CompSci 165 Project #1
SELECTION

due Apr 18 (W week 3)

Write a routine in C that, using as few element comparisons as possible,
finds the indices of array elements that have the `k` largest values, in descending value order, from an array of `n` elements.

Can you use fewer comparisons than used by other CS 165 teams? Fewer than what my routine uses?

You are required to use the provided `COMPARE` subroutine that makes (and counts) element comparisons.

Requirements

1. Implement a subroutine `doalg( n, k, Best )`
   to find the indices of a private array containing `n` distinct elements that have the `k` largest values, in descending value order,
   by using a sequence of `COMPARE()` instructions, and store the `k` indices in an integer array `Best[]`
   - For example, for `n`=7 and `k`=3,
     if the private array (indexed from 1 to 7) contains the values: `3,5,4,7,2,6,1`
     then the integer array `Best[]` (indexed from 0 to 2) should hold: `4,6,2`
   - Your routine can compare the two array elements at indices `x` and `y` by invoking `COMPARE( x, y )`
     - `COMPARE( x, y )` returns 1, if `array[x] > array[y]`
     - `COMPARE( x, y )` returns 2, if `array[y] > array[x]`
   - Your `doalg( n, k , Best )` routine should
     - be capable of handling values of `n` up to 10000 and values of `k` up to 100
     - normally return 1, but return 0 if any errors occurred
     - be implemented to minimize the worst-case number of element comparisons performed
2. Use the provided main routine which does the following
   - Calls the `doalg( n, k, Best )` routine 1000 times for each of the following values of ⟨`n`,`k`⟩ :
     - `n` = 100, `k` = 10
     - `n` = 10000, `k` = 40
   - Invokes `COMPARE( 0, n )` just prior to each call of `doalg()`
     - This initializes a private array of `n` random distinctly-valued elements (indexed from 1 to `n`)
     - (Note that `n` must be at least 10 and at most 10000)
   - Invokes `COMPARE( -1, k, Best[] )` upon return from each call of `doalg()`
     This
     - checks that `Best[]` does indeed contain the largest `k` array element indices in the correct order
     - returns the number of element comparisons performed if `Best[]` is correct

        - returns a negative value if there were any errors
        - (Note that `k` must be at least 1 and at most 100)
    - Outputs the worst case and average (to two decimal places) number of element comparisons performed for each ⟨`n,k`⟩ pair
3. State the worst case and average number of element comparisons performed by your `doalg( n, k, Best )` routine as a function of `n` and `k`, including all constants (do not use O-notation)
    - first, on the basis of empirical observations (including runs using other values of ⟨`n,k`⟩)
    - second, on the basis of algorithmic theory
    - justify your statements
    - explain any discrepancies between theoretical predictions and empirical observations

Routine Usage

- `COMPARE( int arg1, int arg2, ... )`
    - `COMPARE()` requires `dshrandom()` in the module
        - `dshrandom()` generates random floating point values in the range [0,1)
    - `COMPARE()` can have either 2 or 3 arguments
    - `COMPARE( 0, n )` initializes a private array (indexed from 1 to `n`), with random distinct values
        - `n` must be at least 10 and at most 10000
        - returns 0, normally
        - returns -1, if `n` is out of range
    - `COMPARE( x, y )` compares values of private array elements, `array[x]` and `array[y]`
        - `x` and `y` must be unequal integers, at least 1, and at most `n`
        - returns 1, if `array[x] > array[y]`
        - returns 2, if `array[y] > array[x]`
        - returns -1, if `x` or `y` is out of range
    - `COMPARE( -1, k, Best[] )` checks whether `Best[0], ..., Best[k-1]` contain the indices of the largest `k` values in `array[]`
        - `k` must be at least 1 and at most 100
        - returns the number of comparisons performed when indices of largest `k` values are all correctly given
        - returns -1, if `k` is out of range
        - returns -1000, if any of the indices in `Best[]` are wrong

Deliverables

- `doalg.c`
- If you have any, output produced from runs using other values of ⟨`n,k`⟩
- Worst case and average case analysis (statements and justifications)

Evaluation Process

The grader will perform the following tasks

- Place your `doalg.c` file in his directory that contains the class-supplied files: `MAIN.c` and `COMPARE.c`
- Compile the program on Linux using the command: `gcc MAIN.c -o MAIN`
- View the source code (check for malicious or "cheat" code)
- Run the program using the command: `MAIN > output`
- If the program does not finish execution within three minutes then the grader will abort the execution
- Read analysis document
- Compare performance with those of programs from other teams and the Professor
- Points awarding criteria