CSC 381-34: Proj5B (C++)

Swrajit Paul

Due date: Oct. 11, 2018

```
III. Algorithms
******************************

step 0: inFile ← open input file
      - open all output files
      - numRows, numCols, minVal, maxVal ← read from inFile
      - dynamically allocate deCompressedAry and initializing 0's

step 1: output numRows, numCols, minVal, maxVal to outFile_1

step 2: fistPass_deCompress(deCompressedAry)

Step 3: prettyPrint(deCompressedAry, outFile_2)

Step 4: secondPass_deCompress(deCompressedAry)

step 5: prettyPrint(deCompressedAry, outFile_2)

step 6: outputDecompressImg(deCompressedAry, outFile_1)

Step 7: close all files
```

SOURCE CODE

```
// Author: Swrajit Paul

#include <iostream>
#include <fstream>

using namespace std;

ifstream inFile;
ofstream outFile;
ofstream outFiletwo;

class imageProcessing {

        public:

                int numRows;
                int numCols;
                int minVal;
                int maxVal;
                int** deCompressedAry;

           imageProcessing(string in, string out, string outtwo) {

                inFile.open(in.c_str());
                        outFile.open(out.c_str());
                        outFiletwo.open(outtwo.c_str());
                        inFile >> numRows;
                        inFile >> numCols;
                        inFile >> minVal;
                        inFile >> maxVal;

                        deCompressedAry = new int*[numRows+2];
                        for(int i = 0; i < numRows+2; i++){
                                deCompressedAry[i] = new int[numCols+2]; }// set up the array with proper
rows and cols
                        for(int i = 0; i < numRows+2; i++) {
                                for(int j = 0; j < numCols+2; j++) {
                                        deCompressedAry[i][j] = 0; } }// initialize the array

                        outFile << numRows << " " << numCols << " " << minVal << " " << maxVal <<
endl;

                int row, col;
                while (!inFile.eof()){
                        inFile >> row;
                        inFile >> col;
                        inFile >> deCompressedAry[row][col];
                        }

                }
```

```
int max(int a, int b){
	if (a>b){
		return a;
	}
	else if(b>a){
		return b;
	}
	else{
		return a;
	}
}

void fistPass_deCompress (int** imgAry){
	for(int i = 1; i < numRows+1; i++) {
		for(int j = 1; j < numCols+1; j++) {
			if (imgAry[i][j] == 0){
				if(max(imgAry[i-1][j], imgAry[i][j-1]) > 0){
					imgAry[i][j] = max(imgAry[i-1][j], imgAry[i][j-1]) -1;
				}
			}
		}
	}
}

void secondPass_deCompress (int** imgAry){
	for(int i = numRows; i > 0; i--) {
		for(int j = numCols; j > 0; j--) {
			if (imgAry[i][j] < max(imgAry[i+1][j], imgAry[i][j+1])){
				if(max(imgAry[i+1][j], imgAry[i][j+1]) > 1){
					imgAry[i][j] = max(imgAry[i+1][j], imgAry[i][j+1]) -1;
				}
			}
		}
	}
}

void outputDecompressImg (int** imgAry) {
	for(int i = 1; i < numRows+1; i++) {
		for(int j = 1; j < numCols+1; j++) {
			outFile << imgAry[i][j];
		}
		outFile << endl;
	}
	outFile << endl;
}

void prettyPrint (int** imgAry, string pass) {
	outFiletwo << "the result of " << pass << " decompression";
	for(int i = 1; i < numRows+1; i++) {
		for(int j = 1; j < numCols+1; j++) {
```

```
                        if (imgAry[i][j] == 0)
                            outFiletwo << "  ";
                        else {
                            if(imgAry[i][j] / 10 == 0)
                                outFiletwo << imgAry[i][j] << " ";
                            else
                                outFiletwo << imgAry[i][j];
                        }
                    }
                    outFiletwo << endl;
                }
                outFiletwo << endl;
            }

};

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

    imageProcessing img (argv[1],argv[2],argv[3]);

    img.fistPass_deCompress(img.deCompressedAry);

    img.prettyPrint(img.deCompressedAry, "pass-1");

    img.secondPass_deCompress(img.deCompressedAry);

    img.prettyPrint(img.deCompressedAry, "pass-2");

    img.outputDecompressImg(img.deCompressedAry);

    return 0;
}
```

INPUT

INPUT 1

```
25 40 0 9
9 31 9
10 26 5
10 31 9
10 36 5
13 4 1
13 19 1
14 5 2
14 18 2
15 6 3
15 17 3
16 7 4
16 16 4
17 8 5
17 15 5
18 9 6
18 10 6
18 11 6
18 12 6
18 13 6
18 14 6
19 9 6
19 10 6
19 11 6
19 12 6
19 13 6
19 14 6
20 8 5
20 15 5
21 7 4
21 16 4
22 6 3
22 17 3
23 5 2
23 18 2
24 4 1
24 19 1
```

