CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Assignment 4 – October 2, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

due October 14, 2020, 10.00am

- Please submit one single pdf, including code printouts and printouts of results from running your code, in UR Courses.
- Please submit your `C++` code (compilable in Visual Studio or g++) through UR Courses in addition, as a single zip file.

Problem 1 (2+4 marks).

(a) Show the result and intermediate steps of inserting keys 1, 2, 3, 4, 5, 6, 7 into an initially empty leftist heap.

(b) For leftist heaps, to perform `buildHeap` in linear time, one can represent each element as a leftist heap with a single node, place these leftist heaps on a queue, and then execute the following: As long as there is more than one heap on the queue, dequeue two of them and then enqueue the merge of the two.
Prove that this algorithm runs in time $O(N)$ in the worst case. (This is textbook problem 6.25(a).)

Problem 2 (4 marks). Prove that a binomial tree B_k always has binomial trees B_0, B_1, …, B_{k-1} as children of the root. (This is textbook problem 6.30.)

Problem 3 (5 marks). Suppose elements `a[i]` and `a[i+k]` are in the wrong order and we swap them. Prove that this will remove at least 1 inversion but at most $2k-1$ inversions. (This is textbook problem 7.3.) Further explain why both the lower bound of 1 and the upper bound of $2k-1$ can be attained for any i, k, where $k > 0$.

Problem 4 (4+5 marks). Hibbard's gap sequence for Shellsort is defined as follows.

$$2^k - 1,\ 2^{k-1} - 1,\ 2^{k-2} - 1,\ \ldots,\ 7,\ 3,\ 1$$

where k is the largest number satisfying $2^k - 1 < N$.

(a) Illustrate Shellsort with Hibbard's gap sequence on the array with the content 77, 17, 66, 19, 30, 24, 64, 14, 23 and count the number of comparisons made.

(b) Implement Shellsort with Hibbard's gap sequence, with Shell's gap sequence, and with a gap sequence of your own invention, and compare them on three input/output examples for duplicate-free arrays of size 10, 100, and 1000 (one of each size). Report the number of comparisons made in each case.