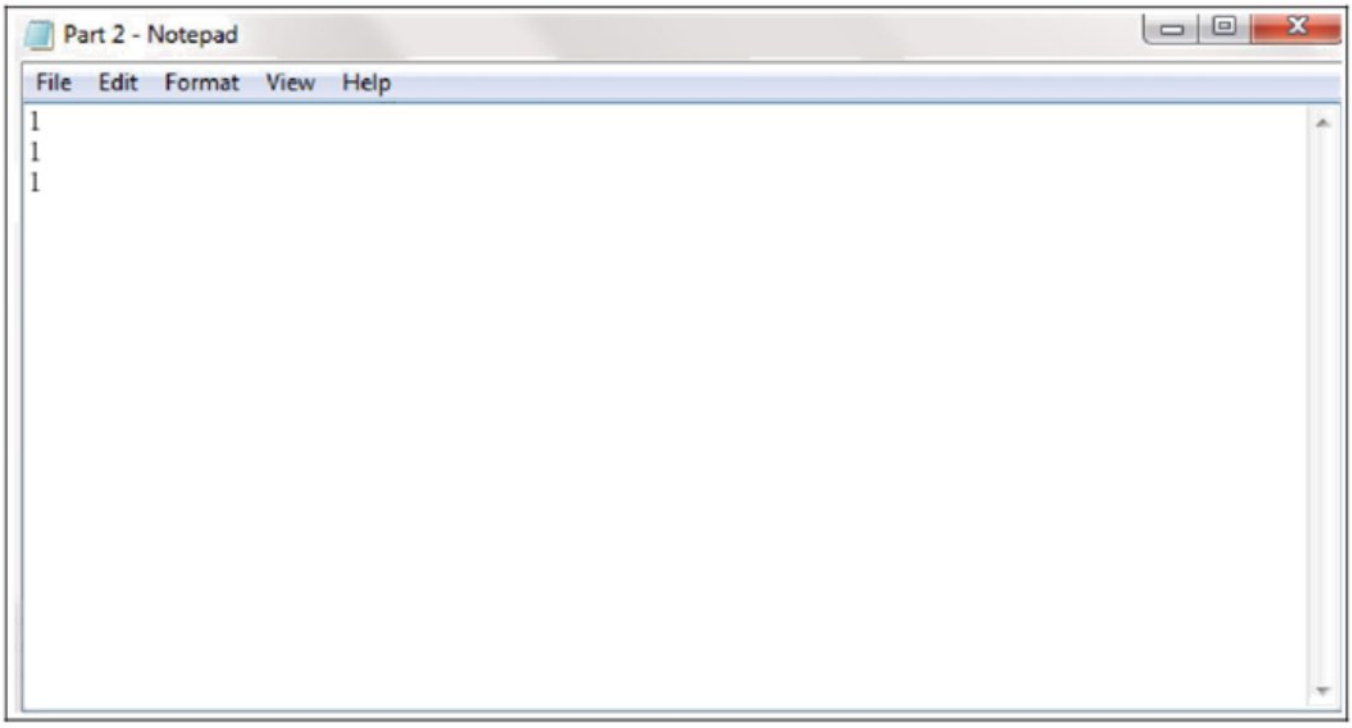
#1. When inserted a matrix of user's choice

[Output Screen]

```
Word Segmentation Program

Do You Want To Insert Your Own Matrix Or Go With The Pre-Created Source File

Press 1. To Insert Your Own Matrix
Press 2. To Go With The Pre-Created Matrix

1

Enter Size Of Row 3

Enter Size Of Column 5

Enter 1 row : 0 1 0 1 0

Enter 2 row : 0 1 0 1 0

Enter 3 row : 0 1 0 1 0

Source File Read Succssfully

Starting Segmentation...

Part 1.txt File Created

Part 2.txt File Created

2 Total Part File(s) Created Successfully

To View The Segmented Files
Go To The Same Location Where The .exe File Of The Program Is Saved

Thank You !
```

