CS 700-34 **RunLengthEncode**

Student: Shuhua Song

Due Date:
Soft Copy: 03/22/2020
Hard Copy: 03/22/2020

Submission Data:
Soft Copy: 03/22/2020
Hard Copy: 03/22/2020

Algorithm Steps:

I. EncodeMethod1()

```
Step 0: read numRows, numCols, minVal, maxVal from give input file
        Output numRows, numCols, minVal, maxVal
        set row = 0
Step1:  set col = 0, length = 1, curVal ←get next pixel from input file
        Output row, col, curVal
Step2:  col++
Step3:  nextVal←get next pixel from input file
Step4:  if nextVal == curVal
            Length++
        Else
            output  length
            curVal←nextVal
            length←1
            output row, col, curVal
Step 5: repeat step2-step4 until col > numCol
Step 6: row++
Step 7: output length
Step 8: repeat Step1-Step7 until row > numRow
```

II. EncodeMethod4()

```
step 1: row ← 0
Step 2: col ← 0
        length← 0
Step 3: read nextVal fron input File
step 3: lastVal ← skipZero (inFile, nextVal, zeroCnt)
step 4: output row, col, lastVal to encodeFile
step 5: length++
Step 6: read nextVal fron input File
Step 7: checking if the nextVal == lastVal
Step 8: repeat Step6-Step7 until nextVal != lastVal
Step 9: output length
Step 10: set length = 0, keep increasing col, col++
Step 11: if col >= numCol, set col = 0, and row++
Step 12: output length
```

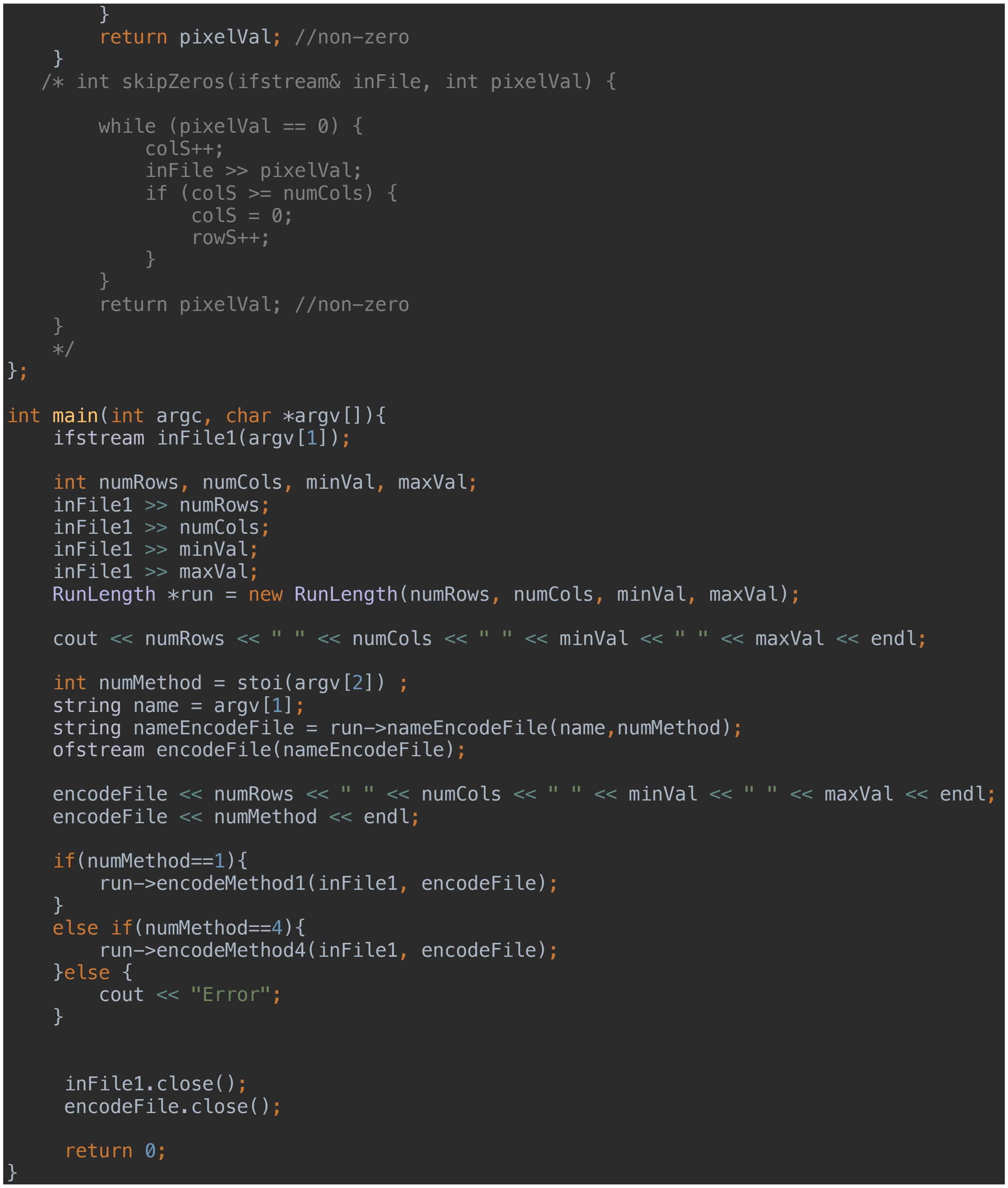
III. skipZeros (inFile, pixelVal, zeroCnt)

```
Step 0: col++
        pixelVal←read the next pixel from input File
Step1: if pixelVal==0 , zeroCnt++
Step2: if col > numCols, col = 0, row++
Step 3: repeat Step 0-Step 2 until pixelVal != 0
Step 4: return pixelVal
```

