6CS005 Learning Journal - Semester 1 2019/20

Bishal Baba Maka 1930143

# Table of Contents

[Table of Contents 1](#_Toc527650969)

[1 POSIX Threads 2](#_Toc527650970)

[1.1 Password Cracking 2](#_Toc527650971)

[1.2 Image Processing 2](#_Toc527650972)

[1.3 Linear Regression 2](#_Toc527650973)

[2 CUDA 3](#_Toc527650974)

[2.1 Password Cracking 3](#_Toc527650975)

[2.2 Image Processing 3](#_Toc527650976)

[2.3 Linear Regression 3](#_Toc527650977)

[3 MPI 4](#_Toc527650978)

[3.1 Password Cracking 4](#_Toc527650979)

[3.2 Image Processing 4](#_Toc527650980)

[3.3 Linear Regression 4](#_Toc527650981)

[4 Verbose Repository Log 5](#_Toc527650982)

# MPI

## Password Cracking

Insert a table that shows running times for the original and MPI versions.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <crypt.h>  #include <time.h>  #include <mpi.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Demonstrates how to crack an encrypted password using a simple  "brute force" algorithm. Works on passwords that consist only of 2 uppercase  letters and a 2 digit integer. Your personalised data set is included in the  code.  Compile with:  mpicc -o password-mpi password-mpi.c -lrt -lcrypt  If you want to analyse the results then use the redirection operator to send  output to a file that you can view using an editor or the less utility:  mpirun -n 3 ./password-mpi  Dr Kevan Buckley, University of Wolverhampton, 2018  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  int n\_passwords = 4;  char \*passwords\_enc[] = {  "$6$KB$36REBSFzq1/dui1N6SgVpJNWbH7.H09nEmDF84hZhfeauI13ShDyaMT/d1tPMVh35NJKKkQrT6OIYDS9VLrIY0",  "$6$KB$TShKJUQHJ/epFrqd8PVtnjB9lsjrRx6Qwv6fY/9ZOr0NHkH3sFrkc0IZlE8ricgOeQjFu56VSq9.BV2kNSpAg.",  "$6$KB$ielcaeww2oJY5cJuYcgdrtKXuWecpbQLcbvNvI32cdGc1SW17x8csdrpeeUN/Pk8i/aXDLw1Kv4ON/obBDmcT1",  "$6$KB$V51Vk5oaq9jlHAOGUICSX1DRt6gOTPS2N9rx8KI9UPCJtcpdSHV5rmY3c9/B5Ow1eQ62.B0x7.RIlljC9ujU71"  };  /\*\*  Required by lack of standard function in C.  \*/  void substr(char \*dest, char \*src, int start, int length){  memcpy(dest, src + start, length);  \*(dest + length) = '\0';  }  void function\_1(char \*salt\_and\_encrypted){  int b,m,c;  char salt[7];  char plain[7];  char \*enc;  int count = 0;  substr(salt, salt\_and\_encrypted, 0, 6);  for(b='A';b<='M'; b++){  for(m='A'; m<='Z'; m++){  for(c=0; c<=99; c++){  printf("Instance 1:");  sprintf(plain, "%c%c%02d", b,m,c);  enc = (char \*) crypt(plain, salt);  count++;  if(strcmp(salt\_and\_encrypted, enc) == 0){  printf("#%-8d%s %s\n", count, plain, enc);  }  else {  printf("%-8d%s %s\n", count, plain, enc);  }  }  }  }  printf("%d solutions explored\n", count);  }  void function\_2(char \*salt\_and\_encrypted){  int b,m,c;  char salt[7];  char plain[7];  char \*enc;  int count = 0;  substr(salt, salt\_and\_encrypted, 0, 6);  for(b='N';b<='Z'; b++){  for(m='A'; m<='Z'; m++){  for(c=0; c<=99; c++){  printf("Instance 2:");  sprintf(plain, "%c%c%02d", b,m,c);  enc = (char \*) crypt(plain, salt);  count++;  if(strcmp(salt\_and\_encrypted, enc) == 0){  printf("#%-8d%s %s\n", count, plain, enc);  }  else{  printf("%-8d%s %s\n", count, plain, enc);  }  }  }  }  printf("%d solutions explored\n", count);  }  int time\_difference(struct timespec \*start, struct timespec \*finish,  long long int \*difference) {  long long int ds = finish->tv\_sec - start->tv\_sec;  long long int dn = finish->tv\_nsec - start->tv\_nsec;  if(dn < 0 ) {  ds--;  dn += 1000000000;  }  \*difference = ds \* 1000000000 + dn;  return !(\*difference > 0);  }  int main() {  struct timespec start, finish;  long long int time\_elapsed;  clock\_gettime(CLOCK\_MONOTONIC, &start);  int size, rank;  int i;  MPI\_Init(NULL, NULL);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  if(size != 3){  if (rank ==0){  printf("this program needs to run on exactly 3 process to run\n ");  }  }else{  if(rank == 0){  int r;  MPI\_Send(&r, 1, MPI\_INT, 1, 0, MPI\_COMM\_WORLD);  MPI\_Send(&r, 1, MPI\_INT, 2, 0, MPI\_COMM\_WORLD);  }else if(rank==1){  int num;  MPI\_Recv(&num, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);  for(i=0; i<n\_passwords; i<i++){  function\_1(passwords\_enc[i]);  }  } else{  int num;  MPI\_Recv(&num, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);  for(i=0; i<n\_passwords; i<i++){  function\_2(passwords\_enc[i]);  }  }  }  MPI\_Finalize();  clock\_gettime(CLOCK\_MONOTONIC, &finish);  time\_difference(&start, &finish, &time\_elapsed);    if (rank ==0){  printf("Time elapsed was %lldns or %0.9lfs\n", time\_elapsed,  (time\_elapsed/1.0e9));  }  return 0;  } |

