String matching using brute force

//string matching

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<time.h>

char text[100],pat[100];

char r[26]="abcdefghijklmnopqrstuvwxyz";

int i,j,count=0,n,m,l,k,x,g;

int string\_match(char text[],char pat[]){

n=strlen(text);

m=strlen(pat);

for(i=0;i<=n-m;i++){

count++;

j=0;

while(j<m&&text[i+j]==pat[j]){

j++;

count++;

}

if(j==m)return 1;

}

return -1;

}

int best(int n){

count=0;

for(l=0;l<g;l++){

k=rand()%26;

text[l]=r[k];

}

for(x=0;x<3;x++)

pat[x]=text[x];

string\_match(text,pat);

return count;

}

int average(int n){

count=0;

for(l=0;l<g;l++){

k=rand()%2;

text[l]=r[k];

}

for(x=0;x<3;x++){

pat[x]=r[rand()%1];

}

string\_match(text,pat);

return count;

}

int worst(int n){

count =0;

for(l=0;l<g;l++)

text[l]='a';

for(x=0;x<3;x++){

pat[x]='a';

if(x==2)pat[x]='b';

}

string\_match(text,pat);

return count;

}

int main(void){

FILE \*bestsm,\*avgsm,\*worstsm;

system("rm bestsm.txt");

system("rm avgsm.txt");

system("rm worstsm.txt");

srand(time(NULL));

bestsm=fopen("bestsm.txt","a");

avgsm=fopen("avgsm.txt","a");

worstsm=fopen("worstsm.txt","a");

for(g=5;g<=50;g+=5){

fprintf(bestsm,"%d\t%d\n",g,best(g));

fprintf(avgsm,"%d\t%d\n",g,average(g));

fprintf(worstsm,"%d\t%d\n",g,worst(g));

}

fclose(bestsm);

fclose(avgsm);

fclose(worstsm);

}

Output file :

