CSCI2100: Regular Exercise Set 6

Prepared by Yufei Tao

Problems marked with an asterisk may be difficult.

Problem 1 (Counting Sort on a Multi-Set—A Linked List Version). Let S be a set of n key-value pairs (k, v). We know that each key k comes from the domain $[1, U]$ where U is an integer. The keys are not necessarily distinct. Design an algorithm to arrange these pairs in non-descending order of key. For example, if $S = \{(35, a), (12, b), (28, c), (12, d), (35, e), (7, f), (63, g), (35, h))$, a possible output of your algorithm is the array $(7, f, 12, b, 12, d, 28, c35, a, 35, b, 35, c, 63, g)$. Your algorithm should terminate in $O(n + U)$ time (note: U may be smaller than n).

Problem 2. Describe an algorithm to achieve the following purposes. Let S be a dynamic set of integers, which is empty at the beginning. Then, you will receive a number of insertions, each of which adds a new integer into S. Eventually, you will receive a signal "end", which indicates that there will be no more insertions. You will need to report the integers in the reverse chronological order, namely, if an integer is received earlier, it should be reported later. If there are n insertions in total, then your algorithm should have total running time $O(n)$. Note that you do not know the value of n until you receive the signal.

For example, if the integers inserted are in this order: 35, 12, 28, 12, 33), then you should produce an array $(33, 12, 28, 12, 35)$.

Problem 3. Describe an algorithm to calculate $x + y$, where x and y are positive integers each of which has n decimal integers. Specifically, the formula is given in an array where the decimal digits of x are followed by $+$, and then followed by the decimal digits of y. For example, the expression $123 + 456$ is given in an array of length 7: $(1, 2, 3, +, 4, 5, 6)$. You should produce the answer in an array that stores the decimal digits of $x + y$. For instance, in the previous example, your output should be $(5, 7, 9)$. Your algorithm must finish in $O(n)$ time.

Problem 4.** Let A be an array of n integers, some of which may be identical. Design a data structure that consumes $O(n)$ space, and supports the following *replacement operation* efficiently:

- `Replacement`(u, v), where u and v are both values in A, The operation replaces all the occurrences of u in A with v.

Your structure must support each such operation in $O(\log n + k)$ time, where k is the number of occurrences of u. Also, you need to design an algorithm to prepare the structure from A in $O(n \log n)$ time.

For example, suppose that $A = (35, 12, 28, 12, 35, 7, 63, 35)$. You can now run your algorithm to create the structure—your algorithm must finish in $O(n \log n)$ time. After this is done, you will be given replacement operations to process. For example, given `replacement`$(12, 28)$, you must update A to $(35, 28, 28, 28, 35, 7, 63, 35)$; for this operation, $k = 2$. Now, given `replacement`$(28, 35)$, you must further update A to $(35, 35, 35, 35, 35, 7, 63, 35)$; for this operation, $k = 3$.

Problem 5* (Textbook Exercise 17.3-7). Suppose that we want to implement the following two operations on a set S of integers (S is empty at the beginning):

- Insert(e): Add a new integer e into S (you are assured that e is not already in S).
- Delete-Half: Delete the $\lceil |S|/2 \rceil$ smallest elements from S.

Describe a data structure that consumes $O(|S|)$ space, and supports any sequence of n operations in $O(\log n)$ time amortized.