Assignment 2
Design and Analysis of Algorithms

Nilanjan Datta

Submission Deadline: 06/03/2022

1. Given a data structure D supports the following sequence operations:

   $$\mathsf{D.insert\_first(x)}, \mathsf{D.delete\_first()}, \mathsf{D.insert\_last(x)}, \mathsf{D.delete\_last()},$$

   each in $O(1)$ time. In addition, D also supports the operations

   $$\mathsf{D.insert\_at(x,i)}, \mathsf{D.delete\_at(i)},$$

   both of which requires $O(\log n)$ time. Device efficient algorithms to implement the following higher level operations using the above lower-level operations:

   (a) $\mathsf{reverse}(D, i, k)$: Reverse in D the order of the k items starting at index i.

   (b) $\mathsf{shift\_left}(D, k)$: Move the first k items in order to the end of the sequene in D.

   (c) $\mathsf{move}(D, i, k, j)$: Move the k items in D starting at index i, in order, to be in front of the item at index j. Assume that the inequalit $i \leq j < i + k$ does not hold.

   Compute the time complexity of each proposed algorithms. Assume that all the delete operations return the value deleted.

2. Asmita is given a data structure D maintaining an extrinsic order on n items, supporting two standard sequence operations: $D.\mathsf{get\_at}(i)$ in worst case $\Theta(1)$ times and $D.\mathsf{set\_at}(i, x)$ in worst case $\Theta(n \log n)$ time. Which of the following sorting algorithm she should choose to sort the n items?

   Insertion Sort, Selection Sort, Merge Sort

3. Suppose Tamojit places n books on top of one another sorted according to the date of publication. Now Rajdeep comes and swaps several pairs of adjacent books. Suggest an efficient sorting algorithm to re-sort the books if Rajdeep performs at most $\log n$ swapping.

4. An integer array A is called (n, k)-type if (i) A contains n distinct integers, (ii) exactly k of them are even, and (iii) the odd integers in A appear in sorted order. Design an efficient algorithm to sort the elements of an $(n, \lceil n/\lg n \rceil)$-type array. Compute the time complexity of your algorithm.