ST340 Programming for Data Science
Assignment 1

Released: Friday week 2, 2021-1-22.
Deadline: 12:00 on Monday week 5, 2021-2-8.

Instructions

- Links and instructions on how to submit can be found on the module website.
- Work in groups of at least one and at most three. **Group work is preferred**. Please note that no special consideration for marking will be given to solo work despite more effort.
- Specify your student numbers and names on your assignment. You need submit only one copy for each group.
- Any programming should be in R. Your report should be created using R markdown. Submit two files: a knitted pdf document and the corresponding R markdown file.
- This assignment is worth16% of your overall mark.

---

1. `Mergesort` is a divide and conquer algorithm for sorting an array.

   (a) Devise an algorithm, `merge`, that takes as input two *sorted* arrays and returns a single array with the elements of the two input arrays in sorted order. For example, with inputs $(1, 3, 3, 6, 7)$ and $(2, 4, 5)$ it should output $(1, 2, 3, 3, 4, 5, 6, 7)$. If the lengths of the input arrays are n_1 and n_2, your algorithm should require $O(n_1 + n_2)$ comparisons in the worst case. Describe, implement and test your algorithm.

   (b) The `mergesort` algorithm is recursive. Taking as input an array of length n it calls itself twice, once on the first half of the array and once on the second half of the array (with appropriate modifications if n is odd). It then uses `merge` to combine the output of the two recursive calls into an array, which it then returns. Describe, implement and test the `mergesort` algorithm.

   (c) Prove by induction that `mergesort` outputs its input array in sorted order.

   (d) Let $T(n)$ be the total number of comparisons made by `mergesort` on an input of size n. Write a recurrence inequality for $T(n)$, assuming n is even. Prove by induction that $T(n) \leq n \log_2 n$ for all $n \in \{2^k : k \in \mathbb{N}\}$.

   (e) Compare and contrast `mergesort` with `quicksort` in a few short sentences.

2. A simple version of the *majority element* problem is as follows. Let $\boldsymbol{a}$ be an array of $n = 2^k$ elements and $\equiv$ be an equivalence relation for those elements. We want to find a majority element of $\boldsymbol{a}$, which is an element that occurs in an equivalence class with strictly greater than $\frac{n}{2}$ elements (e.g. if $n = 8$ then the element is equivalent to at least four others plus itself). Furthermore, we want to do this only using the $\equiv$ relation and no $<$ relations, i.e., we cannot order the elements.

(a) Devise a recursive, divide and conquer algorithm to return the majority element x of $\boldsymbol{a}$ if it exists and which returns "no majority" if it doesn't. At each step, your algorithm should make two recursive calls and perform no more than $2n$ equivalence ($\equiv$) tests outside of these recursive calls. Remember to present your pseudo code. Explain why your algorithm works (by proving something).

(b) Provide an upper bound on the number of equivalence checks that your algorithm makes on an input of size n, along the lines of Q1(d).

3. For this question you are encouraged to re-use as much of the code from Lab 2 as possible (include in your report only the parts you have changed). You are asked to compress the $m \times n = 767 \times 584$ matrix representation of Nightingale in `pictures.rdata` in such a way that the accuracy of the approximation is counterbalanced by the space used. You are to use the greyscale representation computed in Lab 2 as the input matrix $\boldsymbol{A}$. Specifically, you are asked to produce an integer $k \in \{1, \ldots, n\}$, vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$, $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_k$ and values $c_1, \ldots, c_k$ such that you minimize the function

$$f(k, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_k, c_1, \ldots, c_k) = \exp\left(\|\boldsymbol{A} - \boldsymbol{B}\|_F\right) + k(m + n + 1),$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and

$$\boldsymbol{B} = \sum_{i=1}^{k} c_i \boldsymbol{b}_i \boldsymbol{d}_i^{\mathsf{T}}.$$

Use R to find the value of $(k, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_k, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_k, c_1, \ldots, c_k)$ that minimizes f. Explain all computations you have done, *including why your answer minimizes the function above.* (You do not need to ensure that the entries of $\boldsymbol{B}$ are in $[0, 1]$, which we did in the lab only to visualize the matrices.) You should not need to compute any matrices of the form of $\boldsymbol{B}$ to find what is required, nor do you need to print any of $\boldsymbol{B}$, $\boldsymbol{b}_i$, $\boldsymbol{d}_i$, or c_i in your submitted answer.