CS 483 : Programming Assignment (Fall 2018)
Gray-Code Sorting for Data Compression

Read the paper: Dana Richards, "Data Compression and Gray-Code Sorting," *Information Processing Letters*, 22, pp. 201-205, 1986. It was emailed to the whole class.

A type of compression scheme for a file of similar records works as follows: output the first record and for each successive record, output only the fields which differ from the previous record. If most successive records have many identical fields, then the output file should be much shorter than the input file.

All the records of the input file have the same field structure, $m = 20$ fields $f_1, f_2, f_3, \ldots, f_{20}$. Each field has an integer value, between 1 and N_i. You will generate random values for the N_i's in the range 2 to 10. You will create a input "file" of random records fields with 10,000 records. (It is not a real file but is stored in an array of records, and the records are just arrays themselves.)

The *full score* for a file will be computed as follows: for each successive records compute the sum of the (positive) differences of corresponding fields, and for the whole file it is the sum of these numbers. The *binary score* for a file is computed by finding, for each successive records, the number of corresponding fields that are different, and for the whole file it is the sum of these.

Your goal is to sort the input file in Gray-code order. You will implement and evaluate three algorithms. You will need an algorithm A for sorting that runs in $O(n \lg n)$ time; copy one from the book. In particular you should implement these approaches:
(a) use A with the procedure $grayorder(X, Y)$ (mentioned in Section-3)
(b) use A after first calculating the rank of each with the left-to-right "Horner's rule" (mentioned in Section-3) and just use the rank as the sort field (the rank would be a $(m+1)$st field)
(c) use the Radix sorting mentioned in Section-4.

Your will compare these approaches without using time; we use a counting measure instead. Normally we would count comparisons, but radix sort does not make comparisons! So we will count iterations of loops. Every loop will now have a new first statement that increments a (global) counter. You will report for each algorithm the average count over 10 randomized experiments.

In addition, just out of curiosity, you will report the before-and-after scores for the files, both full and binary, for each of the 10 files. This will indicate if data compression is actually improved.

Turn in your code and measures for the different approaches. Your report should be one or two pages, and discuss of the usefulness of the analysis in the paper. Your code should be sufficiently readable that someone grading it can readily determine how you handled the various unspecified algorithmic details (use appropriate comments to make it more comprehensible).

Language: C, Java, Python
Deadline: ~~November 18, 2018, 11:59 PM~~. Email the code and the report to the TA: kkabir@gmu.edu

November 30, 2018, 11:59 PM