Best case

n opcount

5 4

10 4

15 4

20 4

25 4

30 4

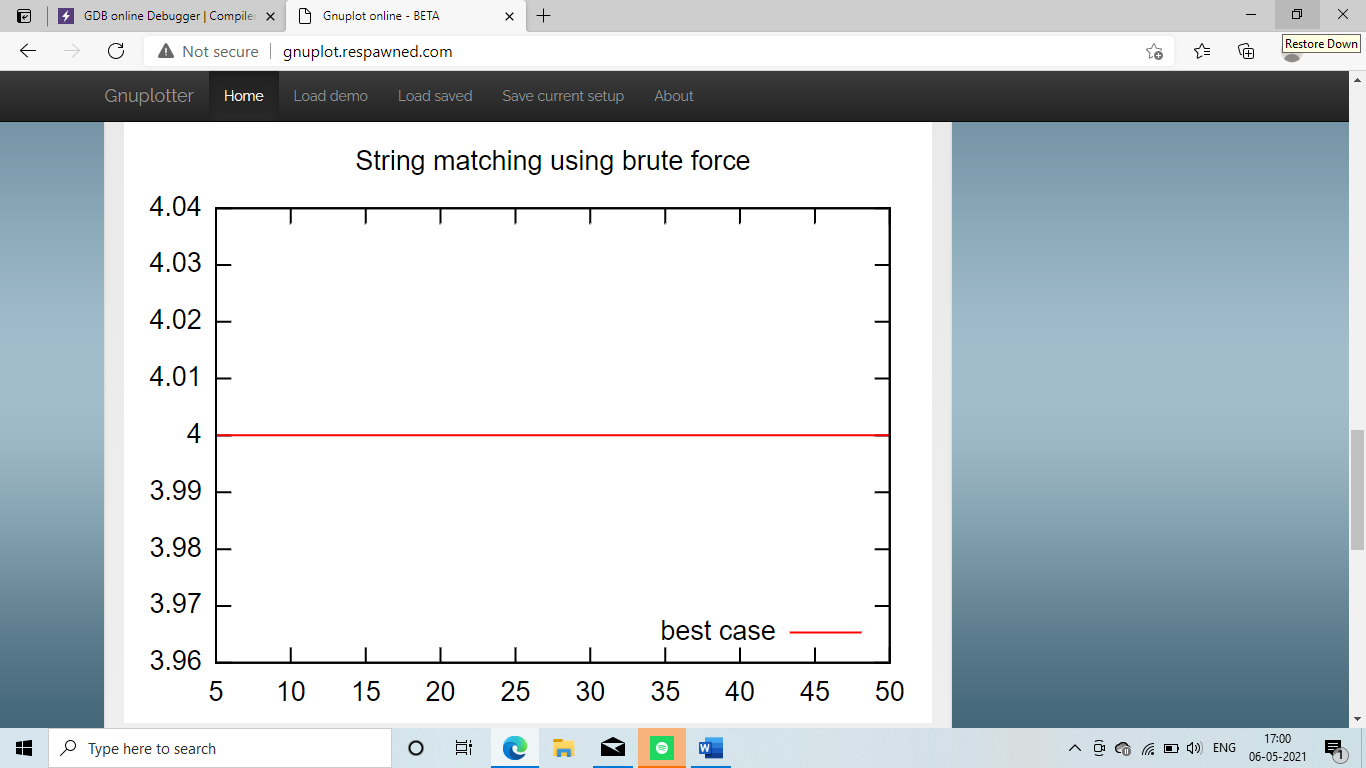
35 4

40 4

45 4

50 4

Graph :



Average case :

n opcount

5 3

10 13

15 23

20 11

25 10

30 4

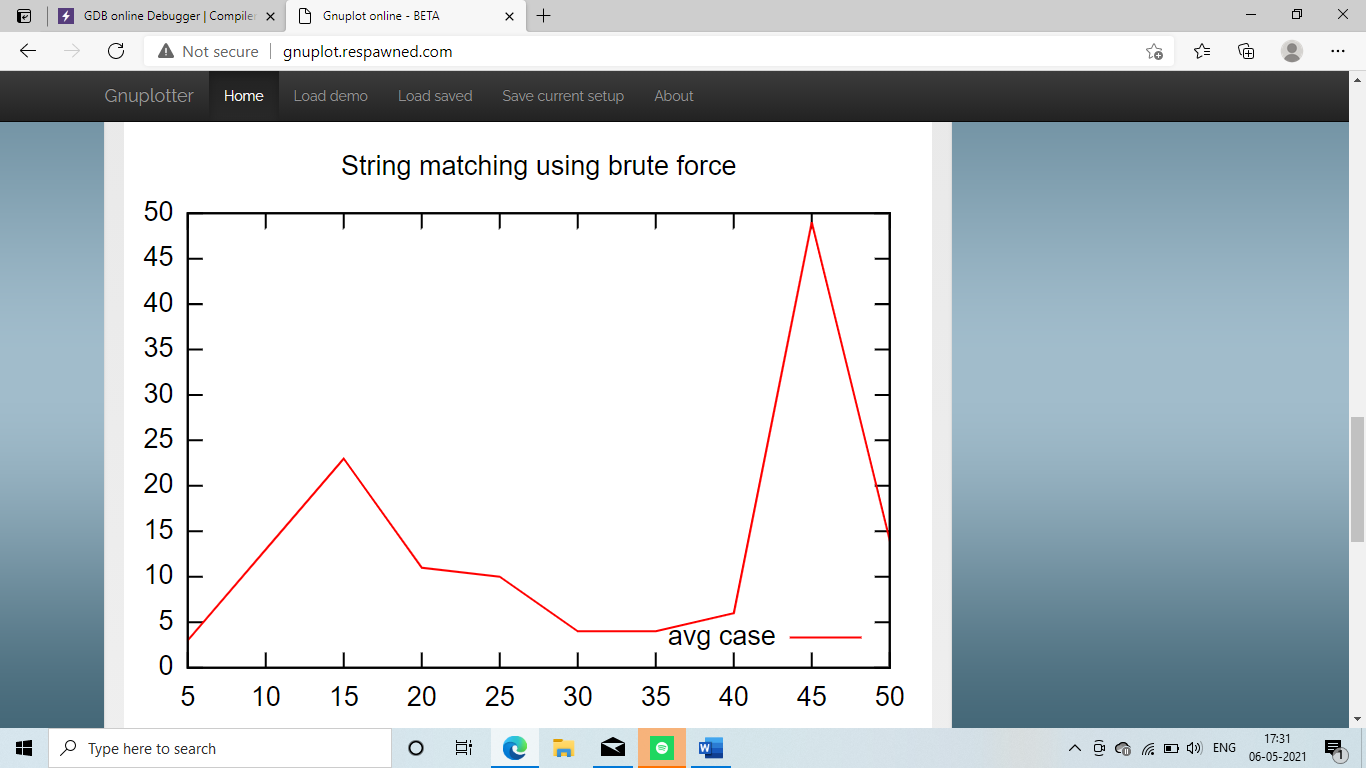
35 4

40 6

45 49

50 14

Graph :