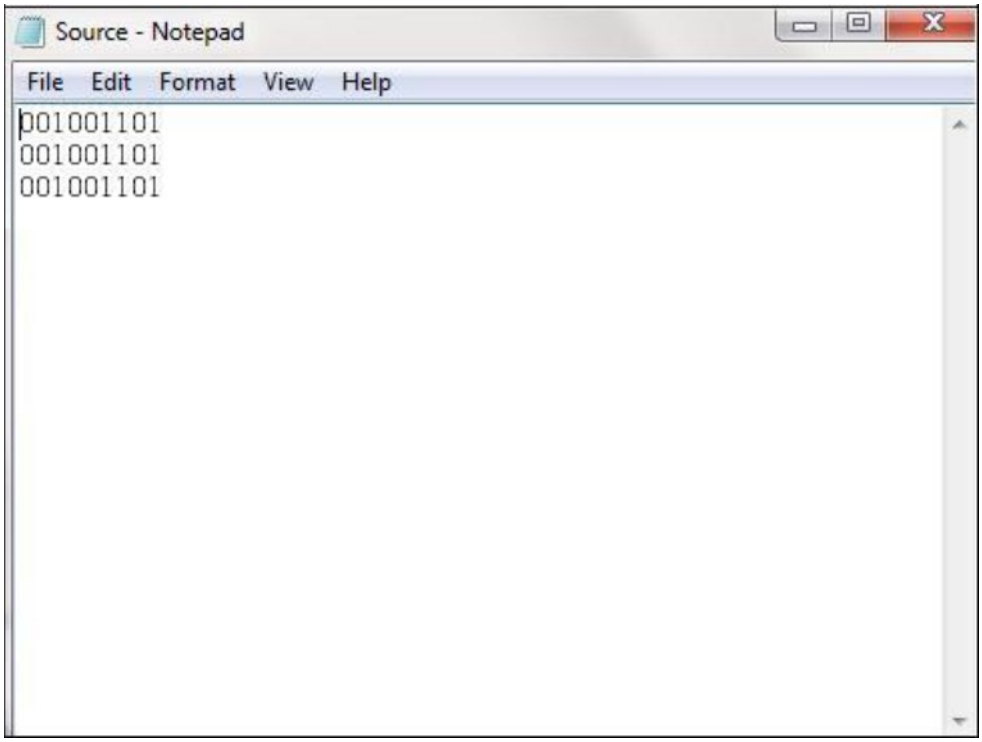
[Part Files Created]





#2. When Operated On A Pre-Defined Matrix

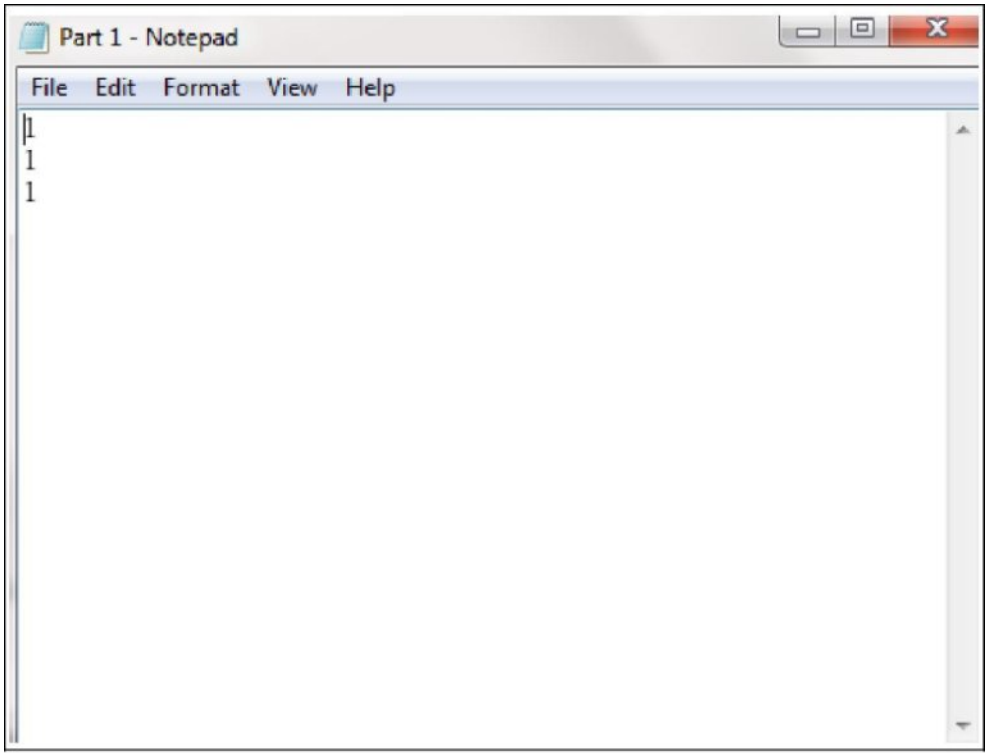
[Source File, Pre-Defined Matrix]