INPUT 2

```
40 22 0 10
11 12 10
30 12 10
```

OUTPUT

OUTPUT For DATA 1

Output file one

```
25 40 0 9
0000000000000000000000000000001000000000
0000000000000000000000000000012100000000
```

```
0000000000000000000000000000123210000000
0000000000000000000000000001234321000000
0000000000000000000000000012345432100000
0000000000000000000000000123456543210000
0000000000000000000000001234567654321000
0000000000000000000000012345678765432100
0000000000000000000000123456789876543210
0000000000000000000001234556789876554321
0000000000000000000000123445678765443210
0000000000000000000000012334567654332100
0001111111111111111100000122345654322100
0001222222222222222100000011234543211000
0001233333333333332100000000123432100000
0001234444444444432100000000012321000000
0001234555555554321000000000012100000000
0001234566666654321000000000001000000000
0001234566666654321000000000000000000000
0001234555555554321000000000000000000000
0001234444444444432100000000000000000000
0001233333333333332100000000000000000000
0001222222222222222100000000000000000000
0001111111111111111100000000000000000000
0000000000000000000000000000000000000000
```

<u>Output file two</u>

`the result of pass-1 decompression`

```
                                                      9 8 7 6 5 4 3 2 1
                                            5 4 3 2 1 9 8 7 6 5 5 4 3 2 1
                                            4 3 2 1   8 7 6 5 4 4 3 2 1
                                            3 2 1     7 6 5 4 3 3 2 1
1                             1             2 1       6 5 4 3 2 2 1
  2 1                       2 1             1         5 4 3 2 1 1
  1 3 2 1                 3 2 1                       4 3 2 1
    2 4 3 2 1           4 3 2 1                       3 2 1
    1 3 5 4 3 2 1     5 4 3 2 1                       2 1
      2 4 6 6 6 6 6 6 5 4 3 2 1                       1
      1 3 6 6 6 6 6 6 5 4 3 2 1
        5 5 5 5 5 5 5 5 4 3 2 1
      4 4 4 4 4 4 4 4 4 4 3 2 1
    3 3 3 3 3 3 3 3 3 3 3 3 2 1
  2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
the result of pass-2 decompression
1
                                                          1 2 1
                                                        1 2 3 2 1
                                                      1 2 3 4 3 2 1
                                                    1 2 3 4 5 4 3 2 1
                                                  1 2 3 4 5 6 5 4 3 2 1
                                                1 2 3 4 5 6 7 6 5 4 3 2 1
                                              1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
                                            1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
                                          1 2 3 4 5 5 6 7 8 9 8 7 6 5 5 4 3 2 1
                                            1 2 3 4 4 5 6 7 8 7 6 5 4 4 3 2 1
                                              1 2 3 3 4 5 6 7 6 5 4 3 3 2 1
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1           1 2 2 3 4 5 6 5 4 3 2 2 1
      1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1             1 1 2 3 4 5 4 3 2 1 1
      1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1                 1 2 3 4 3 2 1
      1 2 3 4 4 4 4 4 4 4 4 4 4 3 2 1                   1 2 3 2 1
      1 2 3 4 5 5 5 5 5 5 5 5 4 3 2 1                     1 2 1
      1 2 3 4 5 6 6 6 6 6 6 5 4 3 2 1                       1
      1 2 3 4 5 6 6 6 6 6 6 5 4 3 2 1
      1 2 3 4 5 5 5 5 5 5 5 5 4 3 2 1
      1 2 3 4 4 4 4 4 4 4 4 4 4 3 2 1
      1 2 3 3 3 3 3 3 3 3 3 3 3 3 2 1
      1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
      1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Output for Data 2

Output file one

```
40 22 0 10
0000000000000000000000
0000000000010000000000
0000000000121000000000
0000000001232100000000
0000000012343210000000
0000000123454321000000
0000001234565432100000
0000012345676543210000
0000123456787654321000
0001234567898765432100
00123456789109876543210
0001234567898765432100
0000123456787654321000
0000012345676543210000
0000001234565432100000
0000000123454321000000
0000000012343210000000
0000000001232100000000
0000000000121000000000
0000000000010000000000
```

```
000000000001000000000
000000000012100000000
000000000123210000000
000000001234321000000
000000012345432100000
000000123456543210000
000001234567654321000
000012345678765432100
000123456789876543210
0012345678910987654321 0
000123456789876543210
000012345678765432100
000001234567654321000
000000123456543210000
000000012345432100000
000000001234321000000
000000000123210000000
000000000012100000000
000000000001000000000
000000000000000000000
```

<u>Output file two</u>
the result of pass-1 decompression

```
109 8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

```
109 8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

the result of pass-2 decompression

```
          1
         1 2 1
        1 2 3 2 1
       1 2 3 4 3 2 1
      1 2 3 4 5 4 3 2 1
     1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
   1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
 1 2 3 4 5 6 7 8 9 109 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
   1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
     1 2 3 4 5 6 5 4 3 2 1
      1 2 3 4 5 4 3 2 1
       1 2 3 4 3 2 1
        1 2 3 2 1
         1 2 1
          1
          1
         1 2 1
        1 2 3 2 1
       1 2 3 4 3 2 1
      1 2 3 4 5 4 3 2 1
     1 2 3 4 5 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
   1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
 1 2 3 4 5 6 7 8 9 109 8 7 6 5 4 3 2 1
  1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1
   1 2 3 4 5 6 7 8 7 6 5 4 3 2 1
    1 2 3 4 5 6 7 6 5 4 3 2 1
     1 2 3 4 5 6 5 4 3 2 1
      1 2 3 4 5 4 3 2 1
       1 2 3 4 3 2 1
        1 2 3 2 1
         1 2 1
          1
```