|  |  |
| --- | --- |
| Original Version | MPI version |
|  |  |

Write a short analysis of the results

## Image Processing

Insert a table that shows running times for the original and MPI versions.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <GL/glut.h>  #include <GL/gl.h>  #include <malloc.h>  #include <signal.h>  #include <mpi.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Displays two grey scale images. On the left is an image that has come from an  image processing pipeline, just after colour thresholding. On the right is  the result of applying an edge detection convolution operator to the left  image. This program performs that convolution.    Things to note:  - A single unsigned char stores a pixel intensity value. 0 is black, 256 is  white.  - The colour mode used is GL\_LUMINANCE. This uses a single number to  represent a pixel's intensity. In this case we want 256 shades of grey,  which is best stored in eight bits, so GL\_UNSIGNED\_BYTE is specified as  the pixel data type.    To compile adapt the code below wo match your filenames:  mpicc -o MPI-img MPI-img.c -lm -lglut -lGL  To run:  mpirun -n 5 ./MPI-img    Dr Kevan Buckley, University of Wolverhampton, 2018  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #define width 100  #define height 72  unsigned char image[], results[width \* height];  int startIndex, endIndex;  int time\_difference(struct timespec \*start,struct timespec \*finish, long long int \*difference) ;  void detect\_edges(unsigned char \*in, unsigned char \*out);  void sigint\_callback(int signal\_number);  static void display();  void tidy\_and\_exit();    void edges\_detect(unsigned char \*in, unsigned char \*out) {  int i;  int n\_pixels = width \* height;  for(i=0;i<n\_pixels;i++) {  int k,e; // the pixel of interest  int a, b, c, d; // the pixels adjacent to x,y used for the calculation  int r; // the result of calculate    e= i / width;  k= i - (width \* e);  if (k == 0 || e == 0 || k == width - 1 || e == height - 1) {  results[i] = 0;  } else {  a = i + width;  b = i - 1;  c = i + 1;  d = i - width;  r = (in[i] \* 4) + (in[a] \* -1) + (in[b] \* -1) + (in[c] \* -1)  + (in[d] \* -1);  if (r > 0) { // if the result is positive this is an edge pixel  out[i] = 255;  } else {  out[i] = 0;  }  }  }  }  void tidy\_and\_exit() {  exit(0);  }  void sigint\_callback(int signal\_number){  printf("\nInterrupt from keyboard\n");  tidy\_and\_exit();  }  static void display() {  glClear(GL\_COLOR\_BUFFER\_BIT);  glRasterPos4i(-1, -1, 0, 1);  glDrawPixels(width, height, GL\_LUMINANCE, GL\_UNSIGNED\_BYTE, image);  glRasterPos4i(0, -1, 0, 1);  glDrawPixels(width, height, GL\_LUMINANCE, GL\_UNSIGNED\_BYTE, results);  glFlush();  }  static void key(unsigned char key, int x, int y) {  switch(key){  case 27: // escape  tidy\_and\_exit();  break;  default:  printf("\nPress escape to exit\n");  break;  }  }  int time\_difference(struct timespec \*start,struct timespec \*finish, long long int \*difference) {  long long int ds = finish->tv\_sec - start->tv\_sec;  long long int dn = finish->tv\_nsec - start->tv\_nsec;  if(dn < 0 ) {  ds--;  dn += 1000000000;  }  \*difference = ds \* 1000000000 + dn;  return !(\*difference > 0);  }  int main(int argc, char \*\*argv) {  signal(SIGINT, sigint\_callback);    printf("image dimensions %dx%d\n", width, height);  int size, rank;    MPI\_Init(NULL, NULL);  MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);  if (size != 5){  if(rank != 0) {  printf("This program needs to run on exactly 4 processes\n");  exit(-1);  }  }  if (rank ==0){  startIndex = 0;  endIndex = 1799;  struct timespec start, finish;  long long int time\_elapsed;  clock\_gettime(CLOCK\_MONOTONIC, &start);    /\*struct timespec start, finish;  long long int time\_elapsed;  clock\_gettime(CLOCK\_MONOTONIC, &start);\*/  edges\_detect(image, results);  clock\_gettime(CLOCK\_MONOTONIC, &finish);  time\_difference(&start, &finish, &time\_elapsed);  printf("Time elapsed was %lldns or %0.9lfs\n", time\_elapsed, (time\_elapsed/1.0e9));      MPI\_Recv(&results[1800], 1800, MPI\_UNSIGNED\_CHAR, 1, 0, MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);  MPI\_Recv(&results[3600], 1800, MPI\_UNSIGNED\_CHAR, 2, 0, MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);  MPI\_Recv(&results[5400], 1800, MPI\_UNSIGNED\_CHAR, 3, 0, MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);    glutInit(&argc, argv);  glutInitWindowSize(width \* 2,height);  glutInitDisplayMode(GLUT\_SINGLE | GLUT\_LUMINANCE);    glutCreateWindow("6CS005 Image Progessing Courework");  glutDisplayFunc(display);  glutKeyboardFunc(key);  glClearColor(0.0, 1.0, 0.0, 1.0);  glutMainLoop();  tidy\_and\_exit();    return 0;  }  else if (rank == 1){  startIndex = 1800;  endIndex =3599;    edges\_detect(image, results);  MPI\_Send(&results[1800], 1800, MPI\_UNSIGNED\_CHAR, 0, 0, MPI\_COMM\_WORLD);  }  else if (rank == 2){  startIndex = 3600;  endIndex =5399;    edges\_detect(image, results);  MPI\_Send(&results[3600], 1800, MPI\_UNSIGNED\_CHAR, 0, 0, MPI\_COMM\_WORLD);  }  else if (rank == 3){  startIndex = 5400;  endIndex =7199;    edges\_detect(image, results);  MPI\_Send(&results[5400], 1800, MPI\_UNSIGNED\_CHAR, 0, 0, MPI\_COMM\_WORLD);  }  MPI\_Finalize();  //clock\_gettime(CLOCK\_MONOTONIC, &finish);  //time\_difference(&start, &finish, &difference);  //printf("run lasted %9.5lfs\n", difference/1000000000.0);  return 0;  }  unsigned char image[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,0,0,  255,0,0,0,0,255,255,0,0,255,255,0,0,0,0,0,0,0,0,  0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,  255,255,255,255,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,0,255,255,255,255,255,255,255,0,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,255,0,0,0,0,0,0,0,0,  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,0,255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,  255,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,0,0,0,0,0,0,255,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,0,  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,0,0,0,  0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,  255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  0,0,0,0,0,0,0,0,0,255,255,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,255,0,255,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,  255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,  255,255,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,255,  255,255,255,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,0,0,255,0,255,255,0,0,0,255,0,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,0,0,0,0,0,0,0,0,255,0,0,255,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,255,255,  255,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,0,0,0,0,0,0,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,  0,0,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,0,0,255,0,0,255,255,255,255,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,  255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,  0,0,0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,0,0,0,0,0,0,255,255,255,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,  0,0,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,0,  0,0,0,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,255,255,255,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,255,255,255,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,  0,0,0,0,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,0,0,0,0,0,255,255,255,255,255,0,0,0,0,255,  255,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,  255,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,255,255,  0,0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,  255,255,255,255,255,0,255,255,255,0,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,  0,0,0,0,255,255,255,255,255,0,255,255,255,255,0,0,0,0,0,  0,0,255,0,0,0,0,255,0,0,0,0,0,0,255,0,255,255,255,  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,  255,0,0,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,0,0,0,0,255,255,255,255,0,255,255,255,255,0,  0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,  255,0,0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,0,255,  255,255,255,0,255,255,0,0,0,0,0,255,255,0,0,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,  0,255,255,0,255,255,255,255,0,0,255,0,0,0,0,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,0,255,0,255,255,0,0,255,255,0,0,0,0,0,0,0,0,  0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,0,0,255,255,255,0,0,255,255,0,0,0,  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,255,0,0,  255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,  0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,255,255,  0,255,255,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,255,  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,0,0,0,0,0,0,0,0,0,0,  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,255,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,0,0,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,255,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,0,0,255,255,255,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,0,255,0,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,  0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,255,0,  0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,  0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,0,255,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,0,0,0,0,0,0,0,0,0,255,0,255,0,0,0,0,0,0,  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,255,  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,255,0,  0,0,0,0,0,0,0,0,255,255,255,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,0,0,0,255,255,0,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  0,0,255,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,  0,0,0,0,0,0,0,0,0,0,255,0,255,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  0,0,255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,255,255,  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  255,255,255,255,255,255,255,255,0,255,255,255,0,0,0,0,0,0,0,  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,255,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,255,255,255,0,255,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,  255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,  0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,255,0,0,255,255,255,0,0,0,0,  0,0,0,0,0,255,255,255,255,255,0,255,255,255,255,0,255,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,  0,0,0,0,0,0,0,0,0,0,255,255,255,255,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  }; |

