\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW0\_5

Author: F74071140 謝育萱 <[shirleysensible@gmail.com](mailto:shirleysensible@gmail.com)>

Class: 甲班

Description:

每次壓縮一個or\_array所需的動作:

1. 呼叫lon這個函式（得目前new\_array的bucket數）
2. 計算該or\_array和所有new\_array有甚麼數字，且有幾個
3. 計算有多少個match的數字
4. 將match數由大排到小（呼叫most這個函式）
5. 測試new\_array是否夠存（呼叫store這個函式）
6. 不夠則開新陣列

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include<stdio.h>

#include<stdlib.h>

int or\_array[100][101]={0},new\_array[100][101]={0};

int num\_array[101][100]={0},x[101][102]={0};

void fprint\_result(int or\_array[][101],int new\_array[][101],int n,int lone)

{

int i,j;

FILE\*fout;

fout=fopen("hw5.out","w+t");

if(fout==NULL)

{

printf("fail to open file hw5.out.");

return;

}

for(i=0;i<n;i++) //隨機存取original array

{

fprintf(fout,"%d/",i);

for(j=1;j<or\_array[i][0]+1;j++)

fprintf(fout,"%d",or\_array[i][j]);

fprintf(fout,"\n");

}

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

{

fprintf(fout,"%d/",i);

for(j=1;j<=new\_array[i][0];j++)

fprintf(fout,"%d",new\_array[i][j]);

fprintf(fout,"\n");

}

fclose(fout);

}

int lon(int new\_array[][101]) //計算new\_array存到第lon個bucket

{ int i=0,lone=0;

while(new\_array[i][1]!=-1)

{ i++;

lone++;

}

return lone;

}

int most(int x[][102],int w) //找最多match的陣列

{ int i,j,temp[1][102]={0};

for(i=1;i<w+2;i++)

{

if(w==0)

return 0; //如果w為0則跳出函式

if(x[i][0]<x[i+1][0]) //將較大的數放置左側

{

for(j=0;j<x[i][0]+2;j++)

temp[0][j]=x[i][j];

for(j=0;j<x[i+1][0]+2;j++)

x[i][j]=x[i+1][j];

for(j=0;j<temp[0][0]+2;j++)

x[i+1][j]=temp[0][j];

}

}

most( x,w-1);

}

int con(int new\_array[][101],int lone) //計算new\_array共有幾個數

{

int i=0,sum\_m=0;

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

sum\_m=sum\_m+new\_array[i][0];

return sum\_m;

}

int store(int new\_array[][101],int or\_array[][101],int x[][102],int a,int lone,int p,int m)

{

int i=1,j=1;

if(a>lone)

return 0; //代表已經與所有的new\_array比完,可以開啟新壓縮bucket

int disc=new\_array[(x[a][1])][0]-(or\_array[p][0]-x[a][0]);

if(disc<=m-new\_array[(x[a][0])][0]) //當new\_array還存的下時

{

for(i=1;i<or\_array[p][0]+1;i++) //取出沒有重複的數字並加以儲存

{

for(j=1;j<new\_array[(x[a][1])][0]+1;j++)

{

if(or\_array[p][i]==new\_array[(x[a][1])][j])

break;

new\_array[(x[a][1])][(new\_array[(x[a][1])][0]+1)]=or\_array[p][i];

new\_array[(x[a][1])][0]+=1; //統計new\_array該bucket已存了多少數

}

return 0;

}

}else if(disc>m-new\_array[(x[a][0])][0]) //當new\_array存不下時

{

store(new\_array,or\_array,x,a+1,lone,p,m); //將match數第二大的拿來計算

}

}

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

{

int n=atoi(argv[1]); //n為bucket的數量

int m=atoi(argv[2]); //m為一個bucket內可存的最大數字量

int d=atoi(argv[3]); //數字的範圍為0~(d-1)

int s=atoi(argv[4]); //seed of rand

srand(s);

int i,j,e=1,w=0,r=2,sum=0,lone=0,a=1,p=1,sum\_n=0;

for(i=0;i<=100;i++)

{

for(j=0;j<=101;j++)

new\_array[i][j]=-1;

}

for(i=0;i<n;i++) //隨機存取original array

{

printf("%d/",i);

or\_array[i][0]=(rand()%m)+1; //將該bucket中有幾個數字存到第一格

for(j=1;j<or\_array[i][0]+1;j++)

{

or\_array[i][j]=(rand()%d);

printf("%d",or\_array[i][j]);

}

printf("\n");

}

for(i=0;i<=or\_array[0][0];i++) //將or\_array[0][0]複製到new\_array[0][0]

new\_array[0][i]=or\_array[0][i];

for(p=1;p<n;p++) //從第p個or\_array壓縮

{

lone=lon(new\_array); //取目前已知的壓縮bucket數

sum\_n=con(new\_array,lone); //計算目前已壓縮的數量

for(e=1;e<or\_array[p][0]+1;e++) //計算該or\_array有甚麼數字,且有幾個

{

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

{

if(or\_array[p][e]==i)

num\_array[0][i]++;

}

}

for(j=0;j<lone;j++) //計算所有new\_array有甚麼數字,且有幾個

{

for(e=1;e<new\_array[j][0]+1;e++)

{

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

{

if(new\_array[j][e]==i)

num\_array[j+1][i]++;

}

}

}

for(w=0;w<lone;w++)

{

x[w+1][1]=w; //存此為第w個new\_array的值

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

{

if(num\_array[0][i]!=0&&num\_array[w+1][i]!=0)

{

sum=num\_array[0][i]<=num\_array[w+1][i]?num\_array[0][i]:num\_array[w+1][i];

x[w+1][0]=x[w+1][0]+sum; //計算match總數量

for(j=1;j<=sum;j++)

x[w+1][r]=i; //存重複那些數

r++; //跳下一個空間

}

}

}

most(x,w);

store(new\_array,or\_array,x,a,lone,p,m);

int sum\_m=con(new\_array,lone);

if(sum\_n==sum\_m) //如果不夠存放新的數字的話

{

for(i=0;i<=or\_array[p][0];i++) //開新陣列

{

new\_array[lone][i]=or\_array[p][i];

}

}

}

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

{

printf("%d/",i);

for(j=1;j<=new\_array[i][0];j++)

{

printf("%d",new\_array[i][j]);

}

printf("\n");

}

fprint\_result( or\_array, new\_array, n, lone);

return 0;

}

Compilation:

gcc hw5.c -o hw5

Execution:

./ hw5 5 5 10 4

Output:

0/34

1/37

2/3

3/5444

4/34211

0/3455

1/37

2/3

F74071140@c-2018-2:~/hw5>