Code:

```cpp
     #include <iostream>
#include <string>
#include <fstream>
using namespace std;

int rowS;
int colS;
int zeroCnt;
class RunLength {
public:
    int numRows;
    int numCols;
    int minVal;
    int maxVal;

    RunLength(){
        numRows = 0;
        numCols = 0;
        minVal = 0;
        maxVal = 0;
    }

    RunLength(int numRows, int numCols, int minVals, int maxVal){
        this->numRows = numRows;
        this->numCols = numCols;
        this->minVal = minVals;
        this->maxVal = maxVal;
    }
    int whichMethod(int numMethod){
     return numMethod;
    }

    string nameEncodeFile(string name,int numMethod){
     string substring = name.substr(0, 0 + name.size()-4);
     string res = substring +  "_EncodeMethod" +  to_string(numMethod) + ".txt";
     return res;
    }

    void encodeMethod1(ifstream& inFile, ofstream& encodeFile){
        int row = 0;
        while(row < numRows){
            int col = 0;
            int length = 1;
            int curVal;
            inFile >> curVal;
            encodeFile << row << " " << col <<  " " << curVal << " ";
```

```
             while(col < numCols){
                 col = col + 1;
                 if(col < numCols){
                     int nextVal;
                     inFile >> nextVal;
                     if(nextVal == curVal){
                         length++;
                     }else{
                         encodeFile << length << endl;
                         curVal = nextVal;
                         length = 1;
                         encodeFile << row << " " << col <<  " " << curVal << " ";
                     }
                 }
             }
         row = row + 1;
         encodeFile << length << endl;
     }
 }
void encodeMethod4(ifstream& inFile, ofstream& encodeFile){
    rowS = 0;
    colS = 0;
    int length = 0;
    int nextVal;
    inFile >> nextVal;
    while(!inFile.eof()){
        int lastVal = skipZeros(inFile, nextVal, zeroCnt);
        encodeFile << rowS << " " << colS << " " << lastVal << " ";
        length++;
        inFile >> nextVal;
        while(nextVal == lastVal){
            length++;
            colS++;
            if(colS >= numCols){
                colS = 0;
                rowS++;
            }
            inFile >> nextVal;
        }
        encodeFile << length << endl;
        length = 0;
        colS++;
        if(colS >= numCols){
            colS = 0;
            rowS++;
        }
        lastVal = nextVal;

    }
    encodeFile << length << endl;
 }

 int skipZeros(ifstream& inFile, int pixelVal,int zeroCnt) {
     while (pixelVal == 0) {
         colS++;
         if (colS >= numCols) {
             colS = 0;
             rowS++;
         }
         zeroCnt++;
         inFile >> pixelVal;
```



```
        }
        return pixelVal; //non-zero
    }
   /* int skipZeros(ifstream& inFile, int pixelVal) {

        while (pixelVal == 0) {
            colS++;
            inFile >> pixelVal;
            if (colS >= numCols) {
                colS = 0;
                rowS++;
            }
        }
        return pixelVal; //non-zero
    }
    */
};

int main(int argc, char *argv[]){
    ifstream inFile1(argv[1]);

    int numRows, numCols, minVal, maxVal;
    inFile1 >> numRows;
    inFile1 >> numCols;
    inFile1 >> minVal;
    inFile1 >> maxVal;
    RunLength *run = new RunLength(numRows, numCols, minVal, maxVal);

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

    int numMethod = stoi(argv[2]) ;
    string name = argv[1];
    string nameEncodeFile = run->nameEncodeFile(name,numMethod);
    ofstream encodeFile(nameEncodeFile);

    encodeFile << numRows << " " << numCols << " " << minVal << " " << maxVal << endl;
    encodeFile << numMethod << endl;

    if(numMethod==1){
        run->encodeMethod1(inFile1, encodeFile);
    }
    else if(numMethod==4){
        run->encodeMethod4(inFile1, encodeFile);
    }else {
        cout << "Error";
    }


     inFile1.close();
     encodeFile.close();

     return 0;
}
```