|  |  |
| --- | --- |
| Original version | MPI version |
|  |  |

Write a short analysis of the results

## Linear Regression

Insert a table that shows running times for the original and MPI versions.

|  |
| --- |
| #include <stdio.h>  #include <math.h>  #include <time.h>  #include <mpi.h>  #include <malloc.h>  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* This program takes an initial estimate of m and c and finds the associated  \* rms error. It is then as a base to generate and evaluate 8 new estimates,  \* which are steps in different directions in m-c space. The best estimate is  \* then used as the base for another iteration of "generate and evaluate". This  \* continues until none of the new estimates are better than the base. This is  \* a gradient search for a minimum in mc-space.  \*  \* To compile:  \* mpicc -o MPI-linear-regression MPI-linear-regression.c -lm  \*  \* To run:  \* mpirun -n 9 ./MPI-linear-regression  \*  \* Dr Kevan Buckley, University of Wolverhampton, 2018  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  typedef struct point\_t  {  double k;  double s;  } point\_t;  int n\_data = 1000;  point\_t data[];  double residual\_error (double k, double s, double m, double c)  {  double e = (m \* k) + c - s;  return e \* e;  }  double rms\_error (double m, double c)  {  int i;  double mean;  double error\_sum = 0;  for (i = 0; i < n\_data; i++)  {  error\_sum += residual\_error (data[i].k, data[i].s, m, c);  }  mean = error\_sum / n\_data;  return sqrt (mean);  }    int time\_difference(struct timespec \*start, struct timespec \*finish,  long long int \*difference) {  long long int ds = finish->tv\_sec - start->tv\_sec;  long long int dn = finish->tv\_nsec - start->tv\_nsec;  if(dn < 0 ) {  ds--;  dn += 1000000000;  }  \*difference = ds \* 1000000000 + dn;  return !(\*difference > 0);  }  int main () {  struct timespec start, finish;    long long int time\_elapsed;  clock\_gettime(CLOCK\_MONOTONIC, &start);  int rank, size;  int i;  double bm = 1.3;  double bc = 10;  double be;  double dm[8];  double dc[8];  double e[8];  double step = 0.01;  double best\_error = 999999999;  int best\_error\_i;  int minimum\_found = 0;  double pError = 0;  double baseMC[2];  double om[] = { 0, 1, 1, 1, 0, -1, -1, -1 };  double oc[] = { 1, 1, 0, -1, -1, -1, 0, 1 };  MPI\_Init (NULL, NULL);  MPI\_Comm\_size (MPI\_COMM\_WORLD, &size);  MPI\_Comm\_rank (MPI\_COMM\_WORLD, &rank);  be = rms\_error (bm, bc);  if (size != 9)  {  if (rank == 0)  {  printf  ("This program is needs to run with exactly 9 processes.\n");  return 0;  }  }  while (!minimum\_found)  {  if (rank != 0)  {  i = rank - 1;  dm[i] = bm + (om[i] \* step);  dc[i] = bc + (oc[i] \* step);  pError = rms\_error (dm[i], dc[i]);  MPI\_Send (&pError, 1, MPI\_DOUBLE, 0, 0, MPI\_COMM\_WORLD);  MPI\_Send (&dm[i], 1, MPI\_DOUBLE, 0, 0, MPI\_COMM\_WORLD);  MPI\_Send (&dc[i], 1, MPI\_DOUBLE, 0, 0, MPI\_COMM\_WORLD);  MPI\_Recv (&bm, 1, MPI\_DOUBLE, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  MPI\_Recv (&bc, 1, MPI\_DOUBLE, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  MPI\_Recv (&minimum\_found, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  }  else  {  for (i = 1; i < size; i++)  {  MPI\_Recv (&pError, 1, MPI\_DOUBLE, i, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  MPI\_Recv (&dm[i-1], 1, MPI\_DOUBLE, i, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  MPI\_Recv (&dc[i-1], 1, MPI\_DOUBLE, i, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);  if (pError < best\_error)  {  best\_error = pError;  best\_error\_i = i - 1;  }  }  printf ("The best m,c is %lf,%lf with an error %lf in direction: %d\n",  dm[best\_error\_i], dc[best\_error\_i], best\_error, best\_error\_i);  if (best\_error < be)  {  be = best\_error;  bm = dm[best\_error\_i];  bc = dc[best\_error\_i];  }  else  {  minimum\_found = 1;  }  for (i = 1; i < size; i++)  {  MPI\_Send (&bm, 1, MPI\_DOUBLE, i, 