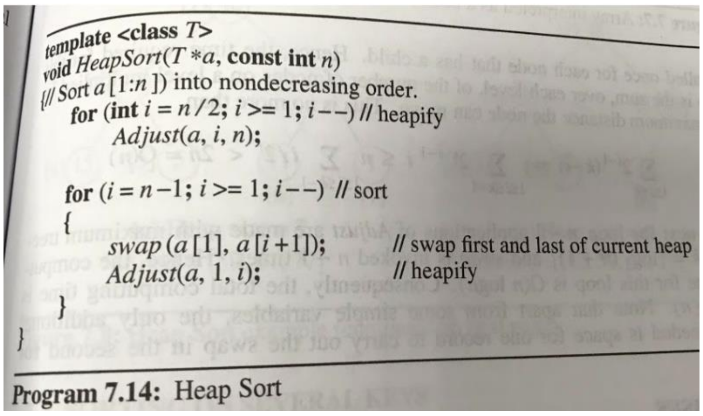
Homework 3

Data Structures and Object-oriented Programming

Instructor: Feng-Jian Wang

Due: 2022/05/22

1. Write a C++ function to implement $BellmanFord$ algorithm. Assume that graphs are represented using adjacency lists in which each node has an additional field called `length` that gives the length of the edge represented by that node. Generate some test graphs and test the correctness of your function.
2. Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of the first **`for`** loop as well as at the end of each iteration of the second **`for`** loop of *HeapSort* (Program 7.14) (No code required)



```
template <class T>
void HeapSort(T *a, const int n)
{// Sort a[1:n]) into nondecreasing order.
   for (int i = n/2; i >= 1; i--) // heapify
      Adjust(a, i, n);

   for (i = n-1; i >= 1; i--) // sort
   {
      swap (a[1], a[i+1]);      // swap first and last of current heap
      Adjust(a, 1, i);          // heapify
   }
}
```

Program 7.14: Heap Sort

3. Give an example of a binary tree that is not a leftist tree. Label the nodes of your binary tree with their *shortest* value. (No code required)
4. Prove that the binomial tree B_k has 2^k nodes, $k \geq 0$. (No code required)
5. Write a C++ function to delete the pair with key k from a binary search tree. What is the time complexity of your function? Test your function and show the results
6. Let T be a tree with root v. The edges of T are undirected. Each edge in T has a non-negative length. Write a C++ function to determine the length of the shortest paths from v to the remaining vertices of T. Your function should have complexity O(n), where n is the number of vertices in T. Test your function and show the results.

7. Into an empty two-pass min pairing heap, insert elements with priorities 20,1,0,5,18,6,12,14,9,8 and 22 (in this order). Show the main pairing heap following each insert. (No code required)
8. Start with an empty AVL tree and perform the following sequence of key insertion: DECEMBER, JANUARY, APRIL, MARCH, JULY, AUGUST, OCTOBER, FEBRUARY, NOVEMBER, MAY, and JUNE. Draw the AVL tree following each insertion and state the rotation type (if any) for each insert. (No code required)
9. Write a C++ function to insert an element into an AVL tree, and another function to list the elements of the AVL Tree in ascending order of key. Test your functions using question number (8)'s list of Keys.
10. Show the result of inserting 8, 9, 12, 10, 15, 4, 1, 13, 17, 3, 5, and 6 into an initially empty B-tree. (No code required)