Worst case :

n opcount

5 9

10 24

15 39

20 54

25 69

30 84

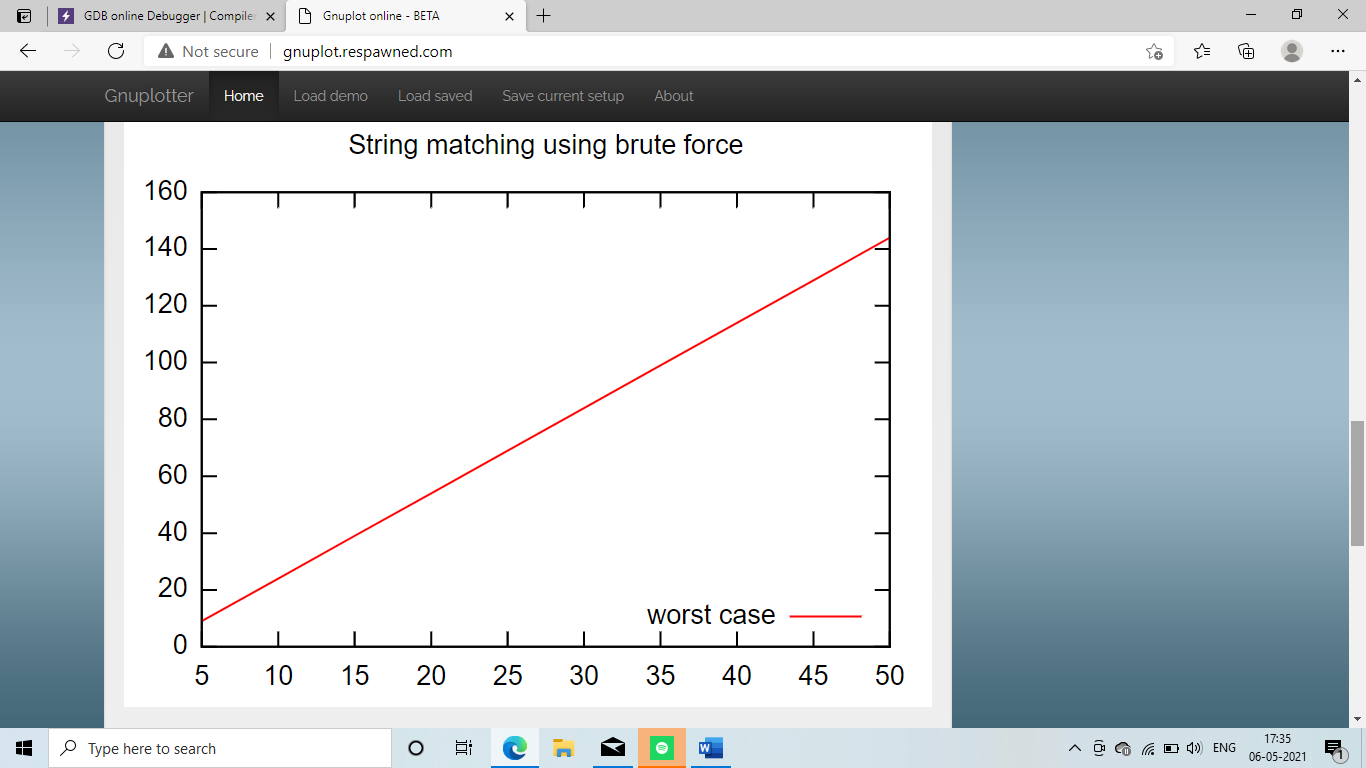
35 99

40 114

45 129

50 144

Graph :



Exponentiation using divide and conquer technique

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int count;

int expo(int x,int y)

{

count++;

if(y==0)

return 1;

if(y%2==0)

return expo(x,y/2)\*expo(x,y/2);

else

return x\*expo(x,y/2)\*expo(x,y/2);

}

void main()

{

int x,y,result;

FILE \*exponent;

system("rm exponent.txt");

srand(time(NULL));

x=rand()%10;

exponent=fopen("exponent.txt","a");

for(y=1;y<=50;y=y+1)

{

count=0;

result=expo(x,y);

fprintf(exponent,"%d\t%d\n",y,count);

}

fclose(exponent);