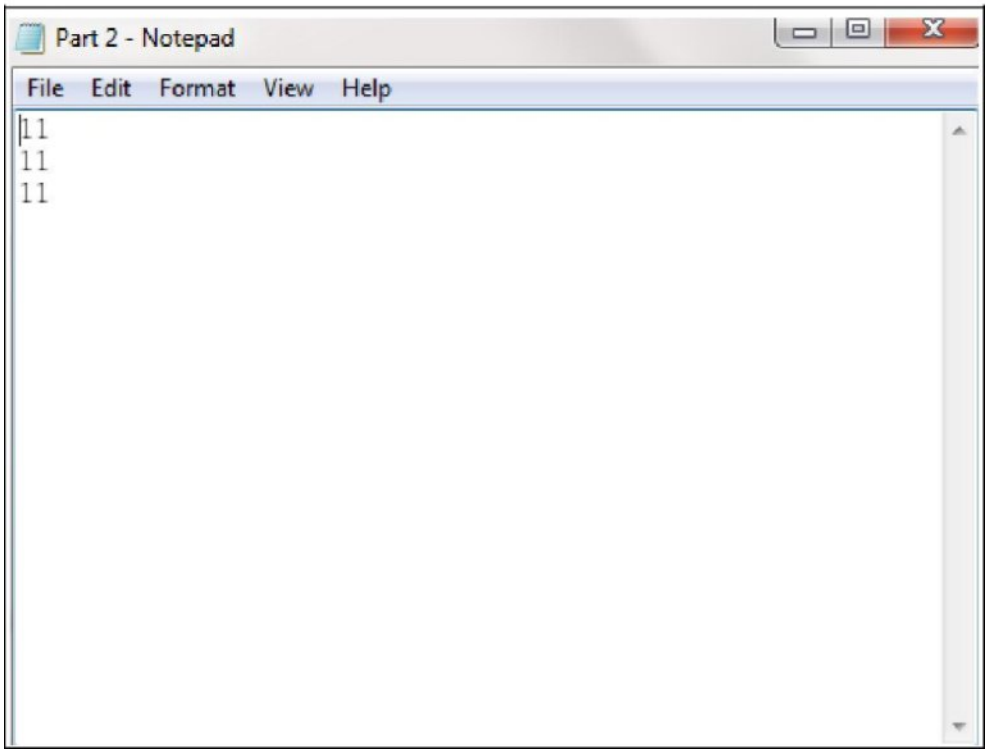


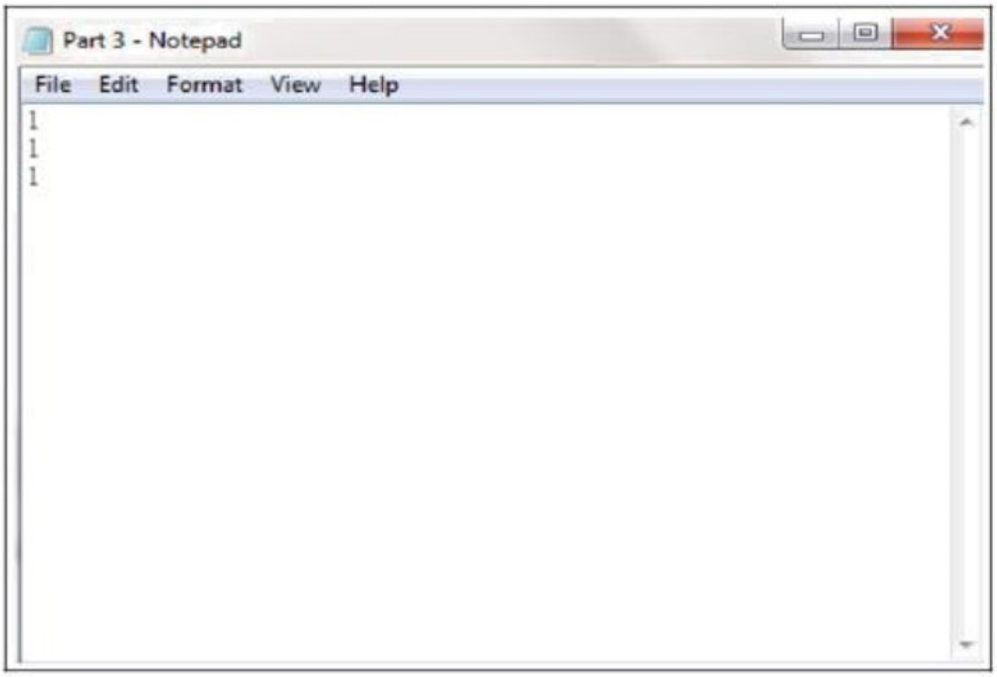
[Output Screen]

```
Word Segmentation Program


Do You Want To Insert Your Own Matrix Or Go With The Pre-Created Source File

Press 1. To Insert Your Own Matrix
Press 2. To Go With The Pre-Created Matrix

2

Source File Read Succssfully


Starting Segmentation...

Part 1.txt File Created

Part 2.txt File Created

Part 3.txt File Created


3 Total Part File(s) Created Successfully


To View The Segmented Files
Go To The Same Location Where The .exe File Of The Program Is Saved


Thank You !
```

[Part Files Created]







Conclusion:

This project is a part of bitmap imaging, i.e.; it gives us knowledge about bitmap images. By working on this project it is understood that a bitmap image works.