ESO207 Programming Assignment 3 Semester II, 2016-17

Due Date: 31st March, 2017

---

1. (**20 points**) The goal of this assignment problem is to compare the efficiency of QuickSort and MergeSort. In class we have seen that the worst case running time of QuickSort is $O(n^2)$ and that of QuickSort is $O(n \log n)$. However on average QuickSort has a running time of $O(n \log n)$. On the other hand QuickSort can be implemented with $O(1)$ extra space whereas MergeSort requires $O(n)$ extra space.

   Implement QuickSort and MergeSort in the most efficient manner possible and answer the following parts.

   (a) Fill in the following table based on your results:

	$n = 10^2$	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
Average running time of QuickSort					
Average running time of MergeSort					
Average number of comparisons in QuickSort					
Average number of comparisons in MergeSort					
No. of times MergeSort had lesser no. of comparisons than QuickSort					

   Write a paragraph explaining the results that you obtained (max 5 lines).

   (b) Fill in the following table based on your results regarding the QuickSort algorithm:

	$n = 10^2$	$n = 10^3$	$n = 10^4$	$n = 10^5$	$n = 10^6$
Average running time of QuickSort					
Average number of comparisons in QuickSort					
Percentage of cases when running time of QuickSort exceeds average by 5%					
Percentage of cases when running time of QuickSort exceeds average by 10%					
Percentage of cases when running time of QuickSort exceeds average by 20%					
Percentage of cases when running time of QuickSort exceeds average by 30%					
Percentage of cases when running time of QuickSort exceeds average by 40%					
Percentage of cases when running time of QuickSort exceeds average by 10%					
Percentage of cases when running time of QuickSort exceeds average by 100%					

   Write a paragraph explaining the results that you obtained (max 5 lines).

   Note:

   - You must implement both the algorithms in the most efficient manner. In particular, for QuickSort you must implement the *Partition*() procedure that should make a single pass and use only $O(1)$ extra space.
   - While carrying out the above experiment, especially for calculating average, you must make sure that you repeat the experiment sufficiently large number of times. In particular, for a given value of n, you must repeat your experiment on at least 1000 different random arrays. In the report, you must mention very explicitly the number of repetitions.
   - Make sure you use a suitable random number generator.
   - You should use sufficiently precise time (which may count microseconds). For large input size, you should mention time in milliseconds or seconds.
   - While writing the codes of respective algorithms, follow good programming practices (indentation, comments, suitable names for variables).

2. Let $A = \{a_0, a_1, a_2, a_3, ..., a_{n-1}\}$ be an array. We define a recursive operation Op on array A as follows

$$\begin{aligned} Op(A) &= Op(two(A)) + Op(one(A)) + Op(zero(A)) \quad \text{if } n > 1 \\ &= A \quad \text{otherwise} \end{aligned}$$

Here, $zero(A) = \{a_0, a_3, a_6, ..\}$ i.e. an array formed by elements whose indices are divisible by 3. Similarly, $one(A) = \{a_1, a_4, a_7, a_{10}, ...\}$ and $two(A) = \{a_2, a_5, a_8, a_{11}..\}$. Also, + is the concatenation operation.

For example, if $A = \{0, 1, 2, 3, 4, 5\}$. Then $Op(A)$ will be calculated as

$$\begin{aligned} Op(A) &= Op(\{2, 5\}) + Op(\{1, 4\}) + Op(\{0, 3\}) \\ &= Op(\{\}) + Op(\{5\}) + Op(\{2\}) + Op(\{\}) + Op(\{4\}) + Op(\{1\}) + Op(\{\}) + Op(\{3\}) + Op(\{0\}) \\ &= \{5, 2, 4, 1, 3, 0\} \end{aligned}$$

We define an query on an array B as taking the sum of all elements b_k where $i \leq k \leq j$ and $l \leq b_k \leq r$.

(a) (**20 points)** For this part you are given an array C and q queries and to have to perform q queries on $B = Op(C)$.

- First line contains size n of array C. ($n \leq 10^5$)
- Second line contains n integers $c_0, c_1, c_2, ...c_{n-1}$. ($|c_i| < 10^6$)
- Third line contains q, number of queries. ($q \leq 10^5$)
- Next q lines contains four integers i, j, l, r. ($0 \leq i < n,\ i \leq j < n,\ l = -10^6 - 1,\ r = 10^6 + 1$)

You have to output q integers corresponding to each query on a separate line.

(b) (**Bonus: Extra 10 points)** For this part we define $C = \{0, 1, 2, ..., n-1\}$. So, you are given n and q queries and to have to perform q queries on $B = Op(C)$.

- First line contains size n of array C. ($n \leq 10^{15}$)
- Second line contains q, number of queries. ($q \leq 10^5$)
- Next q lines contains four integers i, j, l, r. ($0 \leq i < n,\ i \leq j < n,\ 0 \leq l < n,\ l \leq r < n$

You have to output q integers corresponding to each query on a separate line.

For this question, you have to submit your code on spoj for both sub parts. Detailed formal analysis of time and space complexity is required in the report.