}

Output file :

n opcount

1 3

2 7

3 7

4 15

5 15

6 15

7 15

8 31

9 31

10 31

11 31

12 31

13 31

14 31

15 31

16 63

17 63

18 63

19 63

20 63

21 63

22 63

23 63

24 63

25 63

26 63

27 63

28 63

29 63

30 63

31 63

32 127

33 127

34 127

35 127

36 127

37 127

38 127

39 127

40 127

41 127

42 127

43 127

44 127

45 127

46 127

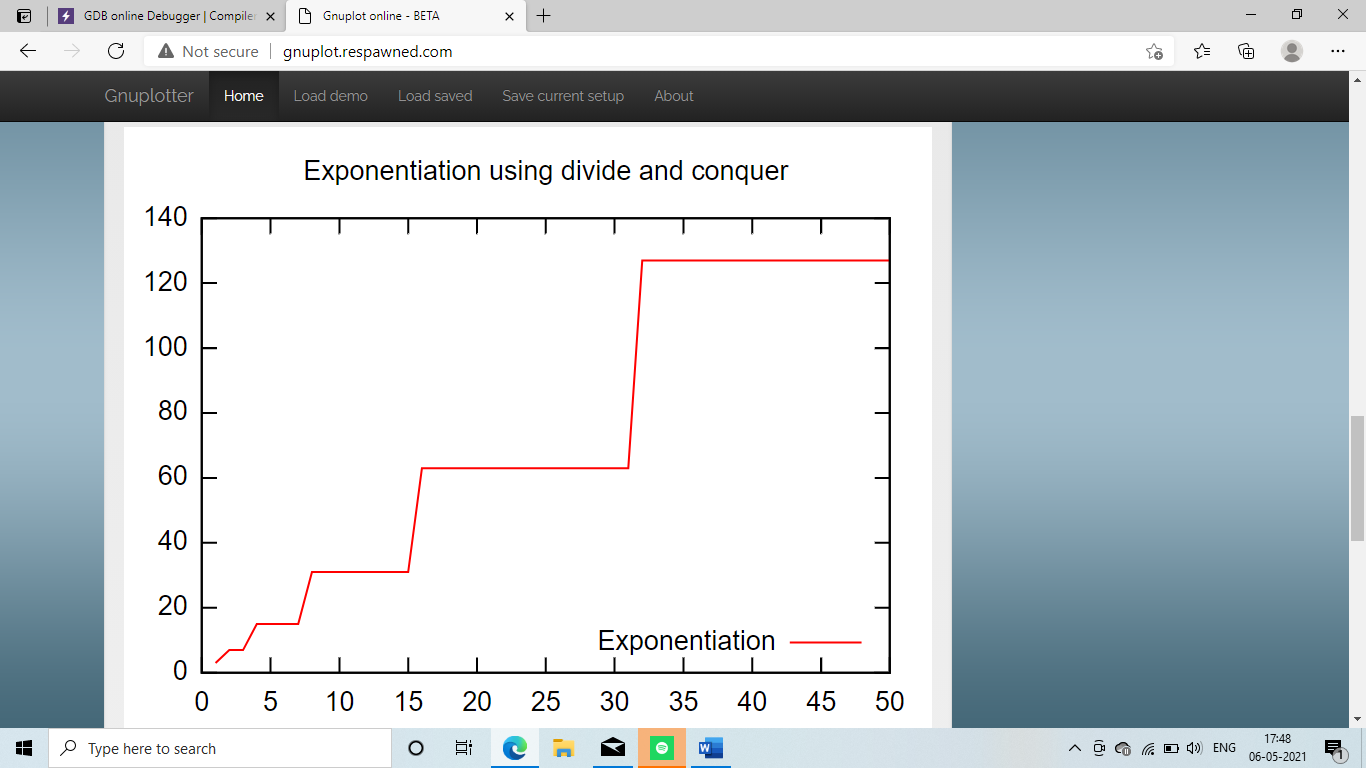
47 127

48 127

49 127

50 127

Graph :



Maximum element in an array using divide and conquer

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int count=0;

int maxele(int [],int,int);

void main()

{

int n,i,max,\*a;

FILE \*maximum;

//system("rm maximum.txt");

maximum=fopen("maximum.txt","a");

for(n=10;n<=200;n=n+10)

{

count=0;

a=(int \*)malloc(n\*sizeof(int));

srand(time(NULL));

for(i=0;i<n;i++)

{

a[i]=rand()%100;

}

max=maxele(a,0,n);

fprintf(maximum,"%d\t%d\n",n,count);

}

fclose(maximum);

}

int maxele(int a[],int f,int r)

{

count++;

if(r-f==1)

return a[f];

int m,y,z;

m=(f+r)/2;

y=maxele(a,f,m);

z=maxele(a,m,r);

return y>z?y:z;

}

Output file :

n opcount

10 19

20 39

30 59

90 179

100 199

110 219

120 239

130 259

140 279

150 299

160 319

170 339

180 359

190 379

200 399

Graph :