0, MPI\_COMM\_WORLD);  MPI\_Send (&bc, 1, MPI\_DOUBLE, i, 0, MPI\_COMM\_WORLD);  MPI\_Send (&minimum\_found, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);  }  }  }  if(rank==0) {    printf ("The minimum searched values for m,c is %lf,%lf with error %lf\n are:", bm, bc, be);  clock\_gettime(CLOCK\_MONOTONIC, &finish);  time\_difference(&start, &finish, &time\_elapsed);  printf("Time elapsed taken was %lldns or %0.9lfs\n", time\_elapsed,  (time\_elapsed/1.0e9));  }  MPI\_Finalize();  return 0;  }  point\_t data[] = {  {83.12,144.47},{65.27,114.80},{65.17,89.01},{68.57,122.90},  {77.57,136.93},{79.84,146.56},{84.42,123.51},{65.34,106.22},  {82.20,120.33},{65.35,142.11},{24.06,53.94},{35.61,87.53},  { 2.02,22.75},{44.01,89.41},{85.58,141.52},{54.14,88.90},  {35.94,84.11},{22.86,45.76},{75.88,111.25},{54.49,105.83},  {94.65,139.29},{74.97,140.29},{46.31,94.00},{48.12,108.88},  {99.29,146.97},{86.76,135.87},{70.11,120.41},{ 5.01,35.32},  {84.56,147.46},{ 0.19,39.41},{13.16,49.52},{34.11,93.57},  {78.99,108.24},{38.38,81.59},{79.20,115.25},{84.38,146.00},  {92.49,166.93},{19.70,61.69},{23.14,82.49},{13.97,44.80},  { 2.30,51.01},{15.33,34.49},{64.82,106.29},{39.99,76.65},  {85.93,162.61},{95.23,172.35},{11.05,60.11},{53.84,106.95},  {71.11,135.65},{33.67,88.76},{ 2.41,41.07},{52.19,108.83},  {30.21,57.75},{69.24,132.80},{96.44,157.86},{87.85,133.87},  {15.51,56.56},{53.81,106.32},{50.03,77.59},{77.05,136.93},  {37.29,81.30},{41.74,95.49},{53.91,109.94},{41.20,67.23},  {76.87,124.78},{39.99,82.29},{21.12,55.37},{34.62,65.13},  {20.91,51.88},{76.70,118.05},{ 4.76,45.66},{ 2.29,26.88},  {27.19,59.89},{ 6.82,36.36},{32.36,78.26},{48.72,99.14},  {80.55,127.01},{91.69,150.94},{ 9.68,29.41},{90.74,165.08},  {35.58,70.65},{90.86,166.10},{99.52,157.98},{15.66,47.55},  {45.23,88.34},{63.46,112.27},{64.21,115.27},{86.10,146.87},  {72.98,119.38},{31.78,67.38},{73.97,135.76},{24.43,70.15},  {74.86,135.38},{18.98,50.05},{49.32,106.88},{93.39,154.91},  { 1.29,39.63},{10.92,61.03},{35.04,64.55},{57.66,111.38},  {42.04,96.64},{ 8.79,40.02},{92.43,147.28},{49.08,85.76},  {30.62,85.66},{51.41,97.98},{88.25,141.92},{27.07,61.14},  {34.88,83.12},{90.82,151.63},{55.07,106.28},{25.73,62.03},  {34.53,63.56},{ 6.61,34.03},{15.62,50.85},{15.32,67.76},  {69.03,114.54},{32.46,56.91},{69.37,123.90},{10.78,57.26},  {10.53,31.37},{53.23,109.49},{ 7.26,44.18},{15.90,63.21},  { 8.53,36.85},{57.16,109.43},{80.74,122.57},{ 7.25,44.88},  {87.53,144.92},{90.70,165.27},{61.17,108.23},{53.14,111.23},  {94.75,138.45},{ 7.60,42.08},{18.83,76.22},{13.48,71.77},  { 0.66,39.45},{35.94,87.05},{88.24,169.85},{22.00,70.26},  {93.97,144.15},{93.09,164.94},{41.88,90.98},{35.68,63.90},  {93.69,160.24},{22.20,53.28},{79.69,118.82},{27.57,57.90},  {24.98,72.67},{86.50,133.90},{40.28,86.21},{14.60,48.01},  {72.54,139.19},{55.30,79.54},{ 3.81,33.25},{ 5.68,53.66},  {17.39,44.50},{82.43,123.95},{26.21,57.88},{50.93,102.91},  {41.54,78.81},{36.41,65.17},{39.67,84.96},{74.19,130.02},  {79.23,147.24},{ 5.43,43.11},{59.04,92.40},{ 4.77,21.65},  {62.12,113.31},{80.55,133.55},{42.32,75.65},{83.01,131.90},  {39.06,88.34},{98.75,175.85},{31.87,62.41},{58.73,96.47},  {10.18,53.65},{12.05,47.02},{77.15,116.12},{17.71,57.77},  {82.98,134.75},{18.11,37.59},{32.30,74.54},{81.96,143.75},  {11.77,47.90},{24.43,78.01},{60.70,116.42},{72.05,123.46},  {42.29,75.74},{ 9.64,53.11},{ 3.20,41.20},{75.68,127.51},  { 7.67,38.82},{ 9.55,45.92},{ 6.22,55.99},{15.01,53.21},  { 2.50,17.99},{30.97,64.75},{15.92,58.06},{39.77,79.31},  {30.30,80.76},{75.71,133.13},{18.68,54.70},{14.33,48.80},  {65.29,112.12},{85.98,156.29},{68.20,115.16},{76.18,127.58},  {12.05,52.54},{ 1.45,26.32},{51.07,91.58},{70.45,131.48},  {46.34,110.44},{86.40,140.67},{62.22,107.05},{39.48,96.73},  {59.28,114.38},{85.33,140.73},{21.85,63.28},{55.32,96.88},  {54.90,99.09},{81.45,134.43},{94.99,152.75},{60.61,91.15},  {85.61,132.87},{54.72,105.30},{ 9.85,37.72},{85.74,133.99},  {30.19,79.45},{87.18,142.65},{27.50,68.06},{48.21,81.13},  {89.60,139.38},{20.45,61.03},{60.56,101.17},{88.41,139.78},  {84.60,146.42},{25.34,45.91},{32.69,104.43},{13.63,53.03},  {80.26,124.62},{97.15,147.49},{99.16,177.78},{81.31,127.71},  {88.58,136.47},{24.77,59.82},{96.93,160.71},{51.92,102.46},  {27.33,67.99},{92.40,156.65},{87.22,135.40},{ 8.66,33.01},  {79.02,137.74},{92.16,158.93},{70.14,117.38},{31.39,83.34},  {98.54,150.47},{81.39,145.14},{32.19,90.89},{49.53,82.60},  {83.19,147.94},{65.68,121.26},{19.73,73.98},{19.26,39.84},  {68.81,127.82},{21.93,64.48},{22.98,67.44},{ 8.19,35.21},  {83.08,134.02},{69.30,124.24},{19.40,46.96},{64.13,120.93},  {61.91,118.90},{31.92,72.59},{97.06,157.02},{69.68,131.99},  {64.02,120.20},{86.75,141.47},{48.62,98.35},{62.34,118.54},  {23.10,73.71},{ 3.22,24.94},{47.03,98.28},{86.10,129.82},  {17.62,41.43},{20.60,62.70},{25.56,79.02},{98.74,168.44},  {25.25,68.33},{ 0.26,17.74},{73.72,125.70},{62.70,101.61},  {86.10,144.15},{ 7.59,38.21},{65.71,118.18},{57.83,104.28},  {48.00,91.86},{59.53,110.64},{75.08,131.55},{66.96,113.45},  {23.44,41.93},{ 7.22,33.51},{22.13,70.49},{20.24,70.87},  {36.57,59.85},{22.89,50.80},{88.83,128.03},{54.08,109.80},  {20.87,65.63},{80.15,132.14},{91.71,142.11},{12.37,46.56},  {31.09,82.71},{ 9.54,28.65},{16.74,44.18},{37.07,73.24},  { 1.67,41.10},{ 0.29,12.09},{34.05,80.10},{64.07,112.30},  {64.66,110.15},{21.74,62.28},{74.39,129.73},{53.67,90.13},  {75.14,147.83},{42.98,82.02},{66.29,121.10},{57.34,102.40},  {96.75,152.13},{13.36,48.35},{21.05,73.53},{81.77,135.48},  {88.21,171.75},{51.53,98.91},{21.88,63.71},{89.27,145.47},  {67.70,125.26},{72.69,126.45},{27.77,58.71},{69.38,115.18},  { 2.59,19.50},{93.93,149.24},{ 4.84,44.09},{19.21,43.14},  {10.58,38.47},{41.51,82.49},{88.02,148.21},{55.22,114.17},  {12.69,79.85},{91.81,160.45},{99.68,162.60},{62.74,103.63},  {10.21,47.93},{ 5.21,28.37},{89.57,148.01},{28.42,54.46},  {61.03,88.74},{73.04,120.93},{71.30,131.03},{ 6.42,27.57},  {82.06,114.82},{50.07,89.66},{76.06,137.34},{69.25,116.77},  {72.62,110.20},{ 8.88,48.25},{24.03,73.68},{52.59,102.23},  {84.77,139.15},{96.75,154.31},{70.15,122.87},{93.18,166.62},  { 6.17,58.46},{92.22,158.34},{74.61,131.25},{67.46,119.20},  {22.98,57.20},{37.45,86.95},{ 1.97,39.59},{48.29,116.20},  {52.60,109.07},{24.17,56.13},{58.56,116.56},{32.87,65.50},  { 0.34,43.67},{87.72,142.21},{37.41,62.88},{64.08,127.92},  {42.54,79.79},{35.53,88.48},{ 2.57,23.24},{77.80,122.09},  { 4.19,35.89},{11.53,28.55},{62.03,82.21},{55.15,93.33},  {63.96,120.79},{73.17,129.77},{57.12,113.60},{32.89,92.86},  {27.89,70.41},{39.21,74.83},{77.58,129.76},{77.44,149.05},  { 2.87,10.13},{11.11,44.31},{77.46,144.46},{45.30,100.95},  { 4.69,30.94},{89.47,157.53},{ 7.61,44.77},{23.09,74.16},  {91.49,156.06},{11.20,52.40},{21.47,77.05},{86.58,141.10},  {24.07,57.57},{76.46,137.23},{84.23,120.97},{96.42,157.37},  {98.02,155.25},{99.42,159.62},{12.67,68.56},{36.27,92.72},  {16.08,50.55},{29.05,58.27},{24.65,58.31},{22.59,71.18},  {54.34,115.03},{44.53,96.50},{50.73,109.29},{10.75,45.32},  {62.06,126.81},{12.61,62.62},{21.94,50.52},{86.83,160.25},  { 9.03,51.65},{73.37,127.89},{54.41,107.85},{95.96,172.35},  {69.67,130.26},{48.73,103.54},{62.30,113.08},{19.39,78.51},  {77.40,124.44},{ 1.63,34.05},{90.02,152.89},{64.47,110.81},  {47.10,103.92},{64.92,116.32},{42.67,73.30},{48.06,76.96},  {35.45,65.22},{98.35,158.55},{17.10,60.38},{29.75,70.75},  {85.75,135.77},{48.27,88.32},{42.05,73.57},{88.04,146.92},  { 9.72,34.51},{66.61,120.50},{52.60,91.06},{78.80,127.29},  {11.69,48.24},{ 2.59,39.39},{84.26,130.65},{10.82,43.81},  {97.33,173.24},{95.78,157.66},{51.35,81.72},{83.75,136.31},  {72.98,114.92},{70.67,120.19},{90.19,147.54},{39.23,71.88},  {35.17,78.15},{84.31,136.47},{ 4.96,37.06},{13.96,55.78},  {51.70,107.90},{48.21,98.95},{90.61,142.67},{ 4.39,50.63},  {76.09,120.85},{72.86,132.97},{69.73,118.54},{60.33,93.71},  { 5.07,42.46},{20.73,60.27},{42.45,89.87},{80.47,166.56},  {16.49,68.34},{97.12,153.22},{19.75,50.44},{75.75,121.87},  {16.84,69.99},{16.59,56.79},{22.78,65.78},{78.48,135.35},  {70.14,122.63},{39.36,74.32},{21.60,75.60},{66.51,101.96},  {62.88,107.89},{50.24,88.20},{60.77,106.24},{86.21,148.74},  { 9.38,44.95},{87.93,141.50},{13.25,49.13},{50.99,106.87},  {84.74,145.24},{91.76,140.41},{81.99,130.91},{58.39,94.20},  {84.02,153.63},{55.36,92.79},{ 2.69,36.03},{65.84,115.04},  {52.09,98.57},{16.14,46.02},{18.37,39.39},{49.37,96.53},  {43.87,80.59},{80.77,130.01},{45.87,98.61},{10.53,37.07},  {46.18,93.03},{24.75,71.96},{85.19,138.24},{66.97,129.60},  { 2.19,44.38},{68.15,89.75},{60.75,117.13},{15.45,62.88},  {59.82,93.68},{14.43,51.77},{46.38,75.94},{86.99,133.36},  {80.16,115.98},{71.51,113.22},{ 8.43,45.23},{36.84,81.44},  {99.22,143.60},{26.46,59.92},{92.97,161.39},{81.44,120.67},  { 4.33,31.81},{81.67,130.81},{34.26,76.67},{76.71,150.31},  {77.99,131.09},{45.96,90.46},{25.87,59.28},{51.79,104.69},  {14.95,41.47},{22.07,67.88},{84.04,152.63},{63.10,114.30},  {94.30,147.86},{56.55,108.74},{ 8.29,55.81},{30.76,84.68},  {68.20,133.71},{ 3.29,50.95},{89.16,145.76},{31.10,67.81},  { 0.88,41.80},{ 7.31,39.34},{51.82,103.09},{13.69,35.21},  {54.12,109.39},{41.60,79.94},{44.78,91.74},{ 0.83,42.82},  {88.24,138.49},{62.16,110.68},{ 7.00,25.60},{80.07,157.43},  {19.82,51.33},{11.07,53.28},{77.57,133.32},{94.77,146.08},  {19.43,67.02},{99.17,165.99},{32.86,70.06},{75.29,142.96},  {37.18,96.22},{37.29,112.25},{84.78,143.59},{93.33,138.44},  {74.44,121.57},{19.51,51.21},{82.81,123.17},{14.24,68.89},  { 3.64,29.43},{18.79,56.15},{97.75,161.17},{71.42,119.80},  { 5.68,42.40},{65.07,120.59},{53.09,109.96},{64.88,117.08},  {64.22,114.47},{22.87,69.56},{26.46,54.11},{38.98,79.57},  {89.71,145.31},{50.80,98.09},{50.17,95.25},{22.41,62.02},  {38.83,81.99},{ 4.82,22.56},{15.01,52.96},{41.12,76.82},  { 5.14,35.46},{40.40,78.76},{76.89,122.53},{99.60,164.21},  {17.56,69.70},{15.47,67.74},{79.33,143.39},{61.38,106.24},  {77.09,145.58},{22.38,57.87},{77.00,146.86},{85.47,139.32},  {78.29,125.77},{56.09,113.82},{29.85,57.95},{68.02,114.98},  {99.80,152.56},{56.13,99.68},{50.87,96.14},{70.92,118.34},  {18.13,52.54},{ 9.65,52.74},{21.14,64.53},{ 5.85,35.25},  { 3.90,35.84},{57.70,113.74},{32.65,79.44},{30.78,57.23},  {15.93,47.90},{94.54,158.57},{15.99,48.42},{54.03,97.67},  {94.56,145.55},{48.42,92.14},{33.50,75.93},{75.31,134.44},  { 7.53,33.84},{48.48,81.91},{62.78,135.05},{22.56,62.72},  {31.12,58.49},{30.90,48.51},{48.27,107.01},{29.57,56.55},  {31.84,67.56},{63.07,115.38},{96.22,146.90},{75.96,125.90},  {78.48,132.71},{ 4.47,19.69},{56.83,94.99},{90.74,136.22},  {18.37,45.45},{43.37,88.50},{75.13,127.54},{91.84,139.83},  {66.99,114.37},{35.62,97.15},{14.32,40.17},{35.62,77.26},  {98.70,157.47},{14.60,46.19},{27.33,82.11},{15.48,46.49},  {82.71,139.29},{17.78,59.32},{37.39,90.82},{29.65,66.51},  {14.27,48.09},{38.27,74.89},{69.32,120.78},{ 3.72,41.25},  { 6.44,62.75},{29.18,70.64},{46.02,71.57},{57.14,115.12},  {45.49,85.00},{38.75,82.52},{58.52,107.65},{54.88,99.55},  {71.98,123.01},{37.71,68.39},{43.32,82.62},{79.11,142.63},  {34.48,81.63},{73.53,130.77},{10.70,50.84},{23.54,68.26},  {63.75,124.89},{ 4.50,31.46},{55.35,99.71},{ 2.26, 1.63},  {65.48,121.04},{65.51,130.58},{74.76,130.05},{61.96,113.45},  {22.75,76.09},{12.11,56.20},{60.19,102.29},{27.93,78.04},  {14.21,40.49},{80.85,130.02},{98.75,163.54},{39.58,101.41},  {75.84,132.72},{ 2.21,14.08},{22.68,65.37},{81.91,138.57},  {71.29,114.89},{90.83,164.22},{94.44,151.59},{82.04,131.07},  {13.66,63.96},{48.38,87.90},{46.38,87.25},{22.28,63.31},  { 2.87,32.37},{10.02,58.24},{49.16,100.16},{86.62,135.56},  {39.26,90.93},{78.34,133.91},{82.53,139.45},{59.77,112.37},  {70.98,130.76},{66.60,114.24},{35.82,90.20},{30.53,71.96},  {69.51,139.87},{94.56,173.33},{21.42,59.83},{58.70,111.28},  {37.44,94.48},{31.15,63.11},{23.53,63.70},{ 5.11,63.57},  {55.81,123.51},{15.80,42.37},{83.53,149.47},{80.35,153.86},  {37.73,102.20},{95.31,133.18},{97.78,155.11},{59.12,116.15},  {10.35,41.60},{65.22,107.71},{54.83,108.60},{91.01,151.20},  {78.63,147.74},{51.16,110.76},{70.28,106.57},{70.08,129.60},  {47.41,99.55},{ 0.52,21.99},{54.85,94.95},{93.87,153.82},  {40.84,67.40},{57.23,116.36},{76.08,140.72},{62.88,107.11},  {23.52,58.75},{86.76,141.34},{76.61,131.49},{69.97,129.62},  { 6.16,24.48},{61.86,114.65},{30.69,88.16},{89.57,147.12},  {42.47,86.94},{29.92,69.93},{36.03,83.92},{90.74,139.60},  {32.22,73.11},{10.79,57.18},{28.87,59.02},{47.85,109.31},  {44.50,87.53},{10.85,44.35},{45.82,85.17},{43.53,93.85},  {57.17,103.94},{86.07,142.47},{97.68,151.83},{85.74,147.44},  { 4.78,35.45},{97.96,154.43},{99.31,154.34},{ 6.00,45.64},  {56.05,115.48},{24.98,66.31},{86.32,152.64},{ 1.08,40.11},  {42.92,80.64},{79.59,132.72},{71.87,107.43},{19.35,47.20},  {38.09,92.45},{18.94,60.66},{30.15,60.80},{19.43,53.20},  {63.91,129.49},{54.38,113.42},{42.06,91.30},{ 1.98,41.20},  { 5.47,23.84},{84.77,133.67},{ 4.93,38.23},{84.19,147.77},  {38.91,67.06},{25.87,60.48},{62.61,110.60},{28.58,84.46},  {92.31,152.06},{61.23,92.60},{82.96,125.80},{15.59,59.43},  {34.88,70.07},{13.29,35.70},{30.92,61.47},{93.31,141.05},  {68.91,126.91},{26.63,59.73},{37.41,72.67},{15.63,44.98},  {27.66,76.55},{99.90,164.33},{87.52,144.03},{ 4.42,29.79},  {30.91,59.24},{ 6.37,47.74},{78.59,133.51},{50.65,94.09},  {69.79,136.05},{60.30,120.16},{53.64,109.72},{ 9.80,62.05},  {84.72,134.75},{90.92,131.16},{70.20,126.34},{19.16,45.57},  {52.85,98.88},{69.27,123.71},{99.94,161.32},{92.46,161.95},  {94.75,159.49},{72.82,126.08},{92.27,145.98},{ 5.93,28.08},  {33.26,72.26},{ 2.12,39.38},{12.99,47.88},{57.53,112.68},  {46.70,94.90},{81.13,126.83},{12.80,69.03},{30.96,68.96},  {24.18,59.11},{ 2.27,41.30},{49.74,82.50},{62.55,126.09},  {48.84,95.14},{72.25,120.77},{ 3.22,24.46},{99.21,167.11},  {87.37,133.05},{82.33,144.86},{95.53,163.89},{94.11,145.19},  {13.11,35.64},{59.44,116.19},{24.27,62.07},{91.53,145.26},  {46.43,82.98},{99.89,151.74},{66.41,102.58},{56.46,114.65},  {62.68,99.59},{77.05,132.15},{47.38,81.81},{64.85,107.58},  {91.24,145.20},{65.69,126.13},{66.98,136.61},{ 4.95,29.94},  {75.39,156.04},{ 7.55,35.93},{29.83,62.85},{91.79,140.73},  {66.56,129.57},{36.16,67.39},{41.25,86.72},{94.82,156.68},  {24.15,66.85},{44.28,97.11},{31.82,69.41},{13.75,53.07},  {81.76,135.27},{23.72,77.94},{24.53,53.47},{23.66,67.62},  {21.90,56.35},{31.58,75.84},{31.28,70.78},{42.78,78.57},  {12.46,42.74},{74.68,148.57},{ 2.58,19.05},{91.39,147.46},  {56.50,121.13},{21.06,54.11},{27.09,57.00},{46.82,87.12},  {45.76,90.04},{85.87,149.19},{40.52,84.52},{72.24,118.46},  { 3.34,27.96},{24.68,51.90},{45.54,98.75},{ 9.05,54.03},  {84.14,127.96},{73.69,129.22},{22.43,56.43},{20.47,67.18},  {21.36,81.39},{88.61,147.49},{88.78,126.41},{36.54,90.18},  {23.39,47.90},{16.16,53.46},{34.88,76.16},{75.58,140.32},  {33.45,88.12},{89.01,142.71},{46.57,96.54},{25.00,56.85},  {99.78,171.85},{82.58,152.64},{13.94,52.87},{46.61,112.56},  {64.76,116.36},{31.86,63.96},{69.61,120.21},{53.72,100.82},  {81.88,142.33},{29.39,66.57},{86.67,143.51},{ 4.13,31.53},  {22.34,58.49},{64.54,116.47},{68.08,129.02},{34.02,98.04},  {55.23,104.11},{19.59,64.50},{84.85,156.51},{94.41,142.74},  {12.49,49.71},{27.81,63.84},{53.94,107.71},{92.25,147.58},  {87.89,148.18},{21.02,69.44},{57.05,97.23},{48.46,94.85},  { 3.81,37.26},{89.90,156.01},{57.31,88.22},{78.39,140.66},  {77.93,149.82},{23.15,62.96},{25.77,55.58},{74.11,141.26},  {21.31,64.10},{46.04,79.80},{65.78,117.56},{41.04,79.20},  {94.38,143.18},{81.52,133.84},{86.12,146.57},{39.38,85.36},  {63.01,110.79},{42.25,92.03},{48.83,86.99},{19.09,65.04}  }; |

|  |  |
| --- | --- |
| Original Version | MPI version |
|  |  |

Write a short analysis of the results

# Verbose Repository Log

Paste your verbose format repository log here. With subversion this can be achieved by the following:

svn update

svn –v log > log.txt

gedit log.txt

Then select, copy and paste the text here