**19F-0228**

**Muhammad Zain**

**QUESTION#1**

CODE:

#include<mpi.h>

#include<stdio.h>

#include<stdlib.h>

void findNumber(int\*recvBuff,int recvCount,int key\_value,int rank)

{

int check = 0;

for(int i=0;i<recvCount;i++)

{

if(recvBuff[i]==key\_value)

{

printf("Process: %d found its occurrence at index: %d\n",rank,i);

check = 1;

}

}

if(!check)

printf("No Key Value Found at Process:%d\n",rank);

}

int main(int argv, char \*argc[])

{

int np,rank;

MPI\_Init(&argv,&argc);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&np);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

int size = atoi(argc[1]);

int \*recvBuff = NULL,\*sendCount = NULL,\*displs = NULL,\*localarr = NULL;

int recvCount = size/np;

if(size%np!=0 && rank==np-1)

{

recvCount+=size%np;

}

recvBuff = (int\*)malloc(recvCount \*sizeof(int));

if(rank==0)

{

FILE \*file;

if(size>100)

size = 100;

localarr = (int\*)malloc(size\*sizeof(int));

if((file = fopen(argc[2],"r"))!=NULL)

{

int num=0;

for(int i=0;i<size;i++)

{

fscanf(file,"%d",&num);

localarr[i] = num;

}

sendCount = (int\*)malloc(np \*sizeof(int));

displs = (int\*)malloc(np \*sizeof(int));

for(int i=0;i<np;i++)

{

if(size%np!=0&&i==np-1)

{

sendCount[i] = size/np;

sendCount[i] += size%np;

}

else

{

sendCount[i] = size/np;

}

}

displs[0]=0;

for(int i=1;i<np;i++)

{

displs[i] = displs[i-1] + sendCount[i-1];

}

printf("INITIAL VALUES = ");

for(int i=0;i<size;i++)

printf("%d, ",localarr[i]);

}

else

{

printf("File Not Found!!!\n");

}

}

MPI\_Scatterv(localarr,sendCount,displs,MPI\_INT,recvBuff,recvCount,MPI\_INT,0,MPI\_COMM\_WORLD);

MPI\_Barrier(MPI\_COMM\_WORLD);

printf("\nI AM RANK = %d WITH VALUES = ",rank);

for(int i =0;i<recvCount;i++)

{

printf("%d, ",recvBuff[i]);

}

printf("\n");

findNumber(recvBuff,recvCount,atoi(argc[3]),rank);

MPI\_Finalize();

}

**QUESTION#2**

CODE:

#include<mpi.h>

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct patternStorage{

int count;

int \*arr;

}patternStorage;

patternStorage findPattern(char\*str,char\*pattern,int sindex,int eindex,int patternlength)

{

patternStorage obj;

obj.arr = (int\*)malloc(((sindex+eindex)/patternlength+1) \* sizeof(int));

int k=0;

obj.count = 0;

for (int i = sindex; i <= eindex; i++)

{

int j;

for (j = 0; j < patternlength; j++)

if (str[i + j] != pattern[j])

break;

if (j == patternlength) // if pat[0...M-1] = txt[i, i+1, ...i+M-1]

{

obj.arr[k] = i;

obj.count += 1;

k++;

}

}

return obj;

}

int main(int argc,char \*argv[])

{

int np,rank;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&np);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

FILE \*file;

if((file = fopen("seq.txt","r"))!=NULL)

{

char str2[]="GC";

char str1[1000];

while(fscanf(file,"%s",str1)!=EOF);

int wordlength = strlen(str1);

int patternlength = strlen(str2);

int sindex=0,eindex=0;

int \*recvBuff1,\*recvBuff2;

int globalcount = 0;

int \*displs;

if(rank==np-1)//LAST PROCESS WILL WORK FOR MORE CHARACTERS THEN THE REST

{

sindex = rank\*(wordlength/np);

eindex = wordlength;

}

else

{

sindex = rank\*(wordlength/np);

eindex = ((rank+1)\*(wordlength/np)) - 1;

}

patternStorage obj = findPattern(str1,str2,sindex,eindex,patternlength);

MPI\_Barrier(MPI\_COMM\_WORLD);

if(rank==0)

{

printf("SEQUENCE: %s\nPATTERN: %s\n",str1,str2);

recvBuff2 = (int\*)malloc(np\*sizeof(int));

MPI\_Gather(&obj.count, 1, MPI\_INT,recvBuff2, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

for(int i=0;i<np;i++)

globalcount+=recvBuff2[i];

printf("TOTAL TIMES THE SEQUENCE IS FOUND: %d\n",globalcount);

