CS 3353 Fall 2024

Program 1

Shell sort

Due: 9/27 (Fri) 11:59pm

The goal of this program is for you to implement shell sort (and your own sorting algorithms).

Class MyObj

You are given a class called MyObj which contains objects to be sorted. For the current implementation, the things to be sorted is an array of 3 integers, but practically I can change the content being stored in the class to anything (or use a template construct) and the sorting program you implemented should still work.

Here is the definition of the class:

- Members:
  - Int a[] : an array of 3 integers
  - Static compareCount: number to keep track how many times lessThan() has been called (see below).
  - Static r: random number generator, used internally only.
- Constructor:
  - MyObj(): Create a MyObj object, with the values of the array being random numbers between 0 and 1000
  - MyObj(int x, int y, int z): create a MyObj object with the array value x, y, z respectively
- Methods:
  - Void Update(int z, int y, int z): update the value of the array of the object to x, y, z respectively
  - boolean lessThan(MyObj x): return true if and only if the object is less than x. (See the code for the current definition of less than). Also increment compareCount by 1
  - Int getCount(): return the current value of compareCount
  - Int reset(): set compareCount to 0
  - Void print(): print the object to the standard output

You are given 3 files:

1. MyObj.java : contains the code for the MyObj class
2. Prog1test.java: a driver program for you to test your implementation
3. ShellSort.java : where you should implement shell sort. See below

Things to do

Part 1: Implementation

You are to implement the ShellSort class. The class contain one static function called ShellSort:

- Public static void ShellSort(MyObj[] arr, int code): Implement shellsort to sort the item inside the array.
    - s: the SortClass object
    - code: an integer denoting how the hlist array (as in the source code) is to be formed:
        - 0: hlist should be [1] (i.e. essentially insertion sort)
        - 1: hlist should be [k^2, $(k-1)^2$, …, 4, 1] where k is the maximum number such that k^2 is still (strictly) smaller than the number of objects to be sorted. (Notice that in this case 1 = 4*1 - 3)
        - 2: hlist should be [2^k-1, $2^{(k-1)}$-1, $2^{(k-2)}$-1 … 3, 1] where k is the maximum number such that $2^k - 1$ is still (strictly) smaller than the number of objects to be sorted. (Notice that in this case 1 = 2^1 - 1)
        - 3: hlist should be [(3^k-1)/2, ($3^{(k-1)}$-1)/2 , … 4 1], where k is k is the maximum number such that (3^k-1)/2 is still (strictly) smaller than the number of objects to be sorted. (Notice that in this case 1 = (3^1-1)/2)
        - If the code is anything either than 0,1,2,3, you should print an error message and exit the program immediately.

Part 2: Analysis

You are to analyze the various of option of shell sort by the following

1. Generate 100 different arrays of MyObj of size (n) = 1000
2. For each case, apply ShellSort for each of the four cases (code = 0, 1, 2, 3) [note that you should sort the same array 4 times)
3. Record the number of comparisons for each case.
4. Calculate the average number of comparisons for each of the four cases. Also calculate the standard deviation.
5. Repeat step 1-4 for n = 3000, 5000, 7000, 9000, 11000, 13000, 15000
6. Use a table to record the average/standard derivation of number of swaps of each case for each different
7. Plot two graphs
    - For graph 1, the x-axis is n, and the y-axis is the average number of swaps over the 100 sets. You should plot 4 lines, each line denotes the change of the number of swaps for code = 0, 1, 2, 3 respectively
    - Graph 2 is the same, with the only exception that the x-axis is log(n) and the y-axis is log(average number of swaps) – you can use either base 2 or base 10.
8. Use the information you collected and the graphs to compare the efficiency of shell sort for the four cases – both which one run fastest, and which one is the most efficient asymptotically.

Notice that you may have to modify Prog1Test.java for this task. And you are welcomed to do that.

What to hand in

You should hand in the following 2 files (and 2 files ONLY):

- ShellSort.java: your updated ShellSort class

- Analysis.pdf : a pdf file that store your table and graphs, plus 1-2 paragraphs for your analysis.

Notice that you do NOT need to include any changes to your Prog1Test.java program. And you cannot make any changes to the MyObj.java class.

You should have a zip file that contains both files and upload it to Canvas.