(a) Input Image1 File

10 22 0 9

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4
4 0 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 0 0 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 0 0 0 0 0 2 3 4 2 2 3 3 4 4 4 4 4 4 0 0
0 0 0 0 0 0 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Input Image1 File

```
20 22 0 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4
4 0 4 4 4 4 4 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 0 0 3 3 3 3 3 3 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 0 0 0 0 0 2 3 4 2 2 3 3 4 4 4 4 4 4 0 0
0 0 0 0 0 0 1 1 1 1 1 9 9 9 9 9 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 6 6 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 0 0 0 0 0 2 3 4 2 2 3 3 4 4 4 4 4 4 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

(b) `image1_EncodeMethod1`

```
10 22 0 9
1
0 0 0 15
0 15 4 7
1 0 4 1
1 1 0 1
1 2 4 9
1 11 0 11
2 0 0 5
```

```
2 5 3 17
3 0 3 3
3 3 0 2
3 5 3 6
3 11 7 11
4 0 7 22
5 0 7 2
5 2 0 5
5 7 2 1
5 8 3 1
5 9 4 1
5 10 2 2
5 12 3 2
5 14 4 6
5 20 0 2
6 0 0 6
6 6 1 5
6 11 9 5
6 16 1 6
7 0 1 10
7 10 6 12
8 0 0 22
9 0 0 22
```

(c) image1_EncodeMethod4

```
10 22 0 9
4
0 15 4 8
1 2 4 9
2 5 3 20
3 5 3 6
3 11 7 35
5 7 2 1
5 8 3 1
5 9 4 1
5 10 2 2
5 12 3 2
5 14 4 6
6 6 1 5
6 11 9 5
6 16 1 16
7 10 6 12
```

(d) image2_EncodeMethod1

```
20 22 0 9
1
0 0 0 15
0 15 4 7
1 0 4 1
1 1 0 1
1 2 4 9
1 11 0 11
2 0 0 5
2 5 3 17
3 0 3 3
3 3 0 2
3 5 3 6
3 11 7 11
4 0 7 22
5 0 7 2
5 2 0 5
5 7 2 1
5 8 3 1
5 9 4 1
5 10 2 2
5 12 3 2
5 14 4 6
5 20 0 2
6 0 0 6
6 6 1 5
6 11 9 5
6 16 1 6
7 0 1 10
7 10 6 12
8 0 0 22
9 0 0 22
10 0 0 22
11 0 0 22
12 0 0 22
13 0 0 22
14 0 7 22
15 0 7 2
15 2 0 5
15 7 2 1
15 8 3 1
15 9 4 1
15 10 2 2
15 12 3 2
```

```
15 14 4 6
15 20 0 2
16 0 0 22
17 0 0 22
18 0 0 22
19 0 0 22
```

(e) image2_EncodeMethod4

```
20 22 0 9
4
0 15 4 8
1 2 4 9
2 5 3 20
3 5 3 6
3 11 7 35
5 7 2 1
5 8 3 1
5 9 4 1
5 10 2 2
5 12 3 2
5 14 4 6
6 6 1 5
6 11 9 5
6 16 1 16
7 10 6 12
14 0 7 24
15 7 2 1
15 8 3 1
15 9 4 1
15 10 2 2
15 12 3 2
15 14 4 6
```