}

else

{

MPI\_Gather(&obj.count, 1, MPI\_INT,recvBuff2, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

MPI\_Barrier(MPI\_COMM\_WORLD);

if(rank==0)

{

recvBuff1 = (int\*)malloc(globalcount\*sizeof(int));

displs = (int\*)malloc(np\*sizeof(int));

displs[0]=0;

for(int i=1;i<np;i++)

displs[i] = displs[i-1] + recvBuff2[i-1];

MPI\_Gatherv(obj.arr,obj.count, MPI\_INT,recvBuff1, recvBuff2,displs, MPI\_INT, 0, MPI\_COMM\_WORLD);

printf("POSITIONS FOR THE OCCURENCES ARE: ");

for(int i=0;i<globalcount;i++)

printf("%d,",recvBuff1[i]);

printf("\n");

}

else

{

MPI\_Gatherv(obj.arr,obj.count, MPI\_INT,recvBuff1, recvBuff2,displs, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

}

else

{

printf("FILE NOT FOUND!!!\n");

}

MPI\_Finalize();

}

**QUESTION#3**

CODE:

#include<mpi.h>

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

int roundOff(double num)

{

double i;

double fraction = modf(num,&i);

if(fraction<0.5)

return num;

else

return num+0.5;

}

int \*countBins(int\*recvBuff,int recvCount,int \*bins)

{

for(int i=0;i<recvCount;i++)

{

if(recvBuff[i] >=0 && recvBuff[i] <=33)

{

bins[0]+=1;

}

else if(recvBuff[i] >=34 && recvBuff[i] <=50)

{

bins[1]+=1;

}

else if(recvBuff[i] >=51 && recvBuff[i] <=75)

{

bins[2]+=1;

}

else if(recvBuff[i] >=76 && recvBuff[i] <=100)

{

bins[3]+=1;

}

}

return bins;

}

int main(int argc,char \*argv[])

{

int np,rank;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&np);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

int recvCount,size;

int \*recvBuff=NULL,\*sendBuff=NULL,\*sendCount=NULL,\*displs=NULL,\*sendBins,\*recvBins;

sendBins = (int\*)malloc(4\*sizeof(int));

recvBins = (int\*)malloc(4\*sizeof(int));

for(int i=0;i<4;i++)

sendBins[i] = recvBins[i] = 0;

sendCount = (int\*)malloc(np\*sizeof(int));

size = atoi(argv[1]);

int tempRank = np;

int tempSize = size;

if(tempSize%tempRank!=0)

{

for(int i=tempRank;i>=1;i--)

{

if(tempSize%i!=0)

{

double num = (double)tempSize/i;

int range = roundOff(num);

tempSize -= range;

sendCount[i-1] = range;

}

else

{

sendCount[i-1] = tempSize/i;

}

}

}

else

{

for(int i=0;i<np;i++)

{

sendCount[i] = size/np;

}

}

if(rank == 0)

{

srand(time(0));

sendBuff = (int\*)malloc(size\*sizeof(int));

for(int i=0;i<size;i++)

sendBuff[i]=rand()%101;

printf("SEND BUFFER VALUES = ");

for(int i=0;i<size;i++)

printf("%d, ", sendBuff[i]);

printf("\n");

displs = (int\*)malloc(np\*sizeof(int));

displs[0] = 0;

for(int i=0;i<np;i++)

displs[i] = displs[i-1] + sendCount[i-1];

}

MPI\_Barrier(MPI\_COMM\_WORLD);

recvCount = sendCount[rank];

recvBuff = (int\*)malloc(recvCount\*sizeof(int));

MPI\_Scatterv(sendBuff,sendCount,displs,MPI\_INT,recvBuff,recvCount,MPI\_INT,0,MPI\_COMM\_WORLD);

sendBins = countBins(recvBuff,recvCount,sendBins);

MPI\_Reduce(sendBins,recvBins,4,MPI\_INT,MPI\_SUM,0,MPI\_COMM\_WORLD);

if(rank==0)

{

printf("THE FINAL DISTRIBUTION AT P0\n");

for(int i=0;i<4;i++)

{

if(i==0)

{

printf("BIN 1 (0-33): %d, \n",recvBins[i]);

}

else if(i==1)

{

printf("BIN 2 (34-50): %d, \n",recvBins[i]);

}

else if(i==2)

{

printf("BIN 3 (51-75): %d, \n",recvBins[i]);

}

else

{

printf("BIN 4 (76-100): %d, \n",recvBins[i]);

}

}

}

MPI\_